DOI: 10.1111/cgf.12799

COMPUTER GRAPHICS *forum*
Volume 00 (2016), number 00 pp. 1–14

Output-Sensitive Filtering of Streaming Volume Data

Veronika Solteszova[1,2], Åsmund Birkeland[2], Sergej Stoppel[2], Ivan Viola[2,3] and Stefan Bruckner[2]

[1]Christian Michelsen Research, Bergen, Norway
veronika@cmr.no
[2]University of Bergen, Norway
{asmund.birkeland, sergej.stoppel, ivan.viola, stefan.bruckner}@uib.no
[3]TU Wien, Austria

Abstract

Real-time volume data acquisition poses substantial challenges for the traditional visualization pipeline where data enhancement is typically seen as a pre-processing step. In the case of 4D ultrasound data, for instance, costly processing operations to reduce noise and to remove artefacts need to be executed for every frame. To enable the use of high-quality filtering operations in such scenarios, we propose an output-sensitive approach to the visualization of streaming volume data. Our method evaluates the potential contribution of all voxels to the final image, allowing us to skip expensive processing operations that have little or no effect on the visualization. As filtering operations modify the data values which may affect the visibility, our main contribution is a fast scheme to predict their maximum effect on the final image. Our approach prioritizes filtering of voxels with high contribution to the final visualization based on a maximal permissible error per pixel. With zero permissible error, the optimized filtering will yield a result that is identical to filtering of the entire volume. We provide a thorough technical evaluation of the approach and demonstrate it on several typical scenarios that require on-the-fly processing.

Keywords: visibility, volume data processing, object-order imaging

ACM CCS: Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics] Three-Dimensional Graphics and Realism – Visible line/surface algorithms. I.4.3 [Image Processing and Computer Vision] Enhancement – Filtering

1. Introduction

Rapid increases in processing power coupled with advances in ultrasound technology have enabled the real-time capture of high-resolution 3D volumes, hence making ultrasound a true 4D imaging modality. 4D ultrasonography is already an important imaging technique in obstetrics and cardiology and has an ever-increasing range of applications. In echocardiography, 4D ultrasound is used to diagnose the functioning of the valve and in intra-operative guidance it is used as a navigational aid for minimally invasive valve replacement. In obstetrics, 4D ultrasound offers many potential benefits in the prenatal diagnosis of neurological problems by enabling the assessment of grimacing, breathing movements, swallowing, mouthing, isolated eye-blinking and revealing the orientation of the limbs [LV11]. Similarly, 4D ultrasound provides significant advantages in assessing fetal heart defects [Ion10]. Recent developments are even progressing toward portable 4D scanners, such as GE's *Voluson i* system, which open up new possibilities in point-of-care medicine, e.g. for emergency medicine in crisis zones.

Despite its advantages, the visualization of 4D ultrasound data is plagued by several challenges. Other tomographic imaging modalities such as CT or MRI have a much better signal-to-noise ratio, but ultrasound additionally suffers from various acoustic artefacts such as attenuation, focusing and interference [NPH*00]. In volume visualization, these artefacts are particularly problematic, since they can obscure relevant structures, and hence affect the clinical value of the resulting image. A significant body of research has been devoted to the investigation of filtering strategies for the removal of speckle noise and other artefacts, but effective methods typically have a considerable computational cost [SSHW*12]. For regular 3D data or prerecorded 4D sequences, filtering is regarded as a one-time operation, and is thus not considered as performance-critical. However, when data are streamed at 5–20 volumes per second during a live examination, a delay in processing or the skipping of frames is

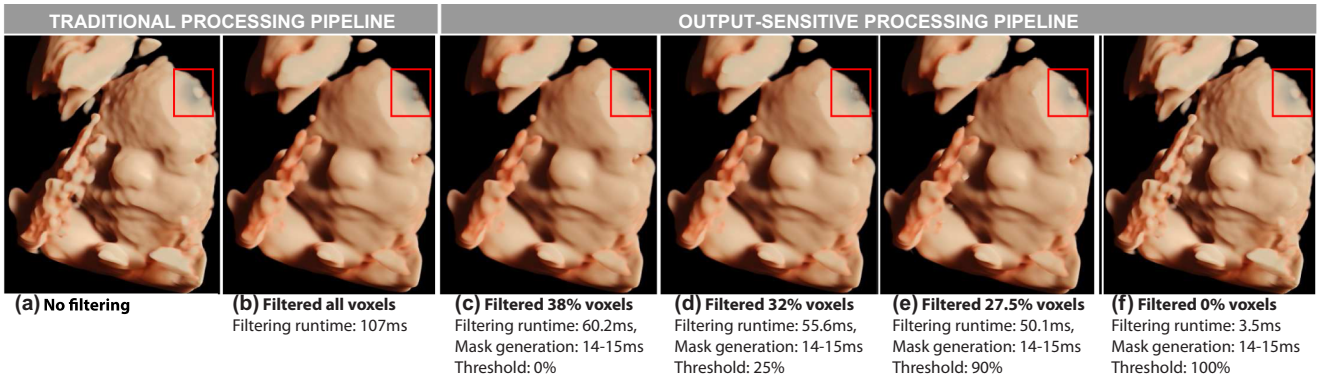


Figure 1: *Fetal ultrasound dataset Anna: (a) raw data, (b) with all voxels filtered and (c) with all voxels that contribute to the visualization filtered (threshold 0%). (b) and (c) yield the same results. In (d), (e) and (f), we are omitting more voxels from filtering based on their contribution to the image. In (d), the difference is not noticeable. An error threshold of 100% corresponds to no filtering and therefore there are no differences between the images in (a) and (f). The output is mostly sensitive in thin and highly transparent regions, here marked in red.*

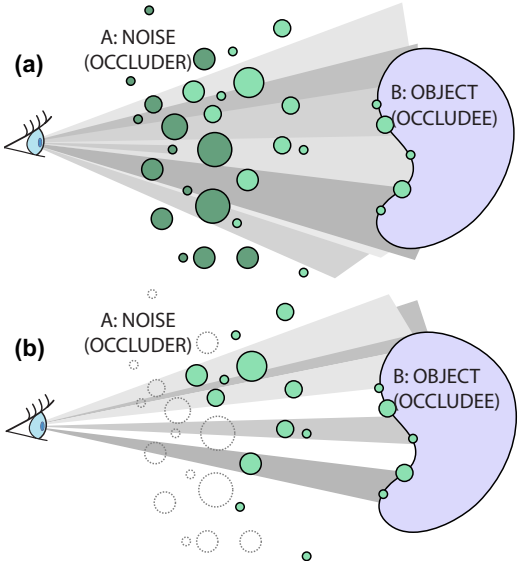


Figure 2: *(a) Noise A (green) is partly visible in the original dataset (dark green) and it completely occludes object of interest B. (b) Data filtering removes the visible noise. Consequently, the object of interest B is now partially visible.*

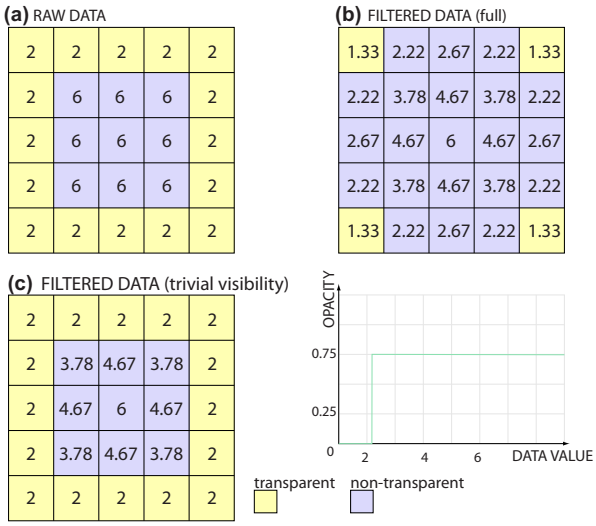


Figure 3: *A 2D example: (a) raw data to be filtered with a simple 3×3 averaging kernel. (b) The ground truth, where all data points are filtered. (c) The result if we filtered only the visible (non-transparent) data.*

clinically not acceptable. Hence, high-quality visualization of live 4D ultrasound raises a need to tackle this problem.

We propose a novel output-sensitive method to address the challenge of time-demanding filtering of 4D ultrasound data (see also Figure 1). Key to reducing the computational load of filtering operations is to give priority to regions of the volume depending on their contribution to the visualization. Typically, only a small fraction of all voxels will be visible, since they are either classified as transparent or occluded by other structures. However, in contrast to widely used culling techniques, which exploit visibility information for improving rendering performance, we need to account for the fact that filtering operations themselves, since they modify the underlying data values, influence the visibility and the appearance of regions in the volume. We give special attention to the following cases:

1. A region in the volume considered to be an occluder before the filtering operation has been applied, but may get new values which are mapped to an opacity of zero during the visualization stage. This means that it no longer acts as an occluder. This case is illustrated in Figure 2.
2. The filtering operation can also change data values in such a manner that previously invisible regions will be non-transparent after its application. An example of such a scenario is shown in Figure 3.

3. Filtering alters data values and after application of the transfer function, the color assigned to a voxel may change.

