Interactive Deformation of Structurally Complex Heart Models Constructed from Medical Images

Kazutaka Nakashima[1] Yuki Koyama[1] Takeo Igarashi[1] Takashi Ijiri[2] Shin Inada[3] Kazuo Nakazawa[3]

[1]The University of Tokyo [2]Ritsumeikan University [3]National Cerebral and Cardiovascular Center

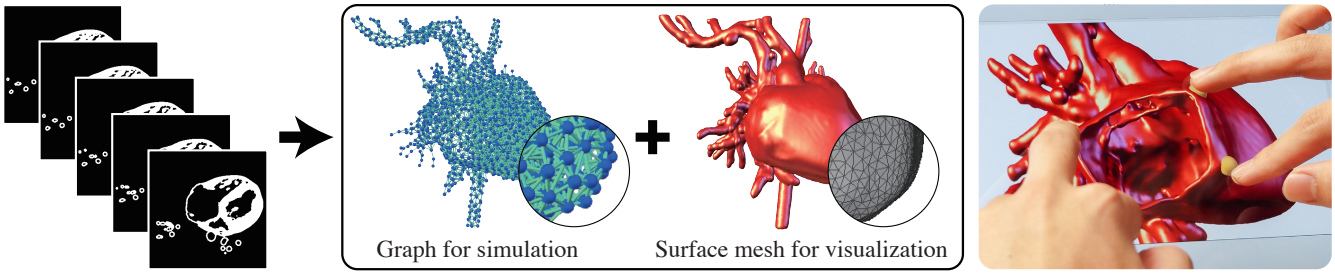

Figure 1: *The proposed framework automatically generates a* dual model *from a medical image. The dual model comprises a graph structure for elastic simulation and a surface mesh for visualization. Using this framework, we developed a multi-touch browser application with interactive deformation function.*

Abstract

We present a data structure for interactive deformation of complicated organ models, such as hearts, and a technique for automatically constructing the data structure from given medical images. The data structure is a dual model comprising of a graph structure for elastic simulation and a surface mesh for visualization. The system maps the simulation results to the mesh using a skinning technique. First, the system generates a dense graph and mesh from input medical images; then, it independently reduces them. Finally, the system establishes correspondence between the reduced graph and mesh by backtracking the reduction process. We also present an interactive browser for exploring heart shapes, and report initial feedback from target users.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications

1. Introduction

Effective visualizations for volumetric medical images (*e.g.,* computed tomography (CT) and magnetic resonance imaging (MRI)) are significant for physicians or patients to understand the complex structures of diseased organs. Especially, for hearts with congenital diseases that this study focuses on, structural irregularities directly affect their functions. Therefore, understanding their structures is necessary for treatment. Cross section visualization or volume rendering have been commonly used for viewing volumetric images. However, such *static* visualizations are insufficient for structurally complex organs because small but important features (*e.g.,* a small hole) may be overlooked by only controlling several parameters, such as view direction and transparency.

In this study, we focus on hearts with congenital diseases and propose a technique to visualize medical images to facilitate a better understanding of their complex structures. The key idea is to bring *interactivity* into visualization by allowing users to freely deform target organs during exploration by leveraging multi-touch interaction and real-time elastic simulation.

Applying real-time elastic simulation to a medical image of a heart is not trivial because thin features and irregular topologies of a complicated organ drastically increase the resulting mesh elements (*e.g.,* tetrahedron). Typical approaches based on the finite element method (FEM) require a tetrahedral mesh that can be automatically constructed from the isosurface of the volume [Si15]; however, this usually results in a mesh that is too fine for real-time applications. The shape matching method on regular grid structures [RJ07, SOG08] is another real-time elastic model. However,

