Immersive Volume Rendering of Blood Vessels

Gregory Long, Han Suk Kim, Alison Marsden, Yuri Bazilevs, Jürgen P. Schulze

University of California San Diego, 9500 Gilman Drive, La Jolla, CA, USA

ABSTRACT

In this paper, we present a novel method of visualizing flow in blood vessels. Our approach reads unstructured tetrahedral data, resamples it, and uses slice based 3D texture volume rendering. Due to the sparse structure of blood vessels, we utilize an octree to efficiently store the resampled data by discarding empty regions of the volume. We use animation to convey time series data, wireframe surface to give structure, and utilize the StarCAVE, a 3D virtual reality environment, to add a fully immersive element to the visualization.

Our tool has great value in interdisciplinary work, helping scientists collaborate with clinicians, by improving the understanding of blood flow simulations. Full immersion in the flow field allows for a more intuitive understanding of the flow phenomena, and can be a great help to medical experts for treatment planning.

Keywords: Volume Rendering, Blood flow, Virtual Reality, octree, tetrahedra

1. INTRODUCTION

Simulation of blood flow for cardiovascular applications provides a wealth of data that is often not clinically available, including wall shear stress, wall motion, and time-dependent three dimensional velocity fields. Accurate prediction of blood flow and interpretation of the results can be crucial for surgical planning in a range of cardiovascular disease applications, including aneurysm treatment, coronary artery disease, and congenital heart disease. Recent advances in fluid structure interaction simulations now enable coupled fluid/solid simulations, producing time dependent blood flow and wall motion. Due to these advances in blood flow simulation, an intuitive, easy to use method of blood flow visualization has become more and more important. Advanced visualization of these results via 3D virtual reality facilitates communication between engineers and clinicians by improving understanding of simulation capabilities. By immersing the viewer in the flow field, the user can experience the full dynamic motion of both the blood flow and wall motion, which is not possible to convey using conventional visualization modalities.

We have developed a novel method for visualizing time series blood flow in a CAVE environment. Structurally, blood vessels have an interesting shape: they are long and thin, with many small branches. Because of this shape, a traditional direct volume rendering approach would create a significant amount of empty space, incurring unnecessary overhead. To solve this issue, we utilize an octree data structure, and render each leaf node, instead of the entire, sparse volume. Our method reads in unstructured tetrahedral data, constructs an octree, and resamples the data, storing it into voxel format via 3D textures. For each node, we use slice based volume rendering with a 1 dimensional look up table for a transfer function. Per users requests, we also implemented various features to aid in the visualization. Users can navigate both around and within the vessel without loss of performance, and can dynamically adjust rendering quality, animation speed, and transfer function values. Time series information is conveyed through animation. Finally, a polygonal mesh of the vessel wall can be toggled, to aid in depth perception, and therefore visualization of the 3D structure.

In this paper, we will discuss our method for rendering blood flow in blood vessels, as well as the implications of our method in domain science. Section 2 discusses related work in the areas of octree based volume rendering, volume rendering in virtual reality, rendering of tetrahedral data, and visualization of blood flow. Section 3 presents in detail our visualization method. Section 4 discusses performance results as well as quality/performance tradeoffs. Section 5 discusses design decisions and the impact of our visualization method on domain science. Finally, Section 6 concludes our discussion and touches upon possible future work.

2. RELATED WORK

Texture based volume rendering is a widely known method for accelerating volume rendering performance,[1] and octree based volume rendering is a widely known method of increasing performance in texture based volume rendering. La Mar et al.[2] present an octree based method for volume rendering, creating levels of detail by filtering the volume data based on the distance of the data from the camera. Plate et al.[3] also present an octree based method that runs at interactive frame rates, using a caching technique to render very large datasets. Our approach does not use the octree to filter the data into multiple resolutions based on camera positions, but instead uses the octree structure only to reduce the amount of data stored by culling spaces that contain no data. However, our algorithm supports time series with separately computed octrees for each time step.

