# Per-pixel Opacity Modulation for Feature Enhancement in Volume Rendering

Stéphane Marchesin and Jean-Michel Dischler and Catherine Mongenet
LSIIT – Louis Pasteur University – Strasbourg, France

***Abstract*— Classical direct volume rendering techniques accumulate color and opacity contributions using the standard volume rendering equation approximated by alpha blending. However, such standard rendering techniques, often also aiming at visual realism, are not always adequate for efficient data exploration, especially when large opaque areas are present in a dataset, since such areas can occlude important features and make them invisible. On the other hand, the use of highly transparent transfer functions allows viewing all the features at once, but often makes these features barely visible. In order to enhance feature visibility, we present in this paper a straightforward rendering technique that consists in modifying the traditional volume rendering equation independently of any transfer function. Our approach is fully automatic and based on a function quantifying the relative importance of each voxel in the final rendering called *relevance function*. This function is subsequently used to dynamically adjust the opacity of the contributions per-pixel. As will be shown by our comparative study with standard volume rendering, this makes our rendering method much more suitable for interactive data exploration at a low extra cost. Thereby, our method avoids feature visibility restrictions without relying on a transfer function and yet maintains a visual similarity with standard volume rendering.**

## I. Introduction

Traditional direct volume rendering techniques consist in integrating opacity and color values along a viewing ray by using a simple light transport model, called the volume rendering equation, inspired by the physics of light traversing a participating media (usually only absorption is considered, with no scattering). Using this approach, direct volume rendering allows one to display multiple values contributing to one pixel, unlike iso-surface extraction methods, for which a single value is considered. Therefore, volume rendering is widely used for the interactive exploration of 3D datasets since it allows one to display all data ”at once”. However, one major issue consists in defining a transfer function which associates opacity and color values to the data. Setting up this transfer function and in particular the opacity function is an intricate task, since it generally requires the expertise of the user and is deeply data dependent.

In order to address this issue, one can attempt to ease the generation of transfer functions, and keep the classical volume rendering equation. Numerous authors have attempted this [KMM*01], [RBS05], [KD98]. This can be done either by hand, which is, however, an excessively time-consuming and painstaking process, or automatically or semi-automatically, by introducing a specific data analysis process. In either case, it cannot be excluded that for some datasets wrong decisions could be made when building the transfer function. This has implications for the common volume rendering techniques:

e-mail: {marchesin,dischler,mongenet}@dpt-info.u-strasbg.fr

- In the case of direct volume rendering, opaque features might occlude some other parts of the dataset for some or even, in the worst case, for all viewing directions. This could make feature visibility in the dataset highly view-dependent. Conversely, setting up a transfer function with insufficient opacity might result in unintelligible pictures. Consequently, classical volume rendering techniques, especially based on the traditional volume rendering equation, might become impractical for efficient data exploration.
- In the case of the additive blending technique, the produced pictures are often under or over saturated. Since this technique uses a simple additive blending equation, over saturation of pictures can happen very quickly and with only a very limited number of contributions. On the other hand, setting a low opacity for these contributions can result in under saturated pictures which do not convey enough information to be useful and is furthermore highly dependent on the viewpoint.

In order to resolve the crucial issue of simultaneous visibility of data features in a view-independent fashion, one can resort to a volume rendering integral modification in the hope of avoiding a potentially incorrect transfer function setup. By doing so, these methods are trading visual realism for pictures providing a better understanding of the data. In particular, the advent of non-photorealistic techniques has allowed a new range of visualization clues to be added to the generated pictures, in particular in the context of volume exploration.

In a previous paper, we introduced such a method, that consists in modifying the volume rendering integral to achieve better visibility of the internal structures in datasets. It is view-dependent and dynamically adjusts the volume rendering integral per-pixel, thereby improving the resulting pictures by conveying more dataset features at once. In order to identify empty areas, a binary classification of the data was required. In this paper, we refine this principle by further introducing a function called *r*elevance function that quantifies the relative importance of each voxel in the final rendering. We experiment with different suitable alternatives for this function, such as using the gradient and second order derivative. As the approach is still automatic, no complex transfer function setup is required. Unlike completely non-photo realistic volume rendering techniques, our approach remains very close to traditional volume rendering, thus providing pictures close to usual visual realism. For example, we use a traditional surface-based shading model. Yet it can be considered a non-photo realistic method since it does not respect the actual physical light transport equation. We have implemented our technique on the GPU, and show that it can thereby reach interactive performance.

In the next section, we detail related works. In section 3, we present our new volume rendering technique. We then study the influence of different *relevance functions* in section 4. In section 5, we make some adjustments to improve visual quality. We discuss preintegration of our volume rendering equation in section 6. The implementation of our technique on the GPU is detailed in section 7. We present results with various datasets and compare our technique to other volume visualization methods in section 8. Finally, we give concluding remarks in section 9.

## II. Related work

In recent years, many non-photorealistic volume visualization techniques have been proposed. These techniques trade picture realism for greater data understanding.

Saito *et al.* [Sai94] propose a first non-photorealistic volume visualization technique. They achieve non-photorealistic previewing of volume fields using simple primitives such as points or lines. Ebert *et al.* [ER00] use non-photorealistic feature enhancement by modifying the color and transparency of the voxels. Viola *et al.* [VKG04], [VKG05] introduce a technique called importance-driven volume rendering. This technique uses predefined priorities between structures in the dataset and then renders accordingly: low priority structures can be occluded by higher priority ones, while high priority structures cannot be occluded by lower priority ones. However, this requires *à priori* knowledge of the dataset in order to segment and prioritize the object features. Bruckner *et al.* [BGKG05] propose a model for preserving contextual information while prioritizing relevant information according to real-time adjustable parameters. Kraus [Kra05] introduces scale-invariant volume rendering. This technique integrates the data in data space instead of doing so in physical space, and thereby achieves scale invariance for volume visualization. By its design, this technique leads to the same color and opacity for structures bearing the same density but different thicknesses. Sato *et al.* [SSN98] modify the maximum intensity projection scheme by selecting the first value above a threshold on a given ray instead of the maximum value over the whole ray. Hauser *et al.* [HMBG00] mix two well-known volume visualization techniques, namely direct volume rendering and maximum intensity projection, into a joint method. The dataset is first segmented into two classes, and those classes are then rendered using either one or the other technique. Csébfalvi *et al.* [CMH$^*$01] present a visualization technique which uses a gradient-based voxel selection, and only renders the relevant voxels. This results in efficient visualization of contours in the dataset. Mora *et al.* [ME04] explore order independent volume rendering and extend existing maximum intensity projection (MIP) [MGK99], [ME05] and X-ray techniques. Sun *et al.* [SRR04] propose to reduce noise in confocal microscopy pictures using an adaptive technique.

