An Evaluation of Peak Finding for DVR Classification of Biological Data

Aaron Knoll, Rolf Westerteiger and Hans Hagen

Abstract In medicine and the life sciences, volume data are frequently entropic, containing numerous features at different scales as well as significant noise from the scan source. Conventional transfer function approaches for direct volume rendering have difficulty handling such data, resulting in poor classification or undersampled rendering. Peak finding addresses issues in classifying noisy data by explicitly solving for isosurfaces at desired peaks in a transfer function. As a result, one can achieve better classification and visualization with fewer samples and correspondingly higher performance. This paper applies peak finding to several medical and biological data sets, particularly examining its potential in directly rendering unfiltered and unsegmented data.

1 Introduction

Direct volume rendering (DVR) is a ubiquitous method for visualizing 3D raster data, including medical and biological scan data. Volume data in the life sciences are often noisy and contain features at numerous different scales. This poses difficulties for many classification schemes and automatic transfer function generators. While gradient-based 2D transfer functions [8] deliver clear improvements, they can be difficult to create and manipulate compared to 1D approaches. As a result, high-quality DVR classification methods are often abandoned in favor of more approximate methods such as unshaded grayscale maximum-intensity projection (MIP), or simple isosurface rendering.

Aaron Knoll
University of Kaiserslautern, Germany, e-mail: knoll@rhrk.uni-kl.de

Rolf Westerteiger
University of Kaiserslautern, Germany e-mail: rwester@informatik.uni-kl.de

Hans Hagen
University of Kaiserslautern, Germany e-mail: hagen@informatik.uni-kl.de

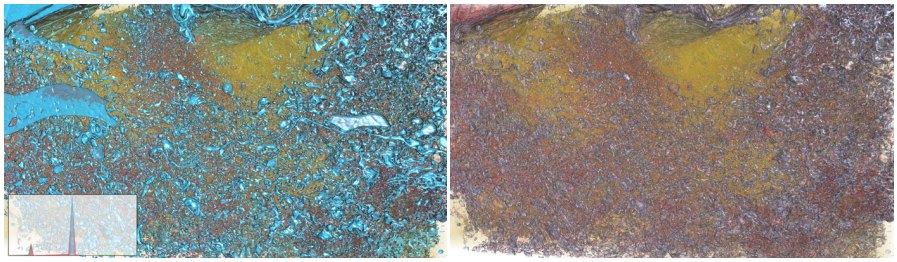

Fig. 1 Zoomed-out zebrafish optic tract rendered with peak finding (top) and preintegration (bottom), at 3.0 and 3.4 fps, respectively with a 1440x720 frame buffer.

Similarly, when rendering discrete isosurfaces within a volume, the preferred method is often to specify the isosurfaces explicitly, or even extract them separately as geometry, rather than to embed them directly in a transfer function. While it handles the most glaring artifacts resulted from point-sampled postclassification, preintegrated classification [4] must still sample adequately with respect to the Nyquist frequency of the data, and omits features when it does not.

Peak finding [10] is a recently-proposed classification scheme that bridges DVR classification and discrete isosurfacing. Rather than sampling uniformly along the ray, peak finding explicitly solves for peaks in the transfer function at their corresponding isovalues in the data field. By sampling and shading directly at these peaks, this method achieves better results than postclassification or preintegration when a transfer function is sufficiently sharp, particularly in the case of Dirac peaks. More significantly, in the case of entropic data, peak finding can successfully identify features at peaks that are omitted by standard methods, even when the chosen transfer function is modest. While the original paper detailing the peak finding algorithm revealed the general appeal of this method in handling noisy data, it only examined one biological data set.

This paper investigates the merits of peak-finding in noisy medical and biological volume data. In particular, we are interested in the application of peak finding as an exploratory classification early on in the data analysis pipeline, especially in visualizing unsegmented and unfiltered data directly from scan source. While peak finding is clearly useful in some scenarios, we are careful not to oversell its merits. The goal of this work is to reveal the strengths as well as limitations of peak-finding as an exploratory classification technique.

2 Related Work

Levoy [13] employed ray casting in the first implementation of direct volume rendering. The advent of z-buffer hardware and built-in texture interpolation units allowed for interactive performance with slice-based rasterization approaches [2, 3]. While slower per-sample than slicing, volume ray casting methods are feasible on programmable GPU hardware [11, 22] and currently represent the state-of-the-art [21].

Isosurface rendering has conventionally been accomplished via mesh extraction using marching cubes [14] or more sophisticated offline methods [24]. Volume-isosurface ray casting methods were first conceived by by Sramek [26]. Parker et al. [20] implemented a tile-based parallel ray tracer and achieved interactive rendering of isosurfaces from large structured volumes, employing a hierarchical grid as a min-max acceleration structure and an analytical cubic root solving technique for trilinear patches. Hadwiger et al. [6] combined rasterization of min-max blocks with adaptive sampling and a secant method solver to ray cast discrete isosurfaces on the GPU. Peak finding [10] is inspired by this approach in that it uses discrete sampling to isolate roots.

