CGI2016 manuscript No.
(will be inserted by the editor)

Adaptive Transfer Functions

Improved Multiresolution Visualization of Medical Models

Jesús Díaz-García[1,2] · **Pere Brunet**[1] · **Isabel Navazo**[1] · **Pere-Pau Vázquez**[1] · **Frederic Perez**[2]

Received: date / Accepted: date

Abstract Medical datasets are continuously increasing in size. Although larger models may be available for certain research purposes, in the common clinical practice the models are usually of up to $512 \times 512 \times 2000$ voxels. These resolutions exceed the capabilities of conventional GPUs, the ones usually found in the medical doctors' desktop PCs. Commercial solutions typically reduce the data by downsampling the dataset iteratively until it fits the available target specifications. The data loss reduces the visualization quality and this is not commonly compensated with other actions that might alleviate its effects. In this paper we propose Adaptive Transfer Functions, an algorithm that improves the transfer function in downsampled multiresolution models so that the quality of renderings is highly improved. The technique is simple and lightweight, and it is suitable, not only to visualize huge models that would not fit in a GPU, but also to render not-so-large models in mobile GPUs, which are less capable than their desktop counterparts. Moreover, it can also be used to accelerate rendering framerates by using lower levels of the multiresolution hierarchy while still maintaining high quality results in a context and focus approach. We also show an evaluation of these results based on perceptual metrics.

Thanks to project TIN2014-52211-C2-1-R of the Spanish Ministry of Economy and Competitiveness and Alma IT Systems, for supporting J. Díaz-García's Ph.D. Thesis.

✉ Jesús Díaz-García - E-mail: jesusdz@cs.upc.edu
Pere Brunet - E-mail: pere@cs.upc.edu
Isabel Navazo - E-mail: isabel@cs.upc.edu
Pere-Pau Vázquez - E-mail: ppau@cs.upc.edu
Frederic Perez - E-mail: frederic.perez@about.me

[1] Universitat Politècnica de Catalunya
[2] Alma IT Systems

Keywords Transfer Function · Multiresolution Volume Model · Direct Volume Rendering · Level of Detail

1 Introduction

Medical imaging is challenged by the continuous increase in size of the datasets produced by capture devices (such as CT scanners). Although GPUs also evolve, their horsepower lies behind the size of the data. This is specially true for physicians that receive the data from radiologists, since their workplaces also commonly shared with other physicians, and, not being their main task the visualization of such data, they are not usually equipped with powerful GPUs. As a result, a common solution applied in many commercial software packages is to downsample the volume iteratively along its largest dimension until it fits a target size for interactive explorations. Although this strategy helps overcoming the limitations of GPUs [3], this also produces ostensible changes in the different multiresolution levels. Commonly, no other improvements are applied. Several advanced methods have already been proposed in the literature, such as compression or data partitioning techniques [1,4]. Unfortunately, most of these advanced techniques still require long preprocessing time and high GPU power that is not commonly available in commodity desktop PCs, or can not even be considered in nowadays mobile GPUs. Nonetheless, physicians can devote limited time to data inspection, so large processes that achieve aggressive compression cannot be applied and would also demand higher powered GPUs for the rendering stage.

With mobile GPUs, which are gaining traction lately, more especially among the medical community, there is an increasing demand on transferring computation tasks to mobile devices. As a consequence, several solu-

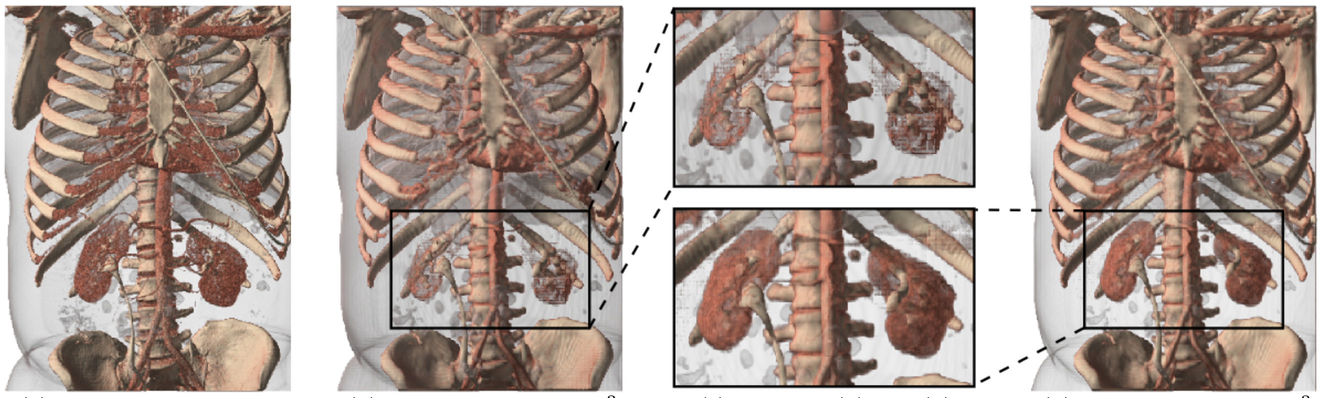

(a) Original dataset at 512^3 (b) Downsampled at 128^3 (original TF) (c) Detail of (b) and (d) (d) Downsampled at 128^3 (**adapted TF**)

Fig. 1 Enhancements for multiresolution rendering. The leftmost image (a) shows the model rendered at full resolution. In (b) the model was downsampled with a standard Gaussian-based filter and rendered using the original transfer function. In (d), the same downsampled model is rendered using an automatically adapted transfer function. Notice how (d) resembles (a) better than (b) approximating the overall colors and opacities with more accuracy.

tions have been developed to tackle volume models in such devices [22,14,23,21,27,20]. However, this pushes forward the pressure on data size and the quality of the renderings, and the reduced mobile GPU power further compromises the complexity of the algorithms to be applied to the compressed data.

As stated above, our research was triggered by a real need of the medical community after observing for many years that the situation is not likely to change anytime soon. The size of the images commonly available in medical practice are usually up to $512 \times 512 \times 2000$ voxels. A downsampling of two levels is usually sufficient to make them tractable in terms of interactivity^{JD} in the typical desktop computers of the physicians. Unfortunately, such an aggressive simplification changes the way the model is visualized, because the downsampling process implies modifying the image values. Thus, the original Transfer Function (TF henceforth) that might have been designed by a radiologist in a high-end computer will now exhibit different information in a downsampled model (see Figure 1-b). Our contribution is a method to automatically adapt the TF to downsampled levels, so that most of the quality of the higher level resolutions is preserved (see Figure 1-d).

