Multimodal Volume Illumination

Erik Sundén, Sathish Kottravel, Timo Ropinski[a,b,c]

Linkping University, Ulm University

[a] *erik.sunden@liu.se.*
[b] *sathish.kottravel@liu.se.*
[c] *timo.ropinski@liu.se.*

Abstract

Despite the increasing importance of multimodal volumetric data acquisition and the recent progress in advanced volume illumination, interactive multimodal volume illumination remains an open challenge. As a consequence, the perceptual benefits of advanced volume illumination algorithms cannot be exploited when visualizing multimodal data – a scenario where increased data complexity urges for improved spatial comprehension. The two main factors hindering the application of advanced volumetric illumination models to multimodal data sets, are rendering complexity and memory consumption. Solving the volume rendering integral by considering multimodal illumination increases the sampling complexity. At the same time, the increased storage requirements of multimodal data sets forbid to exploit precomputation results, which are often facilitated by advanced volume illumination algorithms to reduce the amount of per-frame computations. In this paper, we propose an interactive volume rendering approach that supports advanced illumination when visualizing multimodal volumetric data sets. The presented approach has been developed with the goal to simplify and minimize per-sample operations, while at the same time reducing the memory requirements. We will show how to exploit illumination-importance metrics, to compress and transform multimodal data sets into an illumination-aware representation, which is accessed during rendering through a novel light-space-based volume rendering algorithm. Both, data transformation and rendering algorithm, are closely intervened by taking compression errors into account during rendering. We describe and analyze the presented approach in detail, and apply it to real-world multimodal data sets from biology, medicine, meteorology and engineering.

Keywords: Volume rendering, volumetric illumination, multimodal visualization.

1. Introduction

With the advancements in sensing technology the analysis of multimodal data sets has become more common in many scientific domains. As a consequence, visualization became a key technology that enabled exploration and understanding of the relationships between different modalities. While progress in the area of advanced volume illumination has led to improved spatial comprehension by depicting the relationships between spatial structures more clearly [1, 2], interactive multimodal volume illumination is still an open challenge, and will continue to be as data set sizes increase. This is especially a limitation when dealing with multimodal acquisitions, where the relationship between the structures across modalities is one of the main benefits, and therefore should be preserved for better comprehension during exploration. When for instance performing cranial surgery, the overall structure of the brain, often best visible in MRI data, should be seen within the context of the supplying vessels, often imaged through contrast-enhanced CT acquisitions. Only when spatial relations can be understood across modalities, can the full potential of multimodal data be exploited during interactive exploration.

Several interactive volumetric illumination algorithms have been proposed [3], which increase the degree of realism in single modality visualizations, and many of these algorithms are known to improve the spatial comprehension [1, 2]. While the separation of structures through separate transfer functions can be made for different modalities, a clear spatial comprehension between structures across modalities as well as for each separate modality is not easily achievable without introducing global additions such as occlusion or illumination, designed to enhance depth cues and separate the different structures in the data. Furthermore, the incorporation of light sources, which enable the illumination, does not affect the semantics encoded in a transfer function, and the general user interface for light sources can be considered superior to the user interface of transfer functions, in aspects such as intuitiveness and usability. It should be clearly noted, that lighting does not replace the usage or importance of transfer functions, but complements it for a better separation of the structures in the data. However, the introduction of global illumination can not be made without the global knowledge of what all modalities at a specific grid point (voxel) as a context of all the occlusion from a point towards a respective light, or around the neighborhood of each voxel is at some point required.

When designing such algorithms several challenges need to be overcome, in order to provide interactivity when changing rendering parameters, such as the camera or the transfer

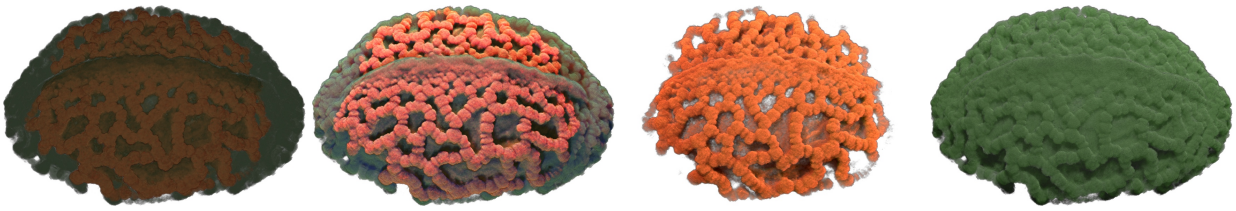
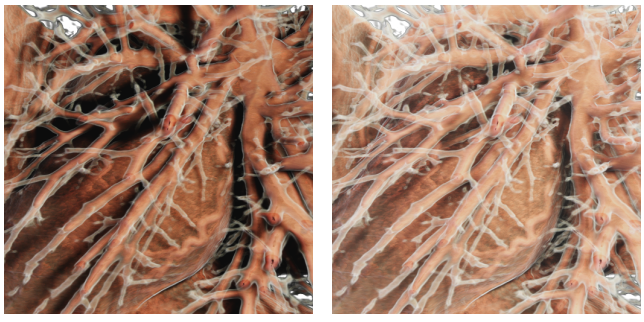


Figure 1: Multimodal volume illumination applied to a two modality raster microscopy scan of a Lily pollen ($367 \times 331 \times 135$ voxels). While standard multimodal volume rendering does not show relations across modalities (*first*), multimodal volume illumination with two light sources depicts these relations (*second*). Showing the individual modalities next to each other with advanced volumetric illumination, does also not convey their spatial relationship (*third & fourth*).

function. Due to the global nature of advanced volumetric illumination, each sampling operation is dependent on several other sampling operations, which are used to determine the influence of surrounding structures. As these additional sampling operations result in a severe performance bottleneck, a common approach has been to trade memory complexity for computing time. By using different forms of illumination caches, with varying precomputation or update times (see, for example, [4, 5, 6]) it became possible to store the relevant information in the graphics processing unit's (GPU) memory to enable advanced illumination of single volumetric data sets at interactive frame rates. Due to the increased sampling complexity of multimodal volumetric data, this scenario would benefit from precomputation in order to reduce rendering times. However, as these data sets comprise several volumes, in the future these enlarged memory footprints forbid exploitation of additional illumination caches during rendering. Thus, higher sampling complexity and memory demands go hand in hand for multimodal volume illumination, and must be tackled in an integrated manner.

In this paper, we introduce a novel approach for multimodal volume illumination which reduces the number of sampling operations as well as their complexity, while at the same time reducing the GPU memory footprint. The key idea is to limit illumination computations only to those samples which are visible to the camera as well as the light source. While this can reduce the number of sampling operations drastically, determining these samples is not easy as it is an order-dependent process. While many volume illumination algorithm work in a two pass process (first a light pass and second a camera pass) to deal with this order-dependence, our approach exploits a visibility function which can be queried in constant time in order to defer illumination computations and perform them only if necessary. This order-independent processing enables direct compositing into the camera view without requiring an intermediate illumination cache. Hence we can benefit from the advantages of deferred shading and exploit it for semi-transparent direct volume rendering of multimodal data. As a consequence our approach can incorporate multiple light sources (see Figure 1) at interactive frame rates without increasing GPU memory requirements. While shadows in single light source setups are known to improve the spatial comprehension of volume rendered scenes [1], multiple light source setups come with the benefit that shadowed regions neither appear to be pitch black, nor suffer from a low contrasts introduced by intense ambient lighting. As it can be seen in Figure 2, this leads to vivid shadows which do not cover critical structures, as for instance a tumor in an MRI scan. This observation is also in line with the findings made by Halle and Meng [7], which indicate that multiple light source setups are beneficial in visualization. Furthermore, Lee et al. [8] designed a system for calculating the optimal placement of multiple light sources for rendering of scientific datasets. Thus, to fulfill this goal we need efficient sampling and further reduced memory requirements. Therefore, we transform the multiple modalities into a single illumination-aware data structure from which we can draw samples in most cases in constant time. The error of this compression process is taken into account during rendering, which interlinks data compression and rendering algorithm in a tight manner. To solve the challenge of multimodal volume illumination at interactive frame rates we make the following contributions:

- A novel light-space volume rendering algorithm which supports multimodal volume illumination with multiple light sources by deferring illumination computations.
- An illumination-aware multimodal volume data structure which reduces the memory footprint and the complexity of sampling operations by exploiting illumination-importance metrics.
- An interlinking of the multimodal data structure and the rendering algorithm which takes into account clustering metrics during the visualization process.



(a) Single Light (b) Multiple Lights

Figure 2: Comparison between single light source setup (a) and multiple light source setup (3 point lights) (b).

2. Related Work

Three areas of research are related to our work: interactive volume illumination, multimodal visualization and volume compression.

Multimodal visualization algorithms are becoming increasingly important as multimodal imaging techniques. Many of these algorithms focus on reducing the rendering overhead when dealing with multiple modalities, and various structures for storting the spatial data and associated properties.

