Vis Comput (2009) 25: 339–347
DOI 10.1007/s00371-008-0299-8

ORIGINAL ARTICLE

Interactive HDR lighting of dynamic participating media

Fernando Navarro · Diego Gutierrez · Francisco J. Serón

Published online: 23 September 2008

Abstract In this paper, we present two optimization techniques to light and render volumetric data of inhomogeneous participating media. Both are independent of the lighting model selected. We use an implementation of the ray marching algorithm to approximate the Radiance Transfer Equation. The system can calculate single scattering in time-varying isotropic participating media with the incident field being modeled as a high dynamic range (HDR) environment map. We can use dynamic lighting (with certain restrictions) and free camera movement without using any precomputations while achieving interactive frame rates.

Keywords High dynamic range · Participating media · Lighting · GPU · Real time · Volume density objects

Electronic supplementary material The online version of this article (http://dx.doi.org/10.1007/s00371-008-0299-8) contains supplementary material, which is available to authorized users.

F. Navarro (✉)
Lionhead Studios (Microsoft Games Studios), 1 Occam Court, Surrey Research Park, Guildford GU2 7YQ, UK
e-mail: fnavarrog@gmail.com

D. Gutierrez · F.J. Serón
Grupo de Informática Gráfica, Centro Politécnico Superior, Instituto de Investigación en Ingeniería de Aragón, C/ María de Luna 1, 50018, Zaragoza, Spain

D. Gutierrez
e-mail: diegog@unizar.es

F.J. Serón
e-mail: seron@unizar.es

1 Introduction

Rendering participating media is a computationally expensive process owed to the complexities of the light transport. The Radiative Transfer Equation (RTE) [4, 16], which mathematically models scattering, absorption and emission, must be solved. Usually the ray marching technique is employed, possibly combined with density estimation in a volume photon map. While these approaches yield good results, they are normally performed off-line. Alternatively, interactive rendering has relied on ad hoc methods that impose severe restrictions or force precomputation steps.

In this paper, we present two GPU-based optimizations to light and render participating media by solving a simplified version of the Radiance Transfer Equation. Interactive frame rates are achieved using a view-dependent optimization technique that reduces the number of fragments processed, and a view-independent empty space skipping technique that allows to speed up the ray marching traversals. None of them are based on predetermined illumination models. Lighting comes from an HDR environment, efficiently sampled by previously transforming it into a light constellation. Given that no heavy precomputations are needed, the system can handle time-varying participating media, single scattering including self-shadowing, free camera movement and dynamic lighting, where the incident light field can be arbitrarily rotated.

In this work, the participating media is characterized as volumetric data obtained by sweeping a laser sheet through a volume of real smoke using the method of Hawkins, Einarsson and Debevec [14]. However, since it is general enough, it has also been applied to data generated using numerical simulation.

The rest of the paper has the following structure. In Sect. 2, we briefly present an overview of traditional vol-

ume rendering techniques and recent GPU-based implementations and optimizations. Section 3 describes the lighting model based on the Radiative Transfer Equation. Specific implementation details are covered in Sect. 4. Finally, the last two sections present the results and a brief discussion of the limitations and future directions of our research.

2 Previous work

Light transport in participating media has been a topic of intense research. An important part of the efforts have been focused on different methods for solving the radiative transfer equation (RTE) [4, 16].

A number of analytic solutions have been found. They are frequently based on strong simplifications like assuming homogeneous [2] and optically thick media [29] or infinitely distant light sources [28]. By sampling a number of points inside the media, stochastic methods like Monte Carlo path tracing [40], bidirectional path tracing [24] or photon mapping [17] can generate high quality results. Numerical simulation approaches are based on evaluating the radiance transport integral [18, 37, 41]. The two latter can model complex phenomena as multiple scattering in inhomogeneous media at the cost of simulation times in the order of hours.

Overall performance can be increased by previously storing light or media dependent magnitudes in a way that can be efficiently evaluated at render time. By using precomputed radiance transfer, Sloan et al. [39] handle complex lighting interactions like soft shadows and scattering. Other methods use precalculated forward multiple scattering [13], pre-integrated lighting [45], spherical harmonics for Fourier volume rendering [9], precalculated transmission integrals [15] or reuse light scattering paths in a particle system [43]. Recently, Zhou has used a set of radial basis functions to convert the media into an equivalent sparse representation [48]. While these methods allow complex light interactions, interactivity is limited to static viewports, specific transfer functions or prefixed light types and positions. Furthermore, rendering data sets containing a sequence of frames forces precomputation on each of them.

