IEEE/EG International Symposium on Volume Graphics (2010)
R. Westermann and G. Kindlmann (Editors)

Feature-Driven Ambient Occlusion for Direct Volume Rendering

Alexandre Ancel and Jean-Michel Dischler and Catherine Mongenet†

LSIIT – University of Strasbourg – Strasbourg, France

Abstract

Ambient occlusion techniques were introduced to improve data comprehension by bringing soft fading shadows. They consist in attenuating light by considering the occlusion resulting from the presence of neighboring structures. Recently introduced in volume rendering, we show that the straightforward application of ambient occlusion in direct volume rendering has its limits as rendering a multi-layer volume results in overdarkening the internal layers of the volume. This paper proposes to address the overdarkening issue by computing ambient occlusion according to the features present in the dataset. This allows us to neglect inter-occlusions between features without losing the auto-occlusions that give cues on the shape of the considered features. We use a GPU-based approach with bricking to speed up the computations of our ambient occlusion method. Results show that our approach not only improves the visual quality of images compared to classical ambient occlusion, but it is also less parameter-sensitive, thus furthermore improving usability for every-day users.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1. Introduction

In scientific 3D scalar field visualization, improving feature perception is a major focus for many researchers. In this context, shading is of primary importance, as it eases comprehension by allowing the human eye to perceive relative positions of objects in space. This issue is even more critical in direct volume rendering (DVR) since features, often interleaved in the form of multiple layers, are rendered at the same time with semi-transparency. Shading based on ambient occlusion (AO) was recently brought to volume dataset rendering to help visual analysis. The interest for such a kind of shading is multiple:

1. Phong shading is not always able to restitute the visual cues needed for good data interpretation, especially concerning the relative positions of objects. Ambient occlusion techniques bring smooth fading shadows resulting from auto-occlusions (from the self-geometry) and inter-occlusions (from neighboring geometry), that were empirically tested to be as good as the best direct illumination configuration by Langer et al. [LB00].
2. Ambient occlusion does not rely on normals / gradients and is thus less affected by potential noise in datasets.
3. It requires setting less lighting condition parameters than with direct illumination models (number of light sources, light source positions, directions, intensities, material properties: ambient and diffuse coefficients, etc.). Furthermore, with Phong lighting, the user often has to retune the parameters, when changing the point of view. Ambient occlusion is viewpoint-independent. Hence it is potentially more accessible to non-experts in visualization, like medical staff members, who just want to visualize data without painstaking settings of parameters.

Nevertheless, ambient occlusion is currently more suitable for "isosurface" rendering than it is for DVR. In fact, straightforward ambient occlusion tends to darken rendered images with multi-layered features: when large areas of data are mapped to a non-zero opacity, the layers at the center will be occluded by external layers, thus resulting in overdarkening the contribution of the corresponding features. This paper proposes a solution to address this issue. In fact, we

† {ancel, dischler, mongenet}@unistra.fr

DOI: 10.2312/VG/VG10/085-092

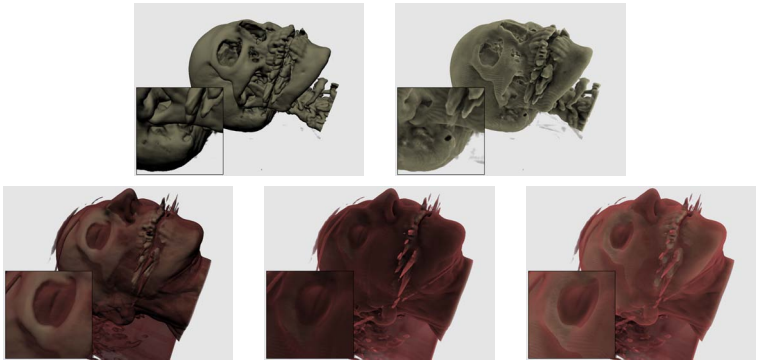

Figure 1: *DVR of a CT scan head dataset (*$256 \times 256 \times 225$*). First row compares Phong lighting (left) with Ambient Occlusion (right) on an isosurface. Second row shows multiple layers respectively rendered from left to right using Phong lighting, Ambient Occlusion and our Feature-Driven Ambient Occlusion.*

avoid overdarkening by separating the features of interest (layers) in the volume, and by computing separate ambient occlusions. These must then be recombined together, so as to preserve auto-occlusions. Figure 1 illustrates this. The first row compares ambient occlusion (right) and Phong lighting (left) applied to an isosurface. Both bring important cues in data comprehension. However, we can notice that ambient occlusion brings better comprehension on shapes and relative positions (see zoom, the little hole in the bottom part of the skull, not visible with Phong shading, see also the relative position and shape of the teeth, which is better rendered with ambient occlusion). The second row illustrates that classical ambient occlusion is less suitable for multi-layered DVR. On the left (Phong shading), the flesh and skull are better rendered than with ambient occlusion (middle), where the skull disappears below the flesh because it is too much occluded (e.g. too dark, as it receives no light). Our feature-driven ambient occlusion (right) corrects this. The skull is visible as for Phong lighting, but one gains extra visual perception of the relative position of the skull: it is clearly under the flesh. With Phong lighting (see zooms) the flesh and skull seem to form a unique surface on some locations, along with a sort of color mixing. Subsequently, it is not obvious how deep the skull is under the flesh.

As for many other methods, we assume the features (layers) of interest have already been identified. Indeed, our concern is not on the creation of transfer functions nor on techniques to identify features. Instead, our focus is on rendering and shading. We propose two contributions: First, we introduce a modified ambient occlusion formulation taking into account separate features in volume datasets (therefore we call our method feature-driven). Second, we propose a method for computing our feature-driven ambient occlusion using hardware acceleration. Globally, our approach improves the quality of images rendered with DVR and ambient occlusion-based shading while preserving the three advantages we enumerated above.

This paper is organized as follows. Section 2 presents related works in the area of ambient occlusion applied to volume rendering as well as feature-driven volume rendering methods. In Section 3, we present the classical ambient occlusion formulation in DVR. Section 4 introduces our feature-driven formulation. Section 5 presents results and performance discussions. Finally, Section 6 concludes and discusses future works.

2. Related Works

Many rendering techniques were developed aiming at improving visibility of features in datasets either through shading or feature-driven compositing as reviewed in the following sections.

2.1. Shading

Basic shading and shadowing – The most commonly used shading models in DVR are the Phong [Pho75] or Blinn [Bli77] models. These are based on gradients and direct illumination computation. However, these models are not always adequate for rendering translucent materials since they neglect light attenuation and scattering through the traversal of the volume. Both add important visual cues. Therefore, Kniss et al. [KPHE02] propose a simple and fast shadowing and light scattering technique by rendering slices in a halfway manner (between viewer and light direction). Hadwiger et al. [HKSB06] present deep shadow maps for DVR, which consists in sampling the visibility function from the light point of view.

Ambient occlusion – Light contributions from surrounding features bring important cues to the human eye as experimentally tested by Langer et al. in [LB00]. Ambient occlusion was introduced by Zhukov et al. [ZIK98] as the notion of obscurance to bring ambient illumination to objects without the cost of global illumination. As opposed

to classical shading or shadowing methods, this technique is independent of lights and viewpoint. Stewart [Ste03] is the first to work on ambient occlusion applied to volume rendering. In his method called "vicinity shading", he proposes to precompute visibility factors based on voxel densities. Ruiz et al. [RBV*08] extend Stewart's work to the obscurance model, in which the distance to the occluder is used. To reduce computations, Hernell et al. [HLY07] compute an ambient occlusion by considering only local spheres around voxels. They throw random rays along which they accumulate opacities. The factors are stored in a 3D texture used in a final rendering pass to modulate colors. Ritschel [Rit07] bases his work on spherical harmonics to compute visibility in the volume dataset. With a precomputation time reduced by using the GPU, he produces soft shadows and attenuation. Desgranges et al. [DE07] try to evaluate the most frequent distances between so called "full" voxels through histogram computation. Then, they compute an occlusion volume based on a summed area volume. The final occlusion volume is obtained by blurring previously obtained occlusion volumes. Ropinski et al. [RMSD*08] compute the interaction of neighboring voxels using histograms, built independently of any transfer function. They combine a reduced number of histograms and a transfer function in the rendering pass to interactively render ambient occlusion and color bleeding effects. Schott et al. [SPH*09] propose an image-space approach, thus making the occlusion method dependent from the viewpoint. No precomputations are required so the method is fast. However, none of these methods uses a feature-driven approach to compute ambient occlusion, which is our objective.

2.2. Feature-driven methods

Apart from shading, the final image computation is also based on a compositing scheme that combines all samples along a cast ray so as to obtain a corresponding pixel-color. Compositing is of primary importance. It can be based on a simplified physical light propagation model, or it can be feature-driven to improve data comprehension. Various non-photorealistic techniques have been recently introduced (e.g. methods that modify the physical semi-transparent compositing scheme). Ebert et al. [ER00] locally modify the rendering integral to enhance feature perception. Viola et al. [VKG04] propose to add an importance parameter to features, which establishes priorities between structures. This allows higher priority ones to be never occluded by lower priority ones. Kraus [Kra05] proposes scale-invariant volume rendering, a method that ensures color and opacity independency with regard to thickness of features. Hauser et al. [HMBG00] segment datasets into two classes, to use for each a different rendering scheme. Marchesin et al. [MDM07] also use a binary segmentation to distinguish between relevant and irrelevant features. In a two-pass scheme, they adapt transparency accordingly. More recently, Chuang et al. [CWM09b] propose to modify the classical over color compositing scheme to preserve the hue of previously identified color regions. Other techniques attempt to optimize parameters, especially transparency, like in [CWM*09a] using perception criteria. These methods often assume the volume has been previously segmented into regions that are then treated as features or structures.

Our approach is in the same spirit. We assume features are already identified, either using the color information provided by the transfer function, or by using user-defined intervals corresponding to features. In fact, the way the features are defined and identified is out of the scope of this paper. Our motivation is rather a new method to compute shading with a GPU approach similar to Hernell et al. [HLY07]. Furthermore, unlike Correa and Ma. [CM09] who use occlusion as a method for classifying data, we use feature classification to improve ambient occlusion-based shading.

3. Ambient occlusion

In classic triangle-based rendering, ambient occlusion computation is based on the analysis of objects neighboring the considered point. This is represented as the integration of a visibility function V in the hemisphere described by the solid angle ω centered on normal N at considered point p:

$$AO_p = \frac{1}{\pi}\int_{\Omega} V_{p,\omega}(N\cdot\omega)d\omega \tag{1}$$

To our knowledge, ambient occlusion in DVR can be computed either by raycasting to sample the neighborhood or by using histograms to analyze the distribution of densities around a voxel.

In this paper, we consider ambient occlusion as Hernell et al. [HLY07] did, which falls into the raycasting category. We thus consider light arriving at the center x_v of voxel v from direction l with formula:

$$I_l(x_v) = \int_a^{R_\Omega} \frac{1}{R_\Omega - a}\cdot\exp\left(-\int_a^s \tau(u)\,du\right)ds \tag{2}$$

with a an initial offset, which allows one to lower the influence of nearest neighbors, R_Ω the considered sphere radius and τ the optical depth. This equation defines an occlusion region which is a shell of thickness $R_\Omega - a$ centered on each voxel. It is approximated using M samples for a front-to-back compositing scheme with an opacity α_i, obtained via the transfer function:

$$I_l(x_v) = \sum_{m=0}^{M}\frac{1}{M}\prod_{i=0}^{m-1}(1-\alpha_i) \tag{3}$$

Obtaining the ambient occlusion factor for voxel v is finally a matter of raycasting and combining the contributions computed per-ray for a set of L rays in the following way:

$$AO(v) = \frac{1}{L}\sum_{l}^{L} I_l(x_v) \tag{4}$$

Following parameters have been identified by Ruiz et al. [RBV*08] to be important for accuracy and quality: the number of cast rays L, the distribution of these rays on the sphere, the initial offset a , the number of steps M and finally the sphere radius R_Ω. While the former parameters control shadowing accuracy and antialiasing and are quite intuitive, the latter is less intuitive and hard to define adequately as it is responsible for overocclusion if chosen too large. On the other side if chosen too small, it would result in poor shading.

In our approach, we also use a sphere and pseudo-random rays uniformly scattered using polar coordinates. But we separate features and modify the previous ambient occlusion formulation, thus reducing the influence of a bad choice of R_Ω while improving visual cues.

4. Feature-driven ambient occlusion

Our proposal consists in splitting the data into features and computing separate occlusion textures for these features. We then recombine these occlusion factors into one final 3D ambient occlusion volume. This can be summarized by the following formula that modifies the computation of $I_l(x_v)$ of equation (3):

$$I_l(x_v) = \odot_{k=1}^{n_f} I_l(x_v,k) \tag{5}$$

$$with\ I_l(x_v,k) = \sum_{m=0}^{M}\frac{1}{M}\prod_{i=0}^{m-1}(1-\alpha_{i,k}) \tag{6}$$

with $\odot$ a recombination operator, n_f the number of features and $\alpha_{i,k}$ a feature depending opacity (i denotes the sample reference along the rays and k the feature corresponding to this sample).

Our approach will be presented as follows:

- Separate occlusion volume computation for each feature;
- Recombination of occlusion textures;
- Computation optimization using bricking;
- Final rendering pass.

4.1. Per-feature ambient occlusion computation

As shown on Figure 2, our objective is to consider occlusions that are due to the feature itself while ignoring interactions with neighboring features.

The fastest and most straightforward approach to ignoring neighboring features consists in making them totally transparent during the computation. This can be done by iterating over all voxels of the dataset. For each voxel v_k, belonging to feature k, we may compute the corresponding ambient occlusion value (according to formulae (4) and (5)). This is done by setting the opacity of all neighboring voxels (i.e. those inside the sphere of radius R_Ω) that do not belong to the same feature k to 0, e.g. for all voxels v_j, $j \neq k$ we set $\alpha_{i,j} = 0$. In this case, operator $\odot$ reduces to a simple sum (since the value of $I_l(x_v, j)$ will be 0 for $j \neq k$).

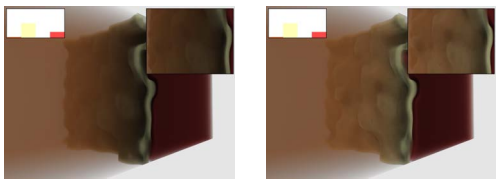

Figure 2: *Left: Traditional AO. Neighboring features can negatively interfere. Notably, the low-opacity brown feature has a darkening effect on the grey one. Right: only interactions of the feature itself are considered with our feature-driven ambient occlusion, which allows us to emphasize the perception of "hills" and "valleys" in the grey medium.*

However, we experienced that such a straightforward approach results in visual artifacts on the borders of the features. This happens when there is a connex area (feature or empty area) that has a different ambient occlusion value, yielding an abrupt change of ambient occlusion value at the feature border. Eventually, this results in artifacts during the rendering pass (see Figure 3a), because of ray sampling and trilinear interpolation. On the zoom (right column), there are some visible white dots in the dataset: these are artifacts. Applying a naive blurring or dilation operator to create smoother transitions would possibly reduce the problem of discontinuity, but it would introduce inaccuracy in terms of visibility, which might in turn result in wrong visual cues.

A more accurate approach therefore consists in computing separately n_f occlusion 3D textures T_k, $k \in [1, n_f]$. As previously done, it consists in associating a 0 opacity to all voxels that do not belong to feature k. An example is shown on the upper row of Figure 3. The visible smoothing effect, that has some resemblance with blurring, is related to the chosen radius R_Ω. The values in this case represent accurate occlusion factors, which decrease as voxels come closer to occluders (features). Because regions are now overlapping, we have to define a good recombination operator $\odot$. Some corresponding rendering results are shown on the right picture of Figure 3b and c. When computing separate ambient occlusion textures T_k, the computation cost is increased by a factor equal to the number of features n_f. So, optimizations must be applied. Both issues, recombination and optimization, will be further discussed in the following.

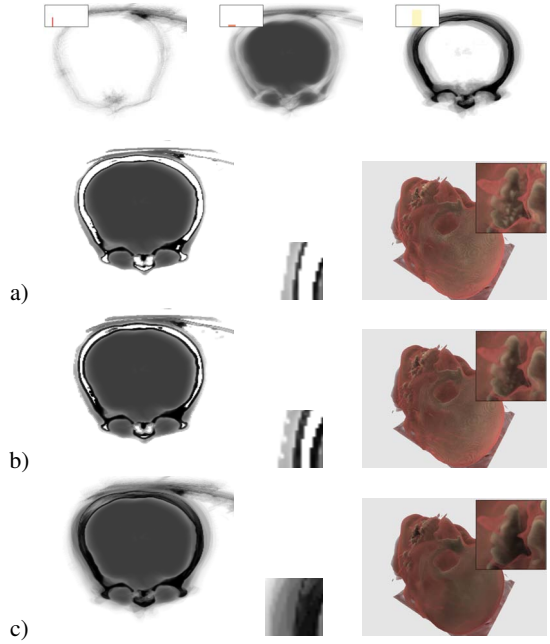

Figure 3: *Top row: AO values for 3 features identified in the dataset. From second to last row are shown different merging techniques and associated results: a) using only the corresponding feature value, b) average by considering 26 neighbors and c) merging using the maximum operator. First column shows a slice, second one a zoom and last one a rendering result (with a close up zoom).*

As usual, the computation of occlusion textures T_k is carried in a raycasting fashion using OpenGL and render-to-texture techniques. The computation is based on 3D textures, that have the same size as the original data volume, mapped onto framebuffer objects. The slices are iteratively mapped and used as rendering targets. Each shader program will then be mapped to a voxel and will be in charge of computing an ambient occlusion factor. This is done by casting L rays in randomly yet uniformly defined directions and attenuating visibility with the opacity of the sampled points. We choose an approach based on a multi-pass algorithm, in which each pass corresponds to the casting of a ray.

4.2. Recombination

Once occlusion textures T_k are computed, we have to merge them into a final ambient occlusion volume AO, while avoiding visual artifacts and maintaining visual cues. The main difficulty in this recombination part occurs on the borders of features, especially when a low occluding feature borders a high occluding one. In conjunction with ray sampling and trilinear interpolation, this results in erroneous values (see Figure 3a). A naive solution that just consists in averaging the values obtained for each individual feature (i.e. an average of all textures T_k), consequently results in inaccurate ambient occlusion values.

To obtain a correct value $AO(v)$ on voxel v, one has to consider all corresponding texture values $T_k(v)$ and potentially the neighborhood of v. We first experimented with neighborhood examination, to determine whether or not there are neighboring features or empty space. The objective was to smooth transitions. Let n_v be the size of the considered neighborhood. We found that good results are obtained by a weighted sum on a small neighborhood as follows. Two cases are considered: if all neighbors belong to the same feature k, we can straightforwardly use the corresponding value of T_k. Otherwise we compute a weighted average:

$$AO(v) = \frac{\sum_{i=1}^{n_v} \alpha_i T_{f(v^i)}(v^i)}{\sum_{i=1}^{n_v} \alpha_i}$$

with v^i the i^{th} neighbor, α_i its corresponding opacity obtained by the transfer function and $f(v^i)$ the feature that v^i belongs to. Empty spaces have an opacity value of 0, and thus will not count in the sum. An example of this kind of combination is shown on Figure 3b. The visual artifacts (white dots) are reduced, yet not totally removed.

We experimented another approach that only considers textures T_k without any neighboring information. Since averaging texture values is not an appropriate solution, we tested keeping the maximum: $AO(v) = max\{T_k(v) \mid \forall k \in [1, n_f]\}$

The bottom of Figure 3 illustrates this. It turns out that this solution gives better results than the one with neighboring information, as it reduces artifacts and improves contrast. We explain this by the fact that the ambient occlusion value becomes higher while getting closer to a given feature. As the best ambient occlusion value for a voxel should be given by the nearby feature with the greatest influence, choosing the maximum function reaches this objective. We note that using the maximum extends the influence of opaque features to neighboring less opaque ones. So, ambient occlusion values used for the less opaque features could be erroneous. However, we experienced that there are generally no visible artifacts because less opaque features are also less visible.

Notice furthermore that this latter solution is more efficient in terms of computation time than the method using neighboring information. In addition, it requires no neighborhood parameter n_v. All further results in this paper therefore use this technique.

4.3. Optimization

During the raycasting pass inside the sphere (this is the costly part of the precomputation), we iterate over every voxel and for all textures to keep continuity on the edge of features. Because we consider separately each feature, there are often large blocks of empty voxels (with respect to that

feature) that will not be used in the recombination part. To exploit those areas and raycast only non empty voxels, we preprocess the volume as follows. The 3D dataset is decomposed into equal sized bricks. Each brick is scanned to determine whether or not it contains only empty voxels. This information is stored as a boolean. Note that this has to be done with some overlapping in order to prevent introducing discontinuities at brick borders. A balance has to be found to optimize the performance: since we need to compute as many bricked volumes as there are features, thus extending preprocess computation time and memory consumption. Indeed, the time gain for computing the occlusion textures might be absorbed by the brick structure computation.

To reduce these costs, one can consider the whole dataset with all of its features. In that case, the boolean associated to a given brick is set according to whether or not it contains only totally transparent voxels. Only one bricked volume needs to be computed instead of n_f ones. However, the amount of empty space that is skipped is likely to be reduced. The choice between these two approaches will mainly depend on the datasets and on the features selected through transfer functions.

4.4. Rendering

The final rendering is done using a traditional volume raycasting algorithm implemented in GLSL. As occlusion factors are precomputed, we only need one additional texture fetch to grab the ambient occlusion factor and two additional ones for preintegrated volume rendering. This is identical to a classical ambient occlusion technique, so the final rendering times are similar.

5. Results

We conducted our work using OpenGL with a viewport of 1024x768 on an AMD opteron 2431 with a Nvidia GTX 295 graphics board (using one GPU). In the following, we first discuss visual quality of our feature-driven ambient occlusion in comparison with classical ambient occlusion and Phong shading. We then analyze performance criteria.

The datasets have been presegmented into a given set of features using an empirical segmentation. Finding an appropriate segmentation can be a difficult task. Nonetheless in some scientific fields, like medical imaging, scientists are used to work with specific data, and have therefore designed efficient transfer functions dedicated to these data. Using these transfer functions, adequate classifications can be more easily obtained.

5.1. Quality Comparison

Our quality test bench is based on two additional datasets: the orange dataset (256x256x64) and the CT knee dataset (379x229x305) (see Table 1). Based on these datasets, we rendered pictures using preintegrated DVR with: a) no shading, b) Phong shading, c) ambient occlusion, d) feature-driven ambient occlusion (using two features for each datasets).

Pictures rendered with no shading provide less cues concerning the shape of features (see for instance the orange). For the Phong shading pictures, the light was empirically set to a point of view bringing correct cues on volume information. The gradients were precomputed and packed with the volume data. But defining an optimal lighting is not a simple task. With classical ambient occlusion, no lighting parameters have to be provided, which simplifies a lot the user's task. But, we can clearly see on figure c) an overdarkening due to interleaved layers and features that negatively interfere with each others. Our result on d) shows a better compromise for the visibility of features.

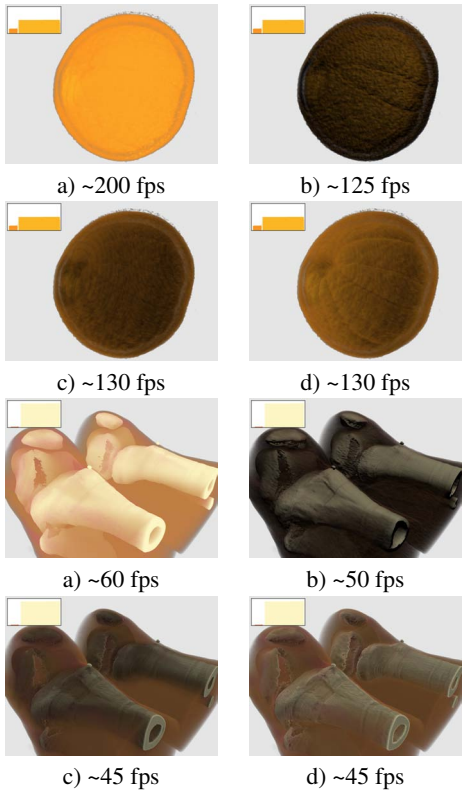

Table 1: *Orange ($256 \times 256 \times 64$) and the CT Knee dataset ($379 \times 229 \times 305$) rendered with respectively from a) to d): no shading, Phong shading, classical ambient occlusion, feature-driven ambient occlusion.*

The amount of parameters is also low. No lighting condition has to be determined. This is a serious issue in DVR as every-day users often do not want to waste time setting lighting parameters. As for classical ambient occlusion, one

main parameter is the radius of influence R_Ω. But in our case, the influence of this parameter is strongly reduced (see Figure 4), which avoids painstaking settings and experimentations. In fact, it impacts less the quality of rendering than the precomputation speed. The top row shows that classical ambient occlusion is sensitive to this parameter. Taking a too small value reduces the perception of shapes, while a high value increases overdarkening a lot. A correct value for R_Ω is not easy to find as it depends on the dataset and proximity of features. In our case (bottom row), the difference induced by R_Ω is just noticeable, so this parameter can be nearly ignored.

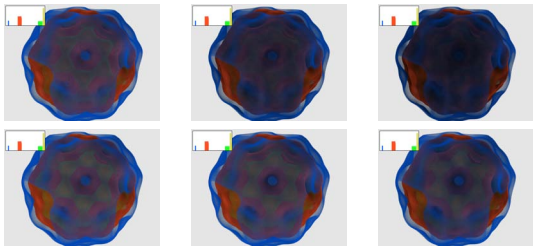

Figure 4: *Influence of R_Ω: top row for classical ambient occlusion, bottom row for feature-driven ambient occlusion. From left to right, R_Ω is 8, 16 and 32 voxels as values.*

One of our main motivation was to avoid overdarkening of features when using AO based shading in DVR. Figure 5 shows that naively increasing luminosity on classical AO (first row) does not improve the visibility of internal layers (see right picture with higher luminosity, but the skull is still not visible). To improve it, features must be considered individually. This is what our approach (second row) does.

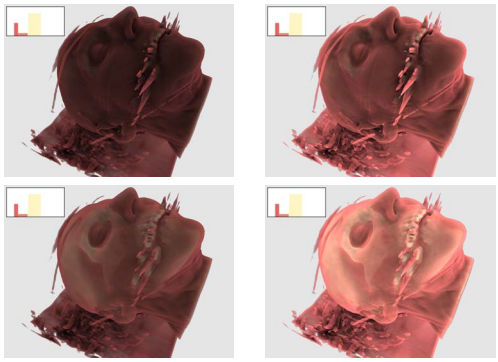

Figure 5: *Increasing brightness by multiplying color by a factor 2 on ambient occlusion (first row) does not produce the same result as with feature-driven ambient occlusion (second row)*

5.2. Performance discussion

Precomputation cost – Precomputation remains the most costly part of ambient occlusion-based shading. Table 2 illustrates this. Timings are given for three datasets in milliseconds and have been obtained without bricking optimization. AO represents classical ambient occlusion, fdAO feature-driven ambient occlusion (sum of timings for all features plus merging step).

	Bucky Ball	Human Head	CT Knee
AO	237	1339	2437
Feature 1	201	1092	1992
Feature 2	178	1092	2046
Feature 3	176	1091	X
Feature 4	173	X	X
Feature 5	179	X	X
Merge	5	7	11
fdAO	912	3282	4049

Table 2: *Precomputation time (in ms) for the following AO parameters: 16 rays launched in random directions with sphere radius 16 voxels. Datasets are the bucky ball with 5 features, the human head with 3 features, the knee dataset with 2 features.*

Table 3 reports timings measured for computing the brick structure as well as the occlusion textures using the two described optimization techniques for the head and knee datasets. We used a brick size of 8^3 voxels. The left column is for per-feature-based optimization (using as many brick structures as features). The right column is for an optimization using a single brick structure. Overlap means the number of voxel overlaps used at brick borders. Both optimizations improve preprocessing timings (compare fdAO with op. fdAO), but a global brick structure provides in both cases the highest gain. We note that optimization depends on the dataset and the amount of empty space.

	Human Head		CT Knee	
	Per-feat. bricking	Base bricking	Per-feat. bricking	Base bricking
Overlap	3	3	2	1
fdAO	3282	3282	4049	4049
Bricking	738	230	399	109
Feature 1	842	880	1733	1773
Feature 2	879	881	1476	1714
Feature 3	810	868	X	X
Merge	6	6	7	8
Op. fdAO	3275	2865	3615	3604
Gain	7	417	434	445

Table 3: *Timings (in ms) using both brick-based optimization techniques for the head and the CT Knee.*

Memory – In addition to the memory used by the volume dataset and the main transfer function, our method binds to each feature a transfer function and an ambient occlusion volume. This implies a maximum memory usage of n_f times the size of original volume data. Notice however that with certain recombination operators (mean, max, ...), we can minimize the required space to the memory needed for the computation of an ambient occlusion texture, plus an additional accumulation buffer used to store information from the previous computations.

6. Conclusions

We proposed a solution to improve ambient occlusion based shading for DVR. Our approach relies on a feature separation in volume datasets, that allows one to consider them separately. Computing ambient occlusion per feature then allows to neglect inter-occlusions between features and to only keep auto-occlusions that emphasize their shape. The visual quality of final images is clearly improved in comparison to classical ambient occlusion, while the influence of the hardly controllable sphere radius is reduced. This comes at the cost of additional memory and time consumption in the precomputation step. The final rendering step is however as in classical DVR, which allows one real-time viewpoint selection.