Chen and Tucker [9], present a method named *"Constructive Volume Geometry (CVG)"*, which does enable fusion of multiple spatial objects (i.e. volumes, geometry etc) through a multi-level tree structure defining composting operations at non-terminal nodes and spatial object structures at terminal nodes. The spatial objects contain associated spatial properties, such as proxy geometry and transfer function. As such, all spatial objects are treated as volume objects. The composting operations are applied during the ray casting process. Our approach instead utilizes a illumination-aware data-structure, which should be more suitable for GPU implementation.

In more recent work, multimodal visualization with a GPU implementation has been applied. Brecheisen et al. exploit depth peeling to generate pairs of entry and exit points, for the various combinations of modalities [10]. Lindholm et al. propose the use of binary space partitioning trees to enable the desired spatial separation [11]. Rössler et al. focus on the generation of dynamic shaders based on the degree of volume overlap such that the executed shaders only access the required volumes for a spatial area [12]. The actual compositing of the available modalities is addressed by Cai and Sakas [13]. Abellán and Tost exploit a slicing-based approach for the composition of two registered volumes [14], which can be extended to support time-varying data, which they store by exploiting run-length encoding [15]. Another GPU-based approach is proposed by Kainz et al., who propose a ray-casting based framework which supports compositing of multiple volumes together with geometry [16]. Beyer et al. describe how to use state-of-the-art GPU-based multiple volume rendering for preoperative planning [17].

While all these approaches enable interactive multimodal volume rendering, and utilized the GPU, volumetric illumination aspects are not addressed. Some methods enable the implementation of illumination approaches designed for regular grids. However, the method memory footprint has been substantially increased as an additional composition grid is required, which dimensions can become very large for spatial varying modalities. This would require significant large storage for many global volumetric illumination techniques. As the incorporation of illumination in scientific visualization is beneficial [2] for multimodal data in the same way as for single modalities, we desire a method that does handle the scalability of data and spatial varying modalities in a more efficient manor.

Interactive volume illumination has been the subject of many research efforts. When analyzing the literature it becomes apparent that a common reoccurring pattern of advanced volumetric illumination algorithms is to trade memory complexity for rendering speed [3]. One approach is to use precomputation of illumination relevant information (see, for example, [4, 5, 6, 18, 19, 20, 21]) that is made available during rendering and used to reduce the number of sampling operations.

Another common approach is to exploit an illumination cache, which is rapidly updated upon illumination changes and thus contains light source visibility or other relevant information, typically for each voxel, e. g., [22, 23, 24]. The downside of both of these approaches is the fact that GPU memory is a limited resource which is storing illumination information. Several volumetric illumination algorithms have been proposed which circumvent the additional memory requirements by applying an iterative processing of the data [25, 26, 27, 28, 29]. While these techniques reduce the additional memory requirements, the additional computational costs for sampling multiple modalities limits their interactive application. Furthermore, as iterative processing is incorporated, multiple light sources are usually not supported. Practically, in the aspect of multimodal volume rendering, it is possible to precompute the illumination for individual modalities separately and then combine these later on. However, in our experience such approach would be much more complex and have a large performance overhead and memory footprint impact. As a consequence none of the techniques discussed here have been applied to interactive multimodal volume rendering.

The only work of which we are aware that discusses multimodality in combination with volume illumination is the fMRI visualization approach presented by Nguyen et al. [30]. In contrast with our work, which incorporates full illumination interaction between potentially semi-transparent structures, their work interprets an fMRI signal as a light source to illuminate a co-registered MRI scan.

Volume compression has been facilitated by several authors in order to deal with large data sets. As compression is not the main contribution of this paper, and a thorough overview would be beyond its scope, we limit ourselves to a few key references and refer to a recent survey by Marcos et al. [31] for other sources. Some of the first approaches used wavelet-based compression which support high compression rates (see, for example, [32, 33]). However, as the reconstruction cannot be tightly coupled with the rendering, wavelet-based compression results in a performance drop during rendering. Therefore other compression schemes, which can be more tightly integrated into the rendering process, have been exploited. Ning and Hesselink were the first to apply vector quantization in the area of volume rendering to enable a block-wise compression of single modalities [34]. In a subsequent paper, they discuss optimization of rendering performance when modifying their storage pattern [35]. Schneider and Westermann have proposed a hierarchical encoding for static and time-varying data sets which also exploits a vector quantization [36], while Fout et al. also incorporate multi-variate data [37]. In this paper, we also employ a vector quantization to compress multimodal data sets whereby we exploit an illumination-aware similarity measure.

3. Deferred Volumetric Illumination

Our multimodal volume illumination approach is based on reducing sampling costs by minimizing the number of sampling operations as well as lowering the complexity of individual sampling operations. In this section we first describe how to reduce the number of sampling operations through a novel rendering method. To reduce the sampling complexity we interlink this method with a data transformation process which is discussed in Section 4.

In polygonal rendering deferred shading is often used to reduce rendering time by computing shading operations only for those parts of the scene which are actually visible [38]. This approach can reduce the number of computional operations drastically while also supporting the integration of multiple light sources without a significant performance impact. Its main disadvantage however, is its incompatibility with rendering semi-transparent structures.

As deferred shading stores visibility information together with normals and other shading relevant information in a screen space buffer, dealing with semi-transparent objects would require a layering of these buffers leading to memory overhead, especially when dealing with multimodal data sets which already have a large memory footprint. This is therefore, not a viable option. This is unfortunate as multimodal rendering could particularly benefit from reducing the number of compute operations, as sampling of multiple modalities leads to increased costs. Therefore, to limit illumination computations to only the necessary parts without requiring a costly shading layer data structure, we propose a light source projection of the visible volume parts in order to evaluate light source impact. While this typically requires an intermediate data structure, given by an illumination volume, we borrow the theory behind adaptive transparency [39] to be able to directly project illumination information into screen space in an order-independent manner. This enables us to reduce the number of sampling computations when determining light source visibility.

3.1. Multimodal Illumination Model

To see how we can reduce the number of sampling operations, we start by analyzing an extension of the standard volume rendering integral which incorporates multiple modalities. Cai and Sakas state that single- and multi-volumes are both mixed media, only with different compounds [13]. As they have identified accumulation level intermixing as a general way to intermix illumination with opacity and intensity, we focus on this intermixing strategy throughout this paper although our approach could also be combined with alternative intermixing strategies. Due to its wide acceptance we have chosen to extend the single scattering model proposed by Max [40], as well as Max and Chen [41], for our purposes. We therefore, assume that a function, $s_i(x)$, is given which samples the intensities of modality i at sampling position x. By considering this multimodal sampling function we can extend the standard volume rendering integral to incorporate multiple modalities as follows:

$$I(x_e, \vec{\omega_o}) = T(x_l, x_e) \cdot I(x_l, \vec{\omega_o}) + \int_{x_l}^{x_e} T(x', x_e) \cdot \int_0^n \tau(s_i(x')) \cdot \rho_i(x', \vec{\omega_o}) di dx' \quad (1)$$

$I(x_e, \vec{\omega_o})$ is the radiance reaching the eye point, x_e, in direction $\vec{\omega_o}$, and $T(x_0, x_1)$ describes the extinction between two points, x_0 and x_1, and n is the number of modalities minus 1. x_l is the exit point where the ray along direction $-\vec{\omega_o}$ leaves the volume, and $\tau(s_i(x'))$ describes the absorption for modality i at location x'. Taking into account multiple modalities is expressed through the second integral, which symbolizes that $\tau(s_i(x')) \cdot \rho_i(x', \vec{\omega_o})$ is dependent on the individual modalities. $\rho_j(x', \vec{\omega_o})$ incorporates the illumination computation and describes scattering at x' towards $\vec{\omega_o}$ as:

$$\rho_j(x, \vec{\omega_o}) = \int_{\vec{\Omega_i}} \phi(s_j(x), \vec{\omega_i}, \vec{\omega_o}) \cdot (T(x_l, x) \cdot I(x_l, \vec{\omega_i}) + \int_{x_l}^{x} T(x', x) \cdot (\int_0^n \tau(s_k(x')) \cdot \iota(s_k(x')) dk) dx') d\vec{\omega_i} \quad (2)$$

Here, we integrate over all incoming light directions $\vec{\Omega_i}$, whereby $\phi(s_j(x), \vec{\omega_i}, \vec{\omega_o})$ is the scattering function at x. While this scattering function can in principle be chosen based on the respective needs, throughout this paper we have applied the convention often used in volume rendering, which makes ϕ linearly dependent on the gradient magnitude. Thus, we linearly interpolate the response of the Henyey-Greenstein phase function, used for low magnitude gradients, with standard gradient-based volume shading, applied for higher magnitude gradients. The anisotropy parameter g of the phase function, as well as the gradient-based shading intensities have been manually adapted, while fixed values are also capable to obtain convincing results. $I(x_l, \vec{\omega_i})$ describes the radiance for direction $\vec{\omega_i}$ as present at x_l, the last point along that direction. $\iota(s_k(x'))$ is the radiance emitted from position x. In the context of multimodal volume rendering the extinction $T(x_0, x_1)$ must also take into account multiple modalities, and thus be redefined as:

$$T(x_0, x_1) = e^{-\int_{x_0}^{x_1} \int_0^n \tau(s_i(x')) di dx'} \quad (3)$$

Solving Equation 2 for every x' during rendering is achievable when dealing with single modalities during interactive frame rates. However, combined with the second integral, which is necessary when dealing with multiple modalities, results in an massive increase in necessary computation time and often makes a desired interactive experience impossible.

3.2. Order-Dependent Sampling

When looking into current volume illumination approaches and analyzing how the equivalents for Equation 1 and Equation 2 are resolved in single volume scenarios, it becomes clear that order-dependence of the sampling operation is a major issue as the numerical solutions for the integrals in these equations are order-dependent. We can identify two main strategies used by existing volume illumination approaches to tackle

order-dependence when applying advanced volumetric illumination. First, *iterative processing* determines a common sampling front based on the camera and light positions to process all samples in the correct order, whereby each sample is only processed once. Several such algorithms have been proposed in combination with slice-based volume rendering [26, 27, 29]. The second strategy for dealing with order-dependent sampling operations is to facilitate a *precomputation* for solving Equation 2. This precomputation is usually performed for a sufficient number of samples, and its results are made available during rendering. Data structures used for storing such precomputation results range from simple illumination volumes [22] to spherical harmonic representations [4, 42].

To enable multimodal volume illumination with multiple light sources we propose an alternative approach where the illumination calculations are not performed as an *iterative processing* or stored after a *precomputation*. Not relying on iterative processing is important as it enables us to avoid slice-based rendering methods for incorporating early ray termination and empty space skipping [43], two mandatory techniques when dealing with costly sampling operations. On the other hand, as multimodal data sets already occupy large amounts of GPU memory, storing precomputated illumination information results available for rendering is not a viable option when performing multimodal volume illumination with multiple light sources.

When considering colored shadows together with multiple light sources, such data structures become particularly big, as they need to store illumination information for all light sources and different wave lengths. In addition, unnecessary computations are performed for occluded parts of the volumes, since the precomputation of Equation 2 is performed independently of Equation 1. As, when taking into account single scattering, illumination only needs to be computed for those sample locations which are visible from the eye point, these additional costly sampling operations should be avoided when dealing with multiple modalities. Thus, if we could avoid slice-based iterative processing and precomputations, we could reduce the number of sampling operations and the GPU memory footprint at the same time. Furthermore, as we are not bound to one common sampling front, the integration of multiple light sources becomes possible.

3.3. Sample Reduction

When using volume ray-casting there are two main possibilities to reduce the number of sampling operations during rendering. First, the number of sampling operations per ray can be reduced and, second, the number of rays can be reduced. Both can be achieved by taking into account two key observations:

Observation 1 Only those samples that are visible from the camera should receive illumination.

Observation 2 Only those samples that are visible from the light source need to be illuminated.

Just by considering these two key observations, the amount of sampling operations can be reduced. We can reduce the cast view rays to those for areas not in shadow and, when considering single scattering, we can reduce the cast light rays to those reaching a visible sample position. The effect is illustrated in Figure 3 where we show the illumination of a semi-transparent volumetric structure. As opacity is accumulated during compositing, only those voxels close to the camera contribute to the visible rendering. Furthermore, only those parts of these volumetric structures, which receive illumination will be visible in the rendering, leaving parts not reached by the illumination out of the illumination computation (Observation 2). As Figure 3 shows, this limits the volumetric regions to be sampled quite drastically. Unfortunately, determining these regions again requires order-dependent sampling, as we would need to simultaneously determine light source influence and camera visibility.

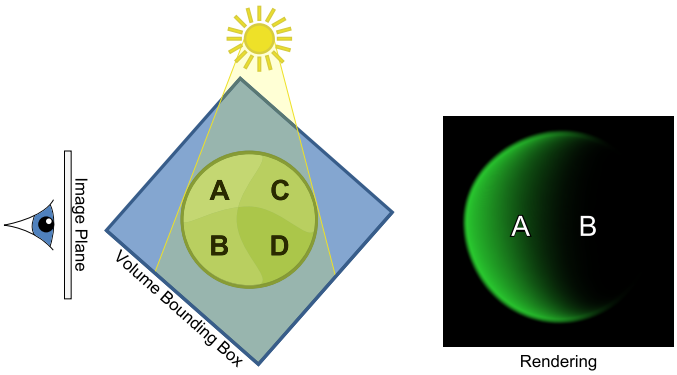


Figure 3: By taking into account two key observations, we can limit the sampling operations for camera view projection and light view influence to the areas which are visible from both, camera and light source (*A*). The other areas are either not visible (*C*), receive no illumination (*B*), or both (*D*).

To cope with this problem we borrow from a theory supporting a refactoring of the order-dependent transparency problem first introduced under the name adaptive transparency by Sintorn and Assarson [44]. They rewrite the alpha-blending equation used for compositing semi-transparent polygons such that they avoid recursion and sorting. In our case we extend this approach to volume rendering and employ it to solve the ray-integrals in Equation 1 and Equation 2. We would like to emphasize that, in contrast to the original approach, we do not apply it to surfaces, and thus limit ourselves to iso-surface representations, but instead apply it to direct volume rendering to enable true, semi-transparent volumetric compositing. To achieve this goal we take into account a visibility function for each sample along a ray. This visibility function represents the total transmittance between the sample and the camera, and enables us to composite all samples along a ray in an order-independent manner. Thus, when numerically solving the ray-integrals in Equation 1 and Equation 2, we can write this order-independent compositing as:

$$\sum_{x=x_l}^{x_e} \rho(x, x_e - x) \cdot \tau(s_i(x)) \cdot v(x_e, x), \tag{4}$$

where $v(x_e, x)$ is the visibility function for x as seen from the camera at x_e.

Sintorn and Assarson have applied adaptive transparency

to hair rendering while assuming a constant alpha value for each strand [44]. More recently Salvi et al. have rendered arbitrary transparent geometries with a similar approach [39]. They have also shown how to apply these principles to shadow mapping [45]. Instead, we exploit the dualism between semi-transparent shadow rendering and semi-transparent volumetric rendering which has first been pointed out by Enderton et al. [46].

In contrast with previous order-independent approaches used for volume rendering [47], this process enables us to achieve a correct direct volume rendering visibility composition beyond MIP and X-ray representations. Thus, we can determine camera space visibility by avoiding the storage of a shadow data structure, and we can enable multimodal volumetric illumination with multiple light sources by considering our two key observations as follows.

Observation 1. Using Equation 4 enables us to project x towards x_e order-independently, as soon as we know the visibility function $v(x_e, x)$. Thus, during rendering we can focus on the numerical solution of the light interactions as expressed in Equation 2, instead of the numerical solution of the primary rays as expressed in Equation 1, since the view ray sample order can be neglected. Thus, it becomes possible to process light sources in an order, which is independent of the camera location. In practice this means that instead of rendering the multiple modalities in an order according to the camera view, we can render the individual modalities unordered as seen from the light view, whereby we project each sample into camera space. During this process the visibility function, $v(x_e, x)$, enables us to achieve correct compositing, without storing the light contribution in an illumination cache. As a consequence, in addition to not needing to order the modalities, colored shadows and multiple light sources are supported without any extensions. By incorporating early ray termination we can also ensure that Observation 1 is taken into account, that is, only those samples which receive illumination are projected into camera space.

Observation 2. While we have limited the number of samples projected into camera space to those receiving illumination by using the visibility function, $v(x_e, x)$, we still compute illumination for all illuminated samples including those which are occluded when looking from the camera. Avoiding this sampling overhead becomes possible by, again, taking into account $v(x_e, x)$. In standard semi-transparent volume rendering the visibility of structures along a view ray decreases, as we march along the ray from the camera position. This is also the reason why early ray-termination can often be exploited, as for many view rays there is a point, x_l, beyond which the visibility is zero for all points further along on the ray. In our scenario, x_l together with the first point hit in camera space, x_f, defines for each pixel the interval of samples for which illumination computations need to be performed. Accordingly, when x_f and x_l are connected through line segments which are projected into light space, the minimal set of entry points for which light view rays have to be cast can be determined. Thus, by facilitating this projection our approach can take Observation 2 into account, that is, only those samples are illuminated which are visible from the camera.

3.4. Approximate Visibility

Knowledge about the visibility function, $v(x_e, x)$, for each camera view pixel is essential for our approach. Salvi et al. could show that an approximate visibility function, with a low number of control points is sufficient to allow transparency effects resulting from a high depth complexity [39]. Thus, in our case, we can also represent volumetric visibility by using a low number of control points, which approximate the decreasing visibility along a view ray marching through a semi-transparent volume. The number of control points used and their influence on the visual quality are discussed in subsection 6.3.

To obtain the visibility information we employ a visibility pass from which we derive an approximate visibility function, $\tilde{v}(x_e, x)$. Within this visibility pass we employ an unshaded ray-casting pass with a sampling frequency of one sample per voxel, while we use a visibility buffer to store $\tilde{v}(x_e, x)$. To further increase the accuracy of the approximate visibility we could utilize peak finding similar to [48] while still keeping the same number of control points. However, this would increase computation time. Additionally during this traversal, we store the first and the last hit point. By projecting the lines formed by these point pairs into light space, we are able to obtain optimized entry points as seen from the light source. Optionally, during this visibility pass we also have the option to extract a multimodal volume rendering of low intensity which can serve as ambient light to account for multiple scattering effects. The implementation details for our visibility pass are discussed in Section 5.