Recent advances in graphics hardware has allowed the implementation of interactive and real-time volume rendering methods. Since analytic solutions rely on evaluating a closed set of mathematical formulas, they can be efficiently implemented using GPUs [42, 47]. Methods without precomputation are usually limited to non-physically based lighting ray marching [23] or are based on 2D texturing [8].

A number of acceleration techniques designed to improve the efficiency of media sampling have been described. Kruger and Westermann [23] state an efficient streaming model over GPUs and implemented different acceleration techniques for volume ray-casting. Early ray termination is based on space classification structures and compression techniques: Guthe et al. use hierarchical wavelets [12], LaMar, Haman and Joy [25] rely on an octree; Westermann and Senenich [46] use CPU precalculated 2D textures. Li, Mueller and Kaufman's [26] empty space skipping technique avoids sampling areas that contain no data. Spatial and temporal data coherence is exploited in [44] and [21]. All these techniques set up acceleration structures in a precomputation step.

The purpose of our work, efficiently rendering time dependent participating media without significant precomputation, limits the applicability of previous methods. Our solution relies on a physically based light transfer model enabling to calculate single scattering with self-shadowing. As this method is not efficient enough, two optimizations, based on the media rendered and the camera position, provide the extra acceleration needed. Moreover, they do not use any pre-computed shading nor depend on the illumination model or lighting environment. Acceleration structures and lighting are calculated on the fly, therefore position and orientation of the environment and camera can be interactively changed.

3 Lighting model

The use of high dynamic range maps has become one of the most popular methods to represent lighting environments. Several image-based algorithms have been developed, based on well-known Monte Carlo methods [19]. These are usually bad shaped for GPU implementation due to intensive sampling of the irradiance map. Different techniques can convert these maps into spherical harmonics [34], light constellations [5, 27] or wavelets [30]. An environment represented with these methods can be efficiently used at the cost of using similar but not equal lighting conditions. All but wavelet-based methods are limited to low frequency environments. For our purposes, this is a mild limitation as the light transport involved in participating media tends to blur the effects of high frequency lights [30].

Light constellation methods replace areas of the irradiance image by simple light emitters. Cohen and Debevec's [5] median-cut technique extracts a set of equal energy lights, each of them representing areas of different sizes. Alternatively, K-means clustering [27] generates lights with different energy but with more regular spatial distribution. Other works have generated light constellations in time slots that are compatible with real-time implementations [22, 33]. Conversion to a finite number of sources has the benefit of imposing a deterministic sampling without the storage and complex representations of traditional

deterministic techniques. Complementarily, noise, principal problem of stochastic methods and origin of flickering artifacts, is eliminated and a desirable coherence along time is added.

Once the light field has been obtained, a solution to the light transport in participating media needs to be provided. Using the integral form of the RTE [4, 16], the radiance L at a point x in direction $\overline{\omega}$ can be written as:

$$\begin{aligned} L(x,\overline{\omega}) &= \tau(x_0,x)L(x_0,\overline{\omega}) \\ &+ \int_{x_0}^{x} \tau(u,x)\kappa_t(u)\big(1-\Omega(u)\big)L_e(u,\overline{\omega})\,du \\ &+ \int_{x_0}^{x} \tau(u,x)\kappa_t(u)\Omega(u) \\ &\times \int_S L(u,\overline{\omega}_i)\rho(u,\overline{\omega},\overline{\omega}_i)\,d\sigma_{\omega i}\,du \end{aligned} \tag{1}$$

where x and x_0 are points in the $\Re^3$ space, κ_t denotes the extinction coefficient and is the sum of the scattering κ_s and absorption κ_a coefficients, $\Omega(x) = \kappa_s(x)/\kappa_t(x)$ is the scattering albedo and $L_e(x,\overline{\omega})$ is the radiance emitted by the media at point x in direction $\overline{\omega}$. S represents the set of directions on the sphere around point x, and the normalized phase function $\rho(x,\overline{\omega},\overline{\omega}_i)$ determines the amount of the incident light arriving at x in direction $\overline{\omega}$ that is scattered in direction $\overline{\omega}_i$. $\tau(x_0,x)$, the transmittance factor along the segment from x_0 to x, is expressed as

$$\tau(x_0,x) = e^{-\int_{x_0}^{x}\kappa_t(\xi)d\xi} \tag{2}$$

The cost of computing (1) can be reduced by using a single scattering model. This simplification narrows the applicability of the model to low albedo or optically thin media. In the same direction, media is supposed to have an isotropic phase function $\rho(x,\overline{\omega},\overline{\omega}_i) = \frac{1}{4\pi}$. Note that participating media such as fog, smoke or clouds show a forward scattering behavior that is not compatible with this assumption. This is a limitation that can be easily overcome using models that efficiently represent anisotropic phase functions [1]. Finally, as common participating media is usually non-emissive, we can safely discard the emission term $L_e(x,\overline{\omega})$.