Direct rendering of unstructured tetrahedral meshes has been covered extensively,[4–6] however our approach differs in that we use barycentric coordinates to resample the tetrahedral data into a regular voxel grid, as opposed to rendering the mesh directly.

Medical volume rendering has mostly been done for diagnosis and virtual surgery. Little work has been done on volume rendering in immersive virtual environments. Schulze et al.[7] presented the volume rendering application for a CAVE-like system with a full transfer function editor, but it uses a single regular grid which is wasteful in the case of blood vessel trees. Blood vessels have been volume rendered before,[8–10] but our method differs in that it uses an octree-optimized data set to skip empty regions of the data volume, and direct volume rendering with a transfer function editor for rendering. Also, our rendering system has been integrated into a virtual reality software framework (COVISE[11]), and thus runs in CAVE and similar, tracked, immersive environments.

A popular approach for data visualization is to use the Visualization Toolkit (VTK),[12] but it does not support octrees for direct volume rendering, and thus requires much more memory for the same rendering quality compared to our algorithm.

3. TECHNICAL DESCRIPTION

Our dataset is composed of two sets of patient specific cardiovascular simulations. The first model is that of a cerebral aneurysm taken from Bazilevs et al.[13] The model is constructed from CT scan data and meshed with tetrahedral elements using the techniques described by Zhang et al.[14] The model was simulated using a fully-coupled fluid-structure intercation (FSI) analysis (see Bazilevs et al.[13] for details). The use of FSI produces vessel wall deformation during the simulated cardiac cycle, which motivates the use of separate octrees at each time step.

The second model corresponds to a patient-specific Kawasaki disease configuration. Kawasaki disease (KD) is a serious pediatric illness affecting the cardiovascular system. There are currently over 6,000 cases of KD diagnosed annually in the US, and KD is the leading cause of acquired heart disease in children.[15] One of the most serious complications of Kawasaki disease, occurring in about 25% of untreated cases, is the formation of large aneurysms in the coronary arteries. These aneurysms put patients at risk for myocardial infarction (MI (heart attack)) due to thrombosis that blocks flow distal to the aneurysm. Aneurysms caused by KD differ from other common types of aneurysms (abdominal, cerebral) in that they typically develop thrombosis (clot) as opposed to a sudden vessel wall rupture.

The KD model is constructed from CT scan data, using SimVascular[16] and MeshSim.[17] Simulation results are generated with a state-of-the-art finite element flow solver PHASTA.[18] The blood vessels are long, winding, tube like structures, often with smaller, branching paths off of a main path. This type of structure leads to a very sparse volume, if the data is naively converted into a single large volume. This motivates the use of an octree with many levels of recursion.

For both models the results are output in VTK format, which is then converted, with no compression, to ASCII COVISE format, one file for position and velocity per time step. The data is in unstructured tetrahedral format, with position and velocity information at each vertex as well as connectivity. The density of the data is non-uniform, with certain areas containing a large amount of small tetrahedras, while others containing fewer, larger tetrahedras. The geometry also changes with each time step. Figure 1 and Figure 2 show two example

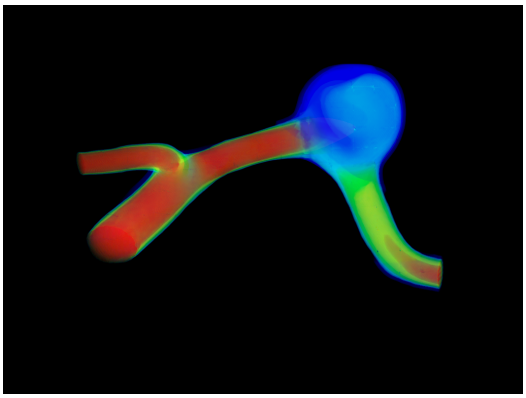


Figure 1: Dataset of an aneurysm. Blue areas indicate low velocity, while red areas indicate high velocity.