We solve these non-trivial problems by quickly identifying a voxel's impact on the visualization, taking into account its potential change in visibility and color after undergoing the filtering process. This paper extends our previous work on visibility-driven filtering [SBVB14]. The original approach only considered the case where the visual result of the pipeline was identical to applying the filter to all voxels. In this paper, we generalize and extend the method by quantifying the visual impact of not filtering a voxel. This allows us to introduce a visual tolerance which can be used to prioritize the voxels to be filtered. We propose an optimization scheme that uses the estimated contribution of voxels to maximize the number of voxels excluded from processing while keeping the summed maximal possible error per pixel below a user-specified threshold. Moreover, we extend our framework to filters that also contain negative values in their kernels.

2. Related Work

The topic of *visibility* is concerned with the classification of all objects in a scene as either being partially visible or totally invisible. The task of rapidly identifying entirely invisible objects is especially important when interactive frame update rates are desired. The surveys by Bittner and Wonka [BW03] and Cohen-Or *et al.* [COCSD03] provide a good introduction into basic techniques. Approaches include image-space methods such as hierarchical z-buffering and hierarchical occlusion maps, spatial subdivision (e.g. octrees, BSP trees, K-D trees), portals, and potentially visible sets.

Another related concept is *deferred shading*, which moves expensive shading operations to later processing stages where they are only applied to those object and image regions that actually need to be processed. First introduced by Deering *et al.* [DWS*88], a common approach uses two rendering passes where the first pass simply outputs the image-space depth and normals per fragment. The second pass then uses these values to compute the final image. The deferred shading pipeline allows for fast computation of differential surface properties, e.g. curvature descriptors. Hadwiger *et al.* [HSBG05] use deferred shading on isosurfaces to realize advanced shading effects such as ridge and valley lines and curvature-based flow advection.

In volume visualization, exploiting visibility information can lead to massive speedups of the rendering process. Early ray termination terminates ray traversal when the accumulated opacity reaches a certain threshold. Levoy [Lev90] proposed a hierarchical subdivision of the volume for empty-space skipping. A very common approach is the use of a min-max octree in combination with a summed area table of the transfer function to quickly identify empty parts of the volume [LL94]. Boada *et al.* [BNS01] proposed a management policy, which uses a hierarchical approach in homogeneous regions and regions of low interest instead of a one-texel-per-voxel logic on the entire dataset. Kraus and Ertl [KE02] introduced adaptive texture maps and applied them in volume rendering. Mora *et al.* [MJC02] proposed the use of hierarchical occlusion maps for hidden volume removal in an object-order volume rendering algorithm.

The elimination of occluded but non-transparent parts of the volume is particularly important in out of core techniques where costly disk-to-main-memory or main-memory-to-GPU-memory transfers need to be avoided. A recent state-of-the-art report [BHP14] provides a good overview of large-scale volume visualization techniques including those using visibility information. Most approaches employ a form of bricking where the volume is subdivided into smaller blocks [BHMF08, LLY06]. Gobbetti and Marton [GM05] introduced a hybrid space-partitioning scheme based on triangles (leaf nodes) and voxels (inner nodes). Their multi-resolution framework for interactive rendering of large and complex models employed visibility culling of the nodes. They coupled visibility culling, out-of-core construction and view-dependent rendering to achieve interactive frame rates. Later Gobbetti *et al.* [GMG08] presented an adaptive out-of-core method for rendering large scalar volumes, also exploiting visibility. Kniss *et al.* [KLF05] use the term deferred filtering to refer to a two-pass volume rendering approach for compressed formats. In the first pass, a local subset of the data is reconstructed and the second pass can then exploit the GPUs native interpolation capabilities. Crassin *et al.* [CNLE09] suggested the use of information extracted from the previous frame to guide data streaming from slower memory units. For the visualization of petascale volume data, Hadwiger *et al.* [HBJP12] presented a visibility-driven rendering method which employs a virtual memory architecture. They use visibility information to only fetch and reconstruct needed bricks from a multi-resolution hierarchy. Fogal *et al.* [FSK13] presented a detailed analysis of ray-guided approaches for large volume data visualization including an evaluation of the effects of common tunable parameters such as brick size. A common limitation of all these approaches, with respect to our application, is that the volume is considered to be static, i.e. its values do not change from one frame to the next.

Westenberg *et al.* [WRW07] used attribute filtering, which allow for the selective removal of connected components, and a max-tree data representation for fast interactive filtering of static volume data. Marton *et al.* [MGDG14] employed a deferred architecture for smoothing of sharp edges that may appear between blocks during block-based rendering of large volume data. Jeong *et al.* [JBH*09] perform on-demand filtering of electron microscopy data only for visible blocks. A local histogram-based edge metric, which is used for the enhancement of tissue boundaries, is recalculated during the filtering pass only for visible blocks. However, this approach is not feasible for streaming volume data, where the entire volume is replaced continuously. In this paper, we present a solution that specifically addresses this scenario. Furthermore, our approach also handles the case where invisible parts of the volume become visible after filtering.

Progressive rendering is often either based on image- or ray sampling distance, and on system responsiveness constraints. Frey *et al.* [FSME14] introduced a space-time error metric for progressive rendering where the spatial error describes the correctness of the pixel values while the temporal error describes the lag of the rendered image. Falk and Weiskopf [FW08] employed the principle of output sensitivity, i.e. the concept that the rendering cost should be primarily determined by the complexity of the output to the image plane, for the on-the-fly computation of 3D line integral convolution. Our approach is based on the same ideas, but we apply them to the filtering of time-dependent volume data.

Only few works have addressed the visualization of 4D streaming ultrasound data. The poster by Bruder *et al.* [BJE*11] shows how the Voreen volume rendering engine can be used to visualize real-time 4D ultrasound data and Elnokrashy *et al.* [EEH*09] present the basic pipeline setup for GPU-based ultrasound rendering, but neither work discusses the important issue of filtering. Bronstad *et al.* [BATK12] used visibility information to locally adapt the opacity transfer function for 4D ultrasound data. Most related to our approach is later work by Elnokrashy *et al.* [EHH*12], where they restrict themselves to single isosurfaces and apply a smoothing filter to the z-buffer. In contrast, our method enables true filtering of the scalar field in an integrated volume rendering pipeline for live streamed 4D ultrasound.

3. Output-Sensitive Volume Filtering

In the standard visualization pipeline, data enhancement (e.g. filtering) directly follows the acquisition stage. The visual mapping and rendering steps then operate on the enhanced data. The problem arises when *in situ* visualization of streamed data is required. In this case, for every rendered frame the data need to undergo an expensive processing operation which may be significantly more costly than the mapping/rendering stages themselves. Examples of such costly common data enhancement algorithms are iterative anisotropic diffusion filtering [PM87], iterative bilateral filtering [TM99], lowest-variance filtering [SSHW*12] and vessel-enhancing filtering [FNVV98].

The term *output-sensitive algorithm* refers to an algorithm whose running time is dominated by the size of its output. In our context, the output of our pipeline are visible color contributions to pixels on a screen, and we aim to exploit the fact that these contributions are not uniform across the voxels of the input volume to reduce the running time of the filtering stage. We observe that in most volume visualization applications, only a fraction of the data is displayed at a time. Transfer functions limit the range of displayed data values and large portions of the volume may be occluded by other structures. Additionally, particularly for ultrasound data, clipping planes or more advanced clipping geometries are commonly used to remove unwanted parts of the data [BBBV12]. Furthermore, when zooming in to investigate small details, large parts of the volume may simply lie outside of the viewing frustum.

Our output-sensitive filtering scheme utilizes the visibility and color-variation information in order to quantify the maximal possible contribution of a voxel to the image. Depending on a user-defined threshold, voxels with zero and small contribution to the image will be excluded from processing. An overview of our approach is depicted in Figure 4. Our pipeline receives one volume of the ultrasound stream at a time. This volume can then undergo multiple sequential processing operations and the result of these operations is then displayed by a volume rendering algorithm. Our strategy to enable on-the-fly processing of live streamed volume data is to prioritize processing of those regions that affect the final displayed image the most, while not exceeding a predefined maximal permissible color error per pixel.

In order to determine the final impact at every position in the volume, we would have to perform the full processing and volume

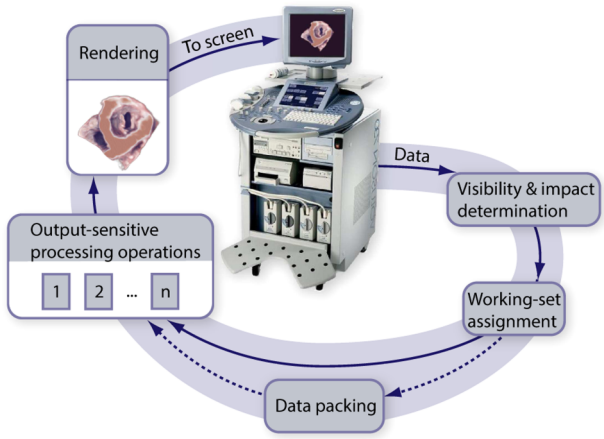


Figure 4: *The pipeline for output-sensitive data filtering: 4D data are streamed directly from the ultrasound scanner. In the first stage, a subset of voxels is tagged for processing based on their impact on the visualization. The neighborhood information is then evaluated and the set of tagged voxels is passed to the next stage. If the data do not fit into the GPU memory, we can optimize memory consumption by performing a visibility-driven data packing before processing (filtering) the data [SBVB14]. Finally, the processed data are rendered.*

rendering steps. However, by making certain assumptions about the nature of the processing operation to be performed (see Section 3.1), we can develop methods to conservatively predict the impact of voxels on the visualization prior to the application of the filtering operation. In Section 3.2, we explain the concept of the impact of a voxel on visualization as the potential error ϵ that would be introduced if this voxel would not be filtered. We also provide a conservative estimate for this error that can be quickly evaluated using a set of simple lookup tables, described in Section 3.3. Then we apply a greedy approach, described in Section 3.4, that aims at maximizing the set of voxels to be excluded from processing, so that the sum of the error ϵ over all voxels that contribute to a pixel is below a user-set threshold. Lastly, in Section 3.5 we discuss how the final working set of voxels is determined and filtered.