As shown in Section 5 significative improvement is obtained both visually or when we numerically compare the results. Our technique is fully automatic, not requiring any user intervention, and is fast: if TF is changed, the new adapted TFs are recomputed in interactive time. Other advantages of our Adaptive TFs are:

1. Physicians can interactively inspect the full resolution level interactively, with seamless integration with the lower resolution levels: no blocking artifacts or seams are visible (see Figure 11).
2. It is independent of the downsampling method and the grid positions used to create the downsampled images.
3. Simple adaptation to rendering algorithms: The system only changes the TF for the corresponding level, and typical post-classification GPU-based rendering is achieved without framerate penalty.
4. It can be combined with other compression techniques.

Our system essentially compares the data of a coarser downsampled level with the original resolution dataset and analyzes how the density values vary. Then, it creates a new TF that approximates the *rendered results* of the downsampled data to the full resolution model. All this process is performed interactively; we need to store only the new 1D TF, and the downsampling histograms in case a new modification of the TF is required, so the extra information required is negligible.

The rest of the paper is organized as follows: Section 2 surveys related work. Section 3 analyzes the optimization procedure. In Section 4 we present our technique for TF adaptation. We then discuss the results in Section 5. We conclude the paper in Section 6 by pointing out some lines of future research.

2 Previous Work

There is a large amount of literature in different aspects of data reduction for volume rendering. On the one hand, many articles focus on lossy or lossless data compression (we refer the interested reader to the recent state of the art by Balsa *et al.* [1]). Another group of techniques partition data in order to make smaller chunks that are then loaded on demand to the GPU. These techniques include bricking, streaming, and other methods [4]. Other authors focus their work on optimized rendering on the CPUs by using specific data structures and instruction-level knowledge [15].

(Paper Isa)^{JD} Multiresolution techniques [29,32,3] use special data structures such as octrees [5,17,10], N^3-trees [6], wavelets [11], 3D mipmaps, hierarchical grids [8], or other sparse representations [26,33], in some cases also combined with compression techniques [9,7] and streaming [28]. We will concentrate on regular representations, as they are the most commonly found in commercial software.

Unfortunately, as previously stated, most of these techniques are not amenable to common commodity desktop PCs that physicians have at their desks, since they would require highly performing CPUs or GPUs, which is not the case, as commented above. Other methods require long pre-processing time which impedes their use since the span the physicians may devote to data exploration is scarce. As a result, simple downsampling is commonly performed. However, the downsampled data often suffers from artifacts due to the lower quality. Surprisingly, little effort has been devoted in literature to address this problem. Although many papers have focused on the important problem of aiding in the definition of the transfer function by automatic or semi-automatic methods (e.g. [25,19,24,13]), less numerous are the papers that concentrate on the effect of data reduction.

Bergner *et al.* [2] perform a spectral analysis of the composition of the scalar field and the transfer function. This way they provide essential understanding to the fundamental problem of proper sampling in volume rendering. With these theoretical tools they are able to provide an adaptive approach that significantly reduces the number of samples. ^{JD}

Wang *et al.* [30] concentrate on the preservation of features in data reduction. This is done by assigning importance to the voxels according to the current transfer function; in our case, we do not change the reduced data, but we adapt the TF to better provide the information on the original data. Ljung *et al.* [18] introduce the use of transfer functions at decompression time to guide a level-of-detail selection scheme. Kraus and Bürger [16] concentrate on the interpolation and reduction of RGBA data. In our case, since we must reduce the data just to make it fit the GPU memory, it is better to rely on the density values; otherwise the size of the data is multiplied by four.

Younesy *et al.* [34] also focus on improving the quality of renderings for coarse multiresolution levels. They state that the original data distribution in coarse levels of detail might be ideally approximated by storing local histograms at each low-resolution voxel. However, as the authors note, this is usually impractical due to its high storage requirements. Thus, they propose a simplification that consists of representing these histograms with a Gaussian basis function, which implies storing an average density (μ) and its standard deviation (σ) along with each voxel. Although they designed an efficient algorithm, the size of the data is increased with respect to traditional downsampling methods. This may be a problem if the available memory is limited. Our solution does not require such extra storage but a simpler small transfer function mapping.

More recently Sicat *et al.* [26] have presented an approach that uses a compact sparse representation of probability density functions (pdf) to capture voxel neighborhood distributions for consistent multiresolution volume rendering. They succeeded in avoiding erroneous data analysis (loss of information) when coarser models are rendered using the same TF than the initial volume. However, the significant precomputation time needed and the increase of storage make this representation impractical in the current clinical practice. Our objective is to obtain consistent visualizations for datasets commonly used in the medical practice but minimizing the modifications needed to the rendering pipeline, the preprocessing time, and the increment of memory storage. Furthermore, our system has no impact in terms of required computational power since we use the same ray casting algorithm with no modifications.

3 Optimal Transfer Function Adaptation for Coarse Levels

Given a volume dataset defined in a space $D \subset \mathbb{R}^3$, $V(x)$ is a scalar function that computes a density value for points $x \in D$:

$$V(x) : D \subset \mathbb{R}^3 \to \mathbb{R}.$$

Let us assume that this volume is evenly sampled in N^3 points x_i and stored in a voxel representation of N^3 resolution. For notation convenience, we use V_0 to refer to this original model and $z = V_0(x_i)$ the density value at x_i.

A multiresolution volume representation is a set of successively coarser resolution models $V_0, V_1, \ldots, V_n$. We assume a reduction factor of two in each dimension from successive resolutions so that, for $k > 0$, V_k is stored in a 1/8th of memory required for V_{k-1}. For $k > 0$, V_k is usually computed by filtering and downsampling a higher resolution representation. It is obvious that the distribution of values in the volume changes among the different levels of resolution.

Without loss of generality and in order to simplify notations, in the rest of this section we will only consider a 1-dimensional scalar field. Downsampling filtering is usually performed through a symmetric weighting function w of finite domain [12]:

$$V_k(x_j) = \sum_i V_0(x_i)\, w(x_j - x_i) \tag{1}$$

where, in uniform voxelizations, we can assume that $x_j = j \cdot 2^k h$ and that $x_i = i \cdot h$, with h being a constant spacing.

By considering the voxelization as a discrete representation of a continuous scalar field we can also write:

$$s = V_k(x) = \int_{-\infty}^{\infty} V_0(y)\, w(x-y)\, dy \tag{2}$$

where s is a density value obtained from the downsampled level V_k at a certain position x. We assume that values retrieved from any discretized volume are usually computed as a linear interpolation of neighbor voxel densities.