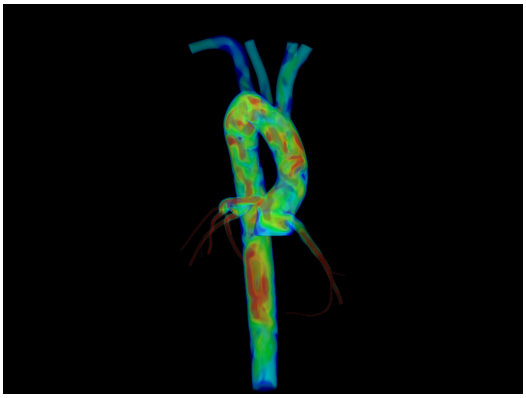


Figure 2: Dataset of a blood vessel exhibiting Kawasaki disease. Blue areas indicate low velocity, while red areas indicate high velocity.

data sets. The data is first read at each node from a single disk and stored in memory. The algorithm is not out of core; data is stored entirely in memory.

An octree is then constructed for each time step. We first find an initial bounding box for the entire data, and then recursively subdivide the bounding box into sub boxes. Nodes that meet the termination criteria are not subdivided further, and nodes that contain no tetrahedra are discarded. Instead of manually specifying a termination level, the octree recursion is terminated either when a minimum size is reached, or a lower bound is reached on the number of tetrahedra intersecting with the octree node. For each node, a list of intersecting tetrahedra is also saved. Figure 3 shows an illustration of the octree node termination criteria. This method of termination is preferable to naively terminating the recursion after a fixed number of time steps because of the variation in data density. With a fixed termination criteria, the octree will be populated with many nodes containing few samples, leading to poor image quality and excess overhead. With this method, we ensure that each node in the octree contains a minimum number of data samples.

Once the octree has been constructed, the data is then resampled into a regular grid and stored as a 3D texture. For each octree node, the list of intersecting tetrahedra is traversed. For each tetrahedra inside the node, the bounding box for that tetrahedra is found, and that bounding box is sampled according to the coordinates of points in the 3D texture. Barycentric coordinates are then used to determine if a point is found within the tetrahedra, and to find the value for that point via interpolation. Figure 4 shows an illustration of the interpolation scheme. This process is repeated for each time step, and the resulting octrees are stored in memory. This allows the initial octree building step to be performed only once at the beginning of the visualization, and possibly offline.

The volume is rendered using a slice based rendering technique with 3D textures. Alpha blending is performed

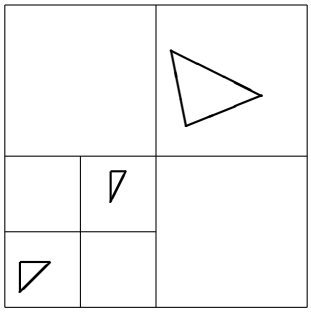


Figure 3: Illustration of the octree node termination criteria. Octree nodes are continually subdivided until a minimum number of tetrahedra are found within the node.

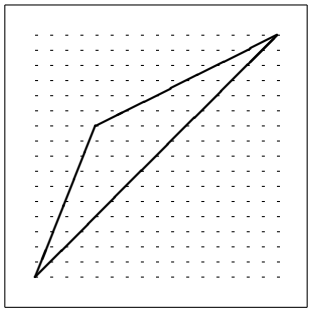


Figure 4: Illustration of the barycentric coordinate interpolation. A bounding box is drawn around the tetrahedra, and the bounding box is regularly sampled according to the 3D texture grid, interpolating the values within by using barycentric coordinates.

using pixel shaders and a look up table to store the transfer function. Each node is rendered separately and slices are oriented based on the position of the viewer given by the head tracker. This allows the user to enter the volume and view it from the inside, allowing for an immersive experience that gives further insight into the flow inside the vessel. This is repeated for each time step and stored independently as its own node in the scene graph. Animation is performed by cycling through the nodes via COVISE's built in animation functionality. Synchronization across all nodes is done by repeating all work on all nodes simultaneously. Culling is done independently on all nodes as well.