3.1. Basic assumptions

To predict the visual impact of a filtering operation, we need to make certain assumptions about its nature. Our input volume is a scalar-valued volumetric function $f : \mathbb{R}^3 \to \mathbb{R}$. In general, a filtering operation $g(p)$ replaces the original value at a voxel position p with a new value. For any such operation, we require that the filtered function value $g(p)$

1. is only dependent on values of f in a finite neighborhood Ω_p around a position p, i.e. that it has compact support, and
2. that $g(p) \in [\min \Omega_p, \max \Omega_p]$, i.e. the new value falls within the minimum and maximum values in that neighborhood.

Both requirements are trivially true for any convolution with normalized non-negative weights (mean, truncated Gaussian, etc.), but also hold for a wide range of other smoothing or denoising operations including nonlinear filters such as the median or

bilateral filters. In the case of an edge-detector that contains negative values in the operator mask, and that does not conform with this requirement, we can amend the estimated range of $g(p) \in [a_1, a_2]$ to the following:

$$a_1 = \left(\sum_{w_i<0} w_i\right) \max \Omega_p + \left(\sum_{w_i \geq 0} w_i\right) \min \Omega_p, \quad (1)$$

$$a_2 = \left(\sum_{w_i<0} w_i\right) \min \Omega_p + \left(\sum_{w_i \geq 0} w_i\right) \max \Omega_p, \quad (2)$$

where $\sum_{w_i<0} w_i$ is the sum of all negative weights $w_i < 0$ of the filter kernel and $\sum_{w_i \geq 0} w_i$ is the sum of all non-negative weights $w_i \geq 0$. Without the above requirements a processing operation could, in principle, be a random number generator and there would be no way to predict the resulting impact on the final image.

3.2. Potential impact of voxels

The idea behind output-sensitive processing is to maximize the set of voxels excluded from processing to decrease the computational workload. The impact of a voxel at position p to the final image is equivalent to the maximal potential color error per pixel $\epsilon(p)$ that would be induced by skipping the processing of this voxel. For each pixel, the accumulated error ϵ of voxels that contribute to that pixel should stay below a user-specified threshold T:

$$\int_R \epsilon(\tilde{s}) d\tilde{s} \leq T, \quad (3)$$

where R is the set of voxels that contribute to a pixel. If $T = 0$ then only voxels that have no contribution to the pixel can be skipped (because they are completely transparent or fully occluded). If $T = 1$, then any change is permissible, i.e. a pixel can change from black to white or vice versa. Our goal is to minimize the set of voxels that will be processed and at the same time conform to this condition. We start the derivation of $\epsilon(p)$ from the emission-absorption volume rendering integral, which defines the light transport from the source at s_0 to the viewpoint s_v [EHK*06]:

$$I(s) = I(s_0) e^{-\tau(s_0, s_v)} + \int_{s_0}^{s_v} q(s) e^{-\tau(s, s_v)} ds. \quad (4)$$

The term $\tau(a, b)$ denotes the extinction between points a and b, and $e^{-\tau(a,b)}$ is the respective transmittance factor. The term $q(a)$ is the emission factor at point a. For simplicity, the following explanation will assume a background intensity of $I(s_0) = 0$. We also use short names for the emission and extinction in the original volume (q_f and τ_f, respectively), and in the filtered volume (q_g and τ_g). The set of voxels that contribute to a pixel is referred to as R, and P is the set of processed voxels. Hence, the set of skipped voxels is $R \setminus P$. We can then write the accumulated error per pixel as:

$$\int_R \epsilon(s) ds = \int_{R \setminus P} \mid q_g(s) e^{-\tau_g(s, s_v)} - q_f(s) e^{-\tau_f(s, s_v)} \mid ds, \quad (5)$$

i.e. the absolute difference between the filtered and unfiltered contributions to the pixel for all skipped voxels. Evaluating $q_g(s)$ and $e^{-\tau_g(s, s_v)}$ would defy our goal of skipping processing operations. Therefore, we instead estimate the upper bound of the error. The estimation is based on the prediction of the maximal possible transmittance between s and a viewpoint s_v in the filtered volume, i.e. $e^{-\tau_{\min}(s, s_v)}$, and a prediction of the maximal emission difference $\Delta q_{\max}(s)$ between the original and filtered volumes:

$$\int_R \epsilon(s) ds \leq \int_{R \setminus P} 2 \Delta q_{\max}(s) \left(e^{-\tau_{\min}(s, s_v)}\right) ds. \quad (6)$$

The proof of the inequality in Equation (6) is discussed in Appendix A. To arrive at a discrete representation of the error term, we use the standard numerical approximation of the transmittance term in the volume rendering integral [EHK*06]:

$$\begin{aligned} e^{-\tau(a,b)} &\approx \prod_{i=0}^{N} \left(e^{-\kappa(i \Delta s)}\right)^{\Delta s} = \\ &= \prod_{i=0}^{N} (1 - \alpha_i)^{\Delta s} = \\ &= 1 - A[0, N], \end{aligned} \quad (7)$$

where $N = (b - a)/\Delta s$, Δs is the sampling distance, κ is the absorption function and α is the opacity. We use the notation $A[a, b]$ for the accumulated opacity between two points a and b along a viewing ray which can be recursively defined as:

$$\begin{aligned} A[0, 0] &= \alpha'_0 \\ A[0, i] &= A[0, i-1] + (1 - A[0, i-1]) \alpha'_i, \end{aligned} \quad (8)$$

where $\alpha'_i = 1 - (1 - \alpha_i)^{\Delta s}$ is the corrected opacity according to the sampling distance Δs. For readability, we will use $V[i] = 1 - A[0, i]$ to denote the visibility of a point from the viewpoint. $V[i] = 0$ means that this point is invisible. We can then write our error term as:

$$\epsilon[p] \leq 2 \Delta C_{\max}[p] V_{\max}[p], \quad (9)$$

where $\Delta C_{\max}$ is the maximal possible color difference, which is the discrete approximation of the maximal difference in emission $\Delta q_{\max}$ and V_{max} denotes the denotes maximum visibility after filtering. Our approach for obtaining $\Delta C_{\max}$ and V_{max} is detailed in the subsequent sections.

3.3. Tables of potential impact

Our error determination method requires the minimal and maximal opacity and a maximal color change that a voxel can obtain after filtering. Based on our assumptions, we know that the voxel value after processing will be $g(p) \in [\min \Omega_p, \max \Omega_p]$ for the neighborhood Ω_p of a voxel at position p. Unlike other approaches, which do not consider dynamically changing data, we cannot rely on any preprocessing. Hence, in order to obtain information about the voxel neighborhood, it needs to be recomputed for every new volume. We

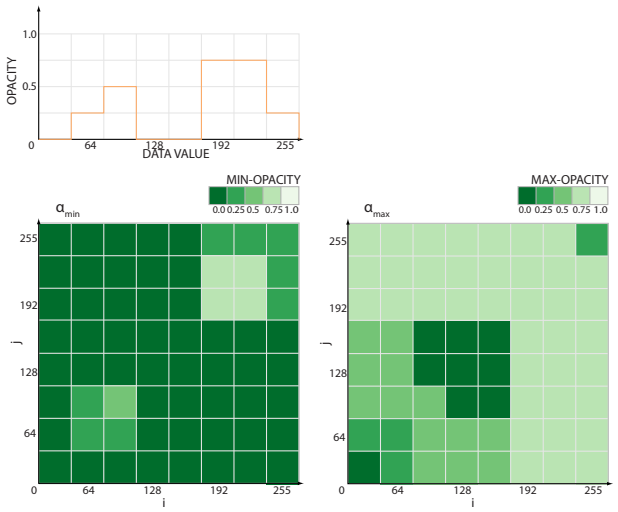


Figure 5: *An opacity transfer function and its corresponding* $\alpha_{\min}$ *and* $\alpha_{\max}$ *tables. For example,* $\alpha_{\min}(64, 192) = 0$*, because 0 is the minimal value of the opacity transfer function* TF_α *in the range* [64, 192]. $\alpha_{\max}(64, 192) = 0.75$*, because 0.75 is the maximum of* TF_α *in the range* [64, 192].

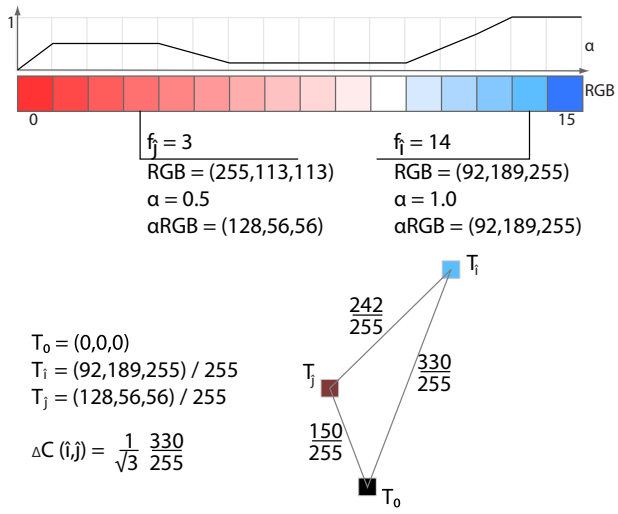


Figure 6: *Explanation of the calculation of* $\Delta C(\hat{\imath}\,\hat{\jmath})$*, the color difference between colors* $TF_{RGB\alpha}(f_{\hat{\imath}})$ *and* $TF_{RGB\alpha}(f_{\hat{\jmath}})$ *on an example of a simplified color-opacity function* $TF_{RGB\alpha}$*.* $\Delta C(\hat{\imath}\,\hat{\jmath})$ *is the length of the longest edge of the triangle* $T_0 T_{\hat{\imath}} T_{\hat{\jmath}}$ *normalized by* $\sqrt{3}$*.*

compute, for each position p in the volume, the minimum $\min \Omega_p$, and maximum value $\max \Omega_p$ with a given neighborhood Ω_p determined by the support of the processing operation (i.e. the size of the kernel for convolution operations). We use an OpenCL kernel in a multi-pass approach which exploits the fact that the min and max filters are separable. While this is not a cheap operation, it is still significantly less costly than the types of filtering operations we want to support.

To quickly determine the relevant opacity contributions for a voxel with a particular neighborhood, we precompute two 2D look-up tables $\alpha_{\min}$ and $\alpha_{\max}$ for all combinations of possible data values f:

$$\alpha_{\min}(i, j) = \min_{f \in [i,j]} \mathrm{TF}_\alpha(f), \tag{10}$$

$$\alpha_{\max}(i, j) = \max_{f \in [i,j]} \mathrm{TF}_\alpha(f), \tag{11}$$

where i, j are in the value range of the volumetric dataset. For example, $i, j \in [0, 255]$ for a dataset with 8 bits per voxel. α_{min} and α_{max} are the minimum and maximum opacity values in the opacity transfer function $\mathrm{TF}_\alpha(f)$ for all values in the interval $f \in [i, j]$. Consequently, the opacity of a processed voxel will be in the range $[\alpha_{\min}(\min \Omega_p, \max \Omega_p), \alpha_{\max}(\min \Omega_p, \max \Omega_p)]$. An example of an opacity transfer function and its corresponding $\alpha_{\min}$ and $\alpha_{\max}$ is shown in Figure 5.

The quantification of the possible color change of a voxel is done in a similar fashion. We generate a third lookup table $\Delta \mathrm{C}_{\max}(i, j)$ where i and j correspond to data values in the volume, and the elements store the maximum color difference $C(\hat{\imath}, \hat{\jmath})$ between any two colors $\mathrm{TF}_{\mathrm{RGB}\alpha}(f_{\hat{\imath}})$ and $\mathrm{TF}_{\mathrm{RGB}\alpha}(f_{\hat{\jmath}})$:

$$\Delta C_{\max}(i, j) = \max_{\hat{\imath} \in [i,j], \hat{\jmath} \in [i,j]} \Delta C(\hat{\imath}, \hat{\jmath}). \tag{12}$$

The calculation of $\Delta C(\hat{\imath}, \hat{\jmath})$ is explained in Figure 6. We define a triangle $T_0 T_{\hat{\imath}} T_{\hat{\jmath}}$ embedded in 3D. Its vertices are defined as the black color and opacity-premultiplied normalized RGB colors $\mathrm{TF}_{\mathrm{RGB}\alpha}(f_{\hat{\imath}})$ and $\mathrm{TF}_{\mathrm{RGB}\alpha}(f_{\hat{\jmath}})$:

$$\begin{aligned} T_0 &= (0, 0, 0), \\ T_{\hat{\imath}} &= (\alpha_{\hat{\imath}} R_{\hat{\imath}}, \alpha_{\hat{\imath}} G_{\hat{\imath}}, \alpha_{\hat{\imath}} B_{\hat{\imath}}), \\ T_{\hat{\jmath}} &= (\alpha_{\hat{\jmath}} R_{\hat{\jmath}}, \alpha_{\hat{\jmath}} G_{\hat{\jmath}}, \alpha_{\hat{\jmath}} B_{\hat{\jmath}}). \end{aligned}$$

$\Delta C(\hat{\imath}, \hat{\jmath})$ is then the length of the longest edge of this triangle, normalized by a factor of $\sqrt{3}$. The normalization factor corresponds to the distance between the black and the white color in the RGB color space. The length of an edge is defined as the Euclidean norm $|.|$ of a vector in 3D space. In the proof that is given in Appendix A, we also use the fact that $\Delta C(\hat{\imath}, \hat{\jmath}) = \max(|T_{\hat{\imath}}|, |T_{\hat{\jmath}}|, |T_{\hat{\imath}} - T_{\hat{\jmath}}|)$.

All three look-up tables ($\alpha_{\min}$, $\alpha_{\max}$, and $\Delta \mathrm{C}_{\max}$) can be computed simultaneously. The computation is straightforward and only consumes a negligible amount of time when the transfer function is modified. This is conceptually similar to pre-integrated volume rendering which employs a lookup table for accumulated colors and opacities along a viewing ray [EKE01]. The tables can be stored as three channels of a single 2D texture. Then, during the determination of $R \setminus P$, values of $\alpha_{\min}[p]$, $\alpha_{\max}[p]$ and $\Delta \mathrm{C}_{\max}[p]$ can be quickly determined for a voxel at position p via a single table-lookup with $\min \Omega_p$ and $\max \Omega_p$. The value of $\alpha_{\min}[p]$ can then be directly used in the accumulation of the maximum visibility $V_{\max}[p] = 1 - A_{\min}[0, p]$, as detailed in the next section. The value of $\alpha_{\max}[p]$ gives us information whether the voxel is completely transparent after filtering, i.e. $\alpha_{\max}[p] = 0$, and hence can be skipped without affecting the per-pixel error.

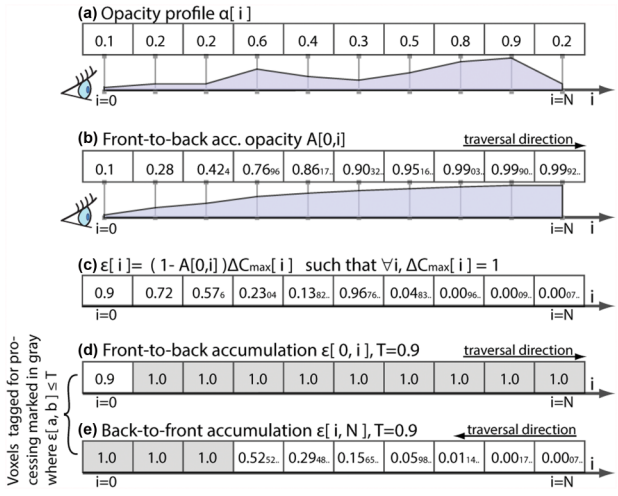


Figure 7: *(a) Opacity profile of an example of a voxel ray α_i and (b) the corresponding front-to-back accumulated opacity $A[0, i]$. (c) The corresponding error factor $\epsilon[i]$. (d) Front-to-back accumulation and (e) back-to-front accumulation of ϵ their corresponding working set of voxels marked in grey. To explain the situation on a simple example, we assumed here that $\forall i,\ \Delta C_{\max}[i] = 1$ and unaltered visibility after processing. The error is then defined as $\epsilon[i] = 1 - A[0, i]$.*

3.4. Greedy accumulation

Being able to evaluate the minimum and maximum opacity and the maximum color difference for any voxel, we can now proceed to identify a set of voxels that complies with Equation (3). As our aim is to reduce overall processing time, we use a greedy approach inspired by object-order volume rendering algorithms such as shear-warp factorization [LL94]. We traverse the volume in a slice-by-slice manner along the principle axis that is most aligned with the viewing direction and store per-ray information in a view-aligned intermediate image. Similar approaches have also been used in the precalculation of illumination information [RDR10].

For the computation of visibility, it would be natural to perform a front-to-back pass through the volume, i.e. starting with the slice that is closest to the viewpoint. However, we would like to accumulate the error ϵ during the same pass. In the example in Figure 7, we see that the front-to-back accumulation of ϵ reaches the threshold T significantly faster than back-to-front accumulation, i.e. $i_1 \ll N - i_2$, where i_1 and i_2 are defined as:

$$i_1 = \max\left\{ i \in \mathbb{R} \;\middle|\; \epsilon[0, i] = \sum_{i=0}^{i_1} \epsilon[i]\Delta s \le T \right\}$$
$$i_2 = \min\left\{ i \in \mathbb{R} \;\middle|\; \epsilon[i, N] = \sum_{i=i_2}^{N} \epsilon[i]\Delta s \le T \right\}. \quad (13)$$

From Equation (4), we see that with decreasing transmittance, we have decreasing visibility. If we disregard the factor $\Delta C_{\max}$ (Equation (9)) as in Figure 7, we observe that smaller errors are expected farther away from the viewpoint. In this respect, it pays off to accumulate errors back-to-front—in this way, a significantly smaller portion of voxels will be tagged for processing.

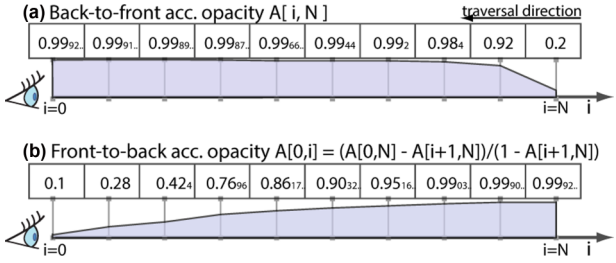


Figure 8: *Calculation of front-to-back accumulated opacity for a ray with an opacity profile same as in Figure 7(a). (a) Corresponding back-to-front accumulated opacities $A[i, N]$. (b) Using Equation 14, we obtain the same results as in Figure 7(b), but with a parallelized transformation and not with a new sequential traversal. The last accumulated value from back-to-front accumulation $A[0, N] = 0.9992..$ is used together with the results of the back-to-front traversal to obtain front-to-back accumulated opacities $A[0, i]$. The supplementary material also contains MATLAB code for this example.*

Since $V_{\max}$ depends on the front-to-back accumulated opacities, the challenge is to combine the computation of these terms and the back-to-front accumulation of ϵ in a single pass through the volume. Our solution is to express the front-to-back accumulated opacity $A[0, i]$ as a function of the back-to-front accumulated opacity $A[i + 1, N]$ and the total accumulated opacity along the ray $A[0, N]$:

$$A[0, N] = A[0, i] + (1 - A[0, i])A[i + 1, N]$$
$$A[0, i] = \frac{A[0, N] - A[i + 1, N]}{1 - A[i + 1, N]}, \quad (14)$$

where $A[i + 1, N]$ is recursively defined using corrected opacities α'_i as:

$$A[i + 1, N] = \alpha'_i + (1 - \alpha'_i)A[i + 2, N]. \quad (15)$$

We illustrate this procedure in Figure 8. A problem of division by zero will occur when $A[i + 1, N] = 1$. This happens when the ray hits a sample with $\alpha'_i = 1$ or due to numerical imprecision. We refer to Appendix B and the MATLAB-script in the supplementary material for details on how the division by zero can be circumvented. Using Equations (9) and 14, ϵ can be written as:

$$\epsilon[i] = 2\Delta C_{\max}[i]\left(\frac{1 - A_{\min}[0, N]}{1 - A_{\min}[i + 1, N]}\right). \quad (16)$$

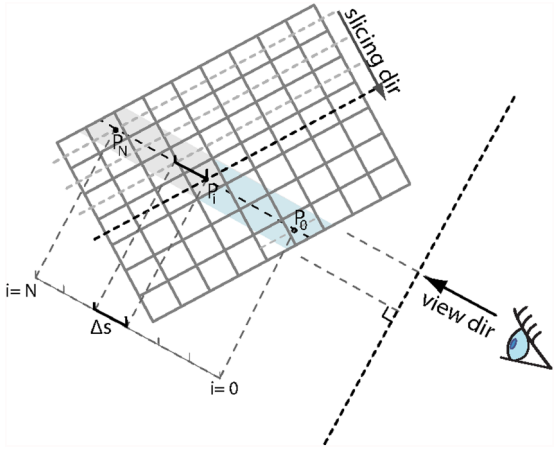


Figure 9: *Back-to-front accumulation of ρ and $A_{\min}$ employs the object-order rendering concept via axis-aligned slicing and ping-pong accumulation buffers. The slicing axis is chosen to be closest to the view direction. The sampling distance Δs depends on the view direction and the voxel size. We store $A_{\min}$ and ρ in an intermediate volume. During the working-set assignment, ρ is multiplied by the factor $(1 - A_{\min}[0, N])$. Both are retrieved from the intermediate volume. For a voxel at position p_i, ρ is fetched from same position p_i, but $A_{\min}$ from position p_0, because it represents $A_{\min}[p_0, p_N]$.*

As the term $A_{\min}[0, N]$ is only known once the back-to-front pass through the volume has been completed, we can instead express this as:

$$\epsilon[i, N] = (1 - A_{\min}[0, N]) \overbrace{\sum_{j=i}^{N} \frac{2\Delta \mathrm{C}_{\max}[j]\Delta s}{1 - A_{\min}[j+1, N]}}^{\rho}. \quad (17)$$

We illustrate the accumulation in Figure 9: ρ and $A_{\min}[i, N]$ can be computed recursively on-the-fly in a single back-to-front pass, because they only depend on data already traversed (the grey corridor). Both ρ and $A_{\min}[i, N]$ are stored in an intermediate volume. Since the accumulated error term depends also on positions which were not traversed yet (the blue corridor), we multiply ρ with a correction term $(1 - A_{\min}[0, N])$ in the next step, i.e. during the working-set assignment. $A_{\min}[0, N]$ is fetched from the intermediate volume, as explained in Figure 9. The greedy-accumulation algorithm is written in pseudo-code in Appendix C.

3.5. Working set assignment and filtering

While the $\epsilon[i, N]$ gives us information about which voxel has sufficiently small summed impact on the final image, we also need to make sure that the results of the processing operation are correct. Many data enhancement techniques require information about the neighborhood and many approaches are carried out in an iterative fashion, e.g. when filters are separable. If we rely only on processing set M defined by $\epsilon[i, N] > T$, iterations can obtain incorrect information from the neighborhood in later passes. This problem is sketched in Figure 10.

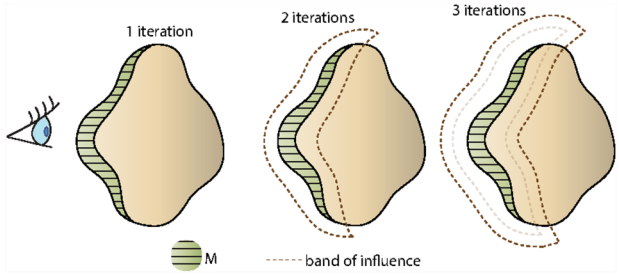


Figure 10: *If data enhancement consists of one operation only, expansion of the original mask (M) is not necessary. If it runs in two iterations, values within the band of influence (BOI) must have been processed in the 1st iteration since the 2nd iteration relies on its results. If the enhancement runs in three iterations, the radius of the BOI is larger accordingly.*

Our solution to this problem is to dilate the binary mask M defined by $\epsilon[i, N] > T$ to obtain a working set of voxels (WSV). Then we define a band of influence (BOI) as WSV$\backslash M$ and BOI $\cap$ $M = \emptyset$. For example, if a filtering operation has m iterations and in each iteration, the operation requires a neighborhood of radius n, then the visibility mask must be dilated by a radius of $m \cdot n$. Using the WSV for filtering calculations will ensure that the values of all voxels $\in$ M will be correct after the final iteration of the filter. In Figure 11, we compare WSVs for different thresholds and different transfer functions of a synthetic dataset (a box composed of a single scalar value). The WSVs were rotated to show that with increasing threshold, the WSV shrinks primarily from behind.

4. Results

Our system was implemented in C++ using OpenCL for the realization of our processing pipeline. The volume rendering itself is performed in OpenGL. Our approach is not tied to any specific volume rendering algorithm, and in fact the current implementation of our pipeline flexibly integrates several different renderers including standard GPU-based ray casting and slice-based multidirectional occlusion volume rendering [SPBV10]. The benchmarks in this paper were performed on a PC with a 3.07 GHz CPU with an NVIDIA Quadro K6000 GPU with 12 GB of dedicated graphics memory running Windows 7, except if explicitly mentioned otherwise.

In Figure 12, we compare the performance of different output-sensitivity thresholds. As can be seen, the number of voxels in the WSV indeed decreases with growing threshold and consequently, the runtime of the filtering operations decreases as well. The most significant improvement of performance compared to processing the entire volume was achieved by excluding voxels with zero impact from processing (T=0). In the case of the cardiac and gallbladder ultrasound in Figure 12, the computation time decreased by 35% (from 30.7 ms to 20.0 ms) and 55% (from 277 ms to 126 ms), respectively. By increasing the sensitivity threshold, we can further improve the performance while still maintaining the main desired properties of the filtering operation. Increasing the sensitivity threshold has a more pronounced effect on highly translucent structures,

