Curvilinear Triangular Discretization of Biomedical Images

Jing Xu and Andrey N. Chernikov

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

Abstract. 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 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^2 smooth boundaries while keeping all elements valid as measured by their Jacobians. The technique is demonstrated with examples.

Keywords: biomedical image processing, high-order mesh generation

1 Introduction

Discretization 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 [12] 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 properties of objects of interest. Another, emerging, area is the use of discretization 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 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 [10, 11, 4].

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 our previous work [10, 11] we

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 the C^2 smoothness requirement, i.e., the first and second derivatives are continuous.
2. A new procedure was developed to efficiently verify the validity. It is formulated to work in an arbitrary polynomial order.
3. Our proposed approach is robust in the sense that all the invalid elements are guaranteed to be 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 cubic spline curve. Once our validity verification procedure detects invalid elements, the meshing procedure 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 been developed and implemented to accomplish the generation of a curvilinear mesh. Sherwin and Peiro [8] 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. However, these strategies are intuitive solutions that are not guaranteed to generate valid high-order meshes. The mesh spacing is decided by a user defined tolerance ϵ related to the curvature and a threshold to stop excessive refinement. Persson and Peraire [6] 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. [5] 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 [7] 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. When the polynomial degree is higher, it is complicated to calculate the *Jacobian*, so we develop a procedure suitable for a polynomial of any degree. Furthermore, none of the above algorithms generates C^1 and C^2 smooth mesh boundaries.

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 Bézier curves and Bézier triangles

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

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

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

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

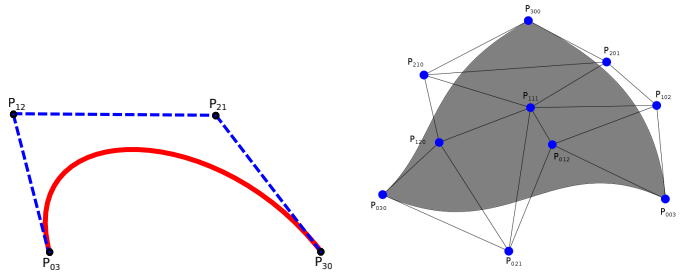
with

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

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 a two-dimensional Euclidean space, and $t \in [0, 1]$, set

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

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.



(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.

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,

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

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

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

where

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

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

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

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

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

$$\begin{aligned}\mathcal{T}^3(u,v,w) &= \sum_{i+j+k=3} B^3_{ijk}(u,v,w)P_{ijk} \\ &= P_{300}u^3 + P_{030}v^3 + P_{003}w^3 + 3P_{201}u^2w + 3P_{210}u^2v \\ &\quad + 3P_{120}uv^2 + 3P_{102}uw^2 + 3P_{021}v^2w + 3P_{012}vw^2 + 6P_{111}uvw.\end{aligned} \tag{1}$$

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

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 global smooth boundary. Fig. 2 illustrates the main steps performed by our algorithm. The details are elaborated below.

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

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

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

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

The two-sided distance is:

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

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.

We transform the linear boundary edges followed by curving the interior edges to eliminate the invalid elements. Bézier curve basis is selected because