The transfer function used is a simple 1 dimensional transfer function based on the scalar velocity magnitude. Velocity is mapped to the HSV spectrum, where blue corresponds to low velocity, and red corresponds to high velocity. The range of values to which the transfer function is mapped is specified with a slider. When the range is limited to a range smaller than the full range of the data, values outside that range are either clamped at the lowest or highest value, or can be set to transparent.

Navigation is performed using the standard COVISE interface.[19] Rotation, translation, and scaling is controlled with a wand. Animation is controlled with the COVISE built in animation controls. Animation speed is controlled with a knob, and individual time steps can also be chosen with a slider.

Our application is running in the StarCAVE, an immersive 3D virtual reality cave environment. One of the main advantages of using the StarCAVE is the 3D stereoscopic vision. In volume rendering, this depth perception can be lost, which can be disorienting for the user, particularly when immersed within the volume. In order to give the user a better point of reference, a triangle mesh of the blood vessel wall is rendered around the volume. Clipping planes are also available to cull specific parts of the volume, allowing further flexibility in examining the data inside the volume. Figure 5 shows various views of the visualization in the StarCAVE.

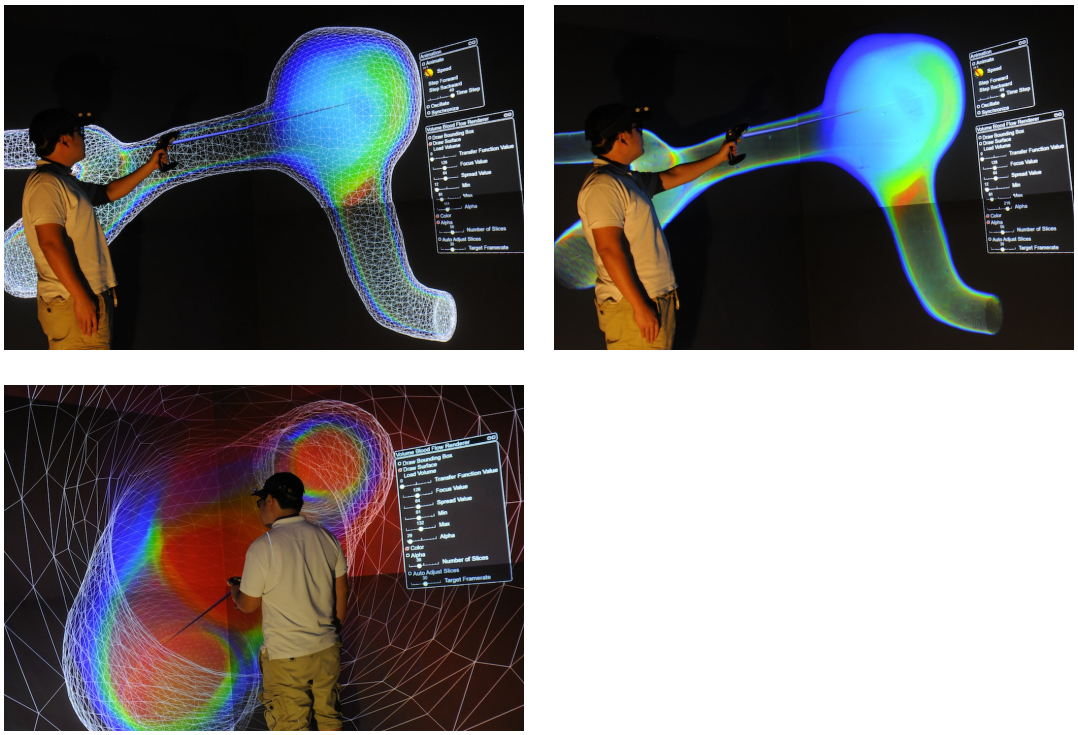


Figure 5: Various views of the blood flow visualization in the StarCAVE.