In order to improve the quality of numerical integration, Engel *et al.* [EKE01] propose a technique called preintegration. This technique increases the visual quality of volume rendering by pre-computing slabs of the volume rendering integral.

In the field of high dynamic range (HDR) volume visualization, Ghosh *et al.* [GTH05] have developed a technique to achieve volume rendering on high dynamic range displays. Yuan *et al.* [YNCP05] visualize high dynamic range datasets on a standard display system using two passes: first the data is rendered into a high dynamic range buffer, and then the resulting image is tone mapped into an image suitable for display.

Work has also been undertaken in order to improve the use of transfer functions. Kniss *et al.* [KMM$^*$01] introduce multi-dimensional transfer functions, which depend not only on the data value, but also on the gradient magnitude. Roettger *et al.* [RBS05] automatically generate multi-dimensional transfer functions and let the user choose among segmented parts of the transfer function to highlight matching object parts. However, setting up a transfer function can be complex for users. In order to avoid this transfer function setup phase, a number of techniques have been developed. Kindlmann *et al.* [KD98] generate transfer functions in a semi-automatic fashion. Fujishiro *et al.* [FAT99] analyze the topology of the data to create a suitable transfer function. Fang *et al.* [FBT98] use an image-based selection tool to ease the creation of transfer functions.

In the field of 2D imaging, the local histogram equalization technique [GW92] enhances the contrast of a picture by performing histogram equalization in the neighbourhood of each pixel.

Even though locally adaptive techniques are widely used in the image processing field, such techniques have been rarely used for visualization. Among all the techniques discussed above, very few take advantage of view-dependent rendering to increase the amount of information in the picture. Those that do so usually require the user to go through a complex data segmentation and/or transfer function setup phase. Since this setup is view-independent, it does not adapt to the current viewing conditions in order to maximize visibility of internal structures. Furthermore, the complexity of the preprocessing phase usually makes volume rendering unsuitable as a data exploration tool. In this paper, our purpose is to move that complex phase further in the visualization pipeline by dynamically and locally adjusting the rendering at runtime. We also aim at dataset exploration, and thus our algorithm should not require any complex setup, and should be able to achieve interactive rendering. Finally, as opposed to Kraus [Kra05], we want to be able to distinguish objects with different thicknesses and therefore thin objects should not appear similarly to thick ones. Therefore, we introduce a new approach which is halfway between classical photo-realistic volume visualization techniques and non-photorealistic volume visualization techniques. It locally adjusts the opacity values of the contributions, and thereby manages to keep a greater number of features visible at the same time. Since opacity values are computed on the fly, our method does not need any opacity transfer function setup, but only a binary thresholding of empty areas.

## III. Reformulating volume rendering using a relevance function

The volume rendering technique we present in this section aims at solving the visibility issues pertaining to volume rendering by dynamically adapting the opacity of the fragment contributions at a per pixel level. Our technique is based on the use of a *relevance function* quantifying the relative importance of each voxel in the final rendering according to a given number of parameters. This function is subsequently used to compute the voxel opacity, and therefore removes the need for an opacity transfer function setup, which can prove time-consuming for the user. Such a function depends only on the data, and can be realized in various ways, based on intrinsic signal properties like the scalar value, or the local gradient. Let us now present our per pixel adaptive technique in both contexts of DVR and additive volume rendering.

### A. *Per-pixel adaptive DVR*

Let $C(x,y)$ represent the final pixel color at $(x,y)$, $s(i)$ the scalar value of the $i^{th}$ sample along a viewing ray cast from $(x,y)$, $c()$ the color transfer function for scalar $s$ so that $c(s(i))$ is the color of the $i^{th}$ sample, $\tau()$ the opacity function so that $\tau(s(i))$ is the opacity of the $i^{th}$ sample, and $L$ the length of the ray. The classical direct volume rendering integral is then as follows:

$$C(x,y) = \int_0^L c(s(t)) \exp(-\int_0^t \tau(s(u))du)dt \tag{1}$$

It can be approximated by the following Riemann sum:

$$\tilde{C}(x,y) = \sum_{i=0}^{n-1} \tilde{c}(\tilde{s}(i))\tilde{\tau}(\tilde{s}(i)) \prod_{j=0}^{i-1}(1-\tilde{\tau}(\tilde{s}(j))) \tag{2}$$

Let us now define the *relevance* of a contribution: we define the *relevance* of a contribution as its importance relative to other contributions. Figure 1 shows the simple case of a binary *relevance* function where *relevant* contributions are samples that fall within segmented parts of the datasets (shown in blue on the figure). Let $\delta()$ be our classification function, so that $\delta(i)$

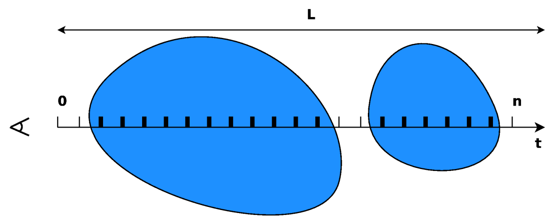

Fig. 1. Counting the relevant contributions (in bold) carried by a ray going through a segmented dataset (in blue). Among the 22 samples, there are only 17 which are relevant.

is defined as greater than 0 when the $i^{th}$ sample on the ray is *relevant*, 0 otherwise. In order to keep all the structures that project to a given pixel visible at the same time, we use the same opacity $\tilde{\tau}_0$ for all the *relevant* contributions made to this pixel times $\delta(i)$, to account for a relevance weight:

$$\tilde{C}(x,y) = \sum_{i=0}^{n-1} \delta(i)\tilde{c}(\tilde{s}(i))\tilde{\tau}_0(1-\tilde{\tau}_0)^{\sum_{j=0}^{i-1}\delta(j)-1} \tag{3}$$