As for most ambient occlusion methods, the drawback of this technique is the precomputation time that hinders real-time transfer function edition. Future works could be conducted to release this constraint. Our technique could benefit from the use of a dynamic structure for speeding up the ambient occlusion textures precomputation and for further improving empty-space skipping. We could also rely on additional GPUs to share the workload, but it has to be put into balance with communication speed that could limit the process. Another track to follow could consist in using object-related parameters such as histograms [RMSD*08], surface curvature [KWTM03], etc. instead of accurate raycasting. But these methods require considering voxel neighborhoods, so the trade-off between quality and performance has to be evaluated.

References

[Bli77] BLINN J. F. : Models of light reflection for computer synthesized pictures. In *SIGGRAPH '77* (1977), pp. 192–198. 2

[CM09] CORREA C., MA K.-L. : The occlusion spectrum for volume classification and visualization. *IEEE Transactions for Visualization and Computer Graphics 15*, 6 (October 2009). 3

[CWM*09a] CHAN M.-Y., WU Y., MAK W.-H., CHEN W., QU H. : Perception-based transparency optimization for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1283–1290. 3

[CWM09b] CHUANG J., WEISKOPF D., MÖLLER T. : Hue-preserving color blending. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1275–1282. 3

[DE07] DESGRANGES P., ENGEL K. : Fast ambient occlusion for direct volume rendering. In *United States Patent Application* (2007), no. #20070013696. 3

[ER00] EBERT D., RHEINGANS P. : Volume illustration : non-photorealistic rendering of volume models. In *Proceedings of the conference on Visualization '00* (2000), pp. 195–202. 3

[HKSB06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K. : Gpu-accelerated deep shadow maps for direct volume rendering. In *GH '06 : Proceedings of the 21st symposium on Graphics hardware* (2006), ACM, pp. 49–52. 2

[HLY07] HERNELL F., LJUNG P., YNNERMAN A. : Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *Eurographics/IEEE-VGTC Symposium on Volume Graphics* (2007), pp. 1–8. 3

[HMBG00] HAUSER H., MROZ L., BISCHI G.-I., GRÖLLER M. E. : Two-level volume rendering — fusing mip and dvr. In *Proceedings of the conference on Visualization '00* (2000), pp. 211–218. 3

[KPHE02] KNISS J., PREMOZE S., HANSEN C., EBERT D. : Interactive translucent volume rendering and procedural modeling. In *Proceedings of IEEE Visualization* (2002), pp. 109–116. 2

[Kra05] KRAUS M. : Scale-invariant volume rendering. In *16th IEEE Visualization Conference* (Oct. 2005), pp. 295–302. 3

[KWTM03] KINDLMANN G., WHITAKER R., TASDIZEN T., MOLLER T. : Curvature-based transfer functions for direct volume rendering : Methods and applications. In *Proceedings of the 14th IEEE Visualization 2003* (2003), p. 67. 8

[LB00] LANGER M., BÜLTHOFF H. : Depth discrimination from shading under diffuse lighting. *Perception 29* (2000), 649–660. 1, 2

[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). 3

[Pho75] PHONG B. T. : Illumination for computer generated pictures. *Commun. ACM 18*, 6 (1975), 311–317. 2

[RBV*08] RUIZ M., BOADA I., VIOLA I., BRUCKNER S., FEIXAS M., SBERT M. : Obscurance-based volume rendering framework. In *Volume and Point-Based Graphics 2008* (aug 2008), pp. 113–120. 3, 4

[Rit07] RITSCHEL T. : Fast GPU-based Visibility Computation for Natural Illumination of Volume Data Sets. In *Short Paper Proceedings of Eurographics* (Sept. 2007), pp. 17–20. 3

[RMSD*08] ROPINSKI T., MEYER-SPRADOW J., DIEPENBROCK S., MENSMANN J., HINRICHS K. H. : Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum 27*, 2 (2008), 567–576. 3, 8

[SPH*09] SCHOTT M., PEGORARO V., HANSEN C., BOULANGER K., BOUATOUCH K. : A directional occlusion shading model for interactive direct volume rendering. In *Computer Graphics Forum* (2009), vol. 28. 3

[Ste03] STEWART A. J. : Vicinity shading for enhanced perception of volumetric data. In *Proceedings of the 14th IEEE Visualization Conference* (2003), pp. 355–362. 3

[VKG04] VIOLA I., KANITSAR A., GRÖLLER M. E. : Importance-driven volume rendering. In *Proceedings of IEEE Visualization'04* (2004), pp. 139–145. 3

[ZIK98] ZHUKOV S., IONES A., KRONIN G. : An ambient light illumination model. In *Rendering Techniques* (1998), pp. 45–56. 2