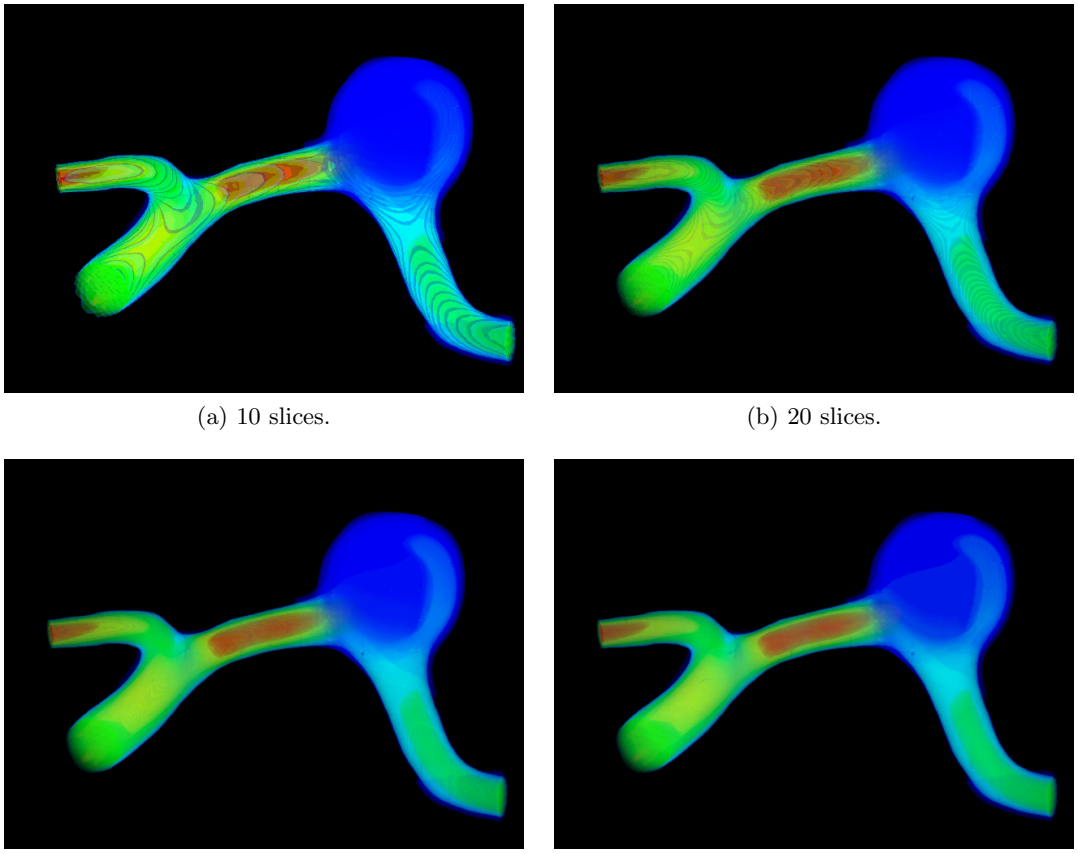
4. RESULTS

Our software is based on the COVISE software environment, and is run on a cluster of 16 nodes, each consisting of an Intel Core 2 Quad 4-core processor running at 2.4 GHz, 4 GB RAM, and dual NVIDIA Quadro 5600 video cards. Measurements were obtained by averaging a series of 10 samples. We measured our performance rendering two datasets, Aneurysm and Kawasaki. The Aneurysm dataset contains 144,612 tetrahedra, while the Kawasaki dataset contains 3,557,011. However, due to our resampling algorithm, the Aneurysm dataset contains 320 leaf nodes, while the Kawasaki dataset contains only 230. The total memory usage at $64 \times 64 \times 64$ resolution for the Aneurysm dataset is 80 MB - as a comparison, if an equivalent texture resolution was used to render the Aneurysm dataset without an octree, the total texture size would be 1.01 GB, or an increase of 12.9×.