In order to keep as many features as possible visible at once, we want to compute $\tilde{\tau}_0$ that maximizes the opacity of the furthest contribution over $[0,1]$ on the screen. The total opacity for the furthest contribution is then:

$$\tilde{\tau}_{furthest}(x,y) = \tilde{\tau}_0(1-\tilde{\tau}_0)^{\sum_{i=0}^{n-1}\delta(i)-1} \tag{4}$$

To maximize this function, we take its derivative and find its zeroes:

$$\frac{\mathrm{d}\tilde{\tau}_{furthest}(x,y)}{\mathrm{d}\tilde{\tau}_0} = \left(1-\sum_{i=0}^{n-1}\delta(i)\tilde{\tau}_0\right)(1-\tilde{\tau}_0)^{(\sum_{i=0}^{n-1}\delta(i)-2)} \tag{5}$$

We are only interested in opacity values in $]0,1[$ since $\tilde{\tau}_0 = 0$ and $\tilde{\tau}_0 = 1$ respectively mean fully transparent and fully opaque:

$$(1-\sum_{i=0}^{n-1}\delta(i)\tilde{\tau}_0)(1-\tilde{\tau}_0)^{(\sum_{i=0}^{n-1}\delta(i)-2)} = 0 \tag{6}$$

$$\Rightarrow 1-\sum_{i=0}^{n-1}\delta(i)\tilde{\tau}_0 = 0 \Rightarrow \tilde{\tau}_0 = \frac{1}{\sum_{i=0}^{n-1}\delta(i)} \tag{7}$$

In order to maximize visibility of the furthest contributions, one should choose an opacity $\tilde{\tau}_0 = \frac{1}{\sum_{i=0}^{n-1}\delta(i)}$. We therefore end up with the following rendering formula:

$$\tilde{C}(x,y) = \sum_{i=0}^{n-1} c(\tilde{s}(i))\frac{1}{\sum_{k=0}^{n-1}\delta(k)}\prod_{j=0}^{i-1}(1-\frac{1}{\sum_{k=0}^{n-1}\delta(k)}) \tag{8}$$

Note that there is no longer any $\tau()$ opacity function in this formula.

### B. *Per-pixel additive volume rendering*

Let $C(x,y)$, $c()$, $\delta()$ and $\tau()$ be defined as previously. The additive blending volume rendering integral is as follows:

$$C(x,y) = \int_0^L c(s(t))\tau(s(t))dt \tag{9}$$

which can be approximated by a Riemann sum in the following way:

$$\tilde{C}(x,y) = \sum_{i=0}^{n-1} c(\tilde{s}(i))\tilde{\tau}(\tilde{s}(i)) \tag{10}$$

In order to avoid saturating colors at a given pixel in our adaptive opacity technique, we want to average the values of the *relevant* contributions gathered over a ray:

$$\tilde{C}(x,y) = \frac{1}{\sum_{i=0}^{n-1}\delta(i)}\sum_{i=0}^{n-1} c(\tilde{s}(i))\delta(i) \tag{11}$$

which can be rewritten as:

$$\tilde{C}(x,y) = \sum_{i=0}^{n-1}\frac{c(\tilde{s}(i))\delta(i)}{\sum_{j=0}^{n-1}\delta(j)} \tag{12}$$

## IV. Relevance functions

In this section, we study the influence of the relevance function $\delta()$ on the final results, and we experiment with multiple relevance functions using different parameters.

- The simplest choice for $\delta()$ consists in using a binary function. That is, $\delta()$ is such that $\delta(i) = 1$ if the contribution of the $i^{th}$ sample is *relevant* and 0 otherwise according to the segmentation depicted on figure 1. In practice, we simply use a binary classification of the scalar value at each sample $s(i)$. This is a solution we have used in a previous paper [MDM07].
- Another solution that we experimented with consists in weighting the relevant contributions instead of considering them as all being equal. A common assumption is to consider regions of high variance, e.g. high gradient, as more important than regions with low variance, e.g. low gradient. It therefore makes sense to use the gradient value as a relevance function. In this case, the relevance of a contribution is defined as $\delta(i) = \|\nabla s(i)\|$.
- We have also tried to use a thresholding of gradient values, that is, the relevance function is defined as $\delta(i) = \begin{cases} 1 & if \|\nabla s(i)\| > t \\ 0 & otherwise \end{cases}$ where $t$ is a user-defined threshold.
- In a similar way to the first order gradient, we have conducted experiments using the second order derivative or Laplacian as a relevance criterion, that is $\delta(i) = \Delta s(i) = \nabla \cdot \nabla s(i)$.

- Feature boundaries are often considered important in datasets. These can also be taken into account by our relevance function by considering both the gradient and second order derivative. Contributions have an increasing gradient as they approach boundaries, while the Laplacian crosses zero near a boundary. We have implemented a simple boundary detection as follows: $\delta(i) = \begin{cases} 1 & if \|\nabla s(i)\| > t_1 \text{ and } \|\Delta s(i)\| < t_2 \\ 0 & otherwise \end{cases}$, where $t_1$ and $t_2$ are user-defined thresholds.

Some of these techniques for choosing relevance functions are fully automatic and require no user specific manipulation, some others require minimal intervention like setting a threshold up or defining a binary classification. We show various examples of these different functions in section 8.

## V. Method adjustments

If the per-pixel volume rendering technique is used strictly as described previously, it produces images that do not clearly depict the object structures as shown in the second image of the top row of Figure 2. In order to efficiently use this method, one has to make some adjustments to it. The first of these adjustments bounds the opacity values. In order to achieve visual continuity, the second adjustment filters what we call the *relevance map*: the sum of *relevance* values for each pixel. If the relevance function $\delta(i)$ is binary, it represents a simple count of the number of relevant contributions. Another adjustment is the addition of depth cues through depth-based coloring and the last one consists in adding a local shading in order to improve the perception of shapes. These last two adjustments produce pictures which have some visual relationship with traditional volume rendering. We now describe these adjustments in detail:

- When the opacity values are used as-is, our technique adjusts even the lowest opacities to higher values, which gives false visual clues. It occurs for instance when there are only few *relevant* contributions to a pixel, that is $\sum_{i=0}^{n-1} \delta(\tilde{s}(i))$ is small and inversely $\tilde{\tau}_0$ becomes high. We therefore decided to add an upper bound $\tilde{\tau}_{max}$ to $\tilde{\tau}_0$. This bound was determined experimentally to be within the $[0.1, 0.2]$ range. The third and fourth images of the top row of Figure 2 show $\tilde{\tau}_{max} = 0.25$ and $\tilde{\tau}_{max} = 0.12$, respectively.
- In order to avoid giving false information about the dataset, spatial continuity should be ensured in screen space. However, when too few contributions are made to a single pixel, it can result in discontinuities in the picture which look like edges. To avoid this, we filter the relevance map using a Gaussian blur. After rendering the relevance map, we run it through a blurring filter that removes most of the high-frequency data. The second pass is then done from that same map. Thanks to that improvement, the noise from the relevance map is successfully removed and internal structures become clear as shown on the bottom left of Figure 2. An alternate implementation of filtering is to render the relevance map at a lower resolution during the first pass, and stretch it using the card's native bilinear filtering capabilities. Using this functionality is faster, at the expense of less accurate results as shown by the second and third images of the bottom row of Figure 2 which show 2 times and 4 times scaling, respectively. These pictures show that it is possible to greatly improve the interactivity of our technique at the expense of visual quality.
- In the case of additive volume rendering, the depth information is lost since the blending method is commutative. This means that two contributions, one on the back of the dataset, and the other on the front, will have the same result on screen. Therefore, it is primordial to restore the depth information. In this purpose, we add depth-based coloring of the fragment samples: as the samples get further from the observer, we modulate their color proportionally to the depth of the sample. This allows an increased depth perception in the produced pictures as shown by the bottom right image of Figure 2.
- In order to improve the perception of shapes it is also important to add a local shading model, such as the Phong one. This consists in replacing the colour $\tilde{c}(\tilde{s}(i))$ with the following formula:

$$\tilde{c}_l = \tilde{c}(\tilde{s}(i)) * (K_a + K_d \times \overrightarrow{L} \cdot \overrightarrow{N} + K_s \times (\overrightarrow{R} \cdot \overrightarrow{V})^{\alpha})$$

where $K_a$,$K_d$ and $K_s$ are the Phong coefficients, $\alpha$ is the specular exponent, $\overrightarrow{L}$ is the light vector, the normal $\overrightarrow{N}$ is assimilated to the gradient of $\tilde{s}(i)$, $\overrightarrow{V}$ is the direction towards the viewer and $\overrightarrow{R}$ the local reflection vector computed according to the local gradient.

## VI. Preintegration

Preintegration is an important feature for volume rendering. It is possible to achieve preintegration in our context of per-pixel additive volume rendering, provided the relevance function depends on a single scalar parameter. In this section, we describe how this can be done in the case of a relevance function depending on the scalar value. Starting from the additive adaptive volume rendering integral we have:

$$C(x,y) = \frac{\int_0^L c(s(t))\delta(s(t))dt}{\int_0^L \delta(s(t))dt} = \frac{\sum_{i=0}^{n-1} \tilde{c}(i)}{\sum_{i=0}^{n-1} \tilde{\delta}(i)} \tag{13}$$

with

$$\tilde{c}(i) = \int_{\frac{L}{n}i}^{\frac{L}{n}(i+1)} c(s(t))\delta(s(t))dt \tag{14}$$

and

$$\tilde{\delta}(i) = \int_{\frac{L}{n}i}^{\frac{L}{n}(i+1)} \delta(s(t))dt \tag{15}$$

which can be approximated as follows by assuming a linear scalar function $s()$ over the $i^{th}$ interval:

$$\tilde{c}(i) \approx \int_{\frac{L}{n}i}^{\frac{L}{n}(i+1)} c(s(i)(1-(t-\frac{L}{n}i)) \\ +s(i+1)(t-\frac{L}{n}i))\delta(s(i)(1-(t-\frac{L}{n}i)) + s(i+1)(t-\frac{L}{n}i))dt \tag{16}$$

and

$$\tilde{\delta}(i) \approx \int_{\frac{L}{n}i}^{\frac{L}{n}(i+1)} \delta(s(i)(1-(t-\frac{L}{n}i)) + s(i+1)(t-\frac{L}{n}i))dt \tag{17}$$

It is therefore possible to preintegrate the values of $\tilde{c}(i)$ and $\tilde{\delta}(i)$ in two separate tables and thus achieve preintegration in the context of adaptive additive volume rendering. Figure 3 shows that doing so greatly improves the quality of the pictures.