DOI: 10.2312/egsh.20161012

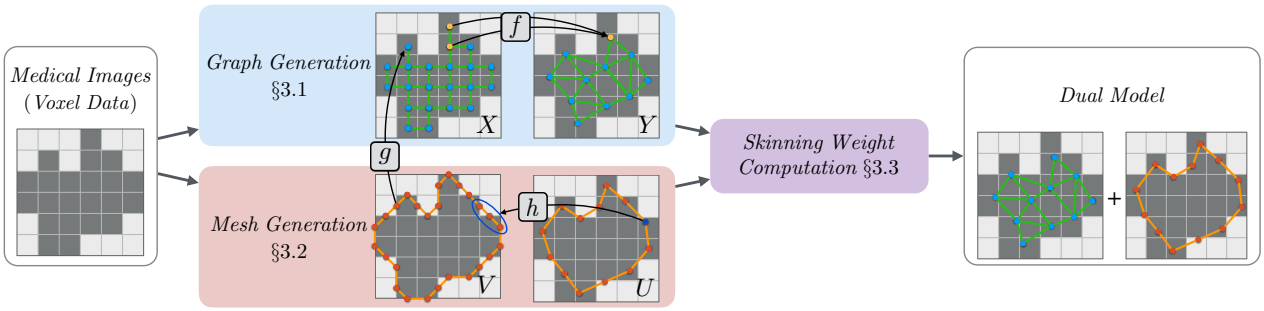

Figure 2: *Overview of our* dual model *construction. We independently generate a graph and a surface mesh. We then compute skinning weights by referencing the graph and surface mesh. Note that this procedure is fully automated.*

due to thin features and detailed topologies, many such acceleration techniques are ineffective for simulating hearts. Consequently, manual fixing the mesh is required so that it is appropriate for real-time use, which is not practical in medical examination scenarios.

Contributions. We present a data structure for interactive deformation of complicated organ models and a technique for automatically constructing it from given medical images. Given segmented CT volumes, the proposed framework automatically generates a *dual model* comprising a *graph* structure for elastic simulation and a surface mesh for visualization (Figure 1). In the graph, each simulation particle is connected to its neighbors. We adopt position-based dynamics [MHHR07, MC11] (PBD) on the graph. Rather than tetrahedral or grid meshes, applying a simple graph structure has some advantages. It allows us to fairly represent a complicated shape with reasonable degrees of freedom and is simple to implement. Deformation in PBD is not always physically correct; however it is acceptable in our case because the proposed framework is not for simulating surgical operations, but for facilitating exploration of complex shapes. To demonstrate the feasibility of the proposed framework, we present an interactive browser for exploring complex heart shapes, where the user can deform and cut the model using multi-touch interactions. We have applied the proposed framework to three heart models obtained from patients and received initial feedback from expert doctors. Note that the proposed method is not limited to heart (see the supplemental video for examples of other CT volumes).

Related Work. Graph structures have been used for deformation. Zhou *et al.* [ZHS*05] presented a method to generate a graph from a watertight surface and used it for deformation. Sumner *et al.* [SSP07] presented how a graph can be used for shape manipulation by space deformation. The method called oriented particles [MC11] simulates deformable objects and can accept graphs as internal structures. Our deformation algorithm is based on these graph-based methods, and our main contribution is an overall framework and its input–output setting, *i.e.,* generating a representation comprising a graph and a surface from medical images.

2. Simulation Framework

The proposed framework comprises two parts. The first part is the pre-computation stage where a dual model, comprising a graph for simulation and a surface mesh for visualization, is constructed from pre-segmented medical images. The generated surface mesh has skinning weights, that define a correspondence to the graph. The use of a graph allows this stage to be fully automatic, even when the input medical image contains many thin or irregular topologies. We describe the details of this stage in the next section.

The second part is the interactive stage where the dual model is simulated as an elastic deformable object, thereby allowing users to interact with it by deformation. For the simulation, we adopt the oriented particles method [MC11], which employs position-based dynamics [MHHR07]. In this simulation, every particle in the graph has a rigid transformation, which allows the use of skinning techniques [JDKL14]. Here, particles of the graph are used as deformation primitives (*i.e.,* bones). In each time step, the system first deforms the graph using simulation and then propagates deformation to the surface mesh using linear blend skinning (LBS).