Tables 1 and 2 show frame rate and preprocessing time for our Aneurysm and Kawasaki datasets, given different numbers of slices and texture sizes. Note that while the Kawasaki dataset contains significantly more tetrahedra than Aneurysm, and therefore has a significantly higher preprocessing time, its frame rate is higher, due to a lower number of leaf nodes. Figures 6 and 7 show the effect of the number of slices and texture size on image quality. Increasing the number of slices improves image quality, at the expense of frame rate. However, the image quality is still acceptable at 10 slices, and increases significantly at 20, without a loss of interactivity. Additionally, because the number of slices can be changed interactively, users can choose a low number of slices to navigate, and increase the slices when visual quality is desired. Increasing texture size can also improve image quality, at the expense of preprocessing time. Because this preprocessing is only done once, and can be done offline and stored for future use, it is desirable to use the highest possible size that can fit into video memory.

5. DISCUSSION

We found that the advanced visualization of cardiovascular simulations provides a new experience in data analysis and interpretation of simulation results which improves the physical intuition of the user, and allows for a more dynamic and physical exploration of the data. The ability to move inside and rotate data in real time is a



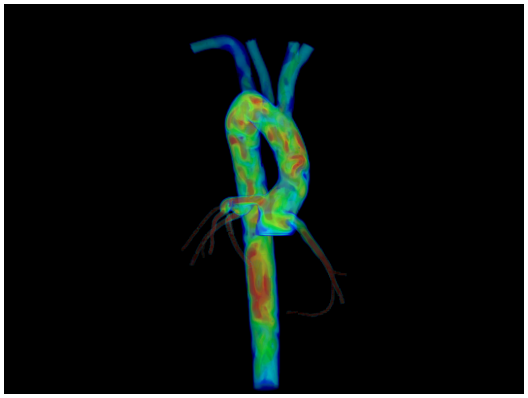
(a) 10 slices. (b) 20 slices.

(c) 50 slices. (d) 100 slices.

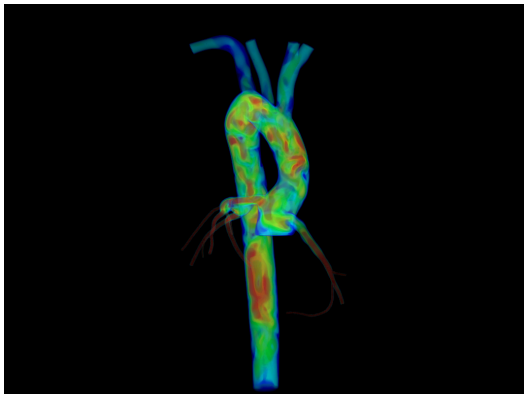
Figure 6: The Aneurysm dataset, rendered with different numbers of slices. Higher slice count results in higher image quality/less artifacts.

number of slices	frame rate (Aneurysm)	frame rate (Kawasaki)
10	26.8	31.2
20	15.4	19.5
30	11.2	13.6
50	7.1	8.9
40	8.6	11.0
60	5.9	7.6
70	5.2	6.5
80	4.6	5.8
90	4.1	5.2
100	3.7	4.7

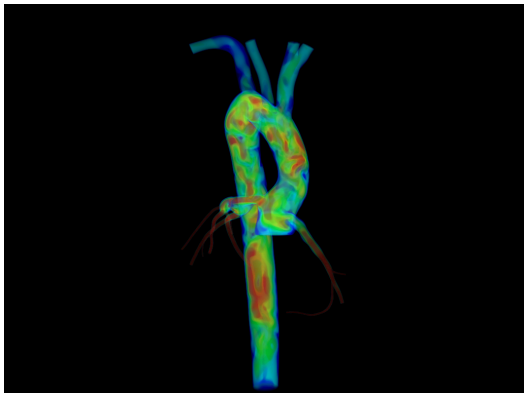
Table 1: Frame rate as a function of the number of slices per octree leaf node. Note that the Kawasaki dataset has a higher performance, due to less leaf nodes after resampling.