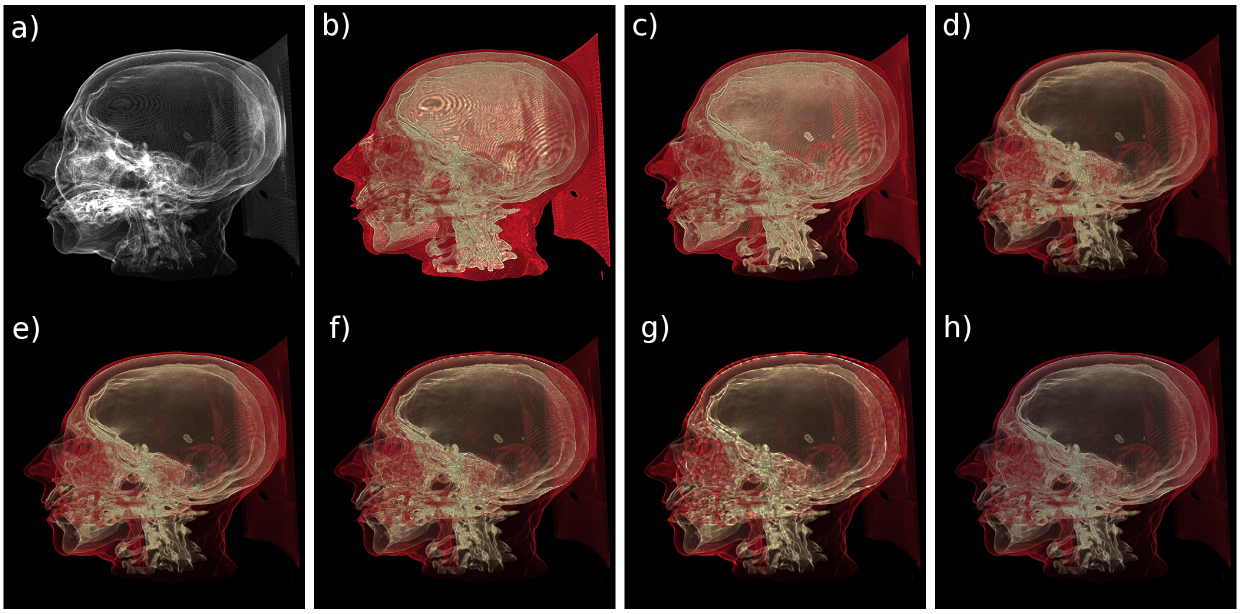

Fig. 2. (a) Relevance map (representing the sum of *relevant* contributions for each pixel), Pictures produced from: (b) naive implementation with $\tilde{\tau}_{max} = 1$ unfiltered (20.5 fps), (c) $\tilde{\tau}_{max} = 0.25$ unfiltered (20.5 fps), (d) $\tilde{\tau}_{max} = 0.12$ unfiltered (20.5 fps), (e) $\tilde{\tau}_{max} = 0.12$ Gaussian filtered (8.5 fps), (f) $\tilde{\tau}_{max} = 0.12$ bilinear with 2 times (24.5 fps) bilinear scaling, (g) 4 times (25.3 fps) bilinear scaling and (h) z-based coloring with $\tilde{\tau}_{max} = 0.12$ and Gaussian filtering (8.5 fps).

Note however that it is not convenient to preintegrate adaptive direct volume rendering since the table would have 3 entries: the two scalar values as with standard preintegration, and the current opacity. This, in turn, would require a 3D texture to store the preintegration table which implies a high video memory usage. Preintegration of relevance functions using multiple scalar parameters further increases the dimensionality requirement for the tables. For example a table preintegrating a relevance function that depends on both the gradient and the Laplacian would require a 5 dimensions in the context of adaptive direct volume rendering.

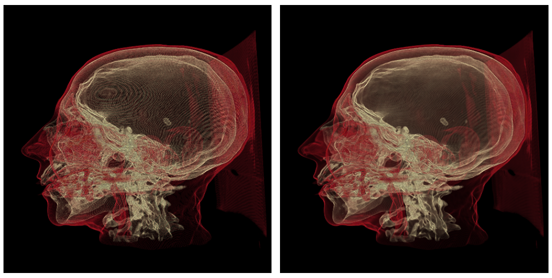

Fig. 3. Our per-pixel modulated volume rendering technique without preintegration (left) and with preintegration (right) using the same sampling rate.

## VII. Implementation

We have implemented per-pixel opacity modulation for volume rendering on graphics hardware for voxel datasets. Our hardware implementation is essentially a 2-pass raycasting-based approach using 3D textures. It is depicted in figure 4 and works as follows:

- Initially, a criterion is chosen that allows discriminating relevant voxels.
- In a **first pass**, the sum of the *relevance* values for each screen pixel is computed using the previously chosen criterion, thereby creating the relevance map. In order to sum the relevance values at each pixel, this pass does a simple GPU-based raycasting of the dataset into a frame buffer object. As the accumulation is done within the fragment shader, full accuracy is retained until the final storing phase.
- In a **second pass**, given the relevance map and the color transfer function, the final rendering is produced. The frame buffer object of the *relevance map* is bound to a texture and the dataset is rendered again as shown in Figure 5. For each pixel, the fragment program first uses the fragment coordinate information to retrieve the sum of the relevance values. Then, a GPU-based raycasting is done through the dataset. Again, full accuracy is retained throughout the raycasting thanks to the use of internal floating point register as accumulators.

## VIII. Results

We have experimented per-pixel modulated volume rendering with both DVR and additive volume rendering, using multiple relevance functions. In all tests, the binary function used is based on the opacity function of the DVR rendering, and is 1 when the opacity function is greater than zero, and 0 otherwise. All thresholds have been set at 0.5.

Figure 6 shows pictures obtained using our technique with different datasets. The first column uses the $256 \times 256 \times 128$ bonsai dataset, the second and third ones use the $256 \times 256 \times 225$ head dataset, and the fourth one the $128 \times 128 \times 128$ bucky ball dataset. Standard rendering is compared to our adaptive technique, and multiple relevance functions are tested. For all these pictures, a local lighting model was used, and a hand-tuned opacity function was created for classical DVR and additive rendering. As can be seen from these pictures, per-pixel modulated volume rendering can significantly enhance details. In particular, borders are made sharper when using a binary classification and DVR, while the other features remain visible. The relevance functions

