Parallel 3D Image Segmentation of Large Data Sets on a GPU Cluster

Aaron Hagan and Ye Zhao

Kent State University

Abstract. In this paper, we propose an inherent parallel scheme for 3D image segmentation of large volume data on a GPU cluster. This method originates from an extended Lattice Boltzmann Model (LBM), and provides a new numerical solution for solving the level set equation. As a local, explicit and parallel scheme, our method lends itself to several favorable features: (1) Very easy to implement with the core program only requiring a few lines of code; (2) Implicit computation of curvatures; (3) Flexible control of generating smooth segmentation results; (4) Strong amenability to parallel computing, especially on low-cost, powerful graphics hardware (GPU). The parallel computational scheme is well suited for cluster computing, leading to a good solution for segmenting very large data sets.

1 Introduction

Large scale 3D images are becoming very popular in many scientific domains including medical imaging, biology, industry etc. These images are often susceptible to noise during their acquisition. Image segmentation is a post processing technique that can show clearer results for analysis and registration. This topic has been a widely studied area of both 2D and 3D image processing and has been explored with a variety of techniques including (not limited to) region growing, contour evolutions, and image thresholding. The method we propose for performing image segmentation is an inherently parallel scheme based on the level set equation. Solving the level set equation is performed by using an extended lattice Boltzmann model (LBM) which provides an alternative numerical solution for the equation. This method has several advantages in that it is very easy and straightforward to implement, implicitly includes the computation of curvatures, has a unique parameter that controls the smoothness of the results, and finally, is parallel which allows it to be mapped to low-cost graphics hardware in a single GPU or GPU cluster environment.

The level set method uses a partial differential equation (PDE) to model and track how fronts evolve in a discrete domain by maintaining and updating a distance field to the fronts. Previous methods based on the level set formulation discretize the PDE with finite difference operators which lead to complex numerical computations. The state-of-the-art narrow-band method applies an adaptive strategy where the level set computation is only performed on a narrow region around the propagating contour. To expedite the narrow band on graphics hardware, Lefohn. et al [1,2] proposed a successful GPU implementation with narrow band packing and virtual memory management that arranges CPU-GPU data communication. The proposed method uses a GPU cluster environment to perform the segmentation of large datasets. With simple local operations, this method is a tool that is easily implemented on distributed machines, with minimal data management and communication through the network. Furthermore, it has the ability to handle curvature flows with its explicit computation, leading to a controllable noise reduction effect in the segmentation results. In detail, previous methods apply the narrow band with the corresponding priority data structure to adaptively propagate fronts to the target regions. A re-initialization of the narrow band is required to maintain the valid distance field. After this step, the new narrow band is packed and reloaded to the GPU. Our method is different in that we do not use a narrow band approach, and therefore, the distance field is always valid in the whole domain and no reload is needed. As a result there is no CPU-GPU crosstalk during segmentation and the data structures are easy to manage as they reside completely in the graphics hardware.

Abandoning the adaptive strategies of this solver may appear unconventional at first, as memory consumption increases and computations are performed globally. There is, however, less management of the data (in terms of narrow band computing) and future work could be expanded to develop an adaptive method for this solver. More importantly, this strategy arises based on the rapidly increasing computational power of GPUs (i.e. speed and memory size). For example, GPU memory is increasing rapidly, current graphics cards are equipped with up to 4 GB of memory on a single unit. Cluster systems that contain many GPUs with large memory capacity are becoming available and being used in many scientific applications. We anticipate that this trend will continue in the future. Further benefit comes from this method's easy implementation with under 100 lines of CPU and GPU code. A knowledgeable graduate student can implement the program in a short period of time. In summary, our approach shows that large volumetric data sets can be segmented in parallel on multiple GPUs with fast performance and satisfying results. To the best of our knowledge this approach is the first to solve level set segmentation of large 3D images on a GPU cluster.

2 Related Work

The level set equation has been used in a wide variety of image processing operations such as noise removal, object detection, and modeling equations of motion [3]. The level set equation can be used to perform the segmentation by creating an initial contour surface in the target image and having it evolve to regions of interest normally defined as target intensity values or gradients to attract the curve [4]. For GPU acceleration, a handful of work [5,1,6] has successfully applied the level set equation for image segmentation by solving it on the GPU.