A large body of volume rendering literature deals with transfer functions, both in how to construct them and employ them in classification. To limit artifacts when sampling high-frequency features of a transfer function, the best existing approaches are preintegration [4,15,23] and analytical integration of specially constructed transfer functions [9]. Hadwiger et al. [5] analyze the transfer function for discontinuities to generate a pre-compressed visibility function employed in volumetric shadow mapping. Our approach is similar except that we search for local maxima, and use these directly in enhancing classification.

Higher-order filters have previously been the subject of study for structured [17, 19] and unstructured [12] volumes. In our higher-order reconstructions, we employ a method similar to [25], in which a 4-point stencil filter is efficiently reconstructed on the GPU using two linearly interpolated fetches. We also experiment with more rigorous filters which are processed offline, namely the anisotropic diffusion filter proposed by Weikert [27].

3 Background

In direct volume rendering, irradiance I is computed as a discrete approximation of the radiative light transport equation through a continuous scalar field. On a segment of a ray, this is given by

$$I(a,b) = \int_a^b \rho_E(f(s))\rho_\alpha(f(s))e^{-\int_a^s \rho_\alpha(f(t))dt}ds \tag{1}$$

where ρ_E is the emmissive (color) term and ρ_α is the opacity term of the transfer function; a,b are the segment endpoints, and $f(t) = f(\mathbf{O}+t\mathbf{D}) = f(\mathbf{R}(t))$ is the scalar field function evaluated at a distance t along the ray. Computing this integral entails approximating it with discrete samples. With uniformly spaced sampling, we break up the ray into equally spaced segments and approximate the opacity integral as a Riemann Sum,

$$e^{-\int_a^s \rho_\alpha(f(t))dt} = \prod_{i=0}^{n} e^{-\Delta t\ \rho_\alpha(f(i\ \Delta t))} = \prod_{i=0}^{n}(1-\alpha_i) \tag{2}$$

where Δt is the uniform sampling step, $n = (s-a)/\Delta t$, and

$$\alpha_i \approx 1 - e^{-\Delta t \ \rho_\alpha(f(i\ \Delta t))} \tag{3}$$

By discretizing the integral on $[a,b]$ in Equation 1 as a summation, we have the following discrete approximation for I,

$$I \approx \sum_{i=0}^{n} \rho_E(i) \prod_{j=0}^{i-1} (1-\alpha_j) \tag{4}$$

where $\rho_E(i) = \rho_E(f(i\,\Delta t))$ is given by the transfer function. Evaluating the transfer function after reconstruction is known as postclassification. While it is equally possible to classify before filtering, preclassification results in worse reconstruction of the convolved scalar field and is unnecessary on current hardware.

3.1 Classification methods

Postclassification (Figure 2(a)) usually delivers adequate sampling when the transfer function and data field are both sufficiently smooth. To accomplish this, one generally requires a sampling rate beneath the Nyquist limit of this convolved signal. In rendering high-frequency features, and particularly discrete isosurfaces, postclassification with a uniform step size will invariably fail. Theoretically, an isosurface is a subset of a DVR transfer function, consisting of a discrete Dirac impulse at that isovalue. To visualize this surface in a volume renderer, one must sample that impulse with an infinite (continuous) sampling rate. In practice, it is generally sufficient to specify fuzzy isosurfaces, sacrificing some classification precision in the interest of visual aesthetics.

Engel et al. [4] note that we can effectively sample at the minimum of either the transfer function or the data frequency. Preintegrated transfer functions build upon this notion by integrating separately over transfer function and scalar field domains (Figure 2(b)). This allows isosurface-like features to be rendered more accurately using a lower sampling rate. However, preintegration still assumes the scalar field is sampled adequately. When the data itself is noisy, it is often impractical to sample at the Nyquist limit of the data for reasons of performance. As a result, both postclassification and preintegration can fail to reconstruct high-frequency features of interest when undersampling the scalar field. Moreover, preintegration assumes a piecewise integration over the transfer function domain between two samples, which does not correspond to the actual scalar field value along the ray, particularly when the source data is entropic. Although interpolating between endpoints can remove obvious shading artifacts [15], preintegration can still fail to reproduce features when the data domain is undersampled.

Peak finding [10] performs a similar function as preintegration, but employs standard point-sampled postclassification when the transfer function frequency is low,

and an explicit isosurface root-solving technique when the transfer function contains a peak on a given range. This consists of a precomputation step to identify peaks in the transfer function, and a root-solving method to solve for them during ray casting. As a result, sharp features in the transfer function are always approximated by discrete isosurfaces, which a ray will either hit or miss. In this sense, peak finding can be seen as a modification to postclassification, but using an adaptive sampling mechanism to identify sharp features as necessary (Figure 2(c)).

Unlike preintegration, peak finding relies on Descartes' rule of signs to determine whether a (root) feature lies between two endpoints of a segment. With preintegration, the integral of ρ on $[a,b]$ is a piecewise summation of all ρ values between $f(a)$ and $f(b)$. For this to be accurate, f must vary smoothly between a and b; beyond general continuity it should be Lipschitz, e.g.

$$L(a,b) = \frac{|\rho(f(b)) - \rho(f(a))|}{b-a} < k \tag{5}$$

where $b-a = \Delta t$, and this sample rate is chosen according to k. Conversely, peak finding only requires that the scalar field be monotonic on $[a,b]$ to find and accurately reproduce the high-frequency sample. Clearly, when the data is noisy and sampled below the Nyquist limit, both conditions can fail. However, monotonicity is a far weaker requirement than Lipschitz continuity, and the success of peak finding is due in great part to this difference.