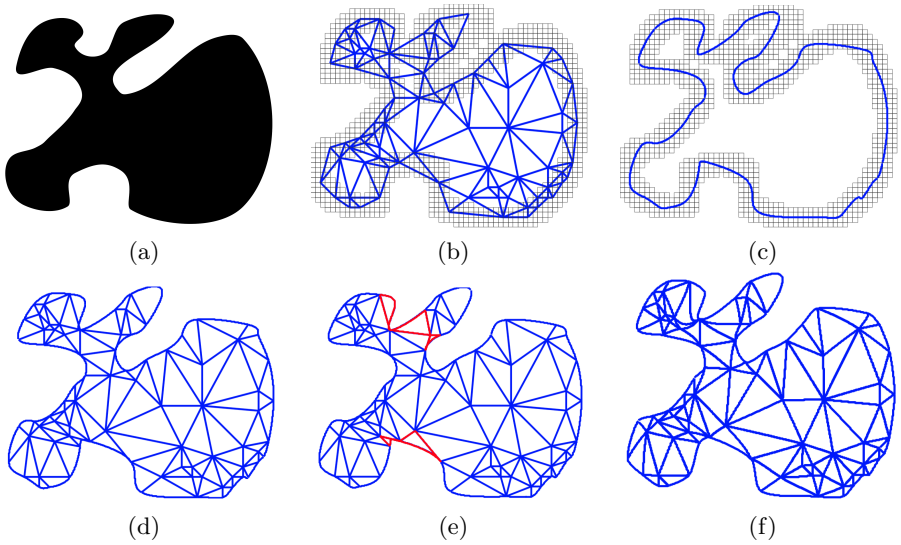


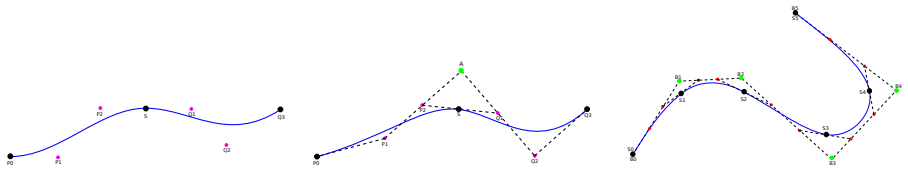
Fig. 2: An illustration of the main steps performed by our algorithm. (a) The input two-dimensional image. It shows a curvilinear domain to be meshed. (b) A linear mesh which satisfies the user specified quality and fidelity tolerances. The shaded region represents the fidelity tolerance. (c) Those edges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

its mathematical descriptions are compact, intuitive, and elegant. It is easy to compute, easy to use in higher dimensions (3D and up), and can be stitched together to represent any shape.

Smoothness of the resulting curve is assured by imposing one of the continuity requirements. 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:

C^0: The curves touch at the joint point;
C^1: First derivatives are continuous;
C^2: First and second derivatives are continuous.

A cubic polynomial is the lowest degree polynomial that can guarantee a C^1 or a C^2 curve. Biomedical objects usually have naturally smooth boundaries, and can be approximated by either a C^1 or a C^2 curve. In our previous work [9], we present how to construct a C^1 Bézier curve; in the following we will address how to generate a C^2 Bézier curve. By counting incorrectly classified pixels (i.e.,



(a) (b) (c)

Fig. 3: 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.

inside vs. outside the shape) in the final mesh, the most suitable curve can be determined.

We aim to find a cubic spline curve passing through all the mesh boundary points given in order. It 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. 3a gives an example of a cubic spline curve.

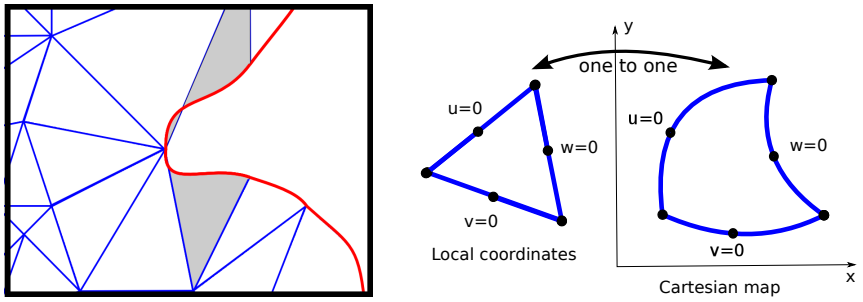
If two Bézier curves with control points P_0, P_1, P_2, S and S, Q_1, Q_2, Q_3 are touched at point S, both their first and second derivatives match at S if and only if their control polygons fit an A-frame, which is a structure in which P_2 is the midpoint of $\overline{AP_1}$, Q_1 is the midpoint of $\overline{AQ_2}$ and S is the midpoint of $\overline{P_2Q_1}$ as Fig. 3b 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. 3c, 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.

The naive high-order mesh generation does not ensure that all the elements of the final curved mesh are valid. Fig. 4a gives an example of this critical issue: some of the curvilinear triangular patches have tangled edges. Thus, it is necessary to verify the validity and eliminate all the invalid elements as a post-processing step once the curved mesh has been constructed. When elements of



(a) (b)

Fig. 4: (a) An example of invalid mesh. 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. (b) 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: an one-to-one correspondence between Cartesian and curvilinear coordinates.

the basic types will be 'mapped' into distorted forms, a general principle is that an one-to-one correspondence between Cartesian and curvilinear coordinates can be established (illustrated in Fig. 4b).