The LBM method has been used in natural phenomena modeling in computer graphics and visualization [7]. The LBM-based diffusion has been used in image processing [8], where an anisotropic 2D image denoising is implemented on the CPU. Recently, Zhao [9] showed how the LBM scheme can be derived to solve volume smoothing, image fairing, and image editing applications. Tölke [10] described the parallel nature of LBM and how it can be mapped to the GPU through the CUDA library for modeling computational fluid dynamics. GPU cluster computing has been a rapidly developing research area which is adopted in many scientific computing tasks. Fan et al. [11] used the classic LBM for fluid modeling with their Zippy programming model for GPU clusters.

3 Introduction to LBM

LBM [12] originates from the cellular automata scheme which models fictitious particles on a discrete grid where each point of the grid contains a particular lattice structure with links to its neighbors. The lattice structures are defined by D3Q19 and D3Q7 which define the dimension and how many links are connected between the lattice and its neighbors. The fictitious particles moving along the links and their averaging behaviors are initially used to simulate traditional fluid dynamics. Using the numerical computing process derived from microscopic statistical physics, this recovers the Naiver-Stokes equations governing flow behaviors. The independent variables in the LBM equation consist of particle distribution functions of each link from a grid point to one of its neighbors. The particle distribution functions model the probability of a packet of particles streaming across one lattice link to its corresponding neighbor. Between two consecutive steps of the streaming computation, the function is modified by performing local relaxation that models inter-particle collisions. We refer the interested readers to a complete physical description [12], and its usage in visual simulations [7].

LBM Computation. The first step in performing a LBM simulation is to discretize the simulation domain to a grid, and generate the lattice structure for each grid point. For LBM simulations each grid point has a variety of different links to its neighbors. During each step of the simulation, collision and streaming computations are performed which are mathematically described as:

$$\textit{collision} \Rightarrow f_i(\boldsymbol{x}, t^*) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau}(f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)), \tag{1}$$

$$\textit{streaming} \Rightarrow f_i(\boldsymbol{x} + \boldsymbol{e}_i, t+1) = f_i(\boldsymbol{x}, t^*), \tag{2}$$

The local equilibrium particle distribution, f_i^{eq}, models collisions as a statistical redistribution of momentum. At a given time step t, each particle distribution function, f_i, along one link vector $\boldsymbol{e}_i$ at a lattice point, $\boldsymbol{x}$, is updated by a relaxation process with respect to f_i^{eq}. The collision process is controlled by a relaxation parameter τ. τ controls the rate at which the equation approaches the equilibrium state. After collision, the post-collision result is propagated to $\boldsymbol{x} + \boldsymbol{e}_i$. Here, $\boldsymbol{x} + \boldsymbol{e}_i$ locates a neighboring lattice point along the link i. This provides the distribution function value at time step t+1. f_i^{eq} can be defined by the Bhatnagar, Gross, Krook (BGK) model as

$$f_i^{eq}(\rho, \boldsymbol{u}) = \rho(A_i + B_i(\boldsymbol{e}_i \cdot \boldsymbol{u}) + C_i(\boldsymbol{e}_i \cdot \boldsymbol{u})^2 + D_i\boldsymbol{u}^2), \tag{3}$$

where A_i to D_i are constant coefficients chosen via the geometry of the lattice links and ρ is the fluid density computed as the accumulation of particle distributions by:

$$\rho = \sum_i f_i. \tag{4}$$

The LBM can be easily extended to incorporate additional micro-physics, such as an external force $\boldsymbol{F}$. This force affects the local particle distribution functions as follows:

$$f_i \longleftarrow f_i + \frac{(2\tau - 1)}{2\tau} B_i(\boldsymbol{F} \cdot \boldsymbol{e}_i). \tag{5}$$

By applying Chapman-Enskog analysis [13], the Navier-Stokes equation can be recovered from the equilibrium equation as:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{6}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = \nu \Delta \boldsymbol{u} + \boldsymbol{F}. \tag{7}$$

Here ∇ defines the gradient operator $(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$ and Δ is the Laplacian $\Delta = \nabla^2 = \frac{\partial^2}{\partial x} + \frac{\partial^2}{\partial y} + \frac{\partial^2}{\partial z}$.