With the previous assumptions and given that the lighting environment has been replaced by a finite set of N light emitters, (1) can be transformed to:

$$\begin{aligned} L(x,\overline{\omega}) &= \tau(x_0,x)L(x_0,\overline{\omega}) \\ &+ \int_{x_0}^{x} \tau(u,x)\kappa_t(u)\frac{\Omega(u)}{4\pi}\sum_{n=1}^{N} L_{ri}(u,\overline{\omega}_n)\,du \end{aligned} \tag{3}$$

where $L_{ri}(u,\overline{\omega}_n)$ is the reduced incidence radiance [3] arriving at u in the direction of the nth light emitter.

Finally, using small integration steps $\Delta x = x - x_0$, the integral equations (3) and (2) can be discretized as

$$\begin{aligned} L(x,\overline{\omega}) &\approx \tau(x_0,x)L(x_0,\overline{\omega}) \\ &+ \tau(x_0,x)\kappa_t(x_0)\frac{\Omega(x_0)}{4\pi}\sum_{n=1}^{N} L_{ri}(x_0,\overline{\omega}_n)\Delta x \end{aligned} \tag{4}$$

$$\tau(x_0,x) = \tau(x_0,x_0+\Delta x) \approx e^{-\kappa_t(x_0)\cdot\Delta x} \tag{5}$$

This equation calculates the radiance arriving at a point x as a function of the radiance and media properties at a near point x_0. If *eye* is the eye position and $\overline{\omega}_{ij}$ the direction from a pixel with coordinates (i,j) to the observer, then $L(eye,\overline{\omega}_{ij})$ will determine the radiance arriving at the observer from each of the image pixels. Given that the initial conditions at the furthest position in the media can be established, this formulation can be mapped to traditional back-to-front ray marching in Δx increments. In the following sections we will explain the details on how (4) can be solved following this schema.

4 Implementation details

CPU-based ray marching algorithms have been commonly used to display volumetric media. However, GPU-based implementations are not so frequent, in part due to restrictions in the logic complexity and number of instructions in fragment programs. To overcome these limitations we use a multi-pass approach. In this section we describe the details of our OpenGL and Cg [31] implementation.

The general overview of our system is outlined as follows: Initially the volumetric data, a 3D array of floating point voxels containing the density of the media, is loaded into GPU memory. The data is analyzed and a view-independent distance map to accelerate media traversals is computed. Taking into account camera position, a different pass builds a validity mask that helps eliminating computations on areas that are to be empty in the final image. Successive passes solve (4) as a back-to-front ray marching. Independently, every time the lighting environment is changed, an equivalent light constellation is applied. For a given HDR image, this set of lights is reused for different frames and eventually rotated. The previous steps are performed using floating point arithmetic and generate a HDR image, so a final tone mapping pass is needed.

4.1 Distance based optimization

Texture fetching operations are one of the bottlenecks of existing GPU architectures. The performance of an algorithm is drastically reduced when high number of memory accesses are executed. Since the ray marching algorithm is

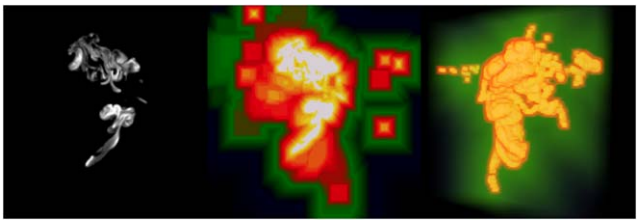

Fig. 1 Distance map optimization: *Left*: One of the 256 slices of smoke from a volume data set. *Middle*: Distance map corresponding to this slice. Brighter regions correspond to lower distances. *Right*: Volumetric representation of the whole distance map