3.2 Peak finding vs. preintegration

Peak finding is attractive in that its algorithm is not significantly different from either volume rendering or isosurface ray casting. Both algorithms employ regular sampling, in the case of DVR to compute the volume rendering integral and in the case of isosurfacing to isolate roots. Peak finding takes advantage of this and does both. As a result, this technique can be implemented quickly by extending existing renderers. Although we propose peak finding in conjunction with differential sampling, the two techniques are orthogonal. It is equally possible to employ peak finding in a uniform sampling ray caster, a slice-based volume renderer, or a shear-warp system. Moreover, as described in Section 4.1, it is simple to extend an existing preintegration scheme to handle peak finding, with a small modification to the lookup table construction and classification algorithm.

Overall, peak finding and preintegration are similar, but make different assumptions about the integral over a given segment. Preintegration assumes this integral can be accurately approximated by piecewise summation. This works well when the transfer function and convolved field are smooth, but encounters difficulties when they are not. Peak finding assumes this integral can be approximated by one or several discrete impulses. This introduces bias, but is better suited for noisy data and sharp C_0 transfer functions for which standard techniques fail.

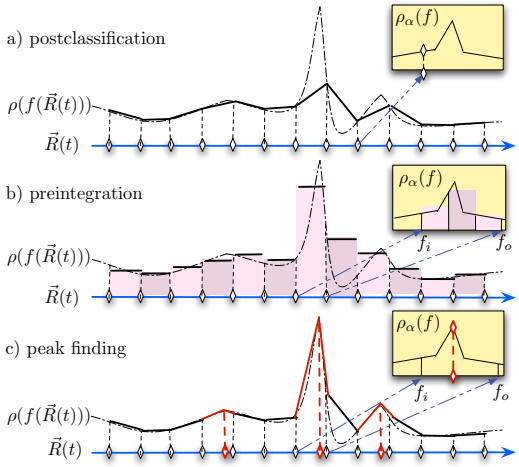

Fig. 2 Postclassification, preintegration and peak finding illustrated.

4 Peak Finding

This section discusses peak finding, which was originally proposed in [10]. We again note this is not a new contribution, but rather a reiteration of the technique that will be evaluated in more detail in the remainder of this paper.

Peak finding combines standard postclassification with an explicit ray-isosurface intersection in high-frequency points of interest at transfer function peaks. Ray-isosurface intersection consists of solving the reconstruction filter function $f(x,y,z) = \upsilon$ at an isovalue υ. Substituting the ray equation $\mathbf{R}(t) = \mathbf{o} + \mathbf{d}t$, we have a one-dimensional equation in terms of the ray parameter t:

$$f(x,y,z) - \upsilon = f(\mathbf{R}(t)) - \upsilon = 0 \tag{6}$$

In peak finding, we simply need to know the isovalue υ if and where a peak exists between any two sample values. Then, we can employ a common root-finding technique to find the t at which to sample.

4.1 Building the peak finding table

The peak finding table is a 2D lookup table as with preintegration. For each discretized segment, we stores an isovalue υ or (optionally) a set of isovalues υ_i that possibly exist within this segment. These values are sorted from the first to last peak value encountered on a given segment defined by the entry and exit values of the scalar field function, $[f_i, f_o]$. In cases where ρ is not monotonic on $[f_i, f_o]$ and multiple peaks are encountered, we reverse the order when necessary.

We first build a sorted 1D array of peaks from the transfer function ρ_α. A peak is simply defined at a a local maximum. The set of peaks consists of at most half the

number of actual data points in our piecewise-linear transfer function, but typically it is far less. Smooth 1D functions such as splines would have relatively fewer peaks, existing at the critical points of these functions. We, we use piecewise-linear transfer functions, as we are primarily interested in specifying and handling sharp features.

The lookup table construction then proceeds as follows: for a range of values $[i, j]$ corresponding to lookup entries from our volume $f(\underline{t}), f(\bar{t})$. If $i < j$, we search our transfer function for the next peak point (or in the case of multiple peaks, next 4 points) such that the opacity $\rho_\alpha(\upsilon) > i$ and $\rho_\alpha(\upsilon) \leq j$. If $i > j$, we search in descending order for peaks with $\rho_\alpha(\upsilon) \leq i$ and $\rho_\alpha(\upsilon) > j$. When necessary, a segment spanning multiple peaks will reverse the sorting order to register all possible peaks within that segment. This process is again similar to preintegration, except that separate discrete peak values are stored instead of a single integral approximation. In each table entry, we store the domain isovalue(s) υ corresponding to each peak. When no peak exists, we use a flag outside of the range of scalar values in the volume. Building the lookup table is relatively undemanding, and proceeds in $O(N^2)$ time, similarly to the improved algorithm of [15] for preintegration. In practice, building a peak-finding table is roughly twice as fast as building a preintegrated table at the same resolution due to the lack of floating point division. More importantly, in most cases a coarser discretization (128 or 256 bins) is sufficient for peak finding, whereas preintegration would require a larger table (512-1024 bins) for comparable quality when rendering near-discrete isosurfaces, limiting interactivity when changing the transfer function.