4. Illumination-Aware Data Transformation

While the proposed rendering algorithm has been designed to keep the number of sampling operations low, the individual sampling operations are still costly and all modalities need to be uploaded to the GPU. To improve this situation we interlink our rendering algorithm with a data transformation step. In this step we exploit intensity distribution similarity together with illumination-aware quality metrics through a k-means based cluster analysis. Therefore, we transform the original data set consisting of multiple modalities, into a common data structure to be able to perform a cluster analysis on the intensity vectors given by all modalities. The output of the cluster analysis is a common data space represented by a reference volume, containing a cluster ID for each voxel. This cluster ID can then be used to look up the multimodal intensity vector from a cluster lookup map, in which we store the set of associated intensities for each cluster ID. In order to keep the GPU memory footprint of this lookup map low, it is essential to find an adequate clustering which needs only a small number of clusters. Therefore, we modify the cluster analysis used to incorporate illumination-aware weighting, such that we can take into account the expected influence of each sample when applying volumetric illumination.

When applying advanced illumination algorithms in computer graphics perceptual properties of the illumination process are often exploited to simplify computation, and thus speedup

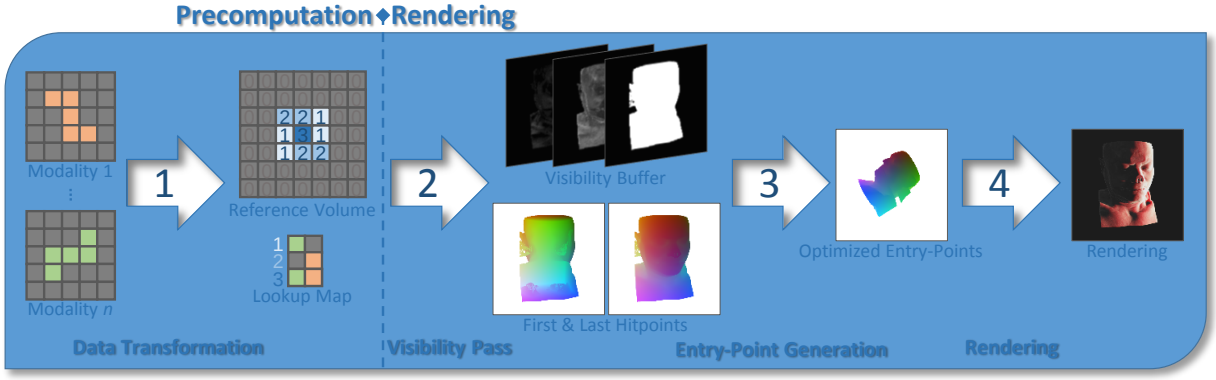


Figure 4: Our multimodal volume illumination algorithm is divided into four stages. After transforming the data (stage 1), we perform a visibility pass (stage 2), which enables order-independent sample composition. Then we derive optimized light entry points for each light source from the results of the visibility pass (stage 3). Finally, light rays are cast in light space and light samples are projected into camera space by exploiting the approximate visibility function (stage 4).

image generation. With our technique, we follow the same approach to reduce the lossiness which is inherent to cluster-based data compression. The goal is to enable perceptually plausible lighting effects while at the same time, reducing the number of clusters and thus the data which needs to be accessed during rendering. We therefore need a mechanism to input illumination-importance for individual voxels into the cluster analysis stage. To do this we have identified two types of weights, which we apply to modify the Euclidean distance metric used in our k-means based clustering approach. These weights are used to emphasize the importance of a particular modality by rewriting the Euclidean distance metric for two intensity vectors, x_i and x_j, as follows:

$$d(i,j) = \sqrt{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + \ldots + w_n|x_{in} - x_{jn}|^2},$$

where w_i are the local weights which we derive from the illumination importance of the current voxel location. While there are many possible ways to modify the clustering we would like to outline the two which we have implemented in our system.

Occlusion-based. As we use the clustered data within a visualization approach, we want to focus on those voxel's which contribute to the final image. We therefore propose the incorporation of occlusion-based weights. While local occlusion has been exploited before in different application scenarios for volume rendering [49], within our approach we exploit a single modality voxel occlusion to derive the weights for the k-means algorithm. The rationale behind this approach is the fact that, when a voxel of a certain intensity is surrounded by many voxels having the same intensity, this voxel is not visible if this intensity is mapped to a high opacity through the transfer function. Therefore, we compute the degree of occlusion within our clustering approach for each voxel and its intensity, $s_i(x)$, by considering a local neighborhood radius, r, and an intensity window parameter, δ. The occlusion of a voxel for a certain modality, i, is then derived based on the number of all voxels within the r neighborhood n_r, and the number of voxels, n_o, having an intensity lying in the range $[s_i(x) - \delta, s_i(x) + \delta]$: $occ(x) = \frac{n_o}{n_r}$. Thus, by using $1 - occ(x)$ as a local weight for each modality, we can ensure that occluded voxels are considered as less important during the clustering.

Boundary-based. The second illumination-aware weighting scheme that we have implemented is based on the rationale that viewers often focus on boundary layers within volumetric data sets. When dealing with illumination, boundaries play an important role as they map to shadow silhouettes. These play an important role in shape recognition, as they are one of the main cues for object from ground separation [50]. To benefit from this observation we map the gradient length of a voxel to a local weight as used in our Euclidean distance function. Therefore, we normalize the gradient magnitudes and map them to weights such that we obtain a higher weighting for larger gradient magnitudes. We would like to point out that this boundary-aware weighting does not mean that we limit ourselves to isosurface representations. Instead we support full semi-transparent direct volume rendering, which potentially contains boundary features as emphasized for instance through 2D transfer functions.

We would like to point out that, in addition to the discussed similarity weights, more interesting adaptations are possible. We could, for instance, use user-defined weights or derive weights based on predominant light source directions. We could also employ mutual information, which has been already exploited in the context of multimodal volume visualization [51].

5. Technical Realization

In this section, we describe the technical realization of the proposed multimodal volume illumination algorithm. The entire workflow consists of four stages. The first stage deals with the data preparation and the remaining three stages are rendering stages. As can be seen in Figure 4, after the data has been clustered (stage 1) we perform the visibility pass (stage 2). Once the visibility information has been obtained, we can generate optimized entry-points given in light space (stage 3), before we perform the actual light ray-casting using these entry-points (stage 4). In the remaining subsections we describe the technical realization of these four stages.

5.1. Data Transformation

To perform the data transformation we apply the occlusion- and boundary-based weights (see Section 4) within a CPU-based k-means clustering. Once the clustering converges we have obtained a single volumetric representation of the available modalities, as illustrated in Figure 4 (stage 1). This reference volume contains one cluster ID entry for each voxel, while each of these IDs corresponds to a row of the 2D cluster lookup map that holds the intensities of the n modalities. In this map each column corresponds to one modality, and each row to one intensity vector used during the clustering. Since this lookup map is uploaded to the GPU for rendering we have to use offsetting to not limit the number of clusters to the maximum texture dimension of the GPU. Furthermore, to enable more rapid data fetches, we always combine four modalities into a single texel consisting of four channels as these can be fetched with a single texture fetch operation. As a consequence, for up to four modalities, we can query our data in constant time of one 3D and one additional 2D texture fetch.

5.2. Visibility Pass

For the main rendering pass we need the approximate visibility function, $\tilde{v}(x_e, x)$, for each view ray. We use an A-Buffer based data structure to store these functions. This buffer, which is represented as a 2D texture array, is similar to a deep shadow map [52]. Each texture in this array consists of two channels: one stores the depth to the camera and the other the visibility at this depth. Thus, we can represent the accumulated transparency along view rays as an approximate visibility function where each texel stores one control point. By exploiting binary search this data structure enables us to query the visibility for arbitrary depth values along all view rays.

Similarly, as when generating a deep shadow map, the approximate visibility function, $\tilde{v}(x_e, x)$, can be acquired by utilizing an error threshold. As soon as the difference in accumulated opacity, computed during the visibility rendering pass, exceeds this error threshold a new control point is added to the visibility function. An optimal visibility function, for a fixed number of layers, can be determined through an iterative merging process, as suggested by Salvi et al. [39]. The merging process is initiated as soon as a new control point needs to be added to a fully occupied visibility map. In this case the stored values are analyzed and the existing control points decimated by removing the control point introducing the smallest difference, such that a new control point can be introduced. Once the visibility buffer has been obtained, the visibility of a sample can be determined by linearly interpolating the visibility of the closest two control points framing the current sample. To optimize storage requirements we neglect storing the first control point of the visibility function, as the first hit point results in the first visibility change.

A algorithm overview of the visibility pass is outlined in algorithm 1.

Algorithm 1 Pseudo-code for Visibility Pass

```
Unit: GPU (fragment stage, per-pixel)
Define ray from front to back (from camera)
for for all samples s along the ray do
    Read/classify voxel at s to color/opacity c
    if opacity in c > 0 then
        perform compositing front-to-back along ray into r
        if opacity in r > a-buffer storage threshold then
            Calculate distance to camera d (volume space)
            Calculate visibility v = 1 − r.a
             (r.a = accumulated opacity)
            if a-buffer is full then
                Merge smallest threshold distance
            end if
            Store v and d as next entry in a-buffer
            Write l_v to write cache
        end if
        Determine next position
        If "end of ray" or "accumulated opacity is
        considered opaque", then end for loop
    end if
end for
Write first and last hit points into two textures
```

5.3. Entry-Point Generation

Now that we have the approximate visibility functions, we could already employ them to perform our main light-space rendering pass. However, to do this efficiently, we employ empty-space skipping and early ray-termination to ensure that illumination computations are limited to those samples actually affecting the image. We therefore generate optimized entry-points as given in light space. As shown in Figure 3, for a correct illumination computation the area between the first and last hit points as seen from the camera, need to be included. To generate optimized light space entry-points from this information we form line segments based on each first and last hit point pair and project these line segments as seen from the light source. To achieve this we exploit a geometry shader which operates on these hit point pairs, as derived from the visibility pass.

Representing the line segments as line primitives is appropriate, as long as the light direction is not collinear with the camera direction. As this would lead to a degeneration of the line primitives to points, holes would occur in the resulting entry-point texture. To avoid this issue we represent line segments as cuboids, the depth of each cuboid corresponding to the distance between the first and the last hit points, and the width and height being calculated based on the pixel size and the distance to the light source. Furthermore, there is no specific rasterization for the cuboids. The only difference is that the cuboid will cover more pixels and as the synchronization is per-pixel based, it does not matter how many pixels that a cuboid covers in the rasterization.

To incorporate multiple light sources we simply project the instantiated geometry to multiple output targets by taking into account the projection matrix of the respective light source. We could use the same procedure to generate optimized exit-points but preliminary tests indicate that such exit-points yield no improvement in performance when we apply early ray-termination

in the main rendering pass.

5.4. Main Rendering Pass

Once we have obtained the approximate visibility functions and the optimized entry-points for the individual light sources we can initiate the main rendering pass. As light is additive, and we can use the generated visibility function for all light sources, each light pass can be performed individually and with no restriction on computation order nor any increase in memory usage. To perform the actual rendering, we treat light rays similarly to view rays of a co-located camera. A algorithm overview of the main rendering pass is outlined in algorithm 2.

Based on this camera's projection matrix we can support different light source types. The most common would be perspective projections representing point light sources, and orthogonal projections representing directional light sources, although more advanced light representations are possible. Area light sources can, for instance, be simulated using fish-eye lenses. In all cases the main rendering pass is performed by rendering the volume from each light source's position and projecting the light samples into camera space before carrying out the compositing based on the approximate visibility functions. In the following paragraphs we describe these subsequent steps.

Data access. During rendering each light ray is traversed in the same manner as view rays would be in standard volume rendering. During this traversal we access the multimodal data stored in our optimized data representation for each sample, as introduced in Section 4. This data access requires one additional texture fetch to retrieve the clustered intensities. During this dependent 2D texture fetch, we support pre- and post-classification. The former enables us to modulate the dependent 2D texture with the transfer function prior to rendering, and composite the colors for the individual modalities. Thus, we only need to fetch one color value during the dependent lookup, and can use it directly during compositing. When applying post-classified interpolation we take into account the dependent lookup for all neighbors and can, therefore, facilitate an error-aware data sampling. To achieve this we not only weight the intensities of the neighbors based on the sampling positions but also the clustering error, which we can optionally store for each voxel in our reference volume.

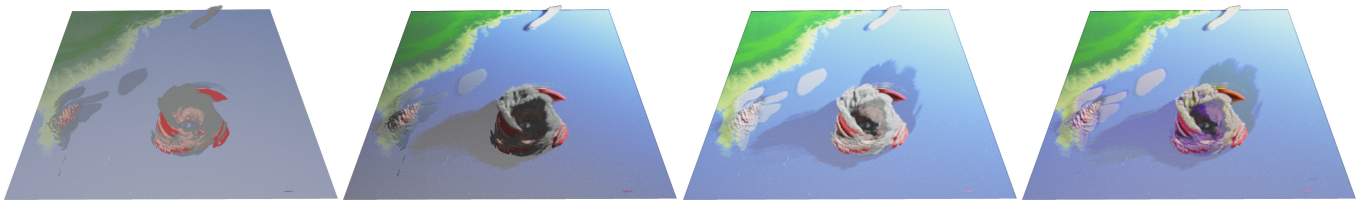
Visibility computation. Once we have fetched the data, before projecting the current light sample value into camera space, we have to perform a binary search in the visibility buffer to determine the sample's camera visibility. To enable this binary search we make sure that we always have $2^k - 1$ layers, thus we know that we always can find the middle layer by calculating $minLayer + (maxLayer - minLayer)/2$. k can be either manually specified or set automatically through an analysis of the variation in the data set. To further optimize visibility queries we assume that the depth extent is larger between the entry and the first hit point as well as between the last hit and the exit point, than between the individual layers in the visibility buffer. Performing a check against the first and last layer before initiating a binary search will, therefore further decrease the average search time.

Algorithm 2 Pseudo-code for Main Rendering Pass

```
Unit: GPU (fragment stage, per-pixel)
Define ray from front to back (from light source)
for for all samples s along the ray do
    Read/classify voxel at s to color/opacity c
    if opacity in c > 0 then
        Project s to camera space and get texture coord t
        Calculate distance d to camera (in volume space)
        Retrieve closest layers to s based on t and d
        (binary search)
        Calculate visibility v through linear interpolation
        between closest layers with respect to d
        Perform compositing front-to-back along ray into r
        Compute color/opacity as seen from camera
        Project s onto camera to determine covered pixels c
        for for all pixels p in c do
            Add contribution to pixels in resulting image
            (atomic add operation needed here)
        end for
        Determine next position or if we are done
        (early ray termination is enabled)
    end if
end for
```

Light sample projection. After a sample's visibility has been determined we can project it into camera space. Depending on the camera and light source position, as well as the light source type, different representations for the cast light rays might be optimal. To compensate for perspective projection we have to change a light sample's footprint. This compensation is very similar in nature to computing the footprint of a splat when using splat-based rendering. Accordingly, we can draw from the pool of solutions developed for volume splatting [53]. Another alternative solution could be to use cone tracing to deal with this varying footprint. However, due do the highly parallel nature of the GPU as well as our efficient multimodal data access structure, in practice we could solve the problem much more easily. By oversampling the amount of light rays, so as to not miss structures seen from the camera, we could achieve the visual results shown throughout this paper while the performance impact would be lower than when dealing with more complex implementations. We would like to explicitly point out, that the reduced multimodal data access costs play a crucial role, as the oversampling requires additional data access.

Compositing. When finally compositing samples in camera space, the order of the light space samples with respect to the camera does not matter as we employ Equation 4. The most obvious way of realizing the addition in Equation 4 would be to use an atomic add operation, accessible since OpenGL 4.0. However, the newest specification at the time, OpenGL 4.4, only supports add operations of single 32-bit signed/unsigned integers. NVIDIA further supports single 32-bit float atomic add operations, through the `GL_NV_shader_atomic_float` extension. By using this extension, we would save a software conversion from float to integer. However, we still require three channels to store the RGB values of the light samples. As mod-



(a) no illumination (b) one light source (c) two light sources (d) two colored light sources

Figure 5: Three modalities (cloud, precipitation, terrain) of the Hurricane Isabel simulation data set (500 × 500 × 100 voxels) visualized without (*a*) and with multimodal volume illumination (*b*). Adding one light source improves the spatial comprehension of the cloud layer, while a second light source can help to better show their structures (*c*). Colored light can be further used to increase the contrast (*d*).

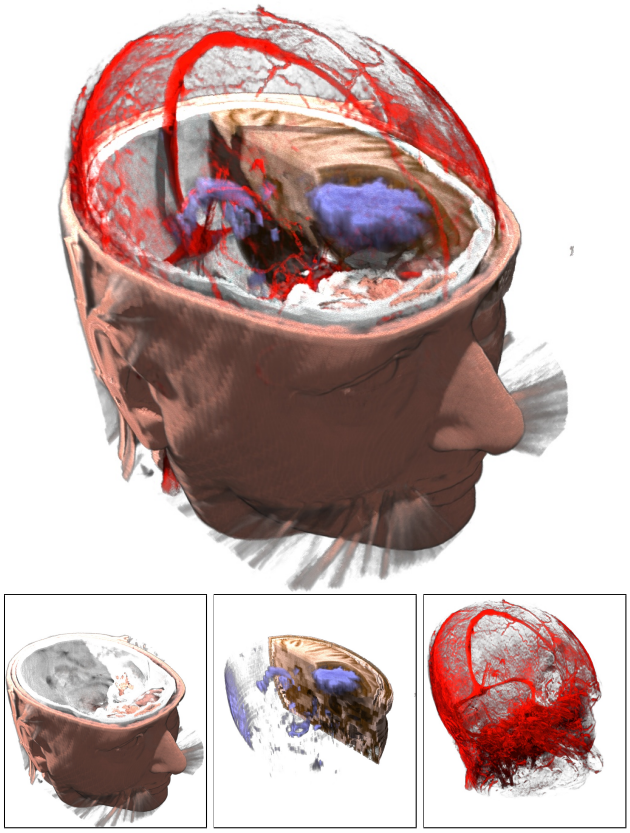


Figure 6: Multimodal volume illumination applied to three modalities (CT, MRI T1, MRI T2) acquired before a cranial intervention (*top*). Advanced illumination helps to spatially comprehend the relation between structures present in the different modalities, which cannot be conveyed through the individual modalities themselves (*bottom*).

ern software implementations of atomic add operations for multiple channels constrain the channel bit depth and have a significant performance impact [54], we perform the addition in Equation 4 for each color channel separately. The impact of the atomic add operation thus directly depends on how many other rays try to access the same pixel in the same iteration. Fortunately, as most entry-points lie at different depths along a view ray, this realization results in interactive behavior for the tested setups. Once all light samples have been projected and composited we can combine the retrieved result with the ambient light layer which we can optionally compute in the visibility pass. Thus, we are able to simulate lighting effects which omit pitch black shadows in the absence of multiple scattering.

6. Results & Discussion

To demonstrate the concepts proposed in this paper we have applied it to real-world data sets from different scientific disciplines. In this section we discuss the achieved results with respect to quality and performance.

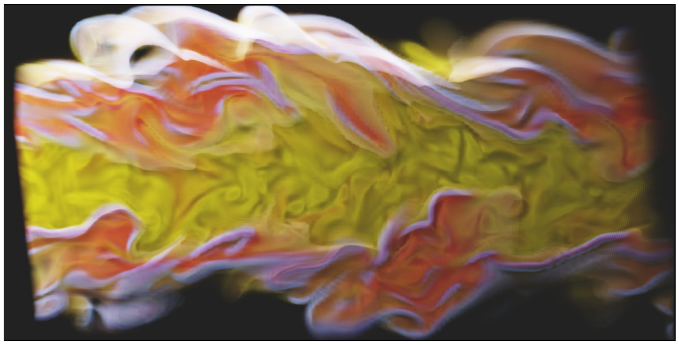


Figure 7: Application to a multi-parametric simulation of temporally-evolving plane jet flames, whereby the parameters chi, hr, mixed fraction, vorticity and Y_OH at time step 41 are shown.

6.1. Rendering Results

For the rendering results shown throughout this paper we have used pre-classification within the visibility pass, while we use post-classification during the main rendering pass. Furthermore, in all cases the maximum available layers in the visibility buffer is set to 32. Figure 1 shows the application to a two modality raster microscopy scan of a Lily pollen grain (367 × 331 × 135 voxels). As can be seen the multimodal illumination approach brings both modalities into a common frame of reference and thus allows inspection of correlations between the two modalities. To demonstrate the applicability to meterology data we have applied our technique to three modalities (cloud, precipitation, and terrain) of the Hurricane Isabel simulation data set (500 × 500 × 100 voxels) as shown in Figure 5. As can be seen the relation between the clouds and the terrain, which are specified in different modalities, only becomes apparent when applying multimodal volume illumination. Furthermore, shape perception benefits from the use of two light sources together

Data set	Modality	Original resolution	ID volume resolution	Cluster count	Data comp	Image size	Standard FPS (0xL)	Ours (L=Lights) FPS (0xL)	FPS (1xL)	FPS (2xL)
Lily-Pollen	EM	$367 \times 331 \times 135$	$400 \times 360 \times 147$	2407	1.55x	256^2	61.8	51.0	18.4	16.2
	EM	$367 \times 331 \times 135$				512^2	33.7	30.8	14.7	10.5
						1024^2	9.4	10.9	7.9	5.4
Hurri-cane	Cloud	$500 \times 500 \times 100$	$549 \times 549 \times 110$	1317	1.08x	256^2	36.2	54.8	17.5	14.8
	Terrain	$500 \times 500 \times 100$				512^2	29.7	44.2	14.6	11.7
	Precip	$500 \times 500 \times 100$				1024^2	9.9	18.7	9.56	6.78
Vis Contest	CT	$414 \times 535 \times 577$	$240 \times 303 \times 297$	5724003	4.70x	256^2	15.6	45.1	16.2	12.2
	T1 MRI	$176 \times 512 \times 512$				512^2	6.9	19.4	9.9	6.2
	T2 MRI	$544 \times 640 \times 24$				1024^2	2.4	5.9	4.0	2.8
Jet Flames	Chi	$480 \times 720 \times 120$	$366 \times 549 \times 91$	4671398	2.26x	256^2	55.1	58.1	16.6	14.2
	HR	$480 \times 720 \times 120$				512^2	30.1	28.7	12.3	8.2
	MixFrac	$480 \times 720 \times 120$				1024^2	8.4	9.2	6.1	4.3
	Vorticity	$480 \times 720 \times 120$								
	Y-OH	$480 \times 720 \times 120$								

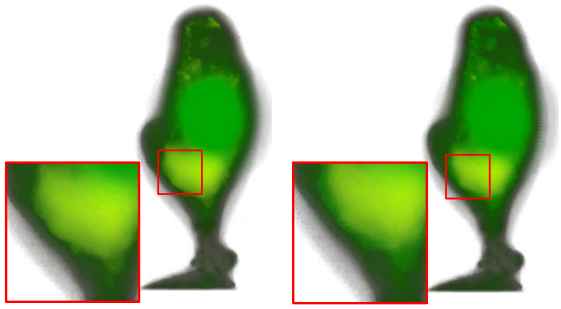
Table 1: Performance measurements and data transformation analysis of our approach.

with colored light further increases the image contrast. Medical visualization is another application area, where multimodal data frequently arises. Figure 6 shows the application of the presented approach to a multimodal data set as acquired prior to a cranial intervention. The data set contains three modalities (CT, MRI T1, MRI T2) and as can be seen in Figure 6, multimodal volume illumination enables to emphasize the spatial relations between these modalities. For instance the relation between the vessels, imaged in the contrast-enhanced MRI data set, and the skull, imaged in CT, becomes apparent. Finally, we have applied the presented approach to a combustion simulation data set ($480 \times 720 \times 120$ voxels), which contains five parameters for a simulation of temporally-evolving plane jet flames (see Figure 7).

6.2. *Performance Analysis*

When analyzing the performance of the presented approach, the performance of the data transformation stage as well as the rendering needs to be taken into account. As for all data sets used throughout this paper, clustering times were below 20 minutes when using an OpenMP version running on a standard desktop PC. GPU acceleration of the clustering algorithm would be highly beneficial in the complete work flow, however, we did not thought this acceleration would be critical for this paper and as we deemed it required much more extensive work, both in examining related work as well as implementation, we thought of this a future work. Thus, we solely focus on the rendering performance in this section.

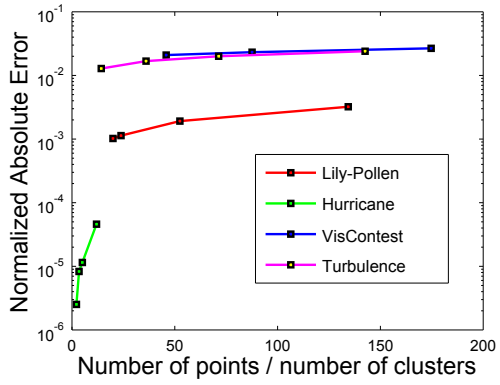
All the images shown have been generated such that the achieved frame rates give us full flexibility when changing the transfer function or other rendering parameters, when excluding the occlusion-based weighting based method which is dependent on the transfer function. To analyze the rendering performance in more detail, we have measured the average frames per second (fps) over 60 frames while performing a camera rotation. We have also varied the screen resolution and number of light sources. Table 1 shows the results achieved on a standard desktop computer, equipped with an Intel Xeon W3550 processor running at 3.07 GHz, 6 GB of RAM and an NVIDIA GeForce GTX 580 GPU.



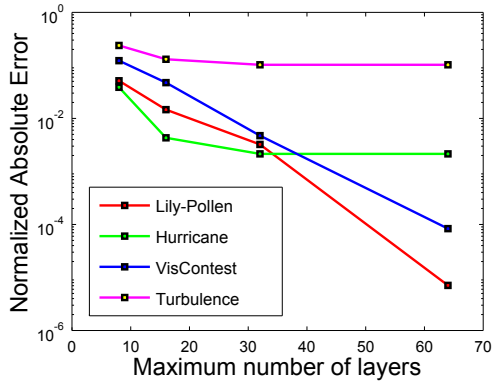
(a) Non-weighted clustering (b) Weighted clustering

Figure 8: Microscopy data set ($685 \times 961 \times 85$ voxels), which has three modalities (DAPI, FITC, TRITC) that are clustered without weighting (a) and with occlusion-based weighting assigned to one of the modalities coloured green (b). The non-weighted and weighted clustering, resulted in clusters of size 13849 and 14386 respectively.

As can be seen in the table we achieve interactive rendering times in all cases. In many cases the achieved frame rates were even higher as compared to standard multimodal volume ray-casting without illumination. The performance benefit of the optimized entry-points has, in our tests, resulted in an average performance increase of 25%. Furthermore, the approach of storing four modalities in one texel instead of in individual texels, as described in Section 4, brought an additional performance increase up to 60%. The measurements in Table 1 benefit from these optimizations.



(a) Visual error in relation to ratio between the number of clusters and the number of points.



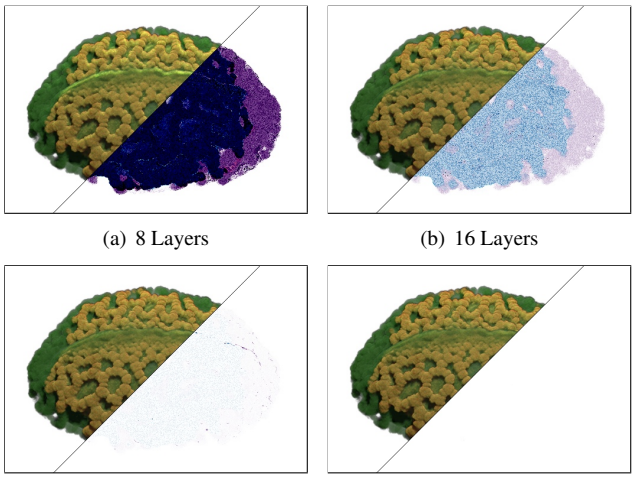
(b) Visual error in relation to maximum number of available visibility layers per pixel.

Figure 9: Comparison of the visual errors resulting from varying the number of clusters (a), and the number of layers in the visibility buffer (b).

6.3. Error Analysis

Clustering error. As can be seen in Figure 8, the weighting introduced during clustering preserved the cluster points with higher weights, as desired. There is no significant change in cluster size but rather it affected the distribution of clusters values. The visual error in relation to the number of clusters has been examined and our findings are plotted in Figure 9 (a). As can be expected the normalized absolute error increases as the ratio between the number of points and the number of clusters increases. The highest compression rate shown in this measurement has an average of 134 points per cluster, which results in a normalized absolute error of 0.0032. Thus we can conclude that a reasonably high ratio between the number of points and the number of clusters can be achieved without introducing any noticeable visual artifacts in the resulting image.

Visibility buffer error. In this evaluation we vary the number of maximum layers available per pixel, l where n is an integer from 3 to 6, and l is equal to 2 to the power of n. As can be expected the normalized absolute error decreases as the maximum number of layers available in the visibility buffer increases. As shown in the graph, allowing a minimum of 32 layers per pixel is required to keep the normalized absolute error below 0.01, which is desired to ensure that no noticeable visual artifacts occur in the resulting image. Accordingly, we have used 32 layers for all results shown throughout this paper. In Figure 10 we show the visual impact for the microscopy case, whereby we show the renderings top left and the visual errors emphasized by 100x in the bottom right of each image. As can be seen, with respect to the required number of layers in the visibility buffer, we can confirm the positive findings made by Salvi et al. [39].



(a) 8 Layers (b) 16 Layers

(c) 32 Layers (d) 64 Layers

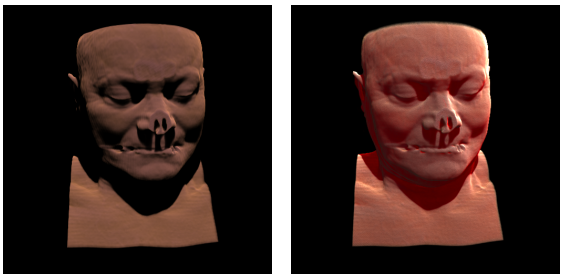
Figure 10: The visual error resulting from varying the number of visibility buffer layers. For each subfigure, top left shows the resulting rendering, and bottom right shows the visual error emphasized 100x.

6.4. Limitations

While the proposed approach works well in the tested scenarios, it has some limitations which would become apparent when dealing with non-overlapping modalities.

While the used reference volume is easy to handle, in cases where no or only a little overlap between the modalities exists it is not optimal and should be combined with other multimodal volume space partitioning techniques to obtain a better memory layout, for example, a binary space partitioning [11]. In cases where low overlap becomes an issue space partitioning data structures can be used to adapt the size of the reference volume. In a similar manner multi-resolution techniques might be used to deal with volumes having vastly different resolutions. In practice however, we could not observe such data sets, and our approach always lead to meaningful results. We believe that this is commonly the case since the goal of multimodal and multi-parametric imaging is, in general, to get a broader set of information for the same spatial positions.

Furthermore, since the optimized entry-points are generated based on camera visibility, the outlined procedure works as long as all volumes lie entirely inside the camera frustum. When using early ray-termination in the visibility pass, however, there could exist cases where the illumination contribution would suffer from light leaks. As there might be occluding scene parts behind the last hit points acquired from the visibility computation, light rays which are not affected by these parts, but contribute to the visual appearance, might appear too bright. As this is only



(a) Half Angle Slicing (b) Our approach

Figure 11: Comparison between half angle slicing (a) and our illumination method (b). Our method can also be applied to unimodal data to achieve a high quality result.

the case when the visibility function has a high variability with respect to the light angle and the penetrated medium, these considerations do not seem to play a role in practice. Nevertheless, these issues could be avoided completely either by choosing a light source position which lies in the same hemisphere as the camera, or by using the exit-points of the proxy geometry seen from the camera as last hit points for the light pass.

6.5. Discussion

The evaluation of performance and visual quality of our clustering indicates that we can decrease the data size by a large factor without noticeable decrease in image quality. While the number of clusters has no severe impact on rendering performance, varying the maximum number of layers in the visibility buffer directly impacts performance through the longer binary search. Thus, while a larger number of layers will decrease the visual error, as detailed in Subsection 6.3, a larger number will unfortunately also decrease rendering performance. Thus based on our experience, we prefer to keep the maximum number of layers per pixel at 32 in order to facilitate a small memory representation of the visibility buffer, while obtaining sufficient accuracy. Thus, the required data is equal to 32 times the screen resolution, times two 32-bit floats which store the visibility and the depth.

Jönsson et al have conducted a survey of various methods for advanced interactive volumetric illumination. They compare the well-known methods in the field with regard to scalability, performance and illumination complexity [3] . While none of the surveyed illumination methods explicitly support multimodal data, our algorithm is quite similar with respect to the supported illumination features.

We have also applied our illuminaton technique to unimodal data, and an example is shown in comparison with half angle slicing [26] in Figure 11, to show that our approach have the same good qualities compared previous work. In the same aspect as the other surveyed methods our approach supports directional and/or point light sources. On top of this we have no limitation in the number of light sources, as memory consumption can be considered low. The scaling with respect to image size is, however, less optimal as it is based on ray-casting and each layer in the visibility buffer will essentially be as large as the image resolution.

7. Conclusions & Future Work

In this paper we have introduced a novel approach for multimodal volume illumination, which supports advanced illumination effects with multiple light sources when interactively exploring multimodal volumetric data sets. We have proposed a novel light space illumination algorithm which reduces the number of sampling operations when sampling multiple modalities. Furthermore, by introducing an illumination-driven volume compression approach, we can transform multimodal volume data into a data structure which simultaneously reduces memory consumption and sampling costs. While the memory reduction enables the application to large data sets, the reduced sampling costs are crucial for all multimodal illumination scenarios as sampling is one of the limiting factors. Besides the data structure, the rendering algorithm also minimizes the GPU memory footprint which is important when dealing with multimodal volumetric data sets. By incorporating the data compression error into the proposed rendering approach we, furthermore, tightly couple the data transformation and the interactive visualization stages in order to obtain high quality rendering results. This interlinking also enables fast switching between sets of modalities. The presented approach supports interactive rendering, while enabling advanced illumination for a large number of modalities and multiple light sources. To our knowledge, this is the first approach, which allows interactive multimodal volume illumination.

In the future we would like to further extend the presented approach in various ways. First, we would like to integrate multiple scattering effects and investigate how varying light space sampling patterns effect the performance. Secondly, we would like to combine our approach with multi-resolution data structures, for example [55], to optimize the memory footprint of modalities with little overlap as well as to optimize sampling when a large difference in data resolution is present.

References

[1] Lindemann F, Ropinski T. About the Influence of Illumination Models on Image Comprehension in Direct Volume Rendering. IEEE Transactions on Visualization and Computer Graphics 2011;17(12):1922–31.

[2] Šoltészová V, Patel D, Viola I. Chromatic Shadows for Improved Perception. In: Non-Photorealistic Animation and Rendering (NPAR). 2011, p. 105–15.

[3] Jönsson D, Sundén E, Ynnerman A, Ropinski T. A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering. Computer Graphics Forum 2014;33(1):27–51.

[4] Kronander J, Jönsson D, Löw J, Ljung P, Ynnerman A, Unger J. Efficient Visibility Encoding for Dynamic Illumination in Direct Volume Rendering. IEEE Transactions on Visualization and Computer Graphics 2012;18(3):447–62.

[5] Schlegel P, Makhinya M, Pajarola R. Extinction-based Shading and Illumination in GPU Volume Ray-Casting. IEEE Transactions on Visualization and Computer Graphics 2011;17(12):1795–802.

[6] Zhang Y, Dong Z, Ma K. Real-time Volume Rendering in Dynamic Lighting Environments Using Precomputed Photon Mapping. IEEE Transactions on Visualization and Computer Graphics 2013;19(8):1317–30.

[7] Halle M, Meng J. Lightkit: A lighting system for effective visualization. In: IEEE Visualization. IEEE; 2003, p. 363–70.

[8] Lee CH, Hao X, Varshney A. Light Collages: Lighting Design for Effective Visualization. In: Proceedings of the Conference on Visualization '04. IEEE Visualization; 2004, p. 281–8.

[9] Chen M, Tucker JV. Constructive Volume Geometry. Computer Graphics Forum 2000;19:281–93.

[10] Brecheisen R, Bartroli AV, Platel B, ter Haar Romeny BM. Flexible GPU-Based Multi-Volume Ray-Casting. In: VMV. 2008, p. 303–12.

[11] Lindholm S, Ljung P, Hadwiger M, Ynnerman A. Fused Multi-Volume DVR using Binary Space Partitioning. In: Computer Graphics Forum; vol. 28. 2009, p. 847–54.

[12] Rössler F, Botchen R, Ertl T. Dynamic Shader Generation for GPU-Based Multi-Volume Ray Casting. IEEE Computer Graphics and Applications 2008;28(5):66–77.

[13] Cai W, Sakas G. Data Intermixing and Multi-volume Rendering. Computer Graphics Forum 1999;18(3):359–68.

[14] Abellán P, Tost D. Multimodal Volume Rendering with 3D Textures. Computers & Graphics 2008;32(4):412–9.

[15] Abellán P, Grau S, Tost D. Time-Varying Multimodal Volume Rendering with 3D Textures. In: GRAPP. 2008, p. 223–30.

[16] Kainz B, Grabner M, Bornik A, Hauswiesner S, Muehl J, Schmalstieg D. Ray casting of multiple volumetric datasets with polyhedral boundaries on manycore GPUs. In: ACM TOG; vol. 28. ACM; 2009, p. 152.

[17] Beyer J, Hadwiger M, Wolfsberger S, Buhler K. High-Quality Multimodal Volume Rendering for Preoperative Planning of Neurosurgical Interventions. IEEE Transactions on Visualization and Computer Graphics 2007;13(6):1696–703.

[18] Ament M, Sadlo F, Weiskopf D. Ambient volume scattering. IEEE Transactions on Visualization and Computer Graphics 2013;19(12):2936–45.

[19] Harris MJ, Lastra A. Real-Time Cloud Rendering. Computer Graphics Forum 2001;20(3):76–85.

[20] Ropinski T, Meyer-Spradow J, Diepenbrock S, Mensmann J, Hinrichs KH. Interactive Volume Rendering with Dynamic Ambient Occlusion and Color Bleeding. Computer Graphics Forum 2008;27(2):567–76.

[21] Stewart AJ. Vicinity Shading for Enhanced Perception of Volumetric Data. In: IEEE Visualization. 2003, p. 355–62.

[22] Behrens U, Ratering R. Adding Shadows to a Texture-Based Volume Renderer. In: IEEE/EG Symposium on Volume Visualization. 1998, p. 39–46.

[23] Nulkar M, Mueller K. Splatting With Shadows. In: Volume Graphics. 2001, p. 35–50.

[24] Ropinski T, Döring C, Rezk Salama C. Advanced Volume Illumination with Unconstrained Light Source Positioning. IEEE Computer Graphics and Applications 2010;30(6):29–41.

[25] Zhang C, Crawfis R. Shadows and Soft Shadows with Participating Media Using Splatting. IEEE Transactions on Visualization and Computer Graphics 2003;9(2):139–49.

[26] Kniss J, Premoze S, Hansen C, Shirley P, McPherson A. A Model for Volume Lighting and Modeling. IEEE Transactions on Visualization and Computer Graphics 2003;9(2):150–62.

[27] Schott M, Pegoraro V, Hansen C, Boulanger K, Bouatouch K. A Directional Occlusion Shading Model for Interactive Direct Volume Rendering. Computer Graphics Forum 2009;28(3):855–62.

[28] Sundén E, Ynnerman A, Ropinski T. Image Plane Sweep Volume Illumination. IEEE Transactions on Visualization and Computer Graphics 2011;17(12):2125–34.

[29] Šoltészová V, Patel D, Bruckner S, Viola I. A Multidirectional Occlusion Shading Model for Direct Volume Rendering. Computer Graphics Forum 2010;29(3):883–91.

[30] Nguyen TK, Eklund A, Ohlsson H, Hernell F, Ljung P, Forsell C, et al. Concurrent Volume Visualization of Real-Time fMRI. In: IEEE/EG Volume Graphics. 2010, p. 53–60.

[31] Balsa Rodriguez M, Gobbetti E, Iglesias Guitián J, Makhinya M, Marton F, Pajarola R, et al. A Survey of Compressed GPU-based Direct Volume Rendering. In: Eurographics State-of-the-art Report. 2013, p. 117–36.

[32] Ihm I, Park S. Wavelet-Based 3D Compression Scheme for Interactive Visualization of Very Large Volume Data. Computer Graphics Forum 1999;18(1):3–15.

[33] Muraki S. Volume Data and Wavelet Transforms. IEEE Computer Graphics and Applications 1993;13(4):50–6.

[34] Ning P, Hesselink L. Vector quantization for volume rendering. In: Workshop on Volume Visualization. 1992, p. 69–74.

[35] Ning P, Hesselink L. Fast Volume Rendering of Compressed Data. In: IEEE Visualization. 1993, p. 11–8.

[36] Schneider J, Westermann R. Compression Domain Volume Rendering. In: IEEE Visualization. 2003, p. 293–300.

[37] Fout N, Ma KL, Ahrens J. Time-varying, multivariate volume data reduction. In: ACM Symposium on Applied Computing. 2005, p. 1224–30.

[38] Deering M, Winner S, Schediwy B, Duffy C, Hunt N. The triangle processor and normal vector shader: a VLSI system for high performance graphics. SIGGRAPH Comput Graph 1988;22(4):21–30.

[39] Salvi M, Montgomery J, Lefohn A. Adaptive Transparency. In: ACM SIGGRAPH Symposium on High Performance Graphics. 2011, p. 119–26.

[40] Max N. Optical Models for Direct Volume Rendering. IEEE Transactions on Visualization and Computer Graphics 1995;1(2):99–108.

[41] Max N, Chen M. Local and Global Illumination in the Volume Rendering Integral. In: Scientific Visualization: Advanced Concepts. 2010, p. 259–74.

[42] Bista S, Zhuo J, Gullapalli R, Varshney A. Visualization of Brain Microstructure Through Spherical Harmonics Illumination of High Fidelity Spatio-Angular Fields. IEEE Transactions on Visualization and Computer Graphics 2014;20(12):2516–25.

[43] Kruger J, Westermann R. Acceleration techniques for GPU-based volume rendering. In: IEEE Visualization. 2003, p. 287–92.

[44] Sintorn E, Assarsson U. Hair self shadowing and transparency depth ordering using occupancy maps. In: Proceedings of the 2009 symposium on Interactive 3D graphics and games. I3D '09; 2009, p. 67–74.

[45] Salvi M, Vidimče K, Lauritzen A, Lefohn A. Adaptive volumetric shadow maps. In: Eurographics Symposium on Rendering. 2010, p. 1289–96.

[46] Enderton E, Sintorn E, Shirley P, Luebke D. Stochastic Transparency. In: Symposium on Interactive 3D Graphics and Games. 2010, p. 157–64.

[47] Mora B, Ebert DS. Instant volumetric understanding with order-independent volume rendering. Computer Graphics Forum 2004;23(3):489–97.

[48] Knoll A, Hijazi Y, Westerteiger R, Schott M, Hansen C, Hagen H. Volume Ray Casting with Peak Finding and Differential Sampling. IEEE Transactions on Visualization and Computer Graphics 2009;15(6):1571–8.

[49] Correa C, Ma KL. The Occlusion Spectrum for Volume Classification and Visualization. IEEE Transactions on Visualization and Computer Graphics 2009;15(6):1465–72.

[50] Dee HM, Santos PE. The Perception and Content of Cast Shadows: An Interdisciplinary Review. Spatial Cognition & Computation 2011;11(3):226–53.

[51] Haidacher M, Bruckner S, Kanitsar A, Gröller E. Information-based transfer functions for multimodal visualization. In: EG VCBM. 2008, p. 101–8.

[52] Hadwiger M, Kratz A, Sigg C, Bühler K. GPU-Accelerated Deep Shadow Maps for Direct Volume Rendering. In: ACM SIGGRAPH/EG Conference on Graphics Hardware. 2006, p. 27–8.

[53] Mueller K, Moller T, Crawlis R. Splatting without the blur. In: IEEE Visualization. 1999, p. 363–544.

[54] Crassin C, Green S. Octree-Based Sparse Voxelization Using The GPU Hardware Rasterizer. In: OpenGL Insights. CRC Press, Patrick Cozzi and Christophe Riccio; 2012, p. 303–19.

[55] 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. 2008, p. 163–70.