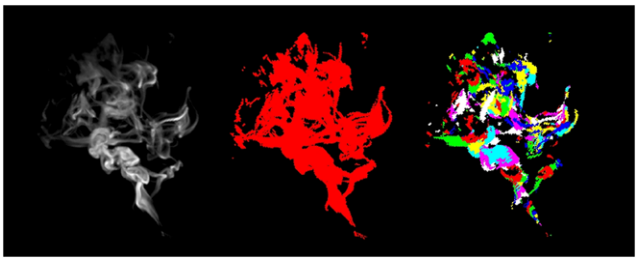

Fig. 2 Values generated by pixel validity mask pass. From *left* to *right*: Smoke volume being rendered, pixel validity mask showing *red* pixels for those fragments containing valid information and color representation of index of the first slice containing non-transparent data ($iFirst$)

based on intensively sampling the media, brute force implementations fail to produce interactive frame rates. In this section, we describe a novel use of an optimization technique that, based on a distance map, reduces the amount of these instructions by skipping volume areas that contain no data.

The distance map, represented in Fig. 1, determines, for each position x inside the data volume, the distance $\delta(x)$ to the nearest non-empty voxel. Other authors have used similar approaches to calculate surface displacement [7] or to compute geometry in shell space [36]. In our case, we apply this technique to optimize lighting calculations by accelerating primary and shadow ray traversals. At runtime, for every sampling point, the next visited position is determined by jumping a distance given by $\delta(x)$ in the sampling direction. Note that no directional information is stored, so each $\delta(x)$ identifies the radius of a sphere centered at x, which contains only empty data. This is a conservative optimization, but given the memory constraints of the hardware framework and since we have observed important speed-ups (see Sect. 5), it seems to be a good tradeoff.

In order to reduce the number of data fetches, the original data and distance maps are stored in the red and green channels of the same 3D texture. This optimization doubles the memory footprint, but allows retrieving both values at no extra cost.

Without loss of generality, our implementation calculates the distance map as a preprocess. Since existing algorithms can calculate it in real time [6], the validity of our results remains the same.

4.2 Validity mask

In this section we explain an optimization method, implemented as a new pass, that uses the data being rendered and the camera settings to determine which areas of the final image are to be empty. Since it does not perform lighting calculations, it can be efficiently applied.

Initially, using the method described in [23], each pixel with coordinates (i, j) is assigned a pixel to eye direction $\overline{\omega}_{ij}$. A traditional slabs test [20] is used to determine P_{near} and P_{far}, the nearest and furthest intersections with media's bounding box. Pixels where the ray does not define intersections are discarded for further computations.

Our sampling strategy is based on using shells or spherical slices centered at camera position [11]. Although this technique eliminates the need of perspective correction, it has been limited to cases where extreme viewports were used [8] due to the cost of building proxy geometry. We determine the position of each sampling point by intersecting a ray with an implicit sphere, so we do not incur in the overhead of creating any geometry. Slices are $i\Delta slice$ units away from the camera position, with i being the index of the slice and $\Delta slice$ the distance between them. To prevent aliasing artifacts $\Delta slice$ is chosen to be smaller than half the side of a voxel. However, given the data varies smoothly, is trilinearly filtered by the hardware and the mask is not directly visible, this value can be relaxed and more sparse sampling can be used. We have found that up to one slice every five voxels yields good results.

$$iNear = ceil\big(|P_{near} - eye|/\Delta slice\big) \tag{6}$$

$$iFar = floor\big(|P_{far} - eye|/\Delta slice\big) \tag{7}$$

In order to determine which ray directions will generate empty image pixels, a ray marching is performed in back–to-front order. Both the indexes of the nearest and furthest slices inside the bounding box are calculated using (6) and (7). A direction will be tagged as containing data as soon as density values above a given threshold are found during the traversal from $iFar$ to $iNear$. The index of this slice will be noted as $iFirst$. To improve performance, this traversal is accelerated using the technique explained in Sect. 4.1.

This pass outputs a half floating point RGBA image composed of each pixel's normalized ray direction $\overline{\omega}_{ij}$ (RGB components) and $iFirst$ index (alpha component). The later is set to -1 for all pixels with no data nor bounding box intersection. In Fig. 2 we can see a false color representation of validity mask and the first slice containing data.

It is worth noting that both the validity mask and the distance map make no assumptions on the physical model

and are calculated before any light computations are started. This is an important improvement in respect to other acceleration methods that only generate time reductions at the cost of making them dependent on specific types of physical phenomena. The single scattering model explained in previous sections should be taken as a convenient example. In general, as our optimizations are focused on speeding up traversals and reducing the number of fragments to be calculated, the heavier the computation imposed by the lighting model, the bigger will be the improvement that will be generated.