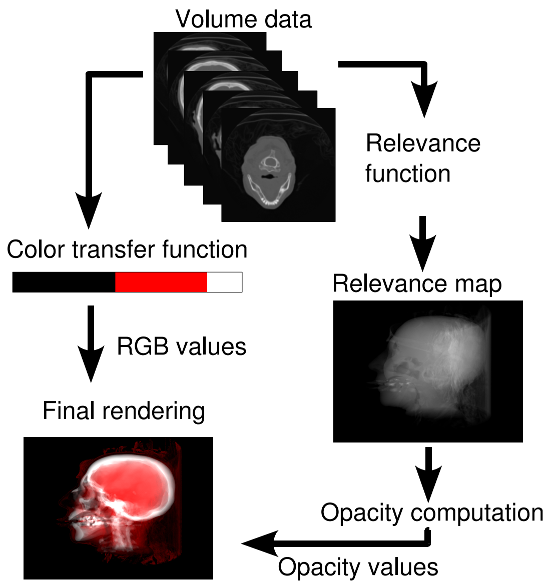

Fig. 4. Outline of our implementation.

```
uniform sampler2D transfer_function;
uniform sampler2D relevance_map;
uniform sampler3D dataset;
uniform vec3 observer;
uniform vec3 sampling_distance;
void main()
{
   vec3 pos=gl_TexCoord[0].xyz;
   float scalar1;
   float scalar2;
   vec4 result = vec4(0.0, 0.0, 0.0, 1.0);
   vec3 progression;
   vec3 steps1,steps2,steps;
   vec4 frag_color;
   vec4 data_sample;
   int nr_steps;
   int i,j;
   float xc;
   float yc;
   float sum_relevance;
   float relevance;

   // compute the progression vector inside the data
   progression.xyz=gl_TexCoord[1].xyz−observer.xyz;
   progression.xyz∗=(sampling_distance.xyz/length(progression.xyz));

   // compute the number of steps inside the data
   steps1.xyz=abs((1.0 − pos.xyz)/progression.xyz);
   steps2.xyz=abs((pos.xyz)/progression.xyz);
   if (progression.x>0.0)
      steps.x=steps1.x;
   else
      steps.x=steps2.x;
   if (progression.y>0.0)
      steps.y=steps1.y;
   else
      steps.y=steps2.y;
   if (progression.z>0.0)
      steps.z=steps1.z;
   else
      steps.z=steps2.z;

   nr_steps=int(min(steps.x,min(steps.y,steps.z)));

   xc=gl_FragCoord.x∗%f;
   yc=gl_FragCoord.y∗%f;
   // look up the sum of the relevance function at the current pixel
   sum_relevance=( float(texture2D(relevance_map, vec2(xc,yc));

   data_sample=texture3D(dataset, pos.xyz);
   scalar2=data_sample.a;
   pos.xyz+=progression.xyz;

   // raycast through the data
   for(j=0;j<256;j++)
   {
   for(i=0;i<256;i++)
   {
      scalar1=scalar2;
      data_sample=texture3D(dataset, pos.xyz);
      scalar2=data_sample.a;
      frag_color.rgba=texture2D(transfer_function, vec2(scalar2,scalar1));
      // find the relevance number for this fragment
      relevance=frag_color.a;
      // modulate the opacity of the fragment
      frag_color.rgba∗=relevance/sum_relevance;

      // accumulate using additive blending
      result.rgba+=frag_color.rgba∗vec4(relevance∗sampling_distance);

      // step into the data
      pos.xyz+=progression.xyz;

      if (j∗256+i>nr_steps) break;
   }
   if (j∗256+i>nr_steps) break;
   }

   gl_FragColor.rgba=result.rgba;

}
```

Fig. 5. Shader code for the second pass of per-pixel modulated additive volume rendering. A simple binary classification criterion is used.

allowing accurate border detection (namely the gradient value and border detection functions), although simpler from an end user point of view, allow visibility of most of the internal structures, especially when combined with additive rendering. On the other hand, the use of the second gradient alone is not sufficient to correctly determine areas of interest inside the dataset, and leads to missing most of the structures. This is exemplified by the pictures from the bonsai dataset where not all leaves are visible, or from the bucky ball dataset where this criterion fails to detect most of the borders. In the case of additive volume rendering, per-pixel modulated volume rendering also globally avoids saturating pictures, but again keeps most details visible. This is shown on the head dataset renderings, where the borders of the bones and skin are more distinguishable than with the other approaches. Notice that the bucky ball dataset is synthetic, and does not feature clear boundaries between the different densities. Nevertheless, our technique is able to achieve good visibility of the internal structure of this dataset. One can also notice from figure 6 that our adaptive method performs similarly or in some cases better than classical volume rendering with a carefully hand-tuned opacity function, while at the same time it requires less user intervention.

Figure 7 compares per-pixel modulated rendering with the classical DVR technique using two opacity transfer functions and different viewpoints, for the $379 \times 229 \times 305$ knee dataset. DVR is used with two opacity functions which are shown on top of the figure: a naive opacity function (the opacity transfer function is 0 everywhere except in the range $[5, 85]$ where it is 1) and a hand-tuned opacity function that segments the skin and bones. The per-pixel modulated technique uses the naive opacity function as a binary classification. These pictures show that even though it uses a naive transfer function, our technique is able to convey most of the bone structure, similarly to what is obtained with a hand-tuned transfer function and DVR. As can be seen on the left column, DVR with a naive transfer function does not allow clearly distinguishing the bone structure, whereas the adaptive approach with this same naive function makes this structure visible, in a way quite similar to the case of a hand-tuned transfer function. Furthermore, figure 7 also shows that viewpoint changes do not impact visibility of features with our technique, as opposed to DVR: for example the bones remain visible from all viewpoints.

Figure 2 shows the influence of applying a $3 \times 3$ Gaussian filter on the relevance map: small features that look like edges because of the adaptive opacity are successfully removed using this filter, while the rest of the features is still visible. Table I shows the performance of our technique compared to the classical volume rendering techniques. These measurements were conducted on a Pentium D 3.4GHz machine with a GeForce 7950 GT graphics card, and no preintegration was used. The classical DVR, additive and all the adaptive methods were all realized using GPU-based raycasting. Note that both DVR and additive volume rendering lead to the same performance results, and changing the relevance function results in a measured performance variation of less than 10%. These measurements show that our technique impacts the rendering interactivity only slightly, and maintain interactivity. These results also demonstrate that replacing the filtering stage by the combination of undersampling and bilinear filtering gives a significant performance gain.

| | Classical | Bilinear 2x | Filter |
|---|---|---|---|
| Head | 10.0 fps | 7.4 fps | 3.1 fps |
| Bonsai | 15.8 fps | 11.2 fps | 4.6 fps |
| Bucky | 23.2 fps | 17.1 fps | 7.0 fps |
| Knee | 6.8 fps | 5.2 fps | 2.6 fps |

TABLE I

PERFORMANCE OF THE CLASSICAL VS PER-PIXEL MODULATED VOLUME RENDERING TECHNIQUES (NOT PREINTEGRATED).

## IX. CONCLUSIONS

In the context of data exploration, non-photorealistic techniques have shown that it is possible to increase the quality of the visualization by showing more data features. In this paper, we have introduced a new simple volume rendering technique that manipulates the opacity values in a view-dependent fashion in order to ensure maximal visibility of the internal data structures. We have compared this technique to other widely used volume rendering methods and hand-tuned transfer functions, and have demonstrated its efficiency. Our technique results in better understanding of the objects features, and furthermore does not require any complex opacity function setup. For some relevance functions it does not require any user intervention, while others require minimal user setup (either a binary classification or a single threshold value). Our method also ensures good visibility of the data features independently of the viewpoint, as opposed to the classical DVR method. Moreover, since our technique has been fully implemented on graphics hardware, we achieve interactive performance, thereby making it efficient in the context of data exploration, and allowing the user to use motion and interaction in order to better understand the internal structures of the dataset.

However, we think a lot of extensions are possible. First, we would like to extend the idea of per-pixel opacity to other volume rendering algorithms, in particular when multiple techniques are used at the same time (for example, DVR and isosurfaces). Second, we would like to derive more complex opacity modification functions, for example taking the depth position of the sample into account. Finally, we would like to experiment combining our technique with an automatic segmentation technique in order to form a fully automatic volume exploration tool.