4.2 Classification

In the main ray casting loop (for example in a fragment shader), we can perform peak finding between samples in the place of a preintegrated lookup. We first query the peak finding table from a 2D texture to determine whether or not there is a peak. If the peak exists, we subtract that isovalue from the entry and exit values, and employ Descartes' rule of signs. If this test succeeds, we assume the segment contains a root. Bracketed by $\underline{t}$,$\bar{t}$, we use three iterations of a secant method (also employed by [6, 16]) to solve the root:

$$t_1 = t_0 - f(t_0)\frac{t_1 - t_0}{f(t_1) - f(t_0)} \tag{7}$$

When the secant method completes, we have an estimate for the root t along the ray segment. We now sample at this position and perform postclassification. However, sampling at the peak requires two subtle choices. First, we do not evaluate our field $f(\mathbf{R}(t))$, but rather assume that the value at this point is our desired isovalue. This works because we are solving for the root position, not its value; moreover for sharp transfer functions it is crucial in avoiding Moire patterns. Second, we do not not scale ρ_α by the segment distance Δt (in Equation 3) but instead use a constant $\Delta t = 1$. Although this may seem counterintuitive, the scaled extinction coefficient is itself a correction mechanism for the inherently discrete approximation of the

volume rendering integral. Lastly, one can choose either to solve for a single peak or up to 4 multiple peaks, depending on their spacing within the volume. In practice, multiple peaks are seldom necessary, and we do not use them in the examples in this evaluation.

4.3 Implementation

As in the original peak finding paper [10], our implementation consists of a straightforward GLSL shader in OpenGL. The 1D transfer function is given as a set of points $\{v,\{r,g,b,a\}\}$, then processed into a fairly wide (8K elements) 1D texture, allowing for rapid access on the GPU and generally sufficient transfer function precision $\Delta f > 1e-4$.

For space skipping, we employ a uniform macrocell grid. This is generated directly from the transfer function on the CPU, stored in a 3D texture on the GPU, and then traversed directly within the shader via a 3DDDA algorithm [1]. We use a simple measure of local gradient to adapt the step size within each macrocell. In this scheme, each macrocell computes a metric based on the ratio of the maximum standard deviation of its voxels to that of the entire volume, and uses this as a rough multiplier for the frequency:

$$m = \sqrt{\lceil \mathrm{Var}(f_{cell}) \rceil / \lceil \mathrm{Var}(f_{vol}) \rceil} \tag{8}$$

As this corresponds to frequency, its inverse can be used to vary the sampling step size Δt. In practice we wish this to be a positive integer, and a multiplier $M = 2m^{-1} + 1$ delivers good results.

For dynamic higher-order filtering, we implemented a tricubic B-spline filter using the method of [25], with the BC smoothing ($B = 2, C = 1$) kernel of [18]. We optionally employ this for both sampling and root solving processes.

5 Results

Results were gathered on an NVIDIA 285 GTX at resolutions around 1 MP (1024x1024). Unless otherwise stated, we disabled differential sampling [10]. While this results in roughly 1.5x-3x worse performance at equivalent quality, we wished to evaluate peak finding alone, without this additional control variable. As a result of this, and of most images requiring over 300 samples per ray for adequate sampling, rendering these data sets is noninteractive (though generally explorable, over 1 fps) for most images in this paper, even on the NVIDIA 285 GTX. We stress that differential sampling is a practical technique to improve performance at little cost in visual accuracy. Peak finding can also be implemented in a slicer framework, which could resulting in improved performance when differential sampling is not employed. Unfortunately, aggressive rasterization-based culling mechanisms would be of little use for these data sets, as there is little exploitable empty space. Moreover, the goal of this paper (and peak finding in general) is to show better exploratory classification rather than ensure interactivity.