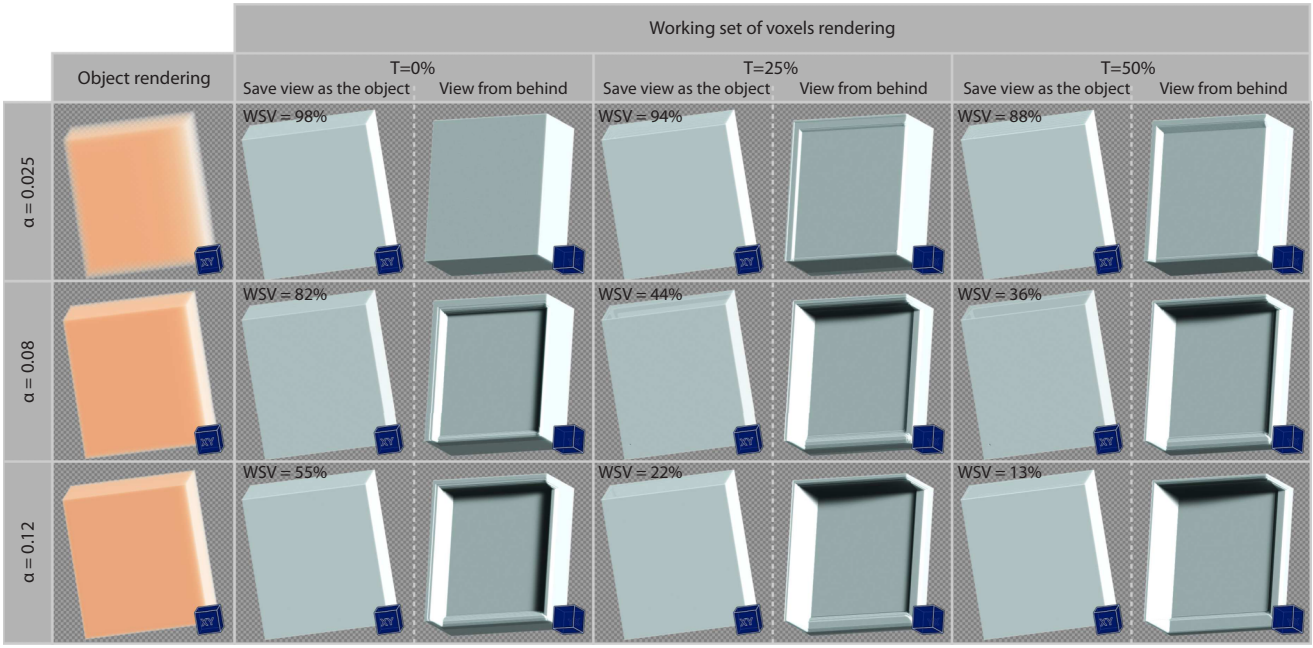


Figure 11: *Comparison of the working set of voxels (WSV) for different thresholds and different transfer function setup. BOI has radius 5 voxels. We used a synthetic dataset—a box composed of a unique scalar value. The transfer function is therefore only one color and opacity* $\alpha = 0.025$, 0.08 *and* 0.12. *Corresponding WSVs (binary masks) are compared for different thresholds* $T = 0\%$, 25% *and* 50% *and shown from the view as the original object and from the back. We also give how the relative number of voxels that are contained in WSV as compared to the original volume.*

e.g. the surrounding tissue of the gallbladder, than for more opaque objects such as in the cardiac ultrasound example.

The performance gain is linearly dependent on the percentage of voxels in the WSV relative to the entire volume. The dependency is also influenced by the size of the dataset and the type of the filter. In Figure 13, we profile the performance of the lowest-variance filter [SSHW*12] as a function of the relative size of the WSV for ultrasound datasets with different sizes. This measurement was conducted on a NVIDIA GTX680 GPU with 2 GB dedicated memory. The horizontal lines show the constant time that is needed to process the full volume. Only if more than 80% of the data is visible, which is very uncommon for real-world ultrasound data, our output-sensitive approach would no longer pay off. The gallbladder dataset was intentionally upsampled to a grid of 256^3 to show the performance on larger datasets, even though ultrasound datasets rarely have this size. In this case, the breaking point is around 60%. We also measured performance using other filters, e.g. Perona-Malik anisotropic diffusion and bilateral filtering, and they showed a very similar trend as lowest-variance filtering. The extra overhead of calculating the accumulated impact of voxels compared to calculating only visibility as described in our previous work [SBVB14] amounts to approximately 1%. Therefore, we can conclude that computation of the working set of voxels for output-sensitive processing does not pose a significant performance overhead in the entire pipeline as compared to visibility-driven processing.

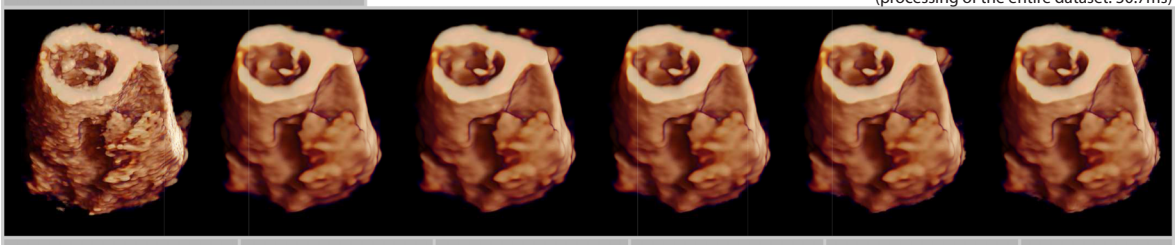
We investigated that our approach for culling invisible regions, i.e. output sensitivity threshold $= 0$, is correct in the sense that it cannot affect the final image. We showed that if we solve the problems of reduced occlusion and increased opacity, the visualizations will be identical by making difference images and by providing a formal proof [SBVB14]. We also calculated the image difference of Figure 1(b) (the ground truth) and 1(c) (T $= 25\%$). The maximal color difference was in fact 5.88%. For the difference between Figures 1(b) and 1(d) (T $= 90\%$), we calculated 10.85%. In addition, we verified the error in visualizations displayed in Figure 12 and plot it in Figure 14. In all cases, the measured error is much less than the maximal permissible error, so our estimator is very conservative.

On modern ultrasound scanners with a 4D cardiac probe, ultrasound volumes are acquired at rates of 10–15 volumes per second [PVMd12], depending on the spatial extent of the acquired volume (sector). The larger the sector, the bigger is the volume and the lower is the acquisition rate. According to our experience, larger sectors are acquired at approximately 15 volumes per second. Figure 12 shows that we allow for higher frame rates which is a relevant step toward *in situ* high-quality processing and visualization for current 3D ultrasound acquisition capabilities.

5. Discussion and Limitations

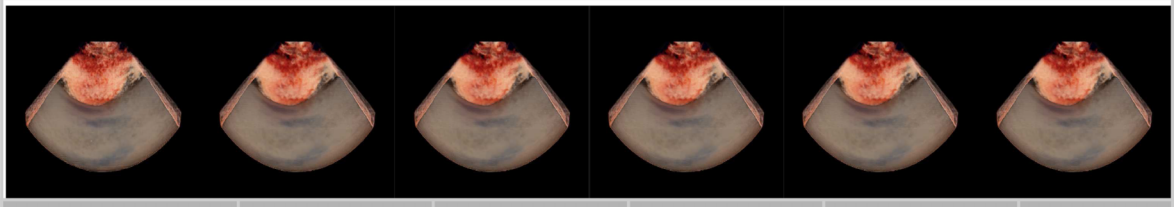
The presented approach was designed to support complex filtering operations for live streaming volume data. Clearly, for very simple filters (such as, trivially, the min and max filters themselves), the overhead of minimum/maximum computation does not pay off, but in those cases filtering the whole volume is easily possible in real

Cardiac ultrasound (128 x 100 x 128) (processing of the entire dataset: 30.7ms)



(a) no filtering	(b) T = 0%	(c) T = 20%	(d) T = 40%	(e) T = 60%	(f) T = 80%
Amount of processed voxels:	14.4%	11.7%	11.0%	10.4%	9.6%
Runtime of processing:	5.4ms	5.1ms	5.0ms	4.8ms	4.8ms
Total time incl. overhead:	20.0ms	19.7ms	20.5ms	19.3ms	19.2ms

Gallbladder ultrasound (256x 256 x 256) (processing of the entire dataset: 277ms)



(g) no filtering	(h) T = 0%	(i) T = 20%	(j) T = 40%	(k) T = 60%	(l) T = 80%
Amount of processed voxels:	36.9%	35.9%	34.8%	34.0%	33.0%
Runtime of processing:	109.7ms	106.9ms	103.9ms	101.9ms	99.5ms
Total time incl. overhead:	126ms	123ms	121ms	118ms	116ms

Figure 12: *Demonstration of filtering runtime and the relative size of the WSV compared to the volume size on two datasets: cardiac ultrasound (upper row) and gallbladder ultrasound (bottom row). In (a) and (g), we show snapshots of unprocessed streams. For the rest, the data are processing using the bilateral filter (heart) and lowest-variance filtering (gallbladder) of kernel size 7. We compare performance for different output-sensitivity thresholds: 0%, 20%, 40%, 60%, 80%. We show the time that was needed for processing the WSV and also the total time that includes the overhead, i.e. generation of the WSV.*

time. Our goal was to enable the application of filters that normally would be too expensive in such a scenario.

Furthermore, naturally our approach becomes less effective if a high percentage of the voxels is tagged for processing. In this case, processing of the full volume might even be faster. However, if we detect a high percentage of tagged voxels in one frame (e.g. due to a highly transparent transfer function and a low sensitivity threshold), we can simply disable our pipeline until the transfer function is modified again. This means that in most practical scenarios, the overall performance will always be as fast as processing of the entire volume.

Currently, there are some limitations in our implementation which we aim to address in the future:

- We start from an already reconstructed regular grid which we receive from the ultrasound scanner's software. However, it would also be possible to integrate our pipeline earlier to potentially eliminate or at least reduce some of the overhead. For instance, it would be easy to compute the minimum/maximum information during the resampling step from the beam space (polar grid) to the regular grid.
- Using minimum/maximum neighborhood information to determine the set of potentially visible voxels is general enough to enable a wide range of practically useful filters. However, it is also quite conservative and can, in some cases, lead to considerable overestimation. For this, a closer analysis of the mathematical properties of individual filters may enable us to determine tighter bounds for special cases.
- We quantify the color differences in the RGB color space, even though it may be more appropriate to use a perceptually uniform color space, such as CIELAB. However, our upper bound depends on the linearity of the RGB color interpolation. To provide a correct maximal distance in CIELAB space, we would have to transform the triangle $T_0 T_{\hat{i}} T_{\hat{j}}$ to CIELAB space first. The result would be a possibly concave set, for which we would have to determine the maximal distance of two points that belong to it—we consider this an interesting challenge for future work.

