A Sketch-based Interface for Modeling Heart Fiber Orientation

Kenshi Takayama[1], Takeo Igarashi[2], Ryo Haraguchi[3] and Kazuo Nakazawa[3]

[1] Department of Computer Science, The University of Tokyo,
Bunkyo-ku, Tokyo, Japan
kenshi@ui.is.s.u-tokyo.ac.jp
[2] Department of Computer Science, The University of Tokyo / PRESTO, JST
takeo@acm.org
[3] National Cardiovascular Center Research Institute
Suita, Osaka, Japan
{haraguch, nakazawa}@ri.ncvc.go.jp

Abstract. This paper proposes a sketch-based interface for modeling muscle fiber orientation of a 3D virtual heart model. Our current target is electrophysiological simulation of heart and fiber orientation is one of the key elements to obtain faithful simulation results. The interface and algorithm are designed based on the observation that fiber orientation is always parallel to the heart surface. The user specifies the fiber orientation on the surface by drawing a freeform stroke on the object surface. The system first builds a vector field on the surface by applying Laplacian smoothing to the mesh vertices and then builds volumetric vector field by applying Laplacian smoothing to the voxels. The usefulness of the proposed method is demonstrated through a user study with a doctor in that area.

1 Introduction

Many people suffer from abnormal cardiac rhythm and effective treatment is much desired. An approach to understand the mechanism of this disease is electrophysiological simulation of heart. Various kinds of parameters are required for this simulation, and muscle fiber orientation is one of the key elements that determines the behavior of signal propagation [1] [2].

Fiber orientation of a living heart is difficult to measure non-invasively, so it is necessary for a doctor to manually design fiber orientation based on his or her expert knowledge. A simple method is to have the user set orientations on discrete slices but it is very difficult to design complicated orientation field using such crude methods. To support this process, we present a sketch-based interface for designing a volumetric vector field that represents muscle fiber orientations inside of a given 3D heart model. We designed the system based on the observation that muscle fiber orientation is parallel to the model's surface. This makes it possible to use simple sketching on the surface as input and to use two-step interpolation (surface and volume) scheme for the construction of volumetric vector field.

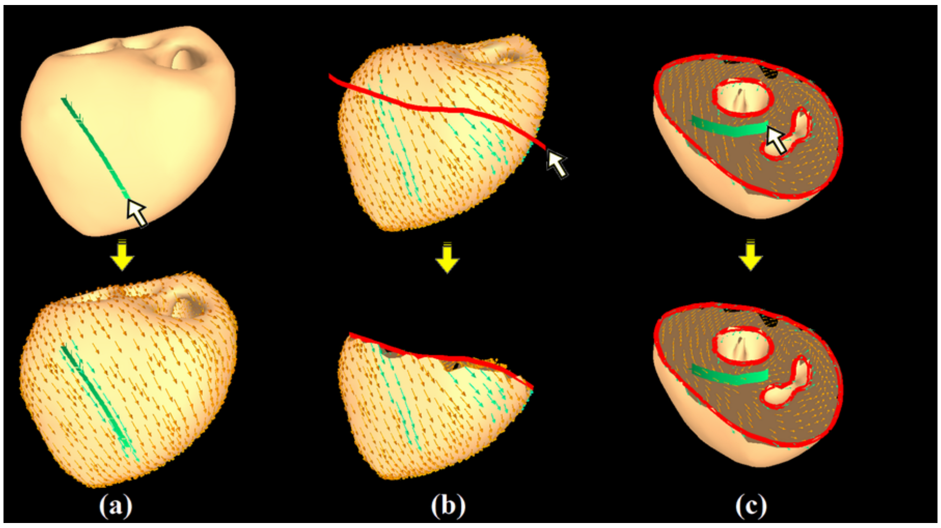

Fig. 1. Designing fiber orientations by sketching. (a) Drawing a stroke on the surface to specify the orientations on the surface. (b) Cutting the model to see the orientations inside the model. (c) Drawing a stroke on the cross-section surface to further control the orientation inside the model.

Figure 1 shows snapshots of our prototype system. The user can draw strokes on the model with common 2D input device such as a mouse or pen. The system applies two-step interpolation to these user-drawn strokes obtaining the volumetric vector field. We use Laplacian smoothing method for the interpolation to enable interactive trial-and-error design. Using our method, the user can design volumetric vector fields quickly and easily.

We asked a doctor in the area of cardiology to test our system and got positive feedback. He appreciated the ability to design fiber orientations quickly and confirms that the proposed system can be a useful tool for practical applications. We also run sample electrophysiological simulation using the heart fiber model created by him to demonstrate the capability of the system.

2 Related Work

Various studies have been done on the *analysis* and *visualization* of vector fields of various kinds [4]. On the other hand, studies on the *design* of vector fields are relatively few. Here we introduce some of the existing methods for designing vector fields.

Salisbury et al. [6] made a simple tool for designing vector fields on 2D plane for the purpose of rendering 2D image with orientable textures. Its interface was more like that of ordinary 'Paint' applications with operations such as 'draw', 'blur' and 'fill'. Praun et al. [5] and Turk [8] used vector fields on surfaces of 3D models to synthesize textures on surfaces. In those papers, they let the user

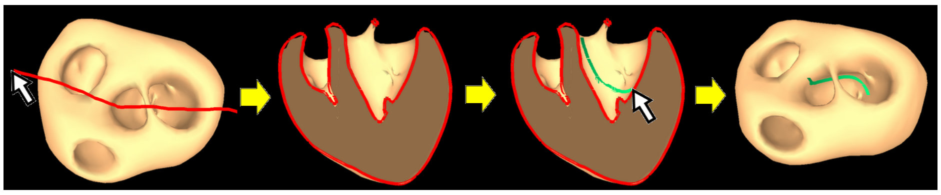

Fig. 2. Drawing a stroke on the internal surface by cutting off the interfering part of the surface.

specify vector values on some of the vertices of the 3D model and assigned interpolated vector values to the remaining vertices. Praun used Gaussian radial basis functions technique for interpolation, while Turk used mesh hierarchy technique.

Topology of vector field is often very important for certain applications. Zhang et al. [9] showed a novel method for designing vector fields on 2D planes and 3D surfaces with consideration of topology.

To our knowledge, however, there has been no study on design of volumetric vector fields. We propose a sketch-based interface for designing volumetric vector fields using two-step approach.

3 User Interface

The system first loads a 3D polygonal model specified by the user. After several precomputations including polygon-to-voxel conversion and calculation of Laplacian matrices, the user can design volumetric vector fields using the sketch-based interface.

The user draws freeform strokes on the surface of the model to specify the local fiber orientation on the surface (Fig. 1a). The user can draw arbitrary number of strokes on the surface to specify the fiber orientation field in detail. The user can cut the model by drawing a crossing stroke. The stroke is extruded to the viewing direction and the system hides the part of the model on the left hand side of the extruded surface (Fig. 1b). The user can also draw strokes on this cross-section surface to specify the fiber orientation inside of the model (Fig. 1c). Cutting is also useful for drawing strokes on the internal surfaces (Fig. 2).

Receiving these user-drawn strokes as inputs, the system performs two-step Laplacian smoothing to obtain the resulting volumetric fiber orientation field. The computation completes within a few seconds and the user can incrementally add or remove strokes until obtaining a satisfactory result.

4 Algorithm

4.1 Overview

When the 3D model is loaded, the system first converts the surface model into voxels using a standard scanning method. Note that each boundary voxel is associated with its neighboring polygon, which is required when setting constraints on the vector field of the volume from vector field of the surface.

The vector field of the surface is defined as unit orientation vectors associated with the mesh vertices, while the vector field of the volume is defined as unit orientation vectors associated with the voxels. After receiving user-drawn strokes as input, the system sets the orientation vector of mesh vertices near the stroke to the direction parallel to the nearest stroke direction (Fig. 3). The system then applies Laplacian smoothing to the surface mesh by using the vertices near the input strokes as constraints (Fig. 1a). The system then sets the orientation vector of boundary voxels to the blending of nearby mesh vertices (Fig. 4). The strokes drawn on the cross-sections are also mapped to the neighboring voxels (Fig. 5). The system finally applies Laplacian smoothing to the inside voxels to obtain the final volumetric vector field (Fig. 1c).

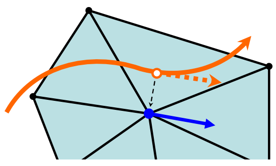

Fig. 3. Orientation vector of n
the stroke and used as constrai

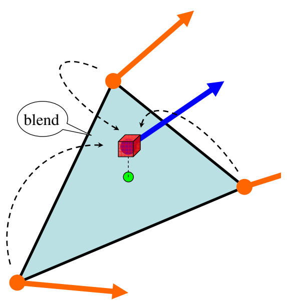

Fig. 4. Setting vector of a boundary voxel to the blending vector of its neighboring vertices.

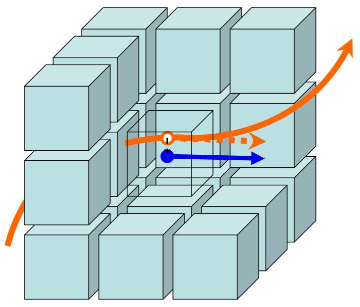

Fig. 5. Orientation vector of voxel near the stroke drawn on the cross section is set to parallel to the stroke and used as additional constraints for applying interpolation to the volume.

Note that the magnitudes of vectors are not considered in the system because our purpose is only to design orientation. Therefore, the system normalizes all the vectors after each smoothing. The interpolated vectors on the surface may not be tangent to the surface. The system therefore projects the vectors associated to the mesh vertices to their tangential planes after each interpolation on the surface.

4.2 Laplacian Smoothing

Here we briefly describe how our Laplacian smoothing works (more details are available in [7]). Let $x_1, \cdots, x_n$ be orientation vectors associated with mesh vertices or voxels. Laplacian of x_i is defined as

$$\delta_i = x_i - \sum_{j \in N_i} w_j^i x_j \tag{1}$$

where N_i is the neighbors of x_i (i.e., 1-ring of the i-th mesh vertex or voxels adjacent to the i-th voxel). We simply set weights as $w_j^i = \frac{1}{|N_i|}$, meaning that δ_i is the difference between x_i and the average of its neighbors. Our goal is to minimize these Laplacians in the least-squares sense while satisfying given constraints:

$$x_{k_i} = b_i \quad (i = 1, \cdots, m) \tag{2}$$

where k_i is the index of the i-th constraint and m is the number of constraints. In the case of the mesh vertices, constraints are given at the vertices near the input strokes (Fig. 3). In the case of the voxels, constraints are given at the voxels near the surface (Fig. 4), as well as those near the additional strokes drawn on the crosssections (Fig. 5).

This goal can be rewritten using vectors and matrices as

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \left| \begin{pmatrix} L \\ C \end{pmatrix} \boldsymbol{x} - \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{b} \end{pmatrix} \right|^2 \right\}$$

where $\boldsymbol{x} = (x_1, \cdots, x_n)^{\mathrm{T}}$ and $\boldsymbol{b} = (b_1, \cdots, b_m)^{\mathrm{T}}$ are vectors, and $L = (l_{ij})$ and $C = (c_{ij})$ are $n \times n$ and $m \times n$ matrices respectively defined by

$$l_{ij} = \begin{cases} -1 & (i = j) \\ w_j^i & (j \in N_i) \\ 0 & (\text{otherwise}) \end{cases}$$

$$c_{ij} = \begin{cases} 1 & (j = k_i) \\ 0 & (\text{otherwise}) \end{cases}$$

This corresponds to solving the following system

$$A^{\mathrm{T}} A \boldsymbol{x} = A^{\mathrm{T}} \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{b} \end{pmatrix} \tag{3}$$

where $A = \begin{pmatrix} L \\ C \end{pmatrix}$. Matrix $A^{\mathrm{T}}A$ and vector $A^{\mathrm{T}} \begin{pmatrix} \mathbf{0} \\ \boldsymbol{b} \end{pmatrix}$ can be rewritten in a simple form as

$$\begin{aligned} A^{\mathrm{T}}A &= (L^{\mathrm{T}}\ C^{\mathrm{T}}) \begin{pmatrix} L \\ C \end{pmatrix} \\ &= L^{\mathrm{T}}L + C^{\mathrm{T}}C \\ A^{\mathrm{T}} \begin{pmatrix} \mathbf{0} \\ \boldsymbol{b} \end{pmatrix} &= (L^{\mathrm{T}}\ C^{\mathrm{T}}) \begin{pmatrix} \mathbf{0} \\ \boldsymbol{b} \end{pmatrix} \\ &= C^{\mathrm{T}}\boldsymbol{b} \end{aligned}$$

Note that the k_i-th diagonal element in $C^{\mathrm{T}}C$ is 1 and the k_i-th element in $C^{\mathrm{T}}\boldsymbol{b}$ is b_i (for $i = 1, \cdots, m$) and all the other elements in $C^{\mathrm{T}}C$ and $C^{\mathrm{T}}\boldsymbol{b}$ are 0.

We solve equation (3) for the mesh vertices every time the user draws a stroke on the surface. Since Laplacian matrix L remains constant for a given mesh, we can precompute $L^{\mathrm{T}}L$ and add 1 to the constrained diagonal elements when solving. In the case of Laplacian smoothing for the voxels, $A^{\mathrm{T}}A$ remains constant as long as the user draws strokes only on the mesh surface, and changes only slightly when the user adds a stroke on crosssection. This matrix is sparse enough to apply an optimized algorithm for sparse matrices. We currently use a fast sparse matrix solver based on LU-decomposition [3].

5 Results and User Experience

Current prototype system is implemented in C++ using OpenGL and runs on Windows PCs. Figure 6 shows some fiber orientations designed using the system. It took about 3 seconds in total to compute a volumetric vector field from user specified strokes for a heart model with 1,992 vertices and 7,646 voxels using a PC with 2.1 GHz CPU and 2.0 GB RAM.

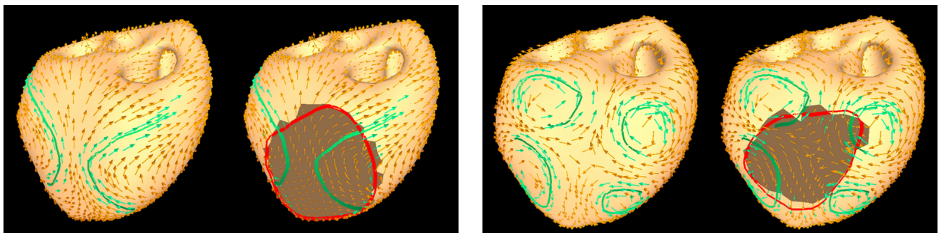

Fig. 6. Example fiber orientations designed using the system.

We asked a doctor in the area of cardiology to try our system in order to obtain feedback for the system. He designed a fiber orientation for a given heart model using our system. The test was performed using a standard laptop PC and a mouse. We used a commercially available 3D polygonal model of heart for this test. We first gave a brief tutorial and he got used to it in about 10 minutes. He then started to design a complete heart fiber orientation and finished it in about 8 minutes. Figure 7 shows the fiber orientations he designed and a sample result of electrophysiological simulation using this model.

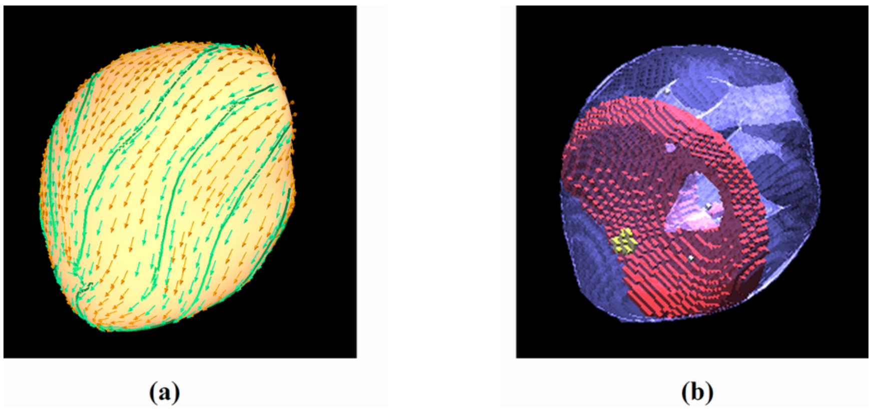

Fig. 7. (a) Heart fiber model designed by the doctor using our system. (b) Sample result of electrophysiological simulation using (a).

We then interviewed him and obtained following feedback. First, he evaluated our system as significant contribution to his research area, because it is the first system that allows the user to directly design 3D fiber structures. Existing methods forced the user to work on 2D slices to specify a 3D vector field and it was very tedious. He was pleased with the resulting heart fiber orientation he created using the system, as it successfully represented typical twisted structure of heart fibers. He stressed that our system is definitely faster than existing methods, even if it may take considerable time for calculation as the number of voxels increases.

It is important, he also noted, that our method defines volumetric vector field using 3D surface geometry and strokes, because this approach indicates the possibility that the user can quickly generate another volumetric vector field from existing one by simply deforming the geometry (which is not yet supported in the current implementation).

He gave us some suggestions for further improvements. First, he noted that it would be great if the user can design heart fiber orientation using some sample images of actual medical data mapped onto the model's surface. This will allow the user to create far more realistic fiber orientation by tracing such sample images. He noted that *tracing* is an important operation in the medical sense, since it achieves human filtering of noisy medical data.

He also pointed out that cross-sectioning is not suitable for the visualization of heart fiber orientation because actual heart fiber consists of number of layers parallel to the surface, and researchers usually associate heart fiber orientations with such layers, not with cross-sections. He proposed a *peeling* interface, with which the user can understand the gradual change of fiber orientation by continuously peeling layers to depth direction. He also noted that such visualization technique would enhance further intuitive design.

6 Conclusion

In this paper, we presented a novel method for designing volumetric fiber orientation field filling a 3D heart model using a sketch-based interface. We apply two-step Laplacian smoothing, on the surface and on the volume, to obtain smoothly varying 3D fiber orientation field from user-specified constraints on the surface. We asked a doctor in the area of cardiology to try our prototype system and confirmed the effectiveness of our method. There are still many points to be improved in our method and we plan to continue working on this problem in the future.

Acknowledgements

We thank Dr. Takashi Ashihara for his cooperation in our user study and his valuable feedback.

References

1. Takashi Ashihara, Tsunetoyo Namba, Takanori Ikeda, Makoto Ito, Masahiko Kinoshita, and Kazuo Nakazawa. Breakthrough waves during ventricular fibrillation depend on the degree of rotational anisotropy and the boundary conditions: A simulation study. *Journal of Cardiovascular Electrophysiology*, 12(3):312–322, 2001.
2. Takashi Ashihara, Tsunetoyo Namba, Takenori Yao, Tomoya Ozawa, Ayaka Kawase, Takenori Ikeda, Kazuo Nakazawa K, and Makoto Ito. Vortex cordis as a mechanism of postshock activation: Arrhythmia induction study using a bidomain model. *Journal of Cardiovascular Electrophysiology*, 14(3):295–302, 2003.
3. Timothy A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):165–195, 2004.
4. Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. State-of-the-art report 2002 in flow visualization.
5. Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
6. Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 401–406, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

7. O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. R&#246;ssl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM Press.
8. Greg Turk. Texture synthesis on surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 347–354, New York, NY, USA, 2001. ACM Press.
9. Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Vector field design on surfaces. *ACM Trans. Graph.*, 25(4):1294–1326, 2006.