Efficient GPU-Accelerated Computation of Isosurface Similarity Maps

Martin Imre* Jun Tao† Chaoli Wang‡

University of Notre Dame

ABSTRACT

We present an efficient GPU-based solution to compute isosurface similarity maps for scientific volume data sets. Our approach first replaces exact isosurface extraction with a binary volume indicating whether each voxel intersects the surface or not. We then employ bounding volume hierarchy (BVH)-trees to speed up the distance field computation. Finally, a self-similarity map is generated from which we identify representative isosurfaces. We apply our approach to compute isosurface similarity maps from different volume data sets of varying sizes and characteristics. The results demonstrate significant speed gain with acceptable loss of accuracy, showing the potential of our solution for handling large-scale time-varying multivariate data sets.

1 INTRODUCTION

Isosurface rendering and direct volume rendering remain two of the most popular techniques for visualizing scientific volume data sets. The former extracts and visualizes surface geometry equal to a given isovalue while the later maps voxels to optical quantities (color and opacity) for back-to-front or front-to-back compositing. In this paper, we focus on isosurface rendering. One key question for this surface-based approach is how to select salient or representative isosurfaces [2, 10, 11] so that the structures of the entire data set can be visually perceived by observing these representatives. A commonly adopted strategy is to select the most distinctive isosurfaces from a set of sample ones based on a similarity measure. Early approaches relied on data histograms and higher order moments [10]. Recently, Bruckner and Möller [2] employed mutual information to identify the similarity among isosurfaces and depict them in a so-called *isosurface similarity map*. Given a discrete set of n sampled isovalues, the self-similarity map is a $n \times n$ symmetric matrix recording the similarity values for each pair of isosurfaces. Haidacher et al. [3] used similarity maps to compare two different variables of a single volume for multimodal surface similarity. In this case, the similarity map is asymmetric.

Most scientific data sets are time-varying and multivariate. To obtain comprehensive insights, we should conduct a complete investigation by considering the time-varying and multivariate nature of scientific data sets and enabling comparative visualization across time and variable. However, the biggest impediment for leveraging isosurface similarity maps to study time-varying multivariate data sets is the huge computation cost involved. The previous work [2] reported around 25 minutes to compute a single similarity map. For a data set with ten variables and a hundred time steps, this amounts to more than 17 days just for getting self-similarity maps for all volumes, not to mention the addition time to compute asymmetric similarity maps for different pairs of variables or time steps.

We therefore present a cost-effective GPU-accelerated solution that optimizes every step of the process to efficiently generate isosurface similarity maps. The contributions of our work are as follows. First, we generate distance fields based on approximations without computing exact isosurfaces. Second, we propose a heuristic to improve the performance of searching the closest points using bounding volume hierarchy (BVH)-trees. Third, we implement our approach in CUDA to leverage the massive GPU parallelism. Finally, we run our solution on different data sets and provide thorough performance comparison. Our GPU-accelerated solution thus paves the way to computing isosurface similarity maps for large-scale time-varying multivariate data sets.

*e-mail: mimre@nd.edu

†e-mail: jtao1@nd.edu

‡e-mail: chaoli.wang@nd.edu

2017 IEEE Pacific Visualization Symposium (PacificVis)
18-21 April, Seoul, Korea
978-1-5090-5738-2/17/$31.00 ©2017 IEEE

2 RELATED WORK

Bruckner and Möller [2] introduced the concept of isosurface similarity map to combat the weaknesses of value frequency based histograms. While employing mutual information via joint histograms yields better insights into the similarity between isosurfaces, it entails a significant amount of computation. Haidacher et al. [3] advanced this approach to compare the isosurfaces of different modalities. Wei et al. [11] used a level-set method to explore the relevance of isosurfaces within subvolumess of a scalar field.