Despite these limitations we have shown that our approach already achieves a considerable reduction in processing times for

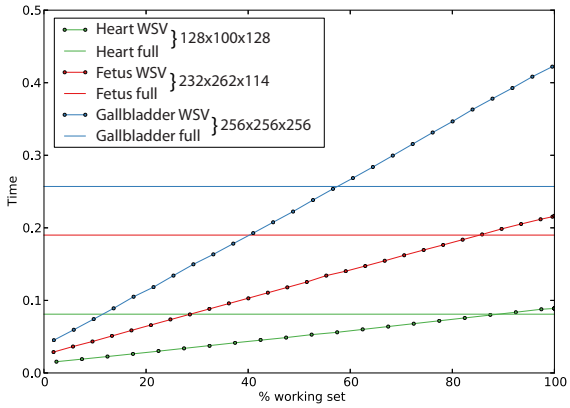


Figure 13: *Performance boost of visibility-driven processing compared to full processing with the lowest-variance filtering with radius 5 on a stream of 3D cardiac ultrasound datasets—a snapshot of which is shown in Figure 12, streamed 3D ultrasound of a foetus shown in Figure 1 and streamed 3D ultrasound of a gallbladder as shown in Figure 12.*

realistic percentages of visible voxels and hence enables real-time filtering and, consequently, high-quality visualization of streaming volume data in cases where this was previously impossible. An interesting direction for future research is to investigate extended schemes for adjusting the processing time. Currently, the decision whether to apply a filtering operation or not is still binary. However, we believe that a more fine-grained approach where the filter quality could be adjusted based on the expected visual impact could also prove useful.

6. Conclusions

In this paper, we presented a novel approach for integrated filtering and visualization of streaming volume data. Our method conservatively estimates visibility information to limit processing operations to regions in the volume which can potentially contribute to the final image and hence can result in a considerable reduction of the processing load. We have demonstrated that the resulting pipeline is an important step toward high-performance integrated filtering and visualization of steaming volume data such as 4D ultrasound.

Acknowledgements

This work has been supported by the MedViz lighthouse project IllustraSound and the ISADAF project (In-Situ Adaptive Filtering, #229352/O70) co-funded by the VERDIKT program of the Norwegian Research Council. The research was also partially supported by the Vienna Science and Technology Fund (WWTF) project VRG11-010 and by the EC Marie Curie Career Integration Grant project PCIG13-GA-2013-618680. Parts of this work have been made possible by the University of Bergen, the Bergen University Hospital, and Christian Michelsen Research AS through their strategic support of the MedViz program. The authors also thank GE Vingmed Ultrasound for the support and Matej Mlejnek for providing the dataset Anna. We also acknowledge the usage of Colorbrewer [BH15] for the color map in Figure 14.

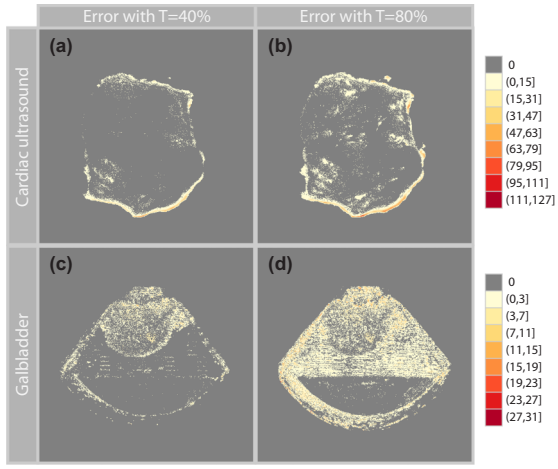


Figure 14: *Difference images for the results shown in Figure 12 scaled to the [0,255] range, where 255 corresponds to the maximal error = 1.0. (a) shows the error of visualization 12(d) as compared to the ground truth visualization 12(b). (b) shows the error in 12(f) as compared to 12(b). (c) shows the error of visualization 12(i) as compared to the ground truth visualization 12(h). (b) shows the error in 12(l) as compared to 12(h).*

Appendix A

Here we provide the proof of Equation (6). We first recall the literals: q_f and q_g represent the emission factors of the original and processed voxel, respectively. They are 3D vectors if we work with RGB emission. The scalar $e^{-\tau_f(s,s_v)}$ is the transmittance between the viewpoint s_v and s (visibility of point s) in the original volume. The scalar $e^{-\tau_g(s,s_v)}$ is the visibility of s in a fully processed volume. We will refer to these quantities as $e^{-\tau_f}$ and $e^{-\tau_g}$. $\Delta q_{\max}(s)$ is the maximal difference of the emission (color) at point s in the processed and original volume, which satisfies $\max(|q_f|, |q_g|, |q_g - q_f|) \leq \Delta q_{\max}$.

Proving Equation (6) is equivalent to proving:

$$|q_g e^{-\tau_g} - q_f e^{-\tau_f}| \leq 2e^{-\tau_{\min}} \Delta q_{\max}. \tag{A.1}$$

Using the triangle inequality, it follows

$$|q_g e^{-\tau_g} - q_f e^{-\tau_f}| \leq |q_g e^{-\tau_g}| + |q_f e^{-\tau_f}|. \tag{A.2}$$

Since $e^{-\tau_g} \leq e^{-\tau_{\min}}$ and $e^{-\tau_f} \leq e^{-\tau_{\min}}$:

$$|q_g e^{-\tau_g}| + |q_f e^{-\tau_f}| \leq |q_g\, e^{-\tau_{\min}}| + |q_f\, e^{-\tau_{\min}}|. \tag{A.3}$$

Using $q_F \leq \Delta q_{\max}$ and $q_g \leq \Delta q_{\max}$, we obtain

$$|q_g e^{-\tau_{\min}}| + |q_f e^{-\tau_{\min}}| \leq |\Delta q_{\max}\, e^{-\tau_{\min}}| + |\Delta q_{\max}\, e^{-\tau_{\min}}|, \tag{A.4}$$

$$= 2|\Delta q_{\max}\, e^{-\tau_{\min}}|. \tag{A.5}$$

This concludes the proof.

Appendix B

In Section 3.2, we introduced Equation 14 that expresses the front-to-back accumulated opacity $A[0, i]$ via the back-to-front accumulated opacity. It is possible that the denominator in this equation becomes zero. In this case, $A[0, i]$ cannot be computed using this equation. To circumvent this problem, we can in addition track slabs that have an accumulated opacity $A[i, j] \geq 1$. We accumulate A[i,N] normally until ≥ 1. Then we store $i_1 \leftarrow i$ and $i_2 \leftarrow N$, reset the accumulation buffer to zero and start accumulating again. After the traversal, when doing the correction of a voxel at position p, we first check if p is farther from the viewer than i_2. If yes, then $A[p_0, p_i] = 1$ and $V[p] = 0$ for this voxel. If p is further from the viewer than i_1, but closer than i_2, we must tag this voxel for processing, unless its corresponding $\alpha'_{\max} = 0$. If p is closer to the viewer than i_1, we proceed with the correction as described in Section 4.

In the supplementary material, there is an example of a back-to-front ray traversal, in which the front-to-back accumulated opacities are calculated. The example uses an arbitrary opacity profile, and is implemented as a Matlab script.

Appendix C

Here we give pseudocode for the greedy accumulation algorithm (Algorithm 1) and its integration into the output-sensitive filtering pipeline (Algorithm 2).

Algorithm 1: Greedy Error Accumulation

Data: Input volume F, view direction **v**, color/opacity transfer function, sampling distance Δs
Result: Volume Vol

a ← determineSlicingAxis(**v**)
x,y ← determineSliceSize(F.size, **a**)
Vol ←init3ChannelVolume(F.size)
Buff[2] ← init2ChannelSwappingBuffers(x,y)
foreach *slice i* **do**
  Buff_{src} ← getSrcBuffer(Buff)
  Buff_{dst} ← getDstBuffer(Buff)
  foreach *voxel v* ∈*slice i* **do**
    $f_{\min}, f_{\max}$ ← getMinMaxInNeighborhood(v)
    $\alpha_{\min}, \alpha_{\max}$ ← getMinMaxOpacity($f_{\min}, f_{\max}$)
    $\Delta C_{\max}$ ← getMaxColorVariation($f_{\min}, f_{\max}$)
    $\alpha'_{\min}, \alpha'_{\max}$ ← opacityCorrection($\alpha_{\min}, \alpha_{\max}$)
    $A_{\min}$ ← getPreviousA(Buff_{src})
    ρ ← getPrevPartialAccumErrors(Buff_{src})
    $\rho \leftarrow \rho + \frac{2\Delta s \Delta C_{\max}}{1-A_{\min}}$
    $A_{\min} \leftarrow \alpha'_{\min} + (1-\alpha'_{\min})A_{\min}$
    updateBuffer($\text{Buff}_{dst}, A_{\min}, \rho$)
    store(Vol, v.pos, $A_{\min}, \rho, \alpha'_{\max}$)
  end
  swapBuffers()
end

Algorithm 2: Output-Sensitive Filtering

Data: F, **v**, Vol, binary mask volume WSV
Result: Partially processed volume *G*

foreach *voxel v*∈F **do**
  $A_{\min}, \alpha'_{\max}, \rho$ ← getData(Vol, v.pos)
  p_0 ← findRayEntryPoint(F.size, v.pos, **v**)
  $A_{\min}$ ← getTotalAccumOpacity(Vol, p_0)
  $\epsilon_{acc} \leftarrow (1 - A_{\min})\rho$
  if $\epsilon_{acc} > T$ *and* $\alpha_{\max} > 0$ **then**
    tagAndDilate(WSV,v.pos)
  end
end
foreach *voxel v*∈F **do**
  if *v* ∈ *WSV* **then**
    g ← doFiltering(v)
    store(G,v.pos,g)
  end
  else
    store(G,v.pos,v)
  end
end

References

[BATK12] Bronstad E., Asen J., Torp H., Kiss G.: Visibility driven visualization of 3D cardiac ultrasound data on the GPU. In *IEEE International Ultrasonics Symposium (IUS)* (2012), pp. 2651–2654.