The Jacobian matrix carries important information about the local behavior of the transformation from linear elements to curved elements. A violation of the condition that the determinant of the Jacobian Matrix is strictly positive everywhere means the violation of the bijection general principle. One way to detect the element validity is evaluating the sign of the determinant of the Jacobian matrix throughout the element. In our previous work [9], we took advantage of the properties of Bézier triangle, formulated the *Jacobian* expression and calculated the tight lower bound of the *Jacobian* value at the element level. However, although we used the third order polynomial, it is computationally and geometrically complicated. To get the tight bound, we recursively refined the convex hull of the *Jacobian* (it is a forth order Bézier triangle) using the Bézier subdivision algorithm. In this paper, an efficient element validity verification procedure is developed for polynomials of arbitrary order. An element is invalid if and only if the control net of the Bézier triangle is twisted, meaning that at least one of the control triangles (shadowed triangles in Fig. 5) of the control net is inverted.

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, 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 [12]. 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.

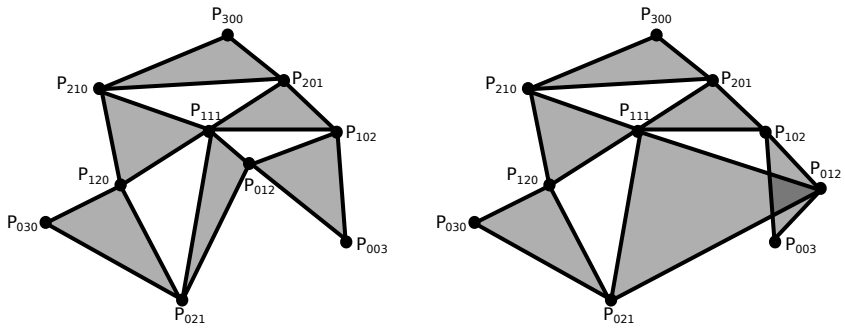


Fig. 5: (a) A valid control net with the uninverted control triangles. (b) A twisted control net with one inverted control triangle $\triangle P_{012}P_{102}P_{003}$.

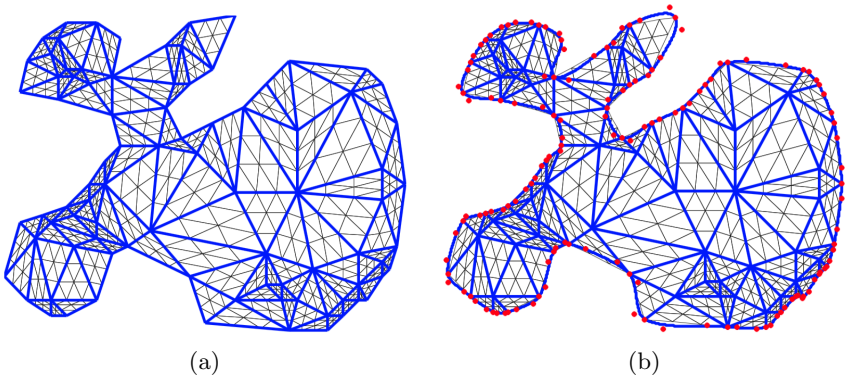


Fig. 6: (a) The control nets of the linear mesh elements is the undeformed geometry. (b) The red control points of the smooth curved boundary edges are the external loadings.

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. 6a). The external loadings are the control points (red points in Fig. 6b) 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. 6 illustrates these steps.

In most 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

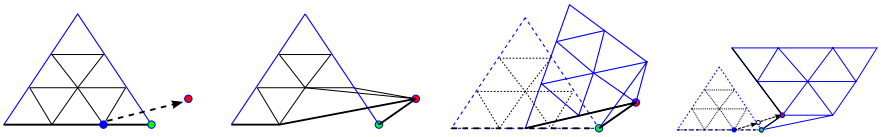


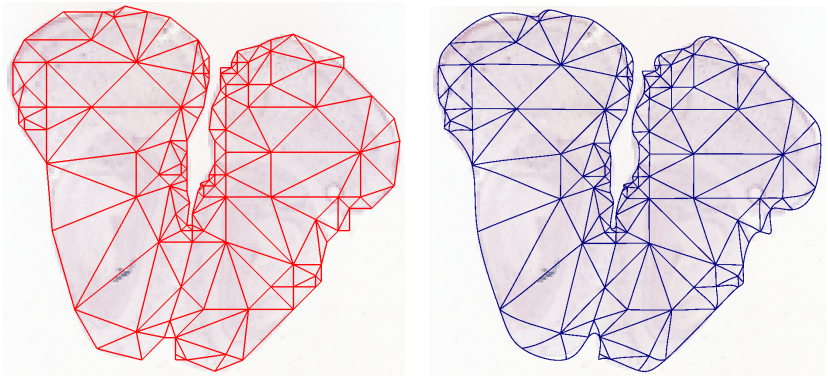
Fig. 7: An illustration of the iterative finite element method. (a) In this example there is only one element in the mesh, the black border line represents the mesh boundary, the position of the blue point represents the original position of one control point of the linear mesh. (b) The position of the red point is the new position of the blue point after deformation. The green point is one of the endpoints of the boundary edge, thus it has to maintain its position. The control net is invalid because there exists an inverted triangle. (c) The one step FEM method was applied, the blue point was directly moved to the position of the red point. After solving for the equilibrium configuration, the control net is still twisted. (d) The iterative way: make the position of the yellow point as the intermediate target, first move the blue point to the position of the yellow point, then move to the position of the red point. The two step FEM method successfully corrected the twisted control net.

invalidity. The iterative finite element method executes the validity check before each round. When it is reported that an invalid element exists, the procedure divides the segment formed by the original position and the new position into two subsegments. The procedure takes the positions of the endpoints of the subsegments one by one as the intermediate targets, and takes the solution of the current target as the input of the next target. The algorithm terminates when all the invalid elements are corrected. Fig. 6 shows an example.

4 Mesh examples

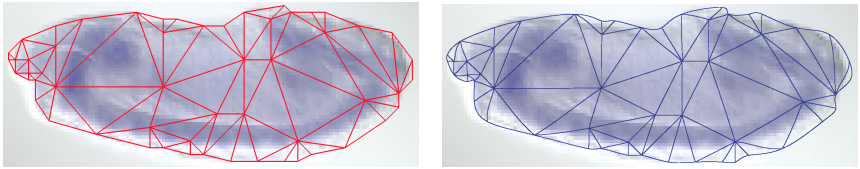
The input data to our algorithm is a two-dimensional image. The procedure for mesh untangling and quality improvement was implemented in MATLAB. All the other steps were implemented in C++ for efficiency.

We meshed a region of a slice of mouse brain atlas [1] and a region of a fruit fly embryo [2]. The size of the first input is $2550 * 2050$ pixels, the size of the second input is $1900 * 950$ pixels. Each pixel has side lengths of 1 unit in x and y dimensions, respectively. In each example, we show the linear mesh result and the high-order mesh result (see Fig. 8 and Fig. 9). For both examples, the fidelity tolerance for the linear mesh was specified by two pixels and the angle quality bound was specified by $20°$. In the first example, after boundary edges were curved, there were two invalid elements in the mesh interior. In the second example, the number of invalid elements is one. After the mesh untangling procedure, all the invalid elements were corrected in both examples. The incorrectly classified pixels (include both background pixels in the mesh elements and tissue



(a) (b)

Fig. 8: Meshing results for a slice of mouse atlas [1]. (a) Linear mesh with two pixels fidelity tolerance and 19° angle bound. (b) Final curvilinear mesh in which all the elements are valid.



(a) (b)

Fig. 9: Meshing results for a fruit fly embryo [2]. (a) Linear mesh with two pixels fidelity tolerance and 19° angle bound. (b) Final curvilinear mesh in which all the elements are valid.

pixels outside the mesh) of the high-order mesh in both examples were improved about 10 percent compared to that of the linear mesh.

5 Conclusion

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

Our future work will include the run time improvement. Compared to the linear mesh generation we present in Section 3, the construction of the high-order mesh is slow. One reason of the inefficiency is the hybrid code we implemented both in MATLAB and C++. The most critical reason is that, in the procedure of invalid mesh correction, we can not anticipate how many iterations we need to eliminate all the invalid elements. As a result, we repeat this procedure until the suitable number of iteration is found. Parallelization may also be used later to enhance the efficiency.

The other concern is further improving the accuracy. Our next step is designing a more suitable linear mesh generation algorithm for curvilinear discretization. For example, besides the two-sided Hausdorff fidelity requirement, if we warp all the mesh boundary vertices to the image boundary, after smoothing, the high-order mesh boundary will be naturally approaching to the boundary of the biomedical objects.

6 Acknowledgments

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

References

1. Allen brain atlas, 2014.
2. Berkeley drosophila genome project, 2014.
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. 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.
6. 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.
7. P.L.George and H.Borouchaki. Construction of tetrahedral meshes of degree two. *Int. J. Numer. Mesh. Engng*, 90:1156–1182, 2012.
8. S.J. Sherwin and J. Peiro. Mesh generation in curvilinear domains using high-order elements. *Int. J. Numer*, 00:1–6, 2000.
9. 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.
10. 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.
11. 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.
12. O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals, 6th edition*. Oxford: Butterworth-Heinemann, 2005.