Let TF_0 be a 1D transfer function specifically designed to map density values of the original dataset V_0 to output color and opacity, $\mathit{TF}_0 : \mathbb{R} \to (R,G,B,\alpha)$; and let $I^{\mathit{TF}_0}(V_0)$ be the image obtained by rendering V_0 using TF_0. Therefore, if V_k is rendered using TF_0, the resulting image $I^{\mathit{TF}_0}(V_k)$ will differ and lose details from the render of the original volume $I^{\mathit{TF}_0}(V_0)$ due to the change of values on downsampling. The ideal after downsampling would be to have a new TF_k with which $I^{\mathit{TF}_k}(V_k) = I^{\mathit{TF}_0}(V_0)$, that is, $\mathit{TF}_k(V_k(x)) = \mathit{TF}_0(V_0(x))$ for all samples x in D.

Several previous papers [34,16] have tried to compute the $RGBA$ color C_k for a point x in the downsampled volume V_k by using a color averaging function w_C and defining:

$$C_k(V_k(x)) = \int_{-\infty}^{\infty} \mathit{TF}_0(V_0(y))\, w_C(x-y)\, dy \tag{3}$$

Notice that, in this equation, $C_k(V_k(x))$ is local, being in fact a function of the original densities around x and the original transfer function TF_0.

We observed that the downsampling process from the original dataset V_0 to a lower-resolution V_k (see equation 2) can be characterized by a 3D point cloud in the (x,z,s) space, where the x-dimension represents the N^3 voxels of V_0. Any point (x,z,s) of the cloud represents that the computation of the downsampled density $s = V_k(x)$ takes into account the density value $z = V_0(y)$ for y in the neighborhood of x (just observe that we have deliberately removed the second order locality of y in equation 2 by ignoring this y value).

We can compact this point cloud in the z-direction by encoding, for each pair (x,s), the occurrences of the z values used to compute $V_k(x)$. We refer to this information as the 2D histograms $H_{x,s}(z)$. Note that $H_{x,s}(z)$ stores the distribution of original density values z within a footprint of x in V_0.

Younesy *et al.* [34], by defining an appropriate weighting of these histograms, transform equation 3 into:

$$C_1(V_1(x)) = \int_{-\infty}^{\infty} \mathit{TF}_0(z)\, H_{x,s}(z)\, dz \tag{4}$$

(They focus on the case $k=1$.) The direct use of equation 4 requires storing one histogram per voxel in downsampled representations, which is unpractical. Thus, the authors approximate each of those histograms by two values μ and σ (mean and standard deviation) to represent the Gaussian curve that better fits their distribution (note that μ encodes the downsampled voxel density s of $V_k(x)$). This is in fact a *projection* of the point cloud in the z direction that allows pre-computing the integral in equation 4 into a 2D TF $\mathit{TF}_k(\mu,\sigma)$.

Our approach was inspired by the experimental behavior of the *discretized projection* of the point cloud in the x-direction. For each pair (z,s) we compute the number of sample points x such that $V_0(x) = z$ and $V_k(x) = s$. We refer to this information as the 2D histogram $H^d(z,s)$ (see Figure 2). We observed that this projection was clearly showing a $z-s$ correlation, even when the locality information on x had disappeared. In other words, the amount of information loss when projecting the point cloud in the x-direction is limited. By using the histograms $H^d(z,s)$ we are able to get rid of the spatial dimension and we still capture most of the downsampling information. By just changing the projection direction of the point cloud $\{(x,z,s)\}$, we move from local histograms $H_{x,s}$ to our global histograms $H^d(z,s)$.

Hence, we decided to use the downsampling global histograms $H^d(z,s)$ in equation 3. As we will see, they significantly improve the visual quality of V_k renderings (see Section 5) without storing extra information in their voxels. The goal is to compute TF_k so that the resulting $I^{\mathit{TF}_k}(V_k)$ is as close as possible to $I^{\mathit{TF}_0}(V_0)$:

$$\mathit{TF}_k(s) = \int_{-\infty}^{\infty} \mathit{TF}_0(z_H)\, H^d_k(z_H,s)\, w_H(z_H - \mu(s))\, dz_H \tag{5}$$

where w_H is a weighting function that allows us to focus on the z_H-interval that contributes with the highest information, and $\mu(s)$ is the average center point for every s, which can be directly estimated from $H^d_k(z_H,s)$. Now, by imposing that $\mathit{TF}_k(s)$ should be equal (or as close as possible) to $\mathit{TF}_0(s)$, we can write:

$$\mathit{TF}_0(z) = \int_{-\infty}^{\infty} \mathit{TF}_0(z_H)\, H^d_k(z_H,s)\, w_H(z_H - \mu(s))\, dz_H \tag{6}$$

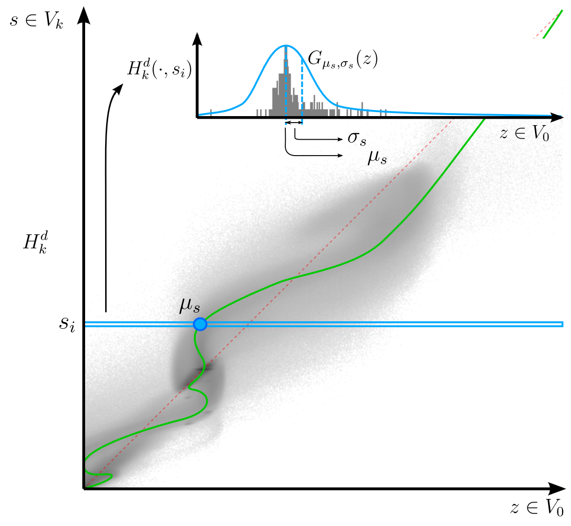
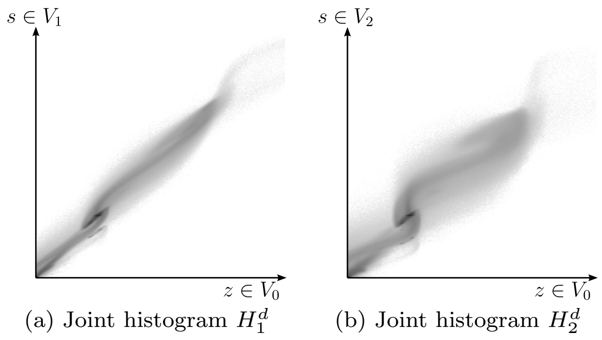


Fig. 2 Histogram of density correspondences between the original volume V_0 and a downsampled volume V_k in the multiresolution pyramid. Gray points lay on zones where the correspondences take place. The green line is a function of s that approximates a path fitting the mean of each individual row 1D histogram; the row histogram for a given s_i contains the overall information about what density used to be s_i in V_0 before having been downsampled.