(a) $16 \times 16 \times 16$ resolution.



(b) $32 \times 32 \times 32$ resolution.



(c) $64 \times 64 \times 64$ resolution.

Figure 7: The Kawasaki dataset, rendered with different texture resolutions. Higher resolution results in smoother looking and more accurately interpolated edges.

texture size	preprocessing time (Aneurysm)	preprocessing time (Kawasaki)
16^3	1,220	31,930
32^3	2,340	65,600
64^3	10,820	334,720

Table 2: Preprocessing time (in ms) as a function of texture size. Note that the Aneurysm dataset has a much lower preprocessing time, due to significantly less tetrahedra.

step forward in increasing overall understanding of blood flow phenomena. These tools are particularly valuable for interdisciplinary work in which simulation results must be conveyed to a clinician or surgeon for treatment planning.

As mentioned in Section 1, the first approach used streamlines and surface rendering to render flow. We found that the volume rendering approach was more effective than streamlines in visualization of the data. Observing small scale flow phenomena with streamlines requires a large number of lines to be rendered, which can occlude other lines and create an overly cluttered visualization. Conversely, fewer streamlines give a more clear visualization, while discarding small scale phenomena. With volume rendering, the user can observe small scale flow phenomena that are often missed when relying on streamlines alone.

There are several values that can be changed to optimize performance for a given dataset. First, the minimum size of an octree node can be set so as to control the number of textures generated. Additionally, the minimum number of tetrahedra contained within a node can be set as well. Depending on the number of tetrahedra, the texture size can also be set, as the size of the texture should be similar to the number of tetrahedra. Finally, the number of polygonal slices used can be changed, to increase image quality at the expense of framerate, as seen in Table 1.

Because of the ability to specify minimum node sizes, our implementation can scale well with the size of the data, keeping performance high at the cost of resolution.

6. CONCLUSION AND FURTHER WORK

In this paper, we presented a novel method of visualizing blood flow in blood vessels with volume rendering in an immersive 3D CAVE environment. We resampled unstructured tetrahedral data into a regular voxel grid, used slice based volume rendering combined with an octree data structure, and included tools such as animation, triangle mesh surfaces, and clipping planes. We are able to achieve real time performance, allowing users to interact with the blood vessel in real time.

We believe our tool conveys understanding of the simulation data very clearly. Due to the high performance, intuitive interface, and novel methods of visualization, we believe this tool is a great help in collaborative, interdisciplinary environments, allowing scientists, engineers, and clinicians all to quickly and clearly view simulation data for use in collaborative efforts.

Possible further work includes visualization of directional components of the flow, through streamlines or other methods, as well as out of core caching schemes, allowing more detailed datasets to be loaded.

7. ACKNOWLEDGMENTS

Support of the UCSD Chancellor's Grant, the Burroughs Wellcome Fund Career Award at the Scientific Interface, and NIH grant number HL102596A is gratefully acknowledged. We would like to acknowledge the contributions of Sasha Koruga and Kenneth Benner, for their work on the first iteration of this project and Dibyendu Sengupta for helping to supply the Kawasaki dataset.

REFERENCES

[1] Cabral, B., Cam, N., and Foran, J., "Accelerated volume rendering and tomographic reconstruction using texture mapping hardware," in [*Proceedings of the 1994 symposium on Volume visualization*], *VVS '94*, 91–98, ACM, New York, NY, USA (1994).

[2] LaMar, E., Hamann, B., and Joy, K. I., "Multiresolution techniques for interactive texture-based volume visualization," in [*Proceedings of the conference on Visualization '99: celebrating ten years*], *VIS '99*, 355–361, IEEE Computer Society Press, Los Alamitos, CA, USA (1999).