3. Automatic Construction of Dual Model

Figure 2 shows an overview of the dual model construction. The input is voxel data that represents the target heart shape. Here, the voxel data need not be a regular isotropic grid; it can have a different pitch for each axis direction. We assume that the target heart shape is pre-segmented from other organs, and the values stored in voxels are binarized into foreground and background. From the input voxels, we first generate an initial graph and an initial surface mesh, both of which are over-resolution in this step. We then reduce them such that their resolutions are suitable for real-time use. After the reductions, we compute skinning weights on the reduced mesh by backtracking the reduction procedures. The resulting pair of the graph and the surface mesh is called a dual model, which is ready for an interactive deformation application.

3.1. Simulation Graph Generation

Our graph structure comprises particles that are connected to their neighborhoods by undirected edges. To generate a graph, we first place a particle at each foreground voxel center. The set of these particles is denoted X. Then, we connect particles such that every particle is connected to its 6-neighborhoods. This graph is called an *initial graph*. We reduce the initial graph by iteratively collapsing the shortest edge until the number of particles becomes equal to a

user-specified parameter. Note that this parameter allows users to control the computational cost in the interactive stage. If there are more than two equally shortest edges, we randomly select one of them. When particles v_0 and v_1 are merged into a particle v_2 during the reduction process, the position of v_2 is calculated by averaging the positions of all particles in X that are merged to v_0 or v_1. Let Y be the set of all particles in the reduced graph. During collapsing, we establish a map $f : X \to Y$ such that $v \in X$ are merged to $f(v) \in Y$ (upper part of Figure 2). This map is used in the skinning weights computation.

3.2. Surface Mesh Generation

We first extract an *initial mesh* by marching cubes from the input voxel data. We denote the set of all vertices in this mesh V. During extraction, we establish a map $g : V \to X$, where $g(v)$ is an initial particle of the foreground voxel that the vertex v faces. This map can be determined uniquely because marching cubes sets every vertex in the extracted mesh such that it faces one of the foreground voxels and one of the background voxels. This initial mesh often contains many vertices (approximately 550,000 for the model shown in Figure 1) and the cost for skinning is too expensive to perform in real time. Thus, we apply a mesh reduction technique for this initial mesh. Any technique based on vertex merging (*e.g.,* QSlim [GH97]) can be used; however, the reduction process itself is out of our scope. In our current implementation, we use a simple strategy. We iteratively collapse the shortest edge and place a new vertex in the middle until all edges become longer than a user-specified length. We simultaneously establish a map $h : U \to \mathcal{P}(V)$ such that all vertices in $h(u)$ are merged to u (lower part of Figure 2), where U is the set of all vertices in the reduced mesh and $\mathcal{P}(V)$ is the power set of V.

3.3. Skinning Weight Computation

Generating appropriate skinning weights is difficult for general cases and they often need to be manually edited such that the correspondences between mesh vertices and deformation primitives are correctly constructed (*e.g.,* [MC11]). In our case, we can robustly generate plausible correspondences automatically using the maps f, g, and h established during the graph and mesh generation.

To compute skinning weights for a vertex u_i of the reduced mesh, the system first identifies $h(u_i)$, the set of vertices of the initial mesh that are mapped to the reduced mesh vertex u_i (Figure 3 (a, b)). Then, it obtains $f(g(v))$ for each vertex $v \in h(u_i)$ (Figure 3 (c, d)). Finally, the skinning weight $w_{i,j}$, the amount of influence of the j-th particle y_j in the graph on the vertex u_i, is computed as

$$w_{i,j} = \left|\{v \mid v \in h(u_i) \wedge f(g(v)) = y_j\}\right| \Big/ |h(u_i)|.$$

4. Results

Application. As an application of our framework, we present an interactive browser for exploring complex heart shapes. This browser runs on consumer-level laptops with multi-touch capability. Figure 4 shows the user interface of the browser. In addition to standard 3D model viewer functions (*e.g.,* camera controls and shader selection), the browser allows users to deform and cut the model interactively through multi-touch interaction.