Ultimately, all of these methods rely on calculating the distance fields, which are a common task in many graphics and visualization applications. As distance fields are often needed for other processes in a large pipeline, a wealth of research has been done to accelerate their computation [5, 6, 7, 9, 12]. Yu et al. [12] introduced the so-called *distance tree* based on distribution of the computation over a cluster of CPU nodes. Liu and Kim [7] combined octrees and BVH-trees to compute distance fields on the GPU. As BVH-trees seem to be the preferred data structure for distance field computation, it is worth mentioning that — to the best of our knowledge — there exists no faster GPU based method of generating a BVH-tree than the one introduced by Karras [4].

3 OUR APPROACH

Our approach to generating isosurface similarity maps consists of three steps: sampling a set of isosurfaces, calculating distance fields, and computing the similarity map. First, we sample over the value range to generate a set of isosurfaces. Instead of computing the actual surfaces, we approximate each surface with a loosely ordered set of points for performance gain. Second, we compute a distance field for every sampled isosurface. We leverage BVH-trees for fast queries of the closest points and use a heuristic approach to improve the performance of traversing the tree based on the loose order of points. Finally, we calculate the similarity of every pair of sampled isovalues and organize all similarity values in a similarity map. The similarity between two sampled isovalues is given by the mutual information of their corresponding distance fields.

3.1 Isosurface Approximation

Isosurfaces are usually generated using the marching cubes algorithm [8], which is not only costly but also unnecessary for distance field calculation. The marching cubes algorithm computes intersection points of the isosurface with each voxel and connects these points for the exact surface. However, to calculate a distance field with reasonable precision, it is unnecessary to compute multiple points for one voxel or know the connections of those points.

Our approach approximates an isosurface S corresponding to a value v by generating one point for each 3D grid cell containing a

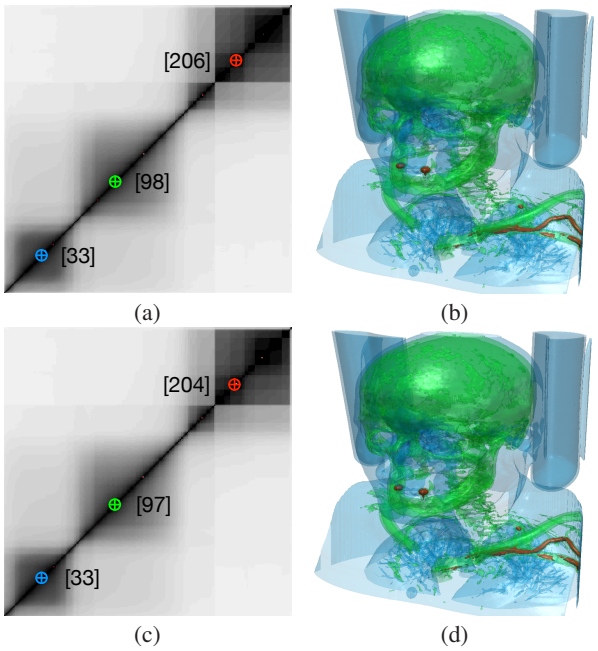

Figure 1: Comparing similarity map and representative isosurfaces generated using actual isosurfaces ((a) and (b)) against our approximation ((c) and (d)) with the CT data set. The isosurfaces are colored in the descending order of their importance: red, green, blue. The number in the brackets indicates the index of sampled isovalue. The marching cubes algorithm is used to extract the actual surfaces shown in (b) and (d) for given representative values.

part of S. A grid cell is a $1 \times 1 \times 1$ cube whose vertices are the eight voxels in a $2 \times 2 \times 2$ spatial region. Whether a grid cell contains a part of S is determined by checking the values of the eight voxels corresponding to it. If some of the values are larger than v and some are smaller than v, linearly interpolated values at some points in the grid cell must be v, which means that the grid cell contains a part of S. In this case, the center of this grid cell will be recorded as a sample point of S (for efficiency, we use the center instead of computing the actual intersection point). Otherwise, the grid cell does not contain any part of S and will be ignored. Our implementation leverages the parallelism of GPU to scan the grid cells simultaneously. Each GPU thread checks a grid cell and stores the result in a binary volume. Then we employ a GPU compaction scheme to convert the binary volume into an array of point coordinates (i.e., the approximation of an isosurface). The points are listed in the scanline order, providing loose spatial relationships among points.