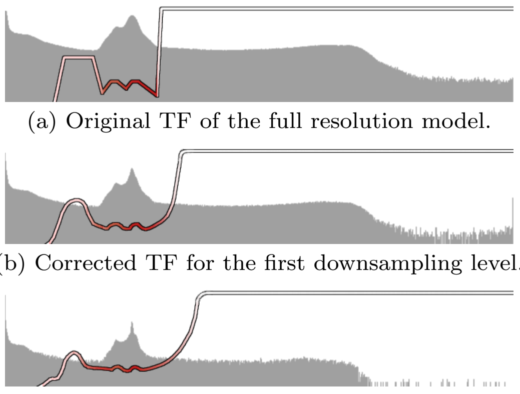


(a) Joint histogram H_1^d (b) Joint histogram H_2^d

Fig. 3 Histograms of correspondences between the full resolution model V_0 and subsampled versions of the Head dataset (see Figure 6). (a) shows the correspondences between V_0 and V_1, and (b) shows the correspondences between V_0 and V_2. Note how the values spread increasingly as long as we go to downsampled levels, because of the averaging functions that dilute the details of the voxels. This clearly suggests that using the original $\mathbb{TF}_0$ on V_k will likely be suboptimal.

By discretizing equations 2 and 6 and using the definition of s in equation 2, a specific equation is directly obtained for every sample position x. Observe that equation 5 computes $\mathbb{TF}_k(s)$, while equation 6 imposes that, for a fixed sample x, $\mathbb{TF}_0(z) = \mathbb{TF}_0(V_0(x))$ should be as close as possible to the resulting value of $\mathbb{TF}_k(s) = \mathbb{TF}_k(V_k(x))$ as evaluated from equation 5. The result of equation 5 is $\mathbb{TF}_k(s)$, while the unknown in equation 6 is the function w_H.

The set of equations 6 tries to force all colors and opacities in points of V_k to be identical to their corresponding locations in V_0. Note that this is an overdetermined linear system of equations on the unknown values that the averaging discrete function w_H takes on its finite domain. An optimal solution of this linear system in the least squares sense could be obtained by quadratic programming, by imposing that all discretized values $w_H(i)$ of the weighting function must be positive ($w_H(i) > 0$). However, for efficiency purposes, we have computed the optimum of equation 6 in a least squares sense on a restricted domain of positive weighting functions by testing a biparametric convex set of functions w_H, as discussed in Section 5. Our results show that a Gaussian averaging function w_H has a good behavior in all cases with small Root Mean Square (RMS) and perceptual errors. The following section describes our implementation using these Gaussian averaging functions.



(a) Original TF of the full resolution model.

(b) Corrected TF for the first downsampling level.

(c) Corrected TF for the second downsampling level.

Fig. 4 Original transfer function (top) and the adapted TFs obtained with our method for two different levels. The bottom one is used for rendering in the Head in Figure 6.

4 Fast Approximation of Optimal Transfer Functions for Coarse Levels

In this section we present our implementation to approximate the optimal TFs for coarse levels. This is achieved by analyzing the distribution of density values at the higher levels in relation with the finest level, once the multiresolution pyramid has been built.

Just after the multiresolution pyramid has been constructed, also as a pre-processing step, we need to compute the downsampling histogram $H_k^d(z, s)$ (the 2D-histogram that relates downsampled densities s of V_k and initial densities z of V_0) for each coarse level k. In Figure 2-top we see one of these histograms created by evaluating one value per voxel at the maximum resolution V_0. Obviously, for the lower resolution level V_k, the obtained values come from the trilinear interpolation (mimicking the behaviour of GPUs accessing 3D textures). Another example of the distribution of such joint histograms is depicted in Figure 3 for the Head dataset (see Figure 6). Gray points lie on zones where the correspondences take place. Notice how the loss of the original information is reflected by the spreading of the points as the downsampling increases. This indicates that the original TF will not work properly for downsampled levels. ^{JD}

We compute TF_k by traversing the vertical axis of H_k^d (s values in V_k) and, for each value, averaging the colors of $\mathit{TF}_0(z)$ using the information along the row (z values in V_0):

$$\mathit{TF}_k(s) \leftarrow \frac{1}{K}\sum_z \mathit{TF}_0(z)\, H_k^d(z,s)\, G_{\mu_s,\sigma_s}(z) \qquad \forall s \qquad (7)$$

Here, G_{μ_s,σ_s} is a Gaussian function centered at μ_s with standard deviation σ_s (see Figure 2), and the fraction $\frac{1}{K}, K = \sum_z H_k^d(z,s)G_{\mu_s,\sigma_s}(z)$, ensures that the weighted sum of colors is normalized. All rows of the histogram are visited, and in our current implementation the algorithm only traverses the values around the mean (we use values of $\pm 3\sigma_s$ around μ_s).

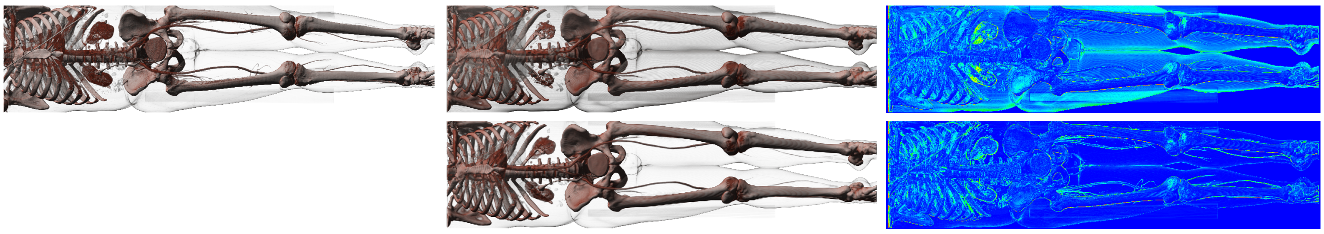
Once our method is applied, fitted transfer functions for V_k can be computed very quickly. The resulting adapted TFs for two coarse levels are shown in Figure 4. These correspond to the TF applied to the Head dataset (Figure 6). Note how it improves the result over the use of the original TF used in (b), and thus our approach yields images that are more similar to the original model in (a). Images (c) and (e) show, as a temperature map, the difference images between the image rendered at full resolution, and the ones obtained with the downsampled models (b) and (d), respectively. The model rendered with the fitted TF used in (d) yields a much better result than the original TF.

5 Results

We have analyzed different candidate weighting functions for equation 5. To ensure positiveness of this weighting function w_H in the least squares solution of equation 6, we have restricted our optimization to a biparametric convex set of weighting functions. We use barycentric coordinates on a triangular domain of functions to interpolate among three basis functions: a normalized Gaussian $G(x)$, a constant function $C(x)$ and a triangular function $T(x) = 0.4 \cdot (1 - 0.4 \cdot |x|)$. The biparametric weighting function is $w_H(x) = s \cdot C(x) + t \cdot T(x) + (1-s-t) \cdot G(x)$, with s and t being defined in the interval $[0,1]$ and x in the centered interval $[-2.5, 2.5]$. All basis functions are normalized, having a unit area in this interval. The equations, once normalized, are used to compute the root-mean-square (RMS) error for any w_H defined by a pair of parameters (s,t). Experiments performed on our test models confirm that the minimum error is obtained at $s = 0$, $t = 0$ in most cases (see Figure 7), while in the rest of the cases the resulting error is almost not sensible to (s,t) and to the shape of w_H.

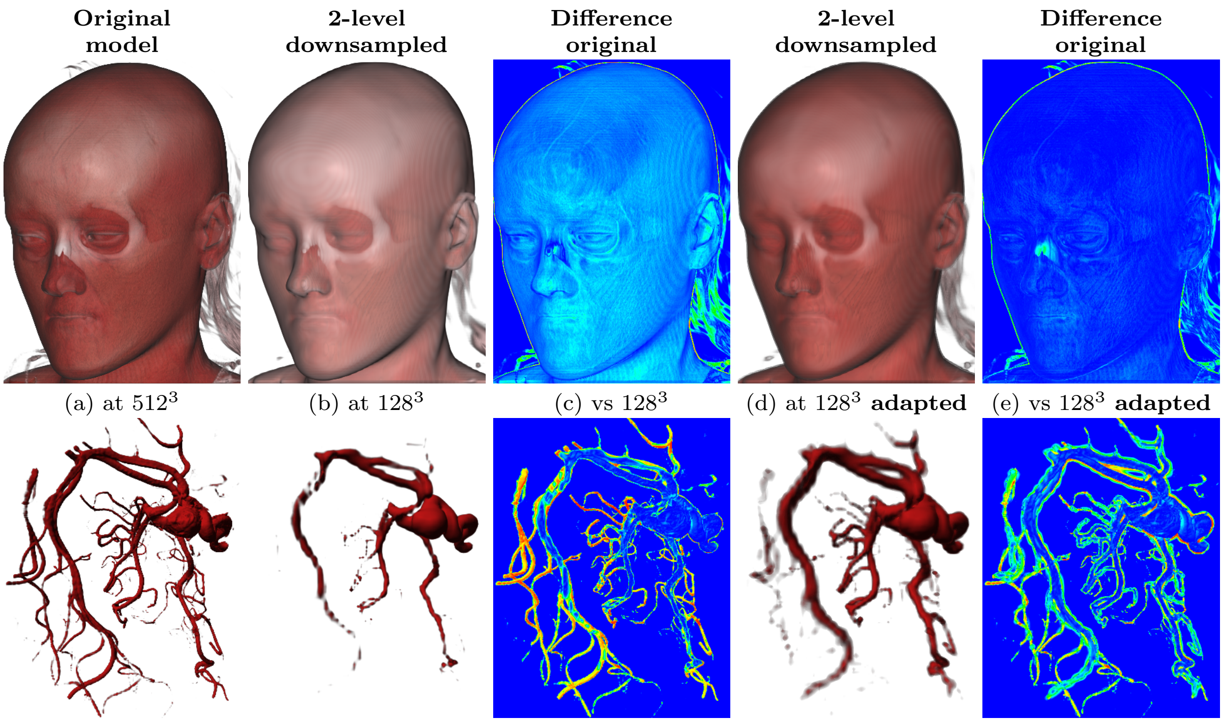
We have also compared the visual quality of different approaches using series of 20 images generated by positioning the camera at the center of all faces of an icosahedron bounding the volume. Each rendering of a downsampled model is compared against the rendering of the original full resolution model using a perceptually-based metric. More concretely, we analyzed the visual quality using the Structural SIMilarity (SSIM) index for image quality assessment [31]. The results using Gaussian and Constant weighting functions for equation 5 are shown in Figure 8, where lower values indicate less error. Errors when using barycentric coordinate interpolation and also using triangle-shaped functions are higher. By analyzing these results, we decided to use Gaussian averaging functions in our implementation. In this way we reduce RMS and perceptual errors while automatically removing outliers in the histogram of density correspondences. In addition, we have performed a set of experiments to show the advantages of using the adapted TF versus the original TF with different models. We have used a Quad Core i7 PC and a Core 2 Duo equipped with a GeForce GTX 470 with 1GB of RAM, and GTX 280 with 1GB of RAM, respectively. The resolutions of the models go up to $512^2 \times 1559$ for the Body model, 512^3 for the Head and the Chameleon, and 256^3 for the Foot and Aneurysm. The rendering algorithm is a GPU-based ray casting with pre-integrated classification and on-the-fly gradient computation, and the sampling step is of the size of the voxel (for the corresponding resolution level). In the first PC, the framerate was interactive and no change was produced with the Adaptive TF. The second PC could only render the large model at 2-3 fps, while our multiresolution rendering with ROI is one order of magnitude faster.

In all the examined cases, the adapted TFs clearly improve the quality of the rendered downsampled model with respect to using the original TF, as shown in Figures 5 and 6. The images shown correspond to a two-level simplification from their full resolution models. This data reduction would allow the models shown along the paper to fit into the GPUs of commodity PCs and most modern tablets and smartphones. We have compared the original models versus three different downsampling levels (see Figure 9). Observe that the pairs



(a) Original model ($512^2 \times 1559$) (b) Downsampled ($128^2 \times 389$) (c) Difference images

Fig. 5 The results of our method applied to a simplification of two levels of a $512^2 \times 1559$ model. The images show the improvement that is very noticeable (zooming in will reveal more details). Top row contains the original model (left), and the downsampling without TF adaptation. The bottom row uses our adapted TF. Note how different structures such as the kidneys are better preserved and the overall color of the image is highly improved. The images on the right illustrate the differences between the low resolution model and the original one.



(a) at 512^3 (b) at 128^3 (c) vs 128^3 (d) at 128^3 **adapted** (e) vs 128^3 **adapted**

(a) at 256^3 (b) at 64^3 (c) vs 64^3 (d) at 64^3 **adapted** (e) vs 64^3 **adapted**

Fig. 6 Results obtained with our modified TFs for two example datasets (Head and Aneurysm). The leftmost column shows models at full resolution. The second column shows the result of a two-level downsampling without TF change. Our TF adaptation method generates better results as shown in the fourth column. We compare the difference maps from the full resolution model in the third and last columns.

of bars corresponding to the V_2 and V_3 levels show that adapted TFs are always better at these levels. The first downsampling level V_1 also improves in three cases while having similar values in the Body and Head cases, with negligible differences. Note that, except for two cases where the first level does not improve (only a very small worsening), in all the rest of the cases the differences are noticeable.

Whenever the user changes the original transfer function, we need to adapt the TFs of the downsampled levels. This entire process takes fractions of a second (less than 0.01 seconds in our workstation)^{JD}, so it is performed interactively (see accompanying video). To obtain this timings, an optimization is needed in some cases; as some medical models require 12 bits per voxel (which means that we have to deal with histograms of 4096 values, and thus downsampling histograms of 4096^2 values) we choose to work with a reduced bit depth of 8 bits per voxels in the coarser representations, a simplification that is in fact much less aggressive than downsampling itself and supposes a great benefit regarding interactivity.^{JD} The achieved computation time is an insignificant amount of time, and it is definitely much faster than the time required by other, more com-

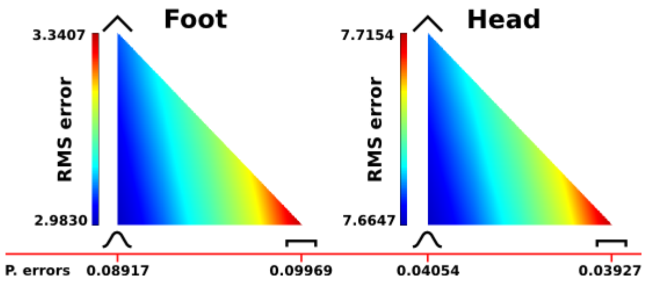


Fig. 7 RMS analysis of tested weighting functions w_H in equation 6. Triangles show the RMS error of all biparametric interpolated functions among a Gaussian (bottom-left corner), a Constant (bottom-right) and a Triangular (top) weighting function are shown for two models. The best behavior corresponds to Gaussian weighting, although the color scales show that differences are more important in the Foot dataset. The bottom of the figure also shows the perceptual errors (computed using SSIM measure on 20 views) to further evaluate the effects of both approximations on rendering (see Figure 8).

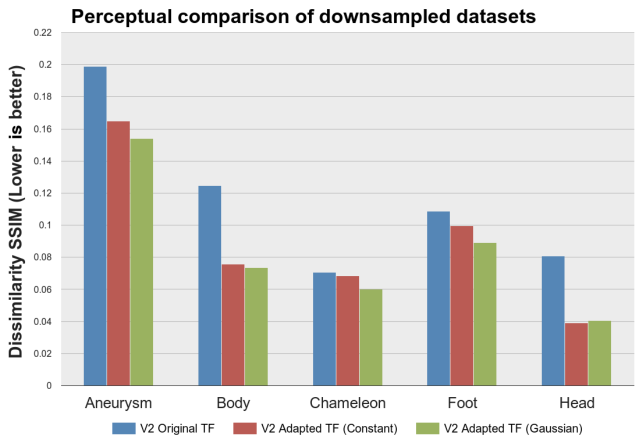


Fig. 8 Perceptual analysis of two tested weighting functions: Constant and Gaussian. Dissimilarities have been computed with the SSIM perceptually-based metric, as in Figure 9. The Gaussian weighting function produces either comparable or better results than the constant function.

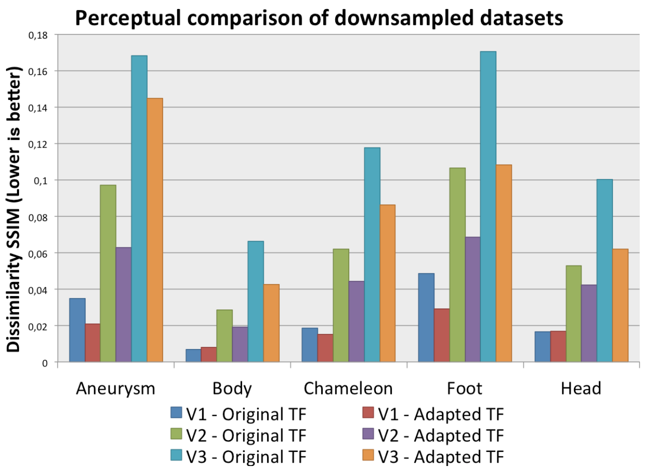


Fig. 9 Comparison of original models vs three different downsampling levels (V_1, V_2, and V_3) of the models used along the paper. The values are computed as the average similarity of 20 regularly spaced views using the SSIM perceptually-based metric [31]. Note that here we use *dissimilarity*, so lower values indicate better performance. Only two models (Body and Head) do not represent an improvement at the first downsampling level, with non significative differences of only 0.0013 and 0.0003.

Method	Data resolution	Total size	Overhead vs $256^3 + 128^3$
[34]	512^3	226MB	75.5MB
[26]	512^3	241MB	90MB
Ours	512^3	151MB	256kB+1kB

Table 1 Storage requirements comparison for a multiresolution of the 8 bits per voxel 512^3 Shepp-Logan model with two levels of downsampling. The original downsampling ($256^3 + 128^3$) requires 151MB. The method by Younesy *et al.* [34] requires 4 bytes per voxel one byte for the average μ, one byte for the standard deviation σ and two bytes for the gradient. The approach by Sicat *et al.* [26] uses a sparse structure. The values here are the ones declared by the authors applied for the 512^3 Shepp-Logan model.

plex techniques. A nice feature of adapted TFs is that they do not require special purpose modification of the rendering algorithm. Thus, the framerates do not decay, while still improving the quality.

Our method has scarce storage requirements. We need to store an adapted TF at each coarse level, and since it needs to be recomputed if the original TF changes, we must also keep the downsampling histograms information. Medical data often uses 12 bits per voxel, but in order to reduce storage and computational complexity, we use 8 bits per voxel in downsampled data, as we have seen that this optimization is in fact much milder than the actual subsampling in these kind of models. Considering this, our technique requires 1 byte per voxel in downsampled levels, plus 256kB (histogram) and $256 \times 4b = 1$kB (adapted TF) per level.

We compare our requirements with the ones by Younesy *et al.* [34] and Sicat *et al.* [26], as declared by the authors in their respective publications. In the first case, they require 4 bytes per voxel: one byte for the average μ, one byte for the standard deviation σ and two bytes for the gradient. In the second case, they store a sparse histogram whose size may vary depending on the model data. In Table 1 we show the requirements for a 512^3 resolution of the Shepp-Logan model. As it can be seen, our method clearly compares favorably against the others.