4.3 Ray marching passes

As stated before, to solve the lighting of the participating media, the radiance arriving at the observer from each of the pixels in the image plane $L(eye, \overline{\omega}_{ij})$, must be determined. To calculate it we have implemented a Cg fragment program that is executed in a series of passes.

The initial steps of the lighting pass perform a simple test on the validity mask to avoid unneeded computations. As soon the pixel is known not to contain data, evaluation of the shader is stopped and a transparent pixel is returned. For the remaining pixels, a primary ray is marched from the position corresponding to *iFirst* slice in the $\overline{\omega}_{ij}$ direction in $\Delta slice$ increments. This traversal is continued until the ray exits media's bounding box. At each point x, a shadow ray is followed from each of the N light emitters to the sampling point itself. Since this ray accounts for absorption and scattering, self-shadowing effects are included in the final image. We assume the space surrounding the bounding box is empty, so the target radiance $L(eye, \overline{\omega}_{ij})$ is $L(P_{near}, \overline{\omega}_{ij})$, the radiance measured when the ray exits the bounding box.

Program 1 shows the pseudocode used to perform lighting calculations on the slice with index *iSlice*. The parameters `vol3D`, `rayDir` and `pathRad` refer respectively to the 3D texture containing volumetric data and distance maps, ray direction $\overline{\omega}_{ij}$ and path radiance accumulated in previous slices. `x` and `x_uvw` are the sampling point's global and texture space coordinates. Function `GetIntersection` returns the intersection of the slice `iSlice` and the ray starting at `eye` with direction `rayDir`. $\tau(x, x + \Delta x)$ is returned by `Transmittance`, whilst $\kappa_t(x)$ and $\Omega(x)$ are calculated by `Extinction` and `ScattAlbedo`. All three functions are implemented as lookup tables, calculated once Δx is known. Function `OneLightRad` calculates the radiance arriving from the `n`th emitter and is based on a shadow ray traversal. `SkipSlices` returns the number of slices that are skipped according to the value stored in the distance map. Both `iSlice` and `pathRad` are output in a render target.

```
function SampleSlice( vol3D,rayDir,iSlice,pathRad )
{
   x = GetIntersection(iSlice, eye, rayDir);
   x_uvw = ToTxtCoords(x);
   volValue,distValue = tex3D(vol3D,x_uvw);

   // accumulate each light contribution
   lightsRad = (0,0,0);
   for (n=0; n<N; n++)
   {
      lightsRad += OneLightRad(x,n);
   }
   lightsRad *= ScattAlbedo(volValue)/(16*pi*pi);
   lightsRad *= Extinction(volValue);

   // add current sample to path radiance
   pathRad += lightsRad;
   pathRad *= Transmittance(volValue,deltaX);

   // determine next sampled slice
   iSlice += SkipSlices(distValue);

   return (iSlice,pathRad);
}
```

Program 1 Slice sampling pseudocode

The OpenGL/Cg FP40 fragment profile, the most flexible available at the date of the implementation, imposes a limit of 65.535 instructions per fragment program. Shaders containing more instructions will generate visible artifacts. Since we use nested loops to compute primary and shadow ray traversals, that limit is easily reached. To solve this, each pass samples m slices and a number of passes are performed until the whole volume is covered. m is assigned an empirical value on a per-scene basis, depending on the number of light sources and the distance between slices. To reduce the overhead, m is chosen to be the highest possible value without exceeding the limit (in our tests, values between 3 and 15). We use the OpenGL framebuffer object extension [32] in conjunction with MRT (Multiple Render Targets) to pipe the results from each pass as initial status for the next pass.

5 Results

Figures 3, 4 and 5a to 5h have been rendered with data scanned from real smoke using the method of Hawkins et al. [14]. This data set, referred to as (i), is composed of 120 full frame volume scans where each frame has been resized to a resolution of 256 × 256 × 64 voxels. Figures 5i to 5p

Fig. 3 *Left to right*: Simple back-to-front alpha compositing, primary rays, primary rays and shadow rays

have been calculated from voxel arrays simulated using the algorithm of Shi and Yu [38]. In this data set, named (ii), each of the 450 frames has been resized to a resolution of $128 \times 128 \times 128$ voxels. For displaying purposes we have implemented the interactive tone mapper by Goodnight et al. [10], which is based on Reinhard's photographic tone mapper [35].

Fig. 4 Image shows the same data set rendered with high (*left*) and low (*right*) extinction coefficients

Constellations of up to 20 lights have been generated for each of the HDR environments. Since our system allows free rotation of the HDR environment, the light constellation is calculated off-line using [5] and conveniently reoriented for each frame rendered. This does not suppose a severe restriction as several real-time conversion techniques [22, 33] are available. Even more, our system can handle dynamic light environments given they are converted to a sequence of light constellations.