Extended LBM. Though initially designed for fluid dynamics, the LBM method can be modified for modeling typical diffusion computations. Equation 3 can be simplified to:

$$f_i^{eq}(\rho) = A_i \rho, \tag{8}$$

which erases momentum terms and in effect removes the nonlinear advection term in the Navier-Stokes equation, which aren't needed for solving diffusion equations. As shown in [9], the parabolic diffusion equation can be recovered by the Chapman-Enskog expansion:

$$\frac{\partial \rho}{\partial t} = \gamma \nabla \cdot \nabla \rho, \tag{9}$$

where γ is a diffusion coefficient defined for a D3Q7 lattice by the relaxation parameter τ as:

$$\gamma = \frac{1}{6}(2\tau - 1). \tag{10}$$

In this case, we can also include the external force in the same way as in Equation 5. And thus, the modified LBM computation can recover the following equation:

$$\frac{\partial \rho}{\partial t} = \gamma \nabla \cdot \nabla \rho + \boldsymbol{F}. \tag{11}$$

Using this equation to compute a distance field (replace ρ by ϕ), we can recover the level set equation, where $\boldsymbol{F}$ is used to accommodate the speed function and the first term relates to the curvature flow effects.

4 Solving Level Set Equation

A distance field defines how far all points in a domain are to an existing surface, where the distance is signed to distinguish between inside and outside the surface. In this way, the surface S is defined as: $\phi : R^3 \rightarrow R$ for $p \in R^3$. The distance function is defined as:

$$\phi(p) = sgn(p) \cdot min\{|p - q| : q \in S\} \tag{12}$$

The surface can be considered as points with a zero distance value.

In image segmentation, the zero level set starts from an arbitrary starting shape and evolves itself by the following level set equation:

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| [\alpha D(x) + \gamma \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}] \tag{13}$$

where ϕ is the distance, $D(x)$ is the speed function that performs as a driving force to move the evolving level set to target regions, with a user-controlling parameter α (we use 0.01 in the examples). The second term represents curvature flow (smoothing). γ determines the level of curvature-based smoothness in the results. For a regular distance field, $|\nabla \phi| = 1$, which leads the last term to $\gamma \nabla \cdot \nabla \phi$. Note that $|\nabla \phi| = 1$ in our framework at all steps, since we do not use an adaptive approach and the distance field is valid in the whole domain. From this, Equation 13 is only a variational formula of Equation 11. It shows that the modified LBM computation leads to a new solution to the level set equation, enabling us to use the simple, explicit, parallel computational process for volume segmentation. In this way, our method also has the potential to be applied to other level set based applications. In our implementation, we apply a simple D3Q7 lattice that uses less memory and improves the performance, comparing with a traditional fluid solver using a D3Q19 lattice. This is made possible since the level set solution does not need to solve the nonlinear advection term as in the Navier-Stokes equations, and the D3Q7 lattice can provide enough accuracy.

Driving Speed Function. Speed functions are designed to make the evolving front of the zero level set propagate to certain target regions. We use a popular approach [3,14] where the speed function is defined by the difference between a target isovalue and a density value at each grid position:

$$D(I) = \epsilon - |I - T|, \tag{14}$$

where I is the voxel/pixel value at the grid position, and T represents the target density isovalue that the front should evolve to. As the front moves closer to the target region, the speed will converge to zero. The speed term also carries properties that allow the front to propagate in either direction, based on the sign of the function. The propagating front will expand if I falls in the T-ϵ or T+ϵ range, otherwise it will contract. The function $D(I)$ is easily applied in our

LBM computation as a body force $\boldsymbol{F}$ in Equation 11. $D(I)$ can also be derived based on the gradient defining the object boundary or other user-specified rules.

Level Set Curvature Computation. As mentioned earlier the LBM scheme inherently contains properties that model curvature during the collision and streaming process. The benefit of the LBM method is that the curvature does not need to be computed explicitly, because it is hidden in the microscopic LBM collision procedure.