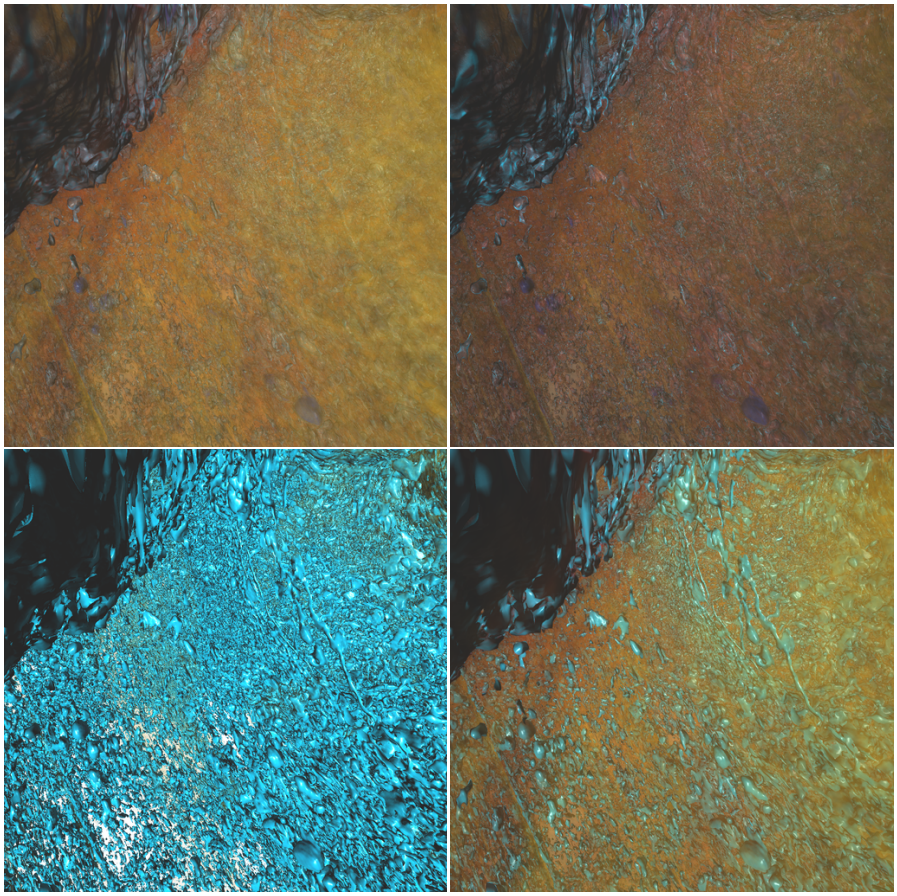

Fig. 3 Axons and glial cells in a zebrafish optic tract acquired through confocal microscopy. Top left to bottom right: postclassification, preintegration, semi-transparent peak isosurfaces, and peak finding. These images rendered at 1.9, 1.7, 2.6 and 2.0 fps, respectively at 1024x1024.

We consider three data sets in this paper that we believe are fairly representative of biological and medical volume data. The first is a moderately large (910x512x910) segmented data set of a zebrafish optic tract acquired through scanning electron microscopy [7]. The next is a highly noisy volume containing a zebrafish embryo, also acquired through microscopy. Finally, we consider a brain MRI before and after segmentation. Most of these datasets benefit from moderately sharp (though not Dirac) transfer functions when performing 1D classification; in most cases peak finding aids in highlighting sharp surface features, though not necessarily in correcting other artifacts stemming from scan, segmentation or reconstruction deficiencies.

5.1 Zebrafish Optic Tract

In Figure 3, we consider a close-up of the zebrafish optic tract with postclassification, preintegration, isosurfacing only, and peak finding. This scene is similar to that in the original peak finding paper [10] but with a more revealing transfer function, and with results considered in greater detail. The transfer function consists of a moderately sharp peak, as shown in the teaser (Figure 1). Both postclassification and preintegration significantly undersample the sharp feature corresponding to the axons. While increasing the sampling rate can ultimately alleviate this problem, it is computationally prohibitive; with preintegration we needed roughly 12 times as high as sampling rate to achieve comparable feature reconstruction, resulting in a frame rate of 0.1 fps for these images.

Not all surface features are omitted by standard DVR when sampling below the data Nyquist limit. Indeed, large membranes such as the feature in the upper-left corner are successfully reproduced by standard methods. However, features defined beneath the sampling frequency are generally omitted. For moderately large data such as the zebrafish, this in fact encompasses most features the user would be interested in, i.e. axons and glial cells. Finally, while rendering of discrete isosurfaces at the transfer function peaks is useful, it tells us nothing about the density of the interstitial media. Rendering with peak finding has the advantage of improved depth perception, and more flexible classification.

5.2 Filtered vs. unfiltered

To evaluate peak finding on filtered data, we consider zebrafish embryo data also acquired through electron microscopy. Though smaller, it is noisier than the optic tract and more difficult to classify. One goal in analyzing this data is to identify distinct cells and their boundaries as they undergo mitosis. Even with a smoothing filter, it is difficult to analyze this data without more sophisticated segementation or analysis. With peak finding, our goal is simply to better visualize cell boundaries where they might exist.

5.2.1 Dynamic filtering

We use a strong smoothing filter from the BC family ($B = 2, C = 1$), which consists of an blend between tri-cubic Catmull-Rom (interpolation) and B-spline (smoothing) bases. More details are discussed by Mitchell [18].

Figure 4 illustrates the difference between standard trilinear interpolation and tricubic BC filtering. In all cases, peak finding helps us find a particularly sharp boundary near the cell membrane, which helps visually separate cell interiors from the outside glial fluid. The original data is sufficiently noisy that it is difficult to discern cells with both standard DVR and semi-transparent isosurfacing. With peak finding, it is still difficult to interpret, but specifying a sharp feature at the cell membrane provides better intuition. Combining this with clipping planes yields reasonable first-glance classification, without needing to classify data offline or employ more complicated 2D transfer functions.