In terms of storage, there are no important memory requirements apart from the volumetric data being rendered and its corresponding distance map. Light constellations use only 6 floating point values (color and position) per emitter. For low frequency lighting, these requirements are comparable to the equivalent of spherical harmonics and wavelets. More significant is the absence of heavy precomputation and storage of precalculated lighting or transfer functions. If we consider that we render sequences of media frames, these

Fig. 5 Images (**a**) to (**p**) from *left-right* and *top-bottom* order. Smoke rendered using eight different light probes

Table 1 Frames per second (speed-ups)

		Alpha	Primary	Self-shadows
No optimiz.	(i)	11	4.1	0.5
	(ii)	11.5	4.7	0.6
Mask	(i)	10 (0.9)	4.7 (1.1)	1.2 (2.4)
	(ii)	9.3 (0.8)	4.5 (0.9)	1.1 (1.8)
Mask + dist.	(i)	36.5 (3.3)	16.3 (3.9)	4.1 (8.2)
	(ii)	35 (3.0)	14.8 (3.1)	4.0 (6.6)

advantages become important improvements in respect to other techniques.

Figure 3 shows a smoke frame rendered using different lighting models: back-to-front alpha compositing, single scattering limited to primary rays and single scattering using primary and shadow rays. Note how in the second and third images the light constellation successfully captures the overall lighting conditions even with a low number of light emitters. However, in the middle image, as the emitters illuminate every point with the same strength, the media looks excessively bright. Last image shows a dimmer and more realistic result due to the attenuation and self-shadowing effects performed in the light to sampling point traversals. Since this example shows optically thin media, limiting our model to single scattering does not prevent getting images that look physically correct.

Figure 4 shows the same participating media with different extinction coefficients. Note how both images—optically thin media with low extinction coefficient and optically thick media with high extinction coefficient—are correctly depicted. Finally, Fig. 5 shows different lighting conditions using a set of HDR maps representing interior and exterior environments ranging from high contrast to smoothly changing conditions. Even in these diverse situations, volumetric data is lit without visible artifacts and is well integrated in the overall scene.

Algorithm performance is represented in Table 1. For each data set (i), (ii) and lighting method (alpha, primary and self-shadowing) the frames per second are shown for three different run modes: brute force ray marching without optimizations, optimized using validity mask, and optimized using validity mask and distance maps. Speed-ups from the brute force method are shown in brackets. All the images have been rendered with a NVidia Quadro FX3500 graphic card, at a resolution of 512×512 pixels. The projection of the bounding box for data sets (i) and (ii) covers approximately 50% of the screen, where 20% and 40% of the pixels correspond to non-transparent smoke respectively. The un-optimized algorithm is capable of generating interactive frame rates, from 11.5 to 0.5 fps, which is a poor performance compared to other algorithms that use texture-based methods. The second data set performs slightly better than the first, probably due to its smaller size. A speed-up of up to 2.4 is obtained when a validity mask optimization is included. More important improvements are obtained with data set (i) whose mask contains less pixels to be processed in the lighting passes. This pass comes with an extra cost, that makes the algorithm run slower with the simplest lighting model (speed-up 0.8). However, when complex models are used (primary and self-shadowing), any initial overhead is surpassed by the improvements of discarding calculations on empty image areas. Finally, including an extra distance map optimization results in a better performance. As both the validity mask and render passes apply this method, the resulting speed-up raises to 8.2 and 4 fps when the full lighting model is applied. In general, the algorithm performs better with data set (i) due to the more sparse distribution of the smoke and its higher number of holes, characteristics that lead to efficiently skipped areas. As shown in these results, our implementation is faster than other algorithms that include self shadowing without the need of including heavy precomputation.

6 Conclusions

In this work, we have presented an implementation of an interactive scattering model for inhomogeneous participating media, adapting the ray marching technique to the GPU. Our method models the incoming light field as a light constellation obtained from a HDR environment map, and allows for dynamic media, dynamic lighting and free camera movement. Any changes in these elements are taken into account by using real-time recalculation.

By combining two novel optimization techniques, a view-dependent validity pass and a view-independent distance map, we achieve speed-ups factors of up to 8. Single scattering with self-shadowing can be rendered at interactive frame rates. Since the optimizations do not make assumptions on the nature of the light interactions, they can be used with other lighting models apