Vision, Modeling, and Visualization (2013)
Michael Bronstein, Jean Favre, and Kai Hormann (Eds.)

# Visualizing Dissections of the Heart in a Dataflow-based Shader Framework for Volume Rendering

S. Arens, M. Bolte and G. Domik

Department of Computer Science, University of Paderborn, Germany

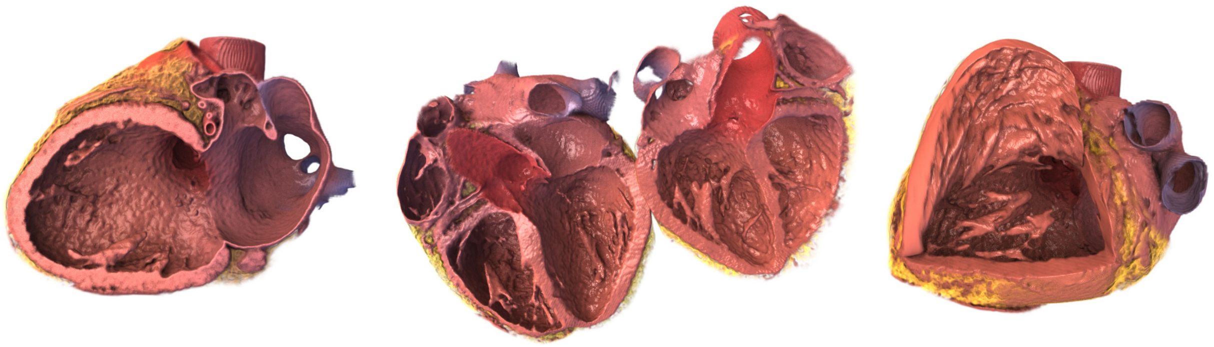

**Figure 1:** *Three volume visualizations of a CT-angiographic dataset, dissected and illuminated according to "Atlas of Human Anatomy" by Frank. H. Netter [Net10]. Left: Simple planar cut. Center: Rigid deformation after a non-planar cut. Right: Elastic deformation. Inner details, like the papillary muscles, can be easily perceived. See Fig. 2 for corresponding dataflow graph.*

**Abstract**

*Dissections in anatomic heart illustrations are an effective visualization technique and support a variety of applications in exploration and communication. In this paper we implemented a set of volume deformations imitating dissections found in anatomy atlases of the human heart based on individual patient data. This allows physicians and surgeons to compare, explore, and discuss volume data in view of atlas illustrations. The main challenge for these illustrations is the flexible real-time combination of various (geometric) deformations and shadows (influenced by the deformations) to depict shapes and structures. The proposed technical solution to these volume visualizations is a novel GPU-based processing technique for dataflow-based shading graphs. Hence, any deformations can easily be combined in our system with immediate influence on shadows and shading. We show the effectiveness and applicability of our approach by imitating illustrations of heart dissections taken from an anatomy atlas using the patient's individual volume data.*

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Dynamic Shader Generation

## 1. Problem

Many GPU-based volume visualization techniques have been presented, that imitate the techniques in traditional anatomy atlases. Nevertheless, the quality of such atlases has not been achieved yet. One reason is the effort to *combine* techniques like deformations and shadows in GPU-based rendering. That is, none of the dataflow-based shader frameworks we found in the literature, is able to cope with a combination of deformations and shadows as in Fig. 1.

*Deformation:* In illustrations of heart's anatomy, cuts in combination with rigid and elastic deformations are used. Several ways exist to imitate such deformations in volume visualizations [CCI*07]. Due to better results, many approaches in literature are inverse deformations, i.e. the data stays untouched, but the sampling of the data is deformed.

*Shadows:* All hand-drawn anatomic illustrations of the heart use shading and shadows to depict shapes and structures. The technical problem is, that shadows and shading have to take into account the aforementioned inverse deformations and cuts.

Technically we can reduce this problem to the ability of nodes being able to modify, skip or add sampling positions for whole subgraphs.

DOI: 10.2312/PE.VMV.VMV13.231-232

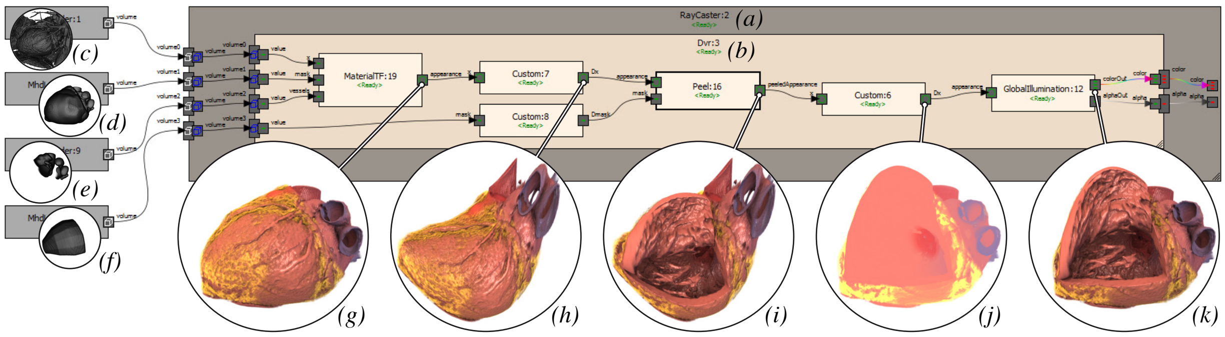

**Figure 2:** *Dataflow graph of Figure 1 (right) in our shader framework. (g) - (i) are illuminated for better contrast, in actual fact they look like (j). (a) Ray casting node with four volume ports as input. Defines rays and awaits color and opacity per ray. (b) Direct volume rendering node. Defines sampling positions on a ray and awaits color and opacity per sampling position. (c) CT angiographic dataset of the human heart. (d) Segmentation of the heart. (e) Masks used for coloring of the vessels. (f) Mask of the cut. (g) Transfer function applied. (h) Deformation applied to shape the data and the cut's mask in a more planar form. (i) Elastic peeling deformation applied, using the cut's mask. (j) Inverse deformation applied to undo the deformation in (h). (k) Global illumination applied taking into account all previous cuts and deformations.*

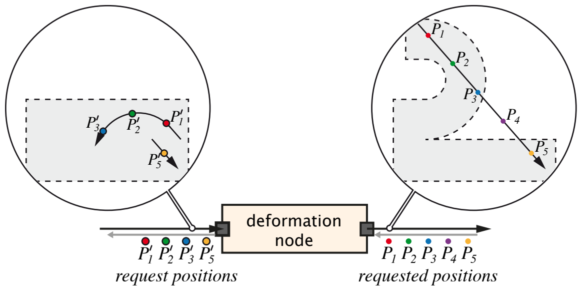

**Figure 3:** *Principle of inverse deformations in our dataflow graph. When sampling in a deformed scene at positions $P_1$ to $P_5$ along a viewing ray (right), the data is looked up at positions $P_1'$ to $P_5'$ (left). In our system nodes can change requested sampling positions in their own request or add or skip requests. Dataflow direction is indicated by black arrows, request direction by gray arrows.*

## 2. Dataflow-based Shading Graphs with Deformations

The nodes in dataflow-graphs represent data processing operations and can consume, produce, and modify data. When the input data of a node changes, its output data could be outdated and has to be updated by evaluating the node. The details about how a node handles its input data and how this affects other nodes is subject to the selected evaluation model. In the *Data-Driven Evaluation model* a node is evaluated as soon as it has new data available on its input ports. All shader frameworks we found in the literature make use of this data-driven model. In the *Demand-Driven Evaluation model*, the demand for output data defines the evaluation order of connected nodes. A node is evaluated when another node requests its output data.

The technical requirement to render dissections of the heart correctly, is a general solution to allow nodes to modify, skip and add sampling positions. Systems with data-driven evaluation come with two limitations that hinder this: 1. Nodes have no control over the evaluation of other nodes that provide them with input data. 2. The volume rendering system feeds the dataflow graph with data prefetched at predefined sampling positions. The individual nodes have no control over the sampling position.

The proposed concept (see Fig. 3) avoids these limitations: The demand-driven evaluation model is used to avoid the first limitation. It gives nodes control over the evaluation of other nodes that provide them with input data. Every node can decide if, and when, and how often to evaluate the preceding nodes to get data for the current sampling position. To avoid the second limitation the nodes are now able to *request data for a specific sampling position*. The shader composer can realize this request as a sampling position parameter to the nested shader function calls.

This makes it possible to insert deforming nodes. There is no global sampling position, instead every node is requested for port-data at a specific sampling position. Only this position can be used by nodes (unchanged or modified) to request data from preceding nodes. This way deformations influence all preceding nodes and several deformations can be combined correctly without the need to adapt any node. See Fig. 2 for an exemplary dataflow graph.

## 3. Conclusion

In this paper we imitate dissections of the heart using GPU-based volume ray casting. We propose a dataflow-based approach for shader generation that combines GPU-power and high flexibility. The approach allows for any combination of deformations with correct shading and shadowing without introducing special case handling.

## References

[CCI*07] CHEN M., CORREA C., ISLAM S., JONES M. W., SHEN P.-Y., SILVER D., WALTON S. J., WILLIS P. J.: Manipulating, deforming and animating sampled object representations. *Computer Graphics Forum 26*, 4 (2007), 824–852.

[Net10] NETTER F.: *Atlas of Human Anatomy*. Netter Basic Science. Elsevier Health Sciences, 2010.