From the LBM-solved diffusion Equation 9, we substitute fluid density ρ by the distance value ϕ. And then by applying $|\nabla\phi| = 1$ for distance field, we get:

$$\frac{\partial\phi}{\partial t} = \gamma\nabla\cdot(\frac{\nabla\phi}{|\nabla\phi|})|\nabla\phi| = \gamma\kappa|\nabla\phi|, \tag{15}$$

where κ represents the mean curvature:

$$\kappa = \nabla\cdot(\frac{\nabla\phi}{|\nabla\phi|}). \tag{16}$$

In summary, the modified LBM can implicitly provide the curvature-based smoothing effects, in contrast to upwinding difference methods that need to explicitly compute curvature components.

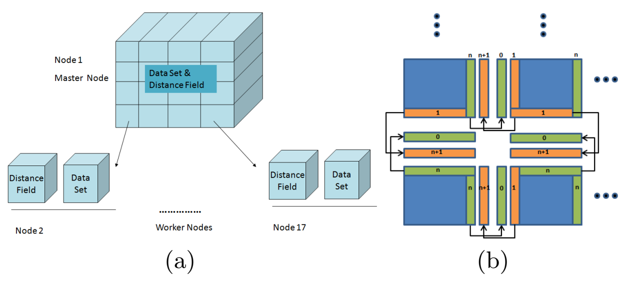

Fig. 1. (a) Data decomposition to cluster nodes for LBM computations. (b) Ghost layers used to transfer boundary data between neighboring nodes in the network.

5 Cluster Computing

Our method can easily be implemented on a single GPU, but to handle large data sets, we extend the algorithm to multiple GPUs organized in a cluster environment. Our cluster is Linux-operated and consists of seventeen nodes, each having a dual core or quad core AMD Opteron processor and a Nvidia 8800 GTX graphics card with 768 MB memory. The 3D volume data set is divided into 16 blocks and sent to the 16 worker nodes in a 4×4 organization of the nodes. One master node is used for managing initial data division, collecting and assembling results, and visualization. The master node assembles separate

Table 1. Performance report: Per step (in seconds) averaging speed to perform LBM computation and ghost layer communication, the memory size (in MB) per node, and the total ghost layer data size on the GPU cluster with 4×4 configuration.

Model	Size	LBM Speed Per Step	Ghost Layer Transfer Per Step	Total Speed Per Step	GPU Mem. Size Per Node	Ghost Layer Data Size
CT Head	$128 \times 128 \times 128$	0.01	0.04	0.05	4.5	5.2
MRI Head	$256 \times 256 \times 256$	0.08	0.21	0.29	36	21
Abdomen	$512 \times 512 \times 174$	0.22	0.51	0.73	97.9	28.5
Colon	$512 \times 512 \times 442$	0.57	1.61	2.18	248.6	72.5
Aneurism	$512 \times 512 \times 512$	0.66	1.96	2.62	288	84
Porche	$559 \times 1023 \times 347$	0.96	2.57	3.53	427	88
Bonsai	$1024 \times 1024 \times 308$	1.57	5.24	6.81	693	101.1

results with correct coordinate transformation and indexing, and uses a Marching Cubes method to render the segmented features of the distance field. Figure 1(a) shows this cluster configuration and data distribution. The data distribution is accelerated by using OpenMP to distribute the data in parallel.

Between consecutive LBM steps, it is necessary for the worker nodes to share the LBM data (i.e., f_i values) residing on the boundaries between each pair of neighboring blocks. We apply a ghost layer method to handle this problem. Each data block contains an extra layer of data, the ghost layer, to communicate with each of its neighbors, which is shown in Fig. 1(b). For example, one node A performs computation on data layer A_1 to A_n. After each step, data in A_1 is transferred to the ghost layer B_{n+1} of its neighbor node B. Meanwhile, B's B_n layer will be transferred to the ghost layer A_0 of A. In the next step, A will use A_0 to implement streaming operation and B will use B_{n+1} as well. The data transfer only involves the boundary layers with a very small amount of data compared with the total data size. With an infiniBand network equipped on our cluster, data can be transferred with a speed at an order of gigabits per second.