[BBBV12] Birkeland Å., Bruckner S., Brambilla A., Viola I.: Illustrative membrane clipping. *Computer Graphics Forum 31*, 3 (2012), 905–914.

[BH15] Brewer C. A., Harrower M.: Colorbrewer - a web tool for selecting colors for maps. http://colorbrewer2.org/, October 2015.

[BHMF08] Beyer J., Hadwiger M., Möller T., Fritz L.: Smooth mixed-resolution GPU volume rendering. In *Proceedings of Point-Based Graphics 2008* (2008), pp. 163–170.

[BHP14] Beyer J., Hadwiger M., Pfister H.: A Survey of GPU-Based Large-Scale Volume Visualization. In *EuroVis - STARs* (2014), pp. 105–123.

[BJE*11] Bruder R., Jauer P., Ernst F., Richter L., Schweikard A.: Real-time 4D ultrasound visualization with the voreen framework. In *Proceedings of ACM SIGGRAPH 2011 Posters* (2011), pp. 74:1–74:1.

[BNS01] Boada I., Navazo I., Scopigno R.: Multiresolution volume visualization with a texture-based octree. *The Visual Computer 17*, 3 (2001), 185–197.

[BW03] Bittner J., Wonka P.: Visibility in computer graphics. *Journal of Environment and Planning B: Planning and Design 5*, 30 (2003), 729–756.

[CNLE09] Crassin C., Neyret F., Lefebvre S., Eisemann E.: Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of Symposium on Interactive 3D Graphics and Games* (2009), ACM, pp. 15–22.

[COCSD03] Cohen-Or D., Chrysanthou Y. L., Silva C. T., Durand F.: A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (2003), 412–431.

[DWS*88] Deering M., Winner S., Schediwy B., Duffy C., Hunt N.: The triangle processor and normal vector shader: A vlsi system for high performance graphics. *ACM SIGGRAPH Computer Graphics 22*, 4 (1988), 21–30.

[EEH*09] Elnokrashy A., Elmalky A., Hosny T., Ellah M., Megawer A., Elsebai A., Youssef A.-B., Kadah Y.: GPU-based reconstruction and display for 4D ultrasound data. In *Proceedings of the IEEE International Ultrasonics Symposium 2009* (2009), pp. 189–192.

[EHH*12] Elnokrashy A., Hassan M., Hosny T., Ali A., Megawer A., Kadah Y.: Multipass GPU surface rendering in 4D ultrasound. In *Proceedings of the Cairo International Biomedical Engineering Conference 2012* (2012), pp. 39–43.

[EHK*06] Engel K., Hadwiger M., Kniss J., Rezk-Salama C., Weiskopf D.: *Real-Time Volume Graphics*. AK Peters, 2006.

[EKE01] Engel K., Kraus M., Ertl T.: High-quality pre-integrated volume rendering using hardware accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EG Workshop on Graphics Hardware 2001* (2001), pp. 9–16.

[FNVV98] Frangi A. F., Niessen W. J., Vincken K. L., Viergever M. A.: Multiscale vessel enhancement filtering. In *Proceedings of Medical Image Computing and Computer–Assisted Intervention* (1998), pp. 130–137.

[FSK13] Fogal T., Schiewe A., Krüger J.: An analysis of scalable GPU-based ray-guided volume rendering. In *IEEE Symposium on Large Data Analysis and Visualization* (2013), pp. 43–51.

[FSME14] Frey S., Sadlo F., Ma K.-L., Ertl T.: Interactive progressive visualization with space-time error control. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2397–2406.

[FW08] Falk M., Weiskopf D.: Output-sensitive 3d line integral convolution. *IEEE Transactions of Visualization and Computer Graphics 14*, 4 (2008), 820–834.

[GM05] Gobbetti E., Marton F.: Far Voxels – a multiresolution framework for interactive rendering of huge complex 3D models on commodity graphics platforms. *ACM Transactions on Graphics 24*, 3 (2005), 878–885.

[GMG08] Gobbetti E., Marton F., Guitián J. A. I.: A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets. *The Visual Computer 24*, 7–9 (2008), 797–806.

[HBJP12] Hadwiger M., Beyer J., Jeong W.-K., Pfister H.: Interactive volume exploration of petascale microscopy data streams using a visualization-driven virtual memory approach. *IEEE Transactions of Visualization and Computer Graphics 18*, 2 (2012), 2285–2294.

[HSBG05] Hadwiger M., Scharsach H., Bühler K., Gross M.: Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum 24*, 3 (2005), 303–312.

[Ion10] Ionescu C.: The benefits of 3D-4D fetal echocardiography. *Maedica (Buchar) 5*, 1 (2010), 45–50.

[JBH*09] Jeong W.-K., Beyer J., Hadwiger M., Vazquez A., Pfister H., Whitaker R. T.: Scalable and interactive segmentation and visualization of neural processes in EM datasets. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1505–1514.

[KE02] Kraus M., Ertl T.: Adaptive texture maps. In *Proceedings of ACM SIGGRAPH/EG Conference on Graphics Hardware* (2002), pp. 7–15.

[KLF05] Kniss J., Lefohn A., Fout N.: *Deferred Filtering: Rendering from Difficult Data Formats*. Addison Wesley, 2005, ch. 41, pp. 669–677.

[Lev90] Levoy M.: Efficient ray tracing of volume data. *ACM Transactions on Graphics 9*, 3 (1990), 245–261.

[LL94] Lacroute P., Levoy M.: Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of ACM SIGGRAPH* (1994), pp. 451–458.

[LLY06] Ljung P., Lundström C., Ynnerman A.: Multiresolution interblock interpolation in direct volume rendering. In *Proceedings of EuroVis 2006* (2006), pp. 259–266.

[LV11] Lebit F.-D., Vladareanu R.: The role of 4D ultrasound in the assessment of fetal behaviour. *Maedica (Buchar) 6*, 2 (2011), 120–127.

[MGDG14] Marton F., Guitián J. A. I., Díaz J., Gobbetti E.: Real-time deblocked GPU rendering of compressed volumes. In *Proceedings of Vision, Modeling and Visualization* (2014), pp. 167–174.

[MJC02] Mora B., Jessel J.-P., Caubet R.: A new object-order ray-casting algorithm. In *Proceedings of IEEE Visualization 2002* (2002), pp. 203–210.

[NPH*00] Nelson T., Pretorius D., Hull A., Riccabona M., Sklansky M., James G.: Sources and impact of artifacts on clinical three-dimensional ultrasound imaging. *Ultrasound in Obstetrics & Gynecology 16*, 4 (2000), 374–383.

[PM87] PERONA P., MALIK J.: Scale-space and edge detection using anisotropic diffusion. In *Proceedings of IEEE Computer Society Workshop on Computer Vision* (1987), pp. 16–22.

[PVMd12] PERRIN D.P., VASILYEV N.V., MARX G.R., DEL NIDO P.J.: Temporal enhancement of 3D echocardiography by frame reordering. *JACC Cardiovascular Imaging 5*, 3 (2012), 300–304.

[RDR10] ROPINSKI T., DÖRING C., REZK-SALAMA C.: Interactive volumetric lighting simulating scattering and shadowing. In *Proceedings of IEEE Pacific Visualization* (2010), pp. 169–176.

[SBVB14] SOLTÉSZOVÁ V., BIRKELAND A., VIOLA I., BRUCKNER S.: Visibility-driven processing of streaming volume data. In *Proceedings of EG Workshop on Visual Computing for Biomedicine* (2014), pp. 127–136.

[SPBV10] SOLTÉSZOVÁ V., PATEL D., BRUCKNER S., VIOLA I.: A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum 29*, 3 (2010), 883–891.

[SSHW*12] SOLTÉSZOVÁ V., SÆVIL-HELLJESEN L. E., WEIN W., GILJA O. H., VIOLA I.: Lowest-variance streamlines for filtering of 3D ultrasound. In *Proceedings of EG Workshop on Visual Computing for Biomedicine* (2012), pp. 41–48.

[TM99] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proceedings of International Conference on Computer Vision* (1999), pp. 839–846.

[WRW07] WESTENBERG M., ROERDINK J., WILKINSON M.: Volumetric attribute filtering and interactive visualization using the max-tree representation. *IEEE Transactions of Visualization and Computer Graphics* (2007), 2943–2952.

Supporting Information

Additional Supporting Information may be found in the online version of this article at the publisher's web site:

Supplementary material Video S1

Matlab script related to Figure 8

Matlab example for circumventing zero divisions in Equation 14