Note that the error of using this approximation to compute the distance from a point p to a surface S is bounded by $\sqrt{3}/2$ (half the diagonal length of the grid cell) due to the triangle inequality. This error is acceptable, since downsampled distance fields are usually used to compute the similarity map [2]. In Figure 1, we can see that our approximation (c) yields a very similar similarity map as the exact marching cubes algorithm does (a). We further select three representative isosurfaces for each similarity map, which are shown in Figure 1 (b) and (d), respectively. Although there are slight differences in the selected isovalues, those are barely visible in the surface rendering. This minor difference is neglectable as the main goal of selecting the most salient values and depicting their surfaces is still achieved.

3.2 Distance Field Computation

A distance field of an isosurface is a volume with every voxel containing the distance from the center of this voxel to the isosurface. In order to compute this, it is necessary to find the closest point of an isosurface for every voxel in the distance field. To decimate the search space for this problem, we downscale the resulting distance field. A typical solution for such a search problem is a tree structure. In our approach, we leverage BVH-trees to accelerate the search for the closest point. A BVH-tree is a binary tree used to subdivide a scene into a tree structure. Each node of a BVH-tree consists of a bounding volume that includes all of its descendants. A leaf consists of a single primitive — in our case a single point. Although other bounding volumes increase the traversal speed, we use bounding boxes for lower construction time and apply Karras's algorithm [4] to build BVH-trees in parallel. This is essential since a tree needs to be built for every isosurface approximation.

To compute the distance fields of a given isosurface approximation, we spawn a CUDA thread to search the closest point in the BVH-tree for each voxel by traversing the tree in a depth-first search manner. During the traversal for a voxel v, we maintain an upper bound of the closest distance from v to the isosurface. The sub-tree rooted at an internal node n will not be traversed, if the distance $d(v,n)$ between v and n is larger than the upper bound. The distance $d(v,n)$ between a voxel v and an internal node n is given by the minimum distance from v to the bounding box of n. The distance $d(v,n)$ between a voxel v and a leaf n is given by the distance between v and the corresponding point of n. Specifically, the traversal procedure can be described in the following four steps:

1. Take a set of sample points P and calculate the initial upper bound $d_u = \min_{p \in P} d(v,p)$.
2. Add the root to a stack.
3. Pop the first node n from the stack. If n is an internal node, for each of its child node n_c, compute the distance $d(v,n_c)$. If $d(v,n_c) < d_u$, push n_c onto the stack. If n is a leaf, compute the distance $d(v,n)$ between v and n, and update d_u with $\min(d_u, d(v,n))$.
4. Repeat Step 3 until the stack becomes empty.

For an efficient traversal, a good initial upper bound is needed. Unlike the existing approaches, which usually initialize the upper bound when the first leaf is encountered, we estimate the upper bound before the traversal by computing the distance from the voxel to a set of sample points. Note that the points approximating an isosurface are listed in the scanline order. By evenly sampling the approximating points in that array, we obtain a set of sample points that is roughly evenly spaced over the isosurface. This provides a tighter estimation of the initial upper bound, allowing more branches to be pruned, especially at the beginning stage of the traversal.

3.3 Similarity Map Generation

The similarity between two isosurfaces are measured by the mutual information of their corresponding distance fields. To calculate the mutual information, we compute a joint histogram of the two distance fields, where each entry (i, j) in the joint histogram contains the number of voxels that fall into bins i and j in the first and second distance fields, respectively. Then, the mutual information can be derived from the joint histogram. In our GPU implementation, we spawn a CUDA thread for each pair of sampled isosurfaces. Each thread reads two distance fields, computes the joint histogram and mutual information, and stores the value of mutual information in the similarity map.