6 Results and Performance

We ran several volumetric data sets with various data sizes on our GPU cluster, where the computation was completely GPU-based. A 3D Aneurism dataset is used in Fig. 2 to show the results. The image sequence demonstrates the process of the level set propagation in different steps. A Marching Cubes method is used to generate a triangle mesh of the zero level set. The total steps used for the level set to reach final results are determined by the position and shape of the initial starting level set (we use a simple sphere in our examples). Fig. 4 shows another example of a CT abdomen data set and Fig. 3 uses a bonsai volume.

Table 1 outlines the performance results of several datasets. The average speed per step is composed of two parts: LBM computation and ghost layer handling. We also report the GPU memory consumption on each node, and the total size

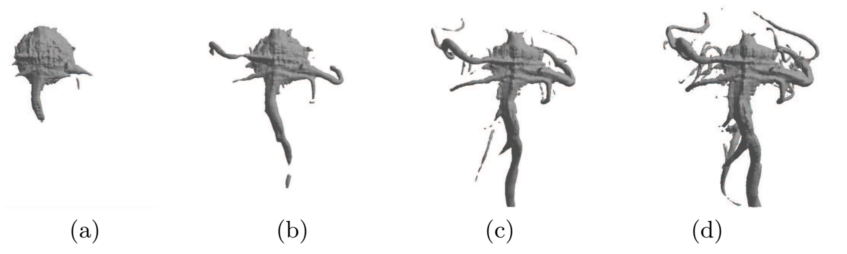

Fig. 2. Results of segmenting an Aneurism dataset with a target iso-value of 32. Data size is 512 × 512 × 512. $\gamma = 1.5$. (a) Level set propagates after 3 steps; (b) After 10 Steps; (c) After 25 Steps; (d) After 50 Steps.

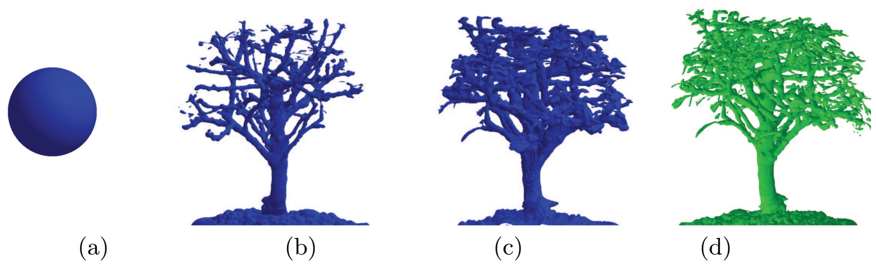

Fig. 3. Results of segmenting a volumetric Bonsai data with a target iso-value of 20. $\gamma = 5.125$. Data size is 1024 × 1024 × 308. (a) Initial level set as a sphere; (b) After 25 Steps; (c) After 45 Steps; (d) Direct rendering of isosurface with density values of 20.

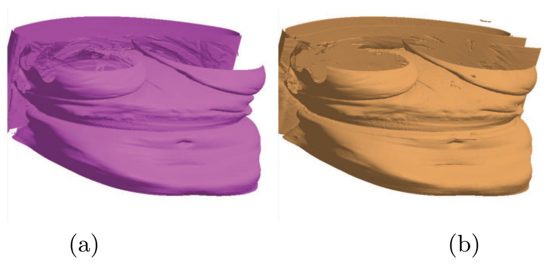

Fig. 4. Results of segmenting a 3D CT addomen dataset with a target iso-value of 62. $\gamma = 1.125$. Data size is 512 × 512 × 174. (a) Segmentation result (distance field) after 25 steps; (b) Direct rendering of isosurface with density values of 62.

of all ghost layers that determines the network traffic speed. It clearly shows that our method achieves very good performance to segment very large data. For the largest Bonsai data, it averages 6.81 seconds per step. The segmentation of the Bonsai completes in 45 steps, leading to a total processing time of around

306 seconds. The segmentation usually uses tens of total steps for large data. With an averaging per step speed at a few seconds, the whole process can generally be accomplished in tens to hundreds of seconds depending on the data sets and the initial distance field.