Although our research has been focused on medical data, the adapted TF can also be successfully applied to other volumetric models. We show an example in Figure 10 where the Nucleon and the Chameleon datasets are shown. Note that even with an aggressive downsampling such as the one in the Nucleon, where the

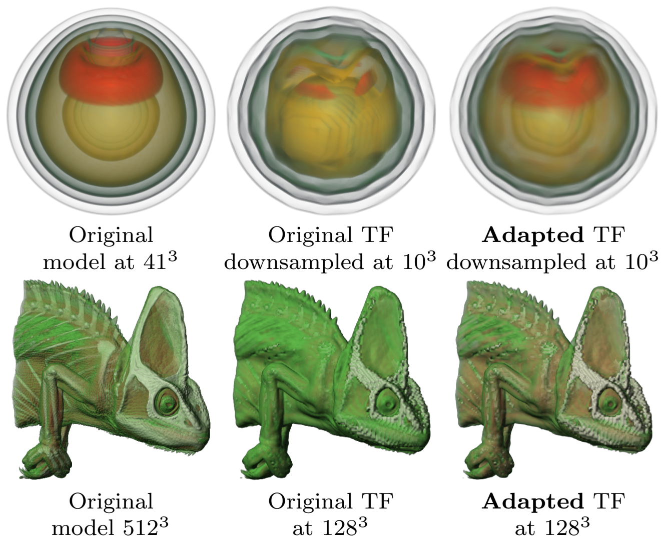


Fig. 10 Using the adapted TF for non-medical models, the Nucleon and the Chameleon, with two levels of downsampling, also achieves good results. The Nucleon dataset is very small; it is only used to illustrate that even with an aggressive downsampled version of 10^3, our fitted TF is able to recover quite a lot of information from the original model.

original model is only of 41^3 and thus the 2-level simplified version is only of 10^3, the information we are able to preserve is quite important. We can see it in the perceptual-based comparison where dissimilarity (computed using 20 views as in the previous chart) for the Nucleon dataset is on average 0.216 when we compare the 2-level downsampled model with the original one while when using our adaptive TF it is reduced to 0.078. For the chamaleon, the 2-level downsampled comparison yields dissimilarities of 0.062 with the original TF and 0.044 with our adapted TF.

Our approach can be used to render a simplified version of the model while showing the model at maximum resolution inside a user-defined Region of Interest -ROI- (see Figure 11 and accompanying video). Notice that the transition between the two resolutions is not perceived. The main advantage of this scheme in client-server architectures is that clients can store low-resolution versions of the model (V_2 in our example), so that only parts of V_0 inside the ROI must be received and rendered in the client devices.

The results obtained outdo the performance of commonly applied algorithms, since they allow us to *recover* information that was lost during the downsampling at a low cost, both in terms of memory and speed.

A *limitation* of our technique is its inherent global character. Different neighborhoods of the voxels in the original dataset may be downsampled to the same value on the lower resolution model. Given this fact, although we did achieve quite successful results in the models we tested, our TF adaption technique is not able to capture little and thin features, its improvements will likely be limited for highly heterogeneous models and in the worst case scenario different textured structures could be equally colored, although it is unlikely that typical medical models will behave this way.

If the original TF is changed, the adapted TFs must be recomputed. For typical models, this process only takes a small fraction of a second; however for 12-bit datasets, without any optimization, the process may take up to a second in our machine if not restricting to use 8 bits to encode downsampled voxels.

6 Conclusions and Future Work

In this paper we have addressed the problem of quality preservation in the visualization of coarser levels of multiresolution datasets through adaptive transfer functions. For each coarser level, we compute the joint histogram of the correlation between the original density values and the downsampled ones; then we automatically build an adapted TF that reduces the effect of data reduction when rendering this coarser model. This technique has four main advantages: *i)* storage costs are negligible (256+1kB per level), *ii)* the rendering is not affected using pre-integrated or post-classification, and the gradient -so shading- may be computed on-the-fly (thus, it can be combined to other data storage management methods), *iii)* the computation requires no manual parameter setting and is fully automatic, and *iv)* it is performed interactively. It is important to note that the method is orthogonal to any downsampling method, so it is not restricted to a subset of the original points, and may use any downsampling filter. It may also be seamlessly combined with any compression technique that generates density values; the decompressed model would be the one used to create the histograms, and during rendering time, it is then just necessary to substitute the TF fetch for a function that fetches the newly created adapted TFs. This approach allows to automatically obtain coarser models that can be used in modest GPU environments and is a good candidate technique for mobile rendering. Obviously, the combination with off-line or bricking techniques is straightforward.

In the future we will address the use of this quality preserving technique in other scenarios such as streaming, in combination with compression or with mobile devices.

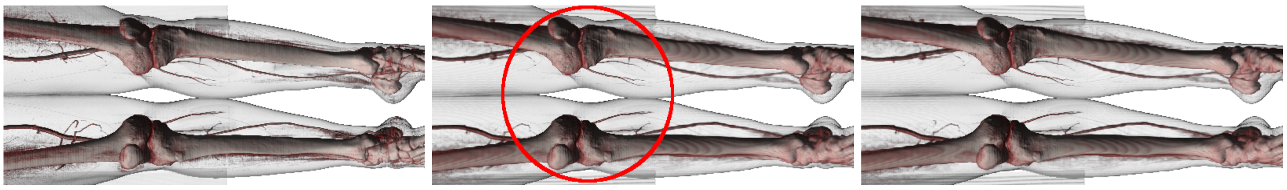

(a) V_0 (b) V_2 with ROI (boundary shown) (c) V_2 with ROI (boundary hidden)

Fig. 11 Part of the model in Figure 5 at the original resolution (a) and two levels of subdivision with the proposed algorithm (b) and (c). In (b) and (c), the simplified model V_2 is shown outside the Region of Interest (ROI), while the ROI shows the original model V_0. The ROI boundary is drawn in (b) for the purpose of comparison with (c).