## REFERENCES

[BGKG05] BRUCKNER S., GRIMM S., KANITSAR A., GRÖLLER M. E.: Illustrative context-preserving volume rendering. In *Proceedings of EuroVis* (2005), pp. 69–76.

[CMH*01] CSEBFALVI B., MROZ L., HAUSER H., KÖNIG A., GRÖLLER M. E.: Fast visualization of object contours by non-photorealistic volume rendering. In *Eurographics* (2001), pp. 452–460.

[EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), ACM Press, pp. 9–16.

[ER00] EBERT D., RHEINGANS P.: Volume illustration: non-photorealistic rendering of volume models. In *Proceedings of the IEEE Visualization conference* (2000), pp. 195–202.

[FAT99] FUJISHIRO I., AZUMA T., TAKESHIMA Y.: Automating transfer function design for comprehensible volume rendering based on 3d field topology analysis (case study). In *VIS '99: Proceedings of the conference on Visualization '99* (Los Alamitos, CA, USA, 1999), IEEE Computer Society Press, pp. 467–470.

[FBT98] FANG S., BIDDLECOME T., TUCERYAN M.: Image-based transfer function design for data exploration in volume visualization. In *VIS '98: Proceedings of the conference on Visualization '98* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 319–326.

[GTH05] GHOSH A., TRENTACOSTE M., HEIDRICH W.: Volume rendering for high dynamic range displays. In *Volume Graphics* (2005), Kaufman A. E., Mueller K., Gröller E., Fellner D. W., Möller T., Spencer S. N., (Eds.), Eurographics Association, pp. 91–98.

[GW92] GONZALEZ R. C., WOODS R. E.: *Digital Image Processing*. Addison Wesley, 1992.

[HMBG00] HAUSER H., MROZ L., BISCHI G.-I., GRÖLLER E.: Two-level volume rendering-fusing mip and dvr. In *Proceedings of the IEEE Visualization conference* (2000), pp. 242–252.

[KD98] KINDLMANN G. L., DURKIN J. W.: Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium On Volume Visualization* (1998), pp. 79–86.

[KMM*01] KNISS J., MCCORMICK P., MCPHERSON A., AHRENS J., PAINTER J., KEAHEY A., HANSEN C.: Interactive texture-based volume rendering for large data sets. *IEEE Computer Graphics and Applications 21*, 4 (2001), 52–61.

[Kra05] KRAUS M.: Scale-invariant volume rendering. In *Proceedings of the IEEE Visualization conference* (2005), IEEE Computer Society, pp. 295–302.

[MDM07] MARCHESIN S., DISCHLER J.-M., MONGENET C.: Feature enhancement using locally adaptive volume rendering. In *IEEE/EG International Symposium on Volume Graphics* (september 2007), vol. to appear, IEEE/EG.

[ME04] MORA B., EBERT D. S.: Instant volumetric understanding with order-independent volume rendering. *Comput. Graph. Forum 23*, 3 (2004), 489–498.

[ME05] MORA B., EBERT D. S.: Low-complexity maximum intensity projection. *ACM Trans. Graph. 24*, 4 (2005), 1392–1416.

[MGK99] MROZ L., GRÖLLER E., KÖNIG A.: Real-time maximum intensity projection. In *Data Visualization*. 1999, pp. 135–144.

[RBS05] ROETTGER S., BAUER M., STAMMINGER M.: Spatialized transfer functions. In *Proceedings of EuroVis 2005* (2005), pp. 271–278.

[Sai94] SAITO T.: Real-time previewing for volume visualization. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization* (1994), pp. 99–106.

[SRR04] SUN Y., RAJWA B., ROBINSON J. P.: Adaptive image-processing technique and effective visualization of confocal microscopy images. In *Microscopy Research and Technique* (2004), pp. 156–163.

[SSN98] SATO Y., SHIRAGA N., NAKAJIMA S.: Local maximum intensity projection (lmip): A new rendering method for vascular visualization. *Journal of Computer Assisted Tomography, Vol. 22, No. 6, November-December 1998* (1998), 912–917.

[VKG04] VIOLA I., KANITSAR A., GRÖLLER M. E.: Importance-driven volume rendering. In *Proceedings of the IEEE Visualization conference* (2004), pp. 139–145.

[VKG05] VIOLA I., KANITSAR A., GRÖLLER M. E.: Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics 11*, 4 (2005), 408–418.

[YNCP05] YUAN X., NGUYEN M. X., CHEN B., PORTER D. H.: High dynamic range volume visualization. In *Proceedings of the conference on Visualization 2005* (2005), IEEE Computer Society, pp. 327–334.