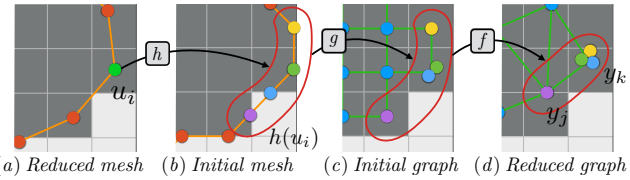

Figure 3: *Skinning weight computation. For each vertex in the reduced mesh, we find a set of corresponding graph vertices using the maps f, g, and h. In this example, the skinning weight $w_{i,j}$, the amount of influence of the j-th particle in the graph on the i-th vertex in the mesh, is $1/4$. Similarly, $w_{i,k}$ is $3/4$.*

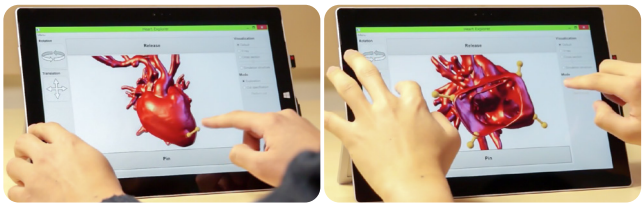

Figure 4: *Our browser application. Users can deform and cut heart models through multi-touch interaction.*

The user can pull the model using multiple fingers. When a touch interaction begins and the finger touches the surface mesh, the system generates a *virtual particle* at the touching point on the surface mesh, and then connects the virtual particle to several nearby model particles in the graph. These connections are interpreted as *distance constraints* [MHHR07] whose target distance is zero. As the finger moves on the display, the virtual particle also moves so that the depth from the screen is constant. When the finger is taken off the display, the virtual particle and its associated connections are removed. The browser also supports a *pinning* function whereby the model particles are pinned by keeping the virtual particles even after the finger is removed from the display. In addition, the user can cut the model by specifying a path on the (non-deformed) surface. To cut the model, we use a simple yet robust approach. We cast rays along the paths and remove the foreground voxels that the rays intersect before they reach the cavity inside the model (*e.g.,* ventricle). Finally, the system computes the model construction process from the beginning. Note that this typically takes approximately 1 min in our current implementation.

Heart Models. We constructed three heart models with congenital diseases from actual medical CT images. Figure 5 shows the models generated using the proposed framework. Before applying our model generation process, we manually segmented the heart part from raw CT data and binarized the values stored in the voxels. We used VoTracer [IYS*13] for segmentation, and it took 15–25 min for each model. All three hearts have irregular structures causing diseases. For example, in Case A, the right ventricle has significantly shrunk; this is difficult to observe in a static figure (see the accompanying video for a dynamic visualization).

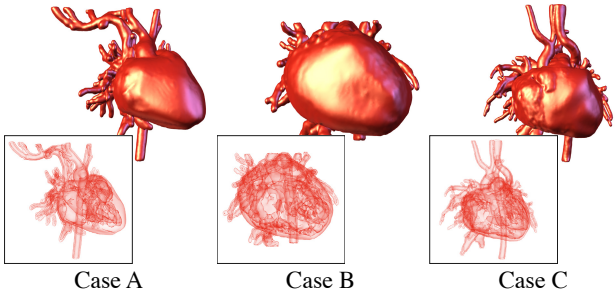

Figure 5: *Three cases available in our browser, all of which were constructed from* real *medical images. The bottom row shows the same models using a semi-transparent visualization technique.*

Table 1: *Statistics of our results*

heart model	Case A	Case B	Case C
voxel resolution	$512^2 \times 139$	$512^2 \times 84$	$512^2 \times 186$
#foreground voxels	955 k	854 k	936 k
graph reduction	11.6 s	9.8 s	11.0 s
mesh reduction	14.2 s	14.0 s	19.9 s
weight calculation	18.6 s	16.9 s	25.7 s
#reduced particles	2100	2100	2100
#reduced triangles	119 k	110 k	163 k
frames per second	32	26	22

Performance. Table 1 shows the performance of our browser running on a Microsoft Surface Pro 3 (1.7 GHz Intel Core i7; Intel HD Graphics 5000). Here, we reduce the graph until the number of particles is less than 2100 and reduce the surface mesh until all edges are longer than 1.9 times the average length of the edges in the initial mesh. Note that our current implementation does not use a GPU for skinning or physical simulation; however, it is possible to improve performance in the runtime session using a GPU.

Case Study. To validate our approach in practical scenarios and obtain feedback for further development of the browser application, we conducted a pilot study using the current implementation. We recruited two pediatricians, P_1 and P_2, who are familiar with the medical treatment of hearts with congenital diseases. We asked them to freely use the browser. We explained the usage of our browser as required and conducted informal interviews. This took approximately 30 min for each participant. Note that Cases A and B (Figure 5) were used for this study.

P_1 agreed that the overall application is useful for actual clinical scenarios, and the function of pulling the heart model specifically helps understand its three-dimensional structure. P_2 commented that this deformation-based exploration could be a new approach for medical examination. He also mentioned that this new approach requires some experience for pediatricians because they have been trained to examine hearts primarily from static two-dimensional cross-sectional images. Relative to deformation quality, P_1 claimed that the current setting is too soft compared to real hearts and is thus unsuitable for clinical use.

5. Limitations and Future Work

Note that the proposed framework has several limitations. The input voxel data creation (*i.e.,* segmentation) requires manual operations, and this remains an unsolved problem. Although this is out of our scope, this issue is important considering the deployment of our framework. In addition, our current graph reduction process does not preserve the original graph topology; thus, detailed topological information could be lost (*e.g.,* thin tubes could be reduced to rods) when the graph is overly reduced. We consider that developing a topology-aware graph reduction process is an important future work. Finally, for using our application in actual medical scenarios, it is necessary to improve performance by, *e.g.,* exploiting a GPU. As revealed in the case study, obtaining and applying more realistic elasticity parameters is an important challenge; measurement-based simulation might be useful for this issue.

Acknowledgements

We thank Hirofumi Seo for providing valuable comments on shader development. We also thank Isao Shiraishi and Ken-ichi Kurosaki for providing the CT volumes used in the paper and for their valuable comments. This study was supporeted by Grants-in-Aid for Scientific Research (B) 25282138 from the Ministry of Education Culture, Sports, Science and Technology, and by Intramural Research Fund (27-4-5) for Cardiovascular Diseases of National Cerebral and Cardiovascular Center.

References

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proc. SIGGRAPH '97* (1997), pp. 209–216. doi:10.1145/258734.258849. 3

[IYS*13] IJIRI T., YOSHIZAWA S., SATO Y., ITO M., YOKOTA H.: Bilateral hermite radial basis functions for contour-based volume segmentation. *Computer Graphics Forum 32*, 2pt1 (2013), 123–132. doi:10.1111/cgf.12032. 3

[JDKL14] JACOBSON A., DENG Z., KAVAN L., LEWIS J. P.: Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses* (2014), pp. 24:1–24:1. doi:10.1145/2614028.2615427. 2

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. *ACM Trans. Graph. 30*, 4 (July 2011), 92:1–92:10. doi:10.1145/2010324.1964987. 2, 3

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. doi:10.1016/j.jvcir.2007.01.005. 2, 3

[RJ07] RIVERS A. R., JAMES D. L.: Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26*, 3 (July 2007). doi:10.1145/1276377.1276480. 1

[Si15] SI H.: Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw. 41*, 2 (Feb. 2015), 11:1–11:36. doi:10.1145/2629697. 1

[SOG08] STEINEMANN D., OTADUY M. A., GROSS M.: Fast adaptive shape matching deformations. In *Proc. SCA '08* (2008), pp. 87–94. doi:10.2312/SCA/SCA08/087-094. 1

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3 (July 2007). doi:10.1145/1276377.1276478. 2

[ZHS*05] ZHOU K., HUANG J., SNYDER J., LIU X., BAO H., GUO B., SHUM H.-Y.: Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph. 24*, 3 (July 2005), 496–503. doi:10.1145/1073204.1073219. 2