References

1. Balsa Rodríguez, M., Gobbetti, E., Iglesias Guitián, J.A., Makhinya, M., Marton, F., Pajarola, R., Suter, S.: State-of-the-art in compressed GPU-based direct volume rendering. Computer Graphics Forum **33**(6), 77–100 (2014)
2. Bergner, S., Möller, T., Weiskopf, D., Muraki, D.J.: A spectral analysis of function composition and its implications for sampling in direct volume visualization. IEEE Trans. Visualization and Computer Graphics **12**(5), 1353–1360 (2006)
3. Beyer, J., Hadwiger, M., Möller, T., Fritz, L.: Smooth mixed-resolution gpu volume rendering. In: Proceedings of the Fifth Eurographics/IEEE VGTC conference on Point-Based Graphics, pp. 163–170. Eurographics Association (2008)
4. Beyer, J., Hadwiger, M., Pfister, H.: A survey of GPU-based large-scale volume visualization. Proceedings EuroVis 2014 (2014)
5. Boada, I., Navazo, I., Scopigno, R.: Multiresolution volume visualization with a texture-based octree. The Visual Computer **17**(3), 185–197 (2001)
6. Crassin, C., Neyret, F., Lefebvre, S., Eisemann, E.: Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games, I3D '09, pp. 15–22. ACM, New York, NY, USA (2009)
7. Fisher, M., Dorgham, O., Laycock, S.D.: Fast reconstructed radiographs from octree-compressed volumetric data. International Journal of Computer Assisted Radiology and Surgery **8**(2), 313–322 (2013)
8. Fogal, T., Schiewe, A., Krüger, J.: An analysis of scalable GPU-based ray-guided volume rendering. In: Large-Scale Data Analysis and Visualization (LDAV), 2013 IEEE Symposium on, pp. 43–51. IEEE (2013)
9. Gobbetti, E., Iglesias Guitián, J., Marton, F.: COVRA: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks. Computer Graphics Forum **31**(3pt4), 1315–1324 (2012). Proc. EuroVis 2012
10. Gobbetti, E., Marton, F., Iglesias Guitián, J.: A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets. The Visual Computer **24**(7-9), 797–806 (2008). Proc. CGI 2008
11. Guthe, S., Straßer, W.: Advanced techniques for high-quality multi-resolution volume rendering. Computers & Graphics **28**(1), 51–58 (2004)
12. Hadwiger, M., Kniss, J.M., Rezk-salama, C., Weiskopf, D., Engel, K.: Real-Time Volume Graphics. A. K. Peters, Ltd., Natick, MA, USA (2006)
13. Jankun-Kelly, T., Ma, K.L.: A study of transfer function generation for time-varying volume data. In: Proceedings of the 2001 Eurographics conference on Volume Graphics, pp. 51–66. Eurographics Association (2001)
14. Kitware, Inc.: VES, the VTK OpenGL ES rendering toolkit (2014)
15. Knoll, A., Thelen, S., Wald, I., Hansen, C.D., Hagen, H., Papka, M.E.: Full-resolution interactive cpu volume rendering with coherent bvh traversal. In: Proc. IEEE Pacific Visualization Symposium, pp. 3–10. IEEE Computer Society, Washington, DC, USA (2011)
16. Kraus, M., Bürger, K.: Interpolating and downsampling RGBA volume data. In: Proceedings of Vision, Modeling, and Visualization 2008, pp. 323–332 (2008)
17. LaMar, E., Hamann, B., Joy, K.I.: Multiresolution techniques for interactive texture-based volume visualization. In: Proceedings of the Conference on Visualization '99: Celebrating Ten Years, VIS '99, pp. 355–361. IEEE Computer Society Press, Los Alamitos, CA, USA (1999)
18. Ljung, P., Lundstrom, C., Ynnerman, A., Museth, K.: Transfer function based adaptive decompression for volume rendering of large medical data sets. In: Proceedings of the 2004 IEEE Symposium on Volume Visualization and Graphics, VV '04, pp. 25–32. IEEE Computer Society, Washington, DC, USA (2004)
19. Martin, S., Shen, H.W.: Interactive transfer function design on large multiresolution volumes. In: 2012 IEEE Symposium on Large Data Analysis and Visualization (LDAV), pp. 19–22. IEEE (2012)
20. Mobeen, M.M., Feng, L.: Ubiquitous medical volume rendering on mobile devices. In: International Conference on Information Society, pp. 93–98. IEEE (2012)
21. Moser, M., Weiskopf, D.: Interactive volume rendering on mobile devices. In: In Vision, Modeling, and Visualization VMV 2008 Conference Proceedings, pp. 217–226 (2008)
22. OsiriX Imaging Software: OsiriX HD (2014)
23. Raster Images: Oviyam - Web DICOM browser (2014)
24. Ruiz, M., Bardera, A., Boada, I., Viola, I., Feixas, M., Sbert, M.: Automatic transfer functions based on informational divergence. IEEE Transactions on Visualization and Computer Graphics **17**(12), 1932–1941 (2011)
25. Šereda, P., Vilanova, A., Gerritsen, F.A.: Automating transfer function design for volume rendering using hierarchical clustering of material boundaries. In: Proc. of Eurographics/IEEE VGTC conference on Visualization, pp. 243–250. Eurographics Association (2006)
26. Sicat, R., Hadwiger, M., Krüger, J., Möller, T.: Sparse PDF volumes for consistent multi-resolution volume rendering. IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization) **20**(12), 2417–2426 (2014)
27. Sousa, R., Nisi, V., Oakley, I.: Glaze: A visualization framework for mobile devices. In: Human-Computer Interaction–INTERACT, pp. 870–873. Springer (2009)

28. Thelen, S., Meyer, J., Ebert, A., Hagen, H.: Giga-scale multiresolution volume rendering on distributed display clusters. In: Human Aspects of Visualization, pp. 142–162. Springer (2011)
29. Wang, C., Gao, J., Li, L., Shen, H.W.: A multiresolution volume rendering framework for large-scale time-varying data visualization. In: Proceedings of the Fourth Eurographics/IEEE VGTC conference on Volume Graphics, pp. 11–19. Eurographics Association (2005)
30. Wang, Y.S., Wang, C., Lee, T.Y., Ma, K.L.: Feature-preserving volume data reduction and focus+context visualization. IEEE Transactions on Visualization and Computer Graphics **17**(2), 171–181 (2011)
31. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. Image Processing, IEEE Transactions on **13**(4), 600–612 (2004)
32. Weiler, M., Westermann, R., Hansen, C., Zimmermann, K., Ertl, T.: Level-of-detail volume rendering via 3D textures. In: Proceedings of the 2000 IEEE Symposium on Volume Visualization, VVS '00, pp. 7–13. ACM, New York, NY, USA (2000)
33. Xu, X., Sakhaee, E., Entezari, A.: Volumetric data reduction in a compressed sensing framework. Computer Graphics Forum **33**(3), 111–120 (2014)
34. Younesy, H., Möller, T., Carr, H.: Improving the quality of multi-resolution volume rendering. In: Proc. Joint Eurographics/IEEE VGTC conference on Visualization, pp. 251–258. Eurographics Association (2006)