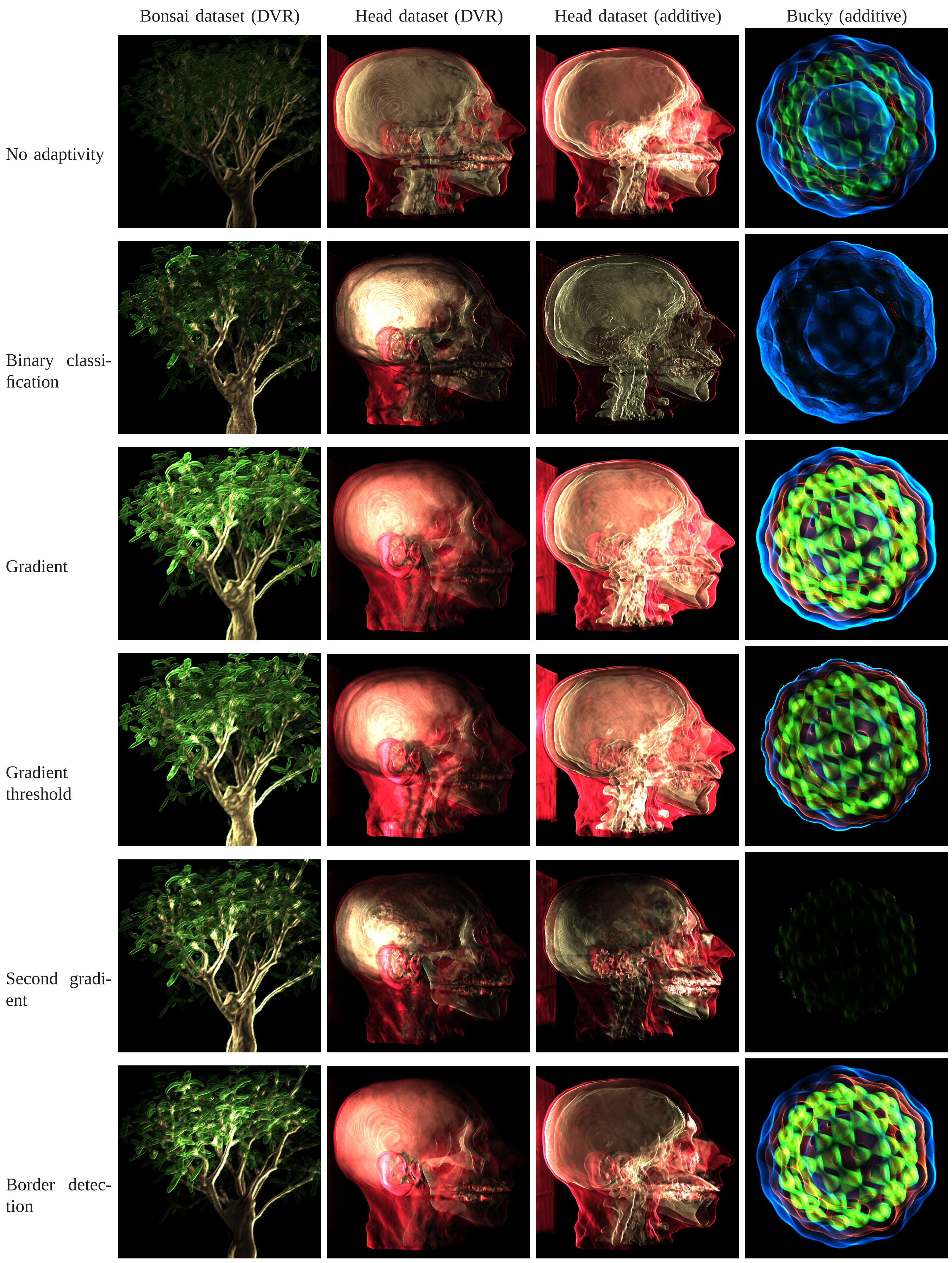

Fig. 6. Comparison of the additive and adaptive volume rendering techniques using different relevance functions.

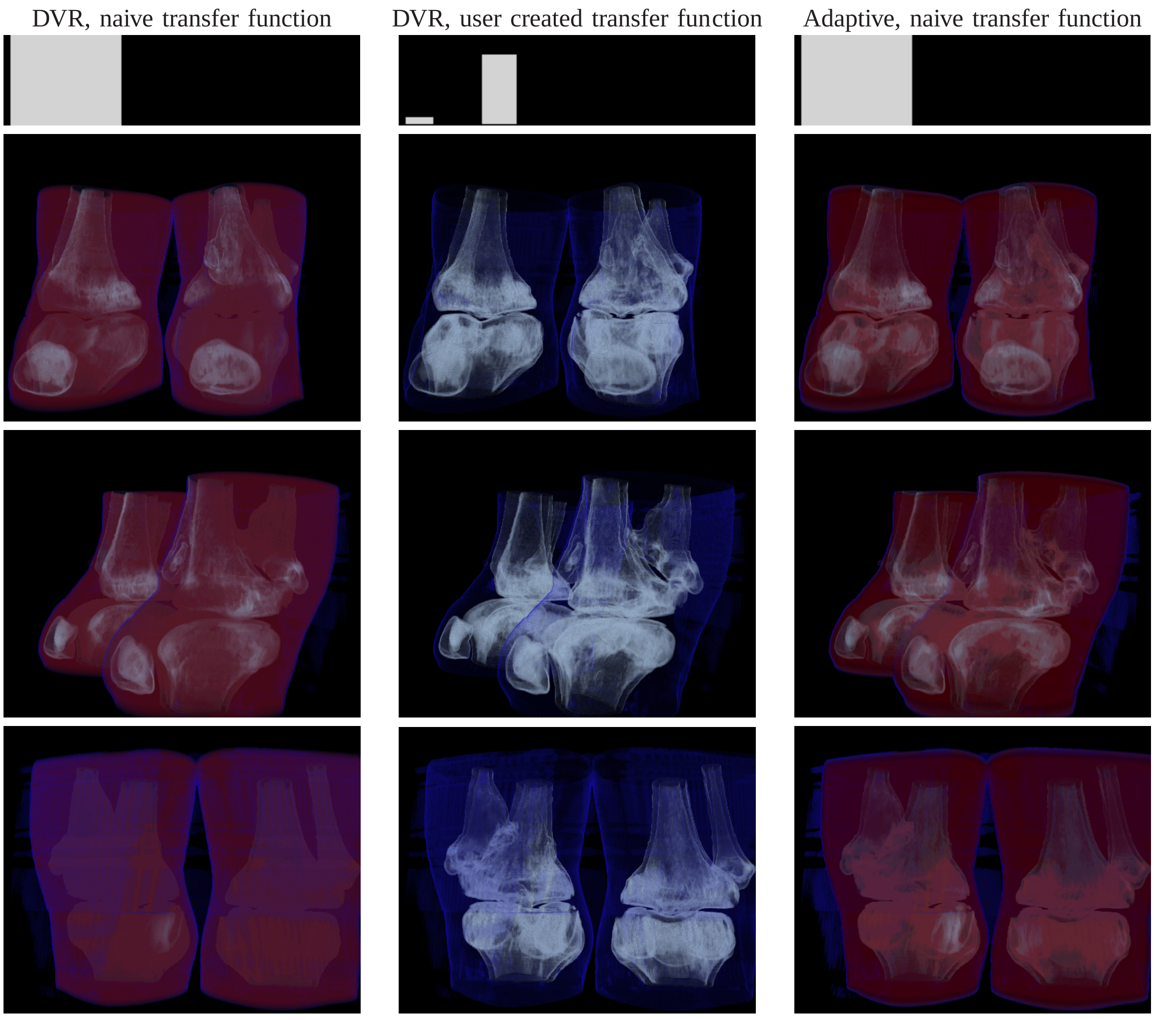
DVR, naive transfer function
DVR, user created transfer function
Adaptive, naive transfer function

Fig. 7. Comparison of DVR (naive and hand-tuned transfer functions) with per-pixel modulated volume rendering (naive binary classification).