[3] Plate, J., Tirtasana, M., Carmona, R., and Fröhlich, B., "Octreemizer: a hierarchical approach for interactive roaming through very large volumes," in [*Proceedings of the symposium on Data Visualisation 2002*], *VISSYM '02*, 53–ff, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2002).

[4] Chopra, P. and Meyer, J., "Incremental slicing revisited: Accelerated volume rendering of unstructured meshes," in [*Proc. IASTED Visualization, Imaging and Image Processing*], *IASTED '02*, 533–538 (2002).

[5] Weiler, M., Kraus, M., and Ertl, T., "Hardware-based view-independent cell projection," in [*Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*], *VVS '02*, 13–22, IEEE Press, Piscataway, NJ, USA (2002).

[6] Callahan, S. P., Ikits, M., Comba, J. L. D., and Silva, C. T., "Hardware-assisted visibility sorting for unstructured volume rendering," *IEEE Transactions on Visualization and Computer Graphics* **11**, 285–295 (May 2005).

[7] Schulze, J., Wössner, U., Walz, S., and Lang, U., "Volume rendering in a virtual environment," in [*Proceedings of 5th IPTW and Eurographics Virtual Environments*], 187–198, Springer Verlag (2001).

[8] Bartz, D., Straßer, W., Skalej, M., and Welte, D., "Interactive exploration of extra- and intracranial blood vessels," in [*Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99)*], *VISUALIZATION '99*, –, IEEE Computer Society, Washington, DC, USA (1999).

[9] Tsuchiya, K., Katase, S., Hachiya, J., and Shiokawa, Y., "Volume-rendered 3d display of mr angiograms in the diagnosis of cerebral arteriovenous malformations," *Acta Radiologica* **44**(6), 675–679 (2003).

[10] Kindlmann, G. L., Weinstein, D. M., Jones, G. M., Johnson, C. R., Capecchi, M. R., and Keller, C., "Practical vessel imaging by computed tomography in live transgenic mouse models for human tumors," *Molecular Imaging* **4**(4), 417–24 (2005).

[11] Rantzau, D., Lang, U., and Rühle, R., "Collaborative and interactive visualization in a distributed high performance software environment," in [*Proceedings of the International Workshop on High Performance Computing for Graphics and Visualization*], (1996).

[12] Schroeder, W., Martin, K., and Lorensen, B., [*Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*], Kitware, fourth ed. (2006).

[13] Bazilevs, Y., Hsu, M.-C., Zhang, Y., Wang, W., Kvamsdal, T., Hentschel, S., and Isaksen, J., "Computational vascular fluid-structure interaction: methodology and application to cerebral aneurysms," **9**, 481–498 (2010). 10.1007/s10237-010-0189-7.

[14] Zhang, Y., Wang, W., Liang, X., Bazilevs, Y., Hsu, M.-C., Kvamsdal, T., Brekken, R., and Isaksen, J., "High-fidelity tetrahedral mesh generation from medical imaging data for fluid-structure interaction analysis of cerebral aneurysms," *CMES: Computer Modeling in Engineering & Sciences* **42**, 141–150 (2009).

[15] Holman, R., Belay, E., Christensen, K., Folkema, A., Steiner, C., and Schonberger, L., "Hospitalizations for kawasaki syndrome among children in the united states, 1997-2007," *Pediatric Infectious Disease Journal* **29**, 483–488 (2010).

[16] "Simvascular: Cardiovascular modeling and simulation application," (2011). http://wiki.simtk.org/simvascular/.

[17] "Meshsim," (2011). http://www.simmetrix.com/products/meshsim/MeshSim.html.

[18] "Parallel hierarchic adaptive stabilized transient analysis (phasta)," (2011). https://www.scorec.rpi.edu/wiki/PHASTA.

[19] "Collaborative vision and simulation environment (covise)," (2011). http://www.hlrs.de/organization/av/vis/covise.