4 Results and Discussion

We used multiple data sets with different characteristics to benchmark our application on a desktop with an Intel i7-4790 quad-core CPU running at 3.6 GHz, 32 GB main memory, and an NVIDIA GeForce GTX 760 GPU with 2 GB memory. Throughout the evaluation we used Bourke's [1] implementation of the marching cubes

data set	variable	time step	dimensions	surface points	approx.	distance field	similarity map	marching cubes surface	marching cubes DF	approximation surface	approximation DF
Ionization	GT	75	$600 \times 248 \times 248$	148,213	78.12	9.54	35.45	2607	14.50	590	9.54
Ionization	GT	146	$600 \times 248 \times 248$	185,897	78.02	10.27	37.35	2617	12.67	590	10.27
Ionization	H	150	$600 \times 248 \times 248$	156,086	77.85	7.32	21.10	2586	9.93	590	7.32
Combustion	CHI	45	$480 \times 720 \times 120$	1,984,240	90.14	57.92	45.66	4033	69.91	695	57.92
Combustion	CHI	90	$480 \times 720 \times 120$	2,823,210	89.45	82.53	46.05	4524	102.77	683	82.53
Combustion	Mixture Fraction	45	$480 \times 720 \times 120$	375,039	88.06	16.69	43.17	3085	20.63	666	16.69
Combustion	Mixture Fraction	90	$480 \times 720 \times 120$	495,256	87.85	20.45	43.72	3137	25.05	666	20.45
Combustion	Vorticity	45	$480 \times 720 \times 120$	1,462,340	88.56	42.62	40.55	3710	52.62	669	42.62
Combustion	Vorticity	90	$480 \times 720 \times 120$	1,980,120	88.87	54.55	41.24	4031	66.72	672	54.55
CT	–	–	$256 \times 256 \times 230$	248,607	32.90	8.05	17.16	1199	10.02	243	8.05
Hurricane	Precipitation Ratio	17	$500 \times 500 \times 100$	639,957	53.81	16.28	24.21	2075	19.60	402	16.28
Hurricane	Precipitation Ratio	36	$500 \times 500 \times 100$	790,787	53.97	20.36	25.06	2164	24.68	402	20.36
Hurricane	Water Vapor Ratio	17	$500 \times 500 \times 100$	408,140	53.56	11.98	25.99	1959	15.66	402	11.98
Hurricane	Water Vapor Ratio	36	$500 \times 500 \times 100$	448,670	53.67	12.91	26.15	1975	16.46	402	12.91

Table 1: Timing and parameters for each data set and variable, as well as the comparison of running time of isosurface generation using marching cubes and our approximation solution. All timing results are in seconds and for all isosurfaces of 256 sampled isovalues. The column "surface points" shows the average numbers of points per approximated isosurface. Distance fields were downscaled by 8 in each dimension. We used 1500 sample points prior to the traversal. The last columns of "surface" and "DF" show the running time for surface computation and distance field calculation, respectively.

algorithm as a baseline. Table 1 shows the total time (in seconds) spent in each step of the similarity map generation pipeline. The timing results were collected with a GPU implementation for all 256 isovalues using each data set.

It took us under 2.5 minutes to generate an isosurface similarity map for most of these data sets. The only exception is the *combustion* data set, which has the largest volumes and the greatest numbers of points in the surfaces. As we can see from Table 1, the three variables (*CHI*, *mixture fraction*, and *vorticity*) of the *combustion* data set have the same dimensions and cost similar time to approximate the isosurfaces. But these variables have a high fluctuation in the number of points in surface approximation, which leads to different running times to compute the distance fields of the same size. Generally, for all data sets, we can observe that the time for surface approximation is proportional to the size of volumes and the time for distance field computation depends on both the dimensions and surface points. Furthermore, the traversal time for the BVH-trees highly depends on the choice of sample points. For this evaluation we opted to choose 1500 as this tends to yield a swift traversal time for all of the data sets.

To demonstrate the quality of the isosurface similarity maps produced by our approach, we identified representative isosurfaces following the work of Bruckner and Möller [2]. Figure 2 shows the three most significant isosurfaces for certain data sets and variables in the first row and the corresponding similarity maps in the second row. The blue isosurfaces are the most important ones, followed by green and orange. The corresponding isovalues are highlighted in the similarity maps as well. For each variable, we can observe that the three selected isosurfaces capture the principal patterns. In addition, Figure 1 shows that we obtained similar similarity maps and representative isosurfaces as reported in [2]. With our acceleration solution and the modern graphics hardware, the performance is greatly improved, as the time to generate the similarity map is reduced from 22.1 minutes (refer to Table 1 in [2]) to less than one minute (refer to the *CT* data set in Table 1).

Furthermore, we examined the difference of similarity maps when applying different levels of downsampling for the distance fields. Specifically, we downscaled the distance fields by 8 along each dimension and compared this to a version with a downscale by 4. While the overall time to calculate 256 distance fields for a data set already shows a 3.5-fold increase, the similarity map computation suffered even more from increasing the accuracy. On average the similarity map calculation took 7.8 times longer than that with a lower resolution of distance fields. We computed the difference of every entry in the similarity maps using the two downscaling factors for all data sets listed in Table 1. Our results show an average difference of 1.5% using the lower resolution distance fields, which is fairly acceptable in exchange for significant performance gain. Further reducing the downsampling rate inevitably leads to an even higher increase in computation time while not improving the results by a reasonable amount.

4.1 Isosurface Approximation

With our approximation, we reduced the time for generating the surfaces points by an ample amount. Table 1 compares the running time of the marching cubes algorithm and our approximation both running on CPU. As we can see, our approximation outperforms the marching cubes algorithm, achieving an average speedup of 5.1 times for surface creation.

Another advantage of the approximation is that the points in the surface are loosely sorted, which allows our heuristic sampling method to find a tight initial upper bound for the tree traversal used in distance field computation. This increases the performance of distance field calculation by 20% on average. Although this only amounts to a difference of a few seconds in a process that takes around 10 minutes on average, once the GPU approximation is realized, this difference would lead to up to 10% of the total time before creating the similarity map.

4.2 Initial Upper Bound Estimation

We examined the impact of the number of samples to the performance of BVH-trees traversal when estimating the initial upper bound. As one could expect, the size of the sample point set has a high influence on the duration of the traversal. With more sample points, the time to estimate an initial upper bound increases, while the traversal time decreases due to a tighter upper bound. Figure 3 shows the average time spent on traversing the trees for a given number of sample points. Without an initial upper bound (i.e., zero sample point), the traversal takes the longest time, as the tree needs to be examined until the first leaf is reached. After that the average duration for the traversal falls rapidly in the beginning and then slowly approaches the minimums. Most data sets have their lowest traversal time between 2000 and 2300 points and just fluctuate by one millisecond.

Interestingly, the two variables *CHI* and *vorticity* of the *combustion* data set reach their minimums early with just 1200 and 1500 sample points respectively. After that, the running time rapidly increases from 94.58 to 380.36 (*CHI*), and from 69.25 to 248.82 (*vorticity*). These two variables have higher numbers of points in surface approximation (refer to Table 1), which makes it sur-

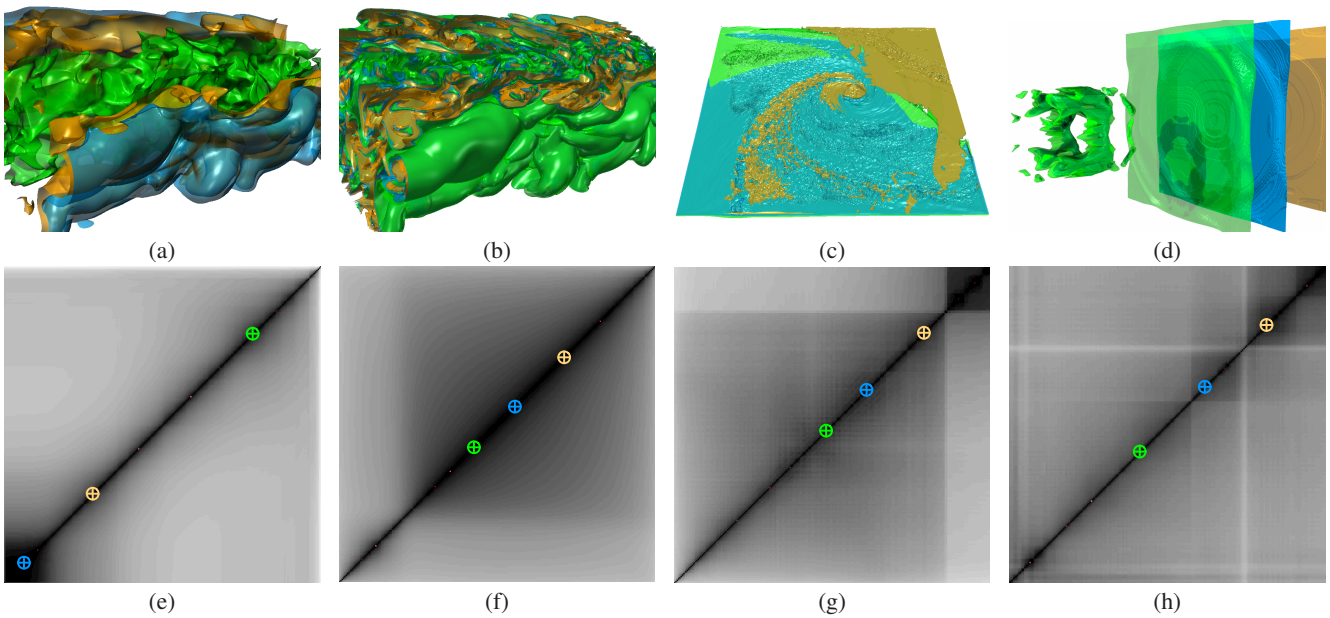

Figure 2: Rendering of the most important isosurfaces (a) to (d), with their corresponding similarity maps (e) to (h). The first and second columns show *mixture fraction* and *vorticity* of the *combustion* data set, respectively. The third column shows *water vapor ratio* of the *hurricane* data set, and the fourth column depicts *GT* of the *ionization* data set.

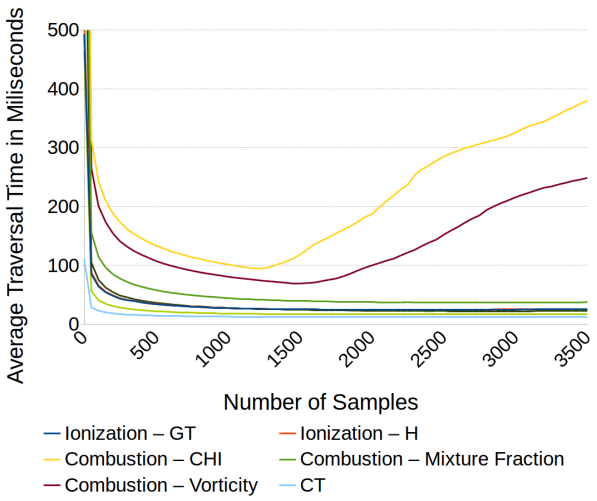

Figure 3: Average traversal time for different numbers of sample points. We sampled from 0 to 3500 points with a step size of 50.

prising since we expect that in those cases a higher quality upper bound would be more beneficial. The rapid increases of the running time may be related to the different characteristics shown in the isosurfaces of these two variables. We compared the representative isosurfaces of the variables *mixture fraction* and *vorticity* that are shown in Figure 2 (a) and (b), respectively. Other than having tremendously more points per isosurface, the isosurfaces of *vorticity* distribute more evenly over the entire volume than those of *mixture fraction*. The same nature can be observed for isosurfaces of *CHI*. We suspect that the high number of points, coupled with their spatial distribution of *CHI* and *vorticity*, allows an easy identification of tight upper bounds, rendering more examination prior to the traversal unnecessary.

We expect a similar increase of the running time for the other data sets with higher numbers of sample points as this process converges toward a brute force approach. But the performance gained by using a tighter upper bound usually outweighs these costs during the traversal for less than 1500 sample points. As we can see in Figure 3, it is safe to take any value between 1000 and 1500 for the data sets we chose. In general, if the data sets examined are of a reasonable size, a value in that range will always be preferable over a smaller value for selecting the initial upper bound.

5 Conclusions and Future Work

With our approach we manage to achieve a considerable speedup without impacting the quality when creating isosurface similarity maps. Our isosurface approximation allows fast generation of the point sets needed for computing the distance fields with small bounded errors. Furthermore, we use BVH-trees to efficiently compute the distance fields. The loosely sorted points from our approximated surface generation enables the use of our heuristic for finding a tight upper bound before initiating the tree traversal. Our experiment demonstrates that the number of samples can be safely determined to achieve significant speedup.

We notice that the calculation of joint histograms and evaluation of mutual information for the isosurface similarity map still takes about a third of the total running time. This is due to the fact that the sheer amount of data needed for this step does not easily fit into the memory of our GPU. Thus the parallel computational power can not simply be leveraged. Future work could aim on a change in the algorithm or on diverse memory management in order to overcome these obstacles. Our work allows users to extract the most important isosurfaces from a data set within a much short amount of time, opening up the possibility to perform a more thorough analysis employing multiple variables and time steps.

Acknowledgements

This work was supported in part by the U.S. National Science Foundation through grants IIS-1456763, IIS-1455886, and CNS-1629914.

REFERENCES

[1] P. Bourke. Polygonising a scalar field. `http://paulbourke.net/geometry/polygonise/`, 1994. [Online; accessed 24-January-2017].

[2] S. Bruckner and T. Möller. Isosurface similarity maps. *Computer Graphics Forum*, 29(3):773–782, 2010.

[3] M. Haidacher, S. Bruckner, and E. Gröller. Volume analysis using multimodal surface similarity. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1969–1978, 2011.

[4] T. Karras. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In *Proceedings of ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*, pages 33–37, 2012.

[5] A. Krishnamurthy, S. McMains, and K. Haller. GPU-accelerated minimum distance and clearance queries. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):729–742, 2011.

[6] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast distance queries with rectangular swept sphere volumes. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1–8, 2000.

[7] F. Liu and Y. J. Kim. Exact and adaptive signed distance fields computation for rigid and deformable models on GPUs. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):714–725, 2014.

[8] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of ACM SIGGRAPH Conference*, pages 163–169, 1987.

[9] J. Seitzer. Parallel computation of the Euclidean distance transform on a three-dimensional image array. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):203–212, 2003.

[10] S. Tenginakai, J. Lee, and R. Machiraju. Salient iso-surface detection with model-independent statistical signatures. In *Proceedings of IEEE Visualization Conference*, pages 231–238, 2001.

[11] T.-H. Wei, T.-Y. Lee, and H.-W. Shen. Evaluating isosurfaces with level-set-based information maps. *Computer Graphics Forum*, 32(3):1–10, 2013.

[12] H. Yu, J. Xie, K.-L. Ma, H. Kolla, and J. H. Chen. Scalable parallel distance field construction for large-scale applications. *IEEE Transactions on Visualization and Computer Graphics*, 21(10):1187–1200, 2015.