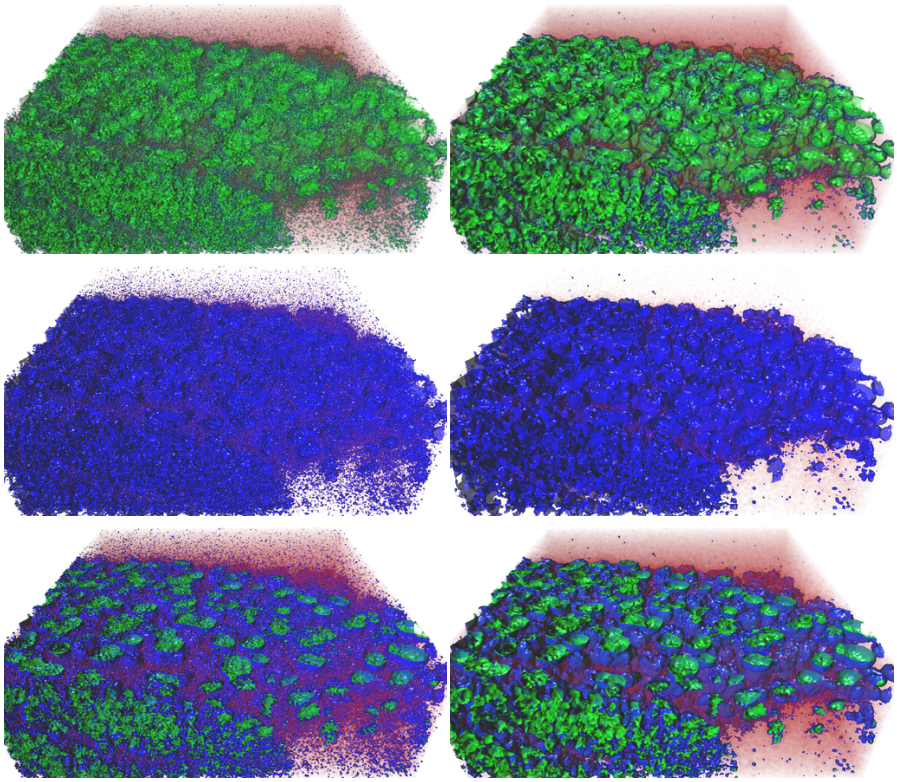

Fig. 4 Zebrafish embryo with trilinear (left), dynamic B-spline (right) filtering, using preintegration (top), isosurfacing (middle) and peak finding (bottom).

One advantage of peak finding is that when dynamic higher-order filters are used to reconstruct data, the root-solving technique can use those as well, effectively allowing for direct raycasting of higher-order isosurfaces within the DVR framework. Conversely, employing a smoothing filter can improve the entropy of the original scalar field function, effectively making $\rho(f)$ more Lipschitz and lessening the need for peak finding. In most cases, peak finding remains useful nonetheless, as strong smoothing reconstructions can remove features from the data before they become renderable with standard techniques at lower frequencies.

5.2.2 Static (preprocessed) filtering

An wider assortment of reconstructions, such as Kalman filters [7] or anisotropic diffusion filtering [27] can be afforded via offline filtering of the volume data. These are often employed as a first step for highly noisy data such as the zebrafish embryo. Unfortunately, we cannot use this filter kernel in DVR sampling or peak finding between discrete voxels. In the example in Figure 5, where data is processed with an anisotropic diffusion filter, the scalar field is smoothed to the point that conventional classification techniques begin to work. Unlike in the previous example (Figure 4), it is possible to distinguish cell boundaries with preintegration, and peak finding helps

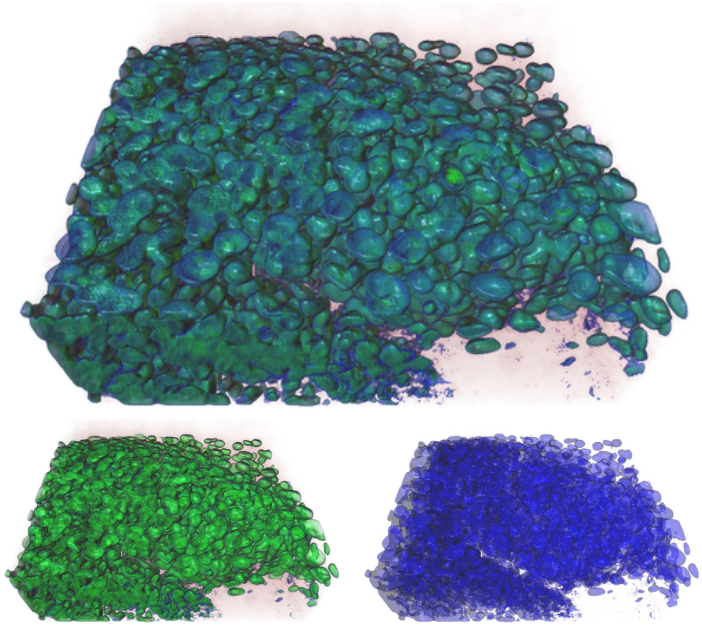

Fig. 5 Embryo dataset preprocessed offline with an anisotropic diffusion filter, with peak finding (top), postclassification (bottom left) and semitransparent isosurfacing (bottom right).

subtly, if at all. It is interesting to note that with high sampling rates and turbulent (though not excessively noisy) data such as the example in Figure 5, peak finding optical effects similar to subsurface scattering, despite only employing straight primary rays. This arguably helps accentuate boundaries.

5.3 Limitations of Peak Finding

When beginning this evaluation, we were hopeful that peak finding would provide useful insights towards two difficult problems in visualization: directly rendering unsegmented volumes, and handling anisotropic gaps in data scanned slice-by-slice. Unfortunately, these pose issues beyond the realm of surface reconstruction from a given filter, and are not addressed by peak finding.

5.3.1 Unsegmented volume data

We applied preintegration and peak finding to an MRI scan of a brain before and after segmentation (Figure 6). Unfortunately, peak finding if anything exhibits worse results than standard DVR, due to the worse occlusion of boundary features. Considering this problem more carefully, it is clear why peak finding is not helpful: segmentation is really an art of identifying connected 3-manifolds and labelling them correctly, rather than simply identifying fine 2-manifold boundaries (which peak-finding successfully does).

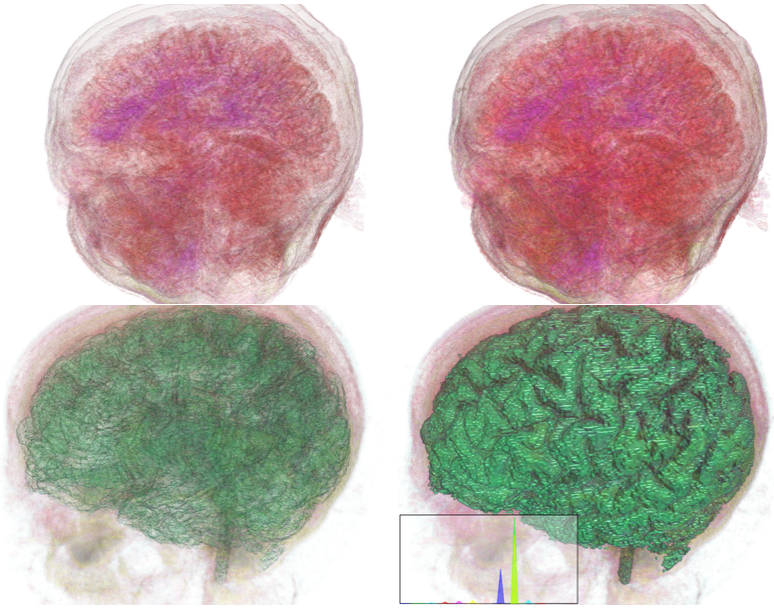

Fig. 6 Top: unsegmented MRI scan. Neither preintegration (left) nor peak finding (right) result in useful visualization of boundaries. Bottom: in some cases, preintegration fails to reconstruct sharp boundaries that peak finding detects.

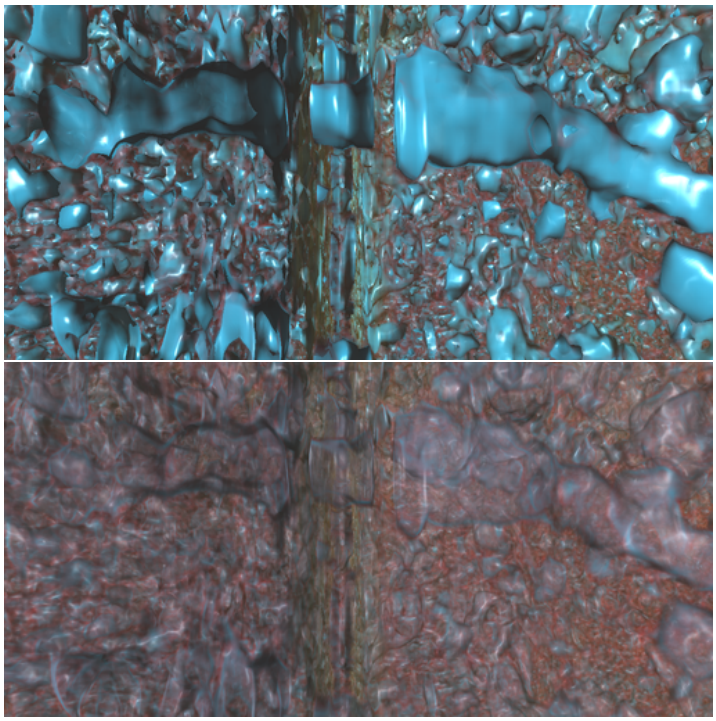

Fig. 7 Anisotropic gaps in the volume data, with peak finding (top) and preintegration (bottom).

5.3.2 Anisotropic Gaps

Individual slices can often be ommitted in CT or microscopy acquisition, leaving conspicuous gaps in the volume. Ideally, it would be useful to reconstruct features between these gaps. For small gaps, we considered using peak finding and a higher-order smoothing filter to simply interpolate values across neighboring voxels. However, peak finding only accentuates these omissions, as shown in Figure 7. This is not necessarily undesirable; and in a sense is to be expected. It is again worth noting that peak finding does not repair artifacts inherent to the original scalar field.

6 Discussion

Peak finding is a simple classification technique that delivers concrete advantages to biological and medical visualization, as well as any visualization of noisy data with sharp 2-manifold boundaries. When rendering sharp transfer functions and reconstructing surfaces, it poses significant advantages over preintegration, which has been conventionally used for this purpose. While not a replacement for advanced surface reconstruction or poor segmentation, it is a useful exploratory tool for scientific visualization.

The classifications used in this paper were all 1D transfer functions. It is important to note that many of the abilities of peak finding can be achieved with 2D gradient-magnitude classification at even lower sampling rates. However, it is theoretically possible to perform multidimensional peak finding, therefore a comparison of multidimensional classification methods should consider both methods. We are greatly interested in pursuing such an evaluation as future work.

Acknowledgments

This work was supported by the German Research Foundation (DFG) through the University of Kaiserslautern International Research Training Group (IRTG 1131); as well as the National Science Foundation under grants CNS-0615194, CNS-0551724, CCF-0541113, IIS-0513212, and DOE VACET SciDAC, KAUST GRP KUS-C1-016-04. Additional thanks to Liz Jurrus and Tolga Tasdizen for the zebrafish data, to Younis Hijazi, Rolf Westerteiger, Mathias Schott and Chuck Hansen for their assistance, and to the anonymous reviewers for their comments.

References

1. J. Amanatides and A. Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *Eurographics '87*, pages 3–10. Eurographics Association, 1987.
2. B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 91–98, New York, NY, USA, 1994. ACM Press.
3. T. J. Cullip and U. Neumann. Accelerating Volume Reconstruction With 3D Texture Hardware. Technical report, University of North Carolina at Chapel Hill, 1994.
4. K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIG-*

GRAPH/EUROGRAPHICS workshop on Graphics hardware, pages 9–16. ACM New York, NY, USA, 2001.
5. M. Hadwiger, A. Kratz, C. Sigg, and K. Bühler. GPU-accelerated Deep Shadow Maps for Direct Volume Rendering. *Graphics Hardware*, 6:49–52, 2006.
6. M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. Gross. Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces. *Computer Graphics Forum*, 24(3):303–312, 2005.
7. E. Jurrus, M. Hardy, T. Tasdizen, P. Fletcher, P. Koshevoy, C. Chien, W. Denk, and R. Whitaker. Axon tracking in serial block-face scanning electron microscopy. *Medical Image Analysis*, 13(1):180–188, 2009.
8. J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional Transfer Functions for Interactive Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.
9. J. Kniss, S. Premoze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun. Gaussian transfer functions for multi-field volume visualization. In *Visualization 2003*, pages 497–504, Oct. 2003.
10. A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, and H. Hagen. Volume ray casting with peak finding and differential sampling. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2009)*, 15(6):1571–1578, Nov-Dec 2009.
11. J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings IEEE Visualization 2003*, 2003.
12. C. Ledergerber, G. Guennebaud, M. Meyer, M. Bächer, and H. Pfister. Volume MLS Ray Casting. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1372–1379, 2008.
13. M. Levoy. Display of Surfaces from Volume Data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988.
14. W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics (Proceedings of ACM SIGGRAPH)*, 21(4):163–169, 1987.
15. E. Lum, B. Wilson, and K. Ma. High-quality lighting and efficient pre-integration for volume rendering. In *Proceedings Joint Eurographics-IEEE TVCG Symposium on Visualization 2004 (VisSym 04)*, pages 25–34. Citeseer, 2004.
16. G. Marmitt, H. Friedrich, A. Kleer, I. Wald, and P. Slusallek. Fast and Accurate Ray-Voxel Intersection Techniques for Iso-Surface Ray Tracing. In *Proceedings of Vision, Modeling, and Visualization (VMV)*, pages 429–435, 2004.
17. S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of Visualization '94*, pages 100–107. IEEE Computer Society Press, 1994.
18. D. Mitchell and A. Netravali. Reconstruction filters in computer-graphics. *ACM Siggraph Computer Graphics*, 22(4):221–228, 1988.
19. T. M oller, R. Machiraju, K. Mueller, and R. Yagel. Evaluation and design of filters using a taylor series expansion. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):184–199, 1997.
20. S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive Ray Tracing for Isosurface Rendering. In *IEEE Visualization '98*, pages 233–238, October 1998.
21. C. Rezk-Salama, M. Hadwiger, T. Ropinski, and P. Ljung. Advanced illumination techniques for gpu volume raycasting. In *ACM SIGGRAPH Courses Program*. ACM, 2009.
22. S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, and W. Strasser. Smart hardware-accelerated volume rendering. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 231–238, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
23. S. Röttger, M. Kraus, and T. Ertl. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Proceedings of the conference on Visualization'00*, pages 109–116. IEEE Computer Society Press Los Alamitos, CA, USA, 2000.
24. J. Schreiner and C. Scheidegger. High-Quality Extraction of Isosurfaces from Regular and Irregular Grids. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1205–1212, 2006. Member-Claudio Silva.
25. C. Sigg and M. Hadwiger. Fast third-order texture filtering. *GPU Gems*, 2:313–329, 2005.
26. M. Sramek. Fast Surface Rendering from Raster Data by Voxel Traversal Using Chessboard Distance. *Proceedings of IEEE Visualization 1994*, pages 188–195, 1994.
27. J. Weickert. *Anisotropic diffusion in image processing*. ECMI Series, Teubner-Verlag, Stuttgart, Germany, 1998, 1998.