In detail, the LBM level set computation is very fast even for a large data set. It uses 1.57 seconds for the Bonsai data, which runs on one $256 \times 256 \times 77$ volume per node due to our data division scheme. The computation for the ghost layers handling is a little slower, which include (1) data readback from GPUs, (2) network transfer, and (3) data write to GPUs. For the Bonsai data, it costs 5.24 seconds. The total ghost layer data size (on all the nodes) reaches 101.1 MB. Although this data size does not impose a challenge on the infiniBand network, the GPU readback may consume a little more time than the LBM computation which is a known bottleneck of GPU computing. We plan to improve performance with further optimization of the ghost layer processing on faster GPUs and a new cluster configuration.

7 Conclusion

Common segmentation techniques such as isovalue threshholding are not adequate enough to handle complex 3D images generated by medical or other scanning devices. It proves necessary to implement advanced techniques which have the power to give clearer segmentation results by solving level set equations. Popular level set approaches on a single GPU are not easily extended to large volume data sets which are prevalent in practical applications. We have proposed an inherent parallel method to solve the segmentation problem flexibly and efficiently on single and multiple GPUs. Based on an extended LBM method, our method lends itself as a good segmentation tool with easy implementation, implicit curvature handling, and thus controllable smoothness of the segmented data. With its parallel scheme only minimal data processing is required for implementing the method in an GPU cluster compared with the previous single GPU approaches. We have reported good performance of multiple data sets on the cluster. In summary, our scheme provides a viable solution for large-scale 3D image segmentation in adoption of distributed computing technology. It has great potential to be applied in various applications. In the future, we will work on combining parallel visualization techniques with the segmentation, to further augment the ability of this method.

Acknowledgement

This work is partially supported by NSF grant IIS-0916131 and Kent State Research Council. The cluster is funded by the daytaOhio Wright Center of Innovation by the Ohio Department of Development. Please find the color version of this paper at the author's homepage.

References

1. Lefohn, A., Cates, J., Whitaker, R.: Interactive, GPU-based level sets for 3d brain tumor segmentation. In: Medical Image Computing and Computer Assisted Intervention, MICCAI, pp. 564–572 (2003)
2. Cates, J.E., Lefohn, A.E., Whitaker, R.T.: Gist: An interactive, GPU-based level-set segmentation tool for 3d medical images. Medical Image Analysis 10, 217–231 (2004)
3. Sethian, J.: Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science (1999)
4. Malladi, R., Sethian, J.A., Vemuri, B.C.: Shape modeling with front propagation: A level set approach. IEEE Transactions on Pattern Analysis and Machine Intelligence 17, 158–175 (1995)
5. Klar, O.: Interactive GPU based segmentation of large medical volume data with level sets. Diploma Thesis, VRVis and University Koblenz-Landau (2006)
6. Rumpf, M., Strzodka, R.: Level set segmentation in graphics hardware. In: Proceedings of IEEE International Conference on Image Processing (ICIP 2001), vol. 3, pp. 1103–1106 (2001)
7. Zhao, Y., Kaufman, A., Mueller, K., Thuerey, N., Rüde, U., Iglberger, K.: Interactive lattice-based flow simulation and visualization. In: Tutorial, IEEE Visualization Conference (2008)
8. Jawerth, B., Lin, P., Sinzinger, E.: Lattice Boltzmann models for anisotropic diffusion of images. Journal of Mathematical Imaging and Vision 11, 231–237 (1999)
9. Zhao, Y.: Lattice Boltzmann based PDE solver on the GPU. Visual Computer, 323–333 (2008)
10. Tölke, J.: Implementation of a lattice boltzmann kernel using the compute unified device architecture developed by nvidia. Computing and Visualization in Science (2008)
11. Fan, Z., Qiu, F., Kaufman, A.E.: Zippy: A framework for computation and visualization on a gpu cluster. Computer Graphics Forum 27(2), 341–350 (2008)
12. Succi, S.: Numerical Mathematics and Scientific Computation. In: The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press, Oxford (2001)
13. He, X., Luo, L.: Lattice Boltzmann model for the incompressible Navier-Stokes equation. Journal of Statistical Physics 88(3/4), 927–944 (1997)
14. Lefohn, A.E., Kniss, J.M., Hansen, C.D., Whitaker, R.T.: A streaming narrow-band algorithm: Interactive computation and visualization of level sets. IEEE Transactions on Visualization and Computer Graphics 10(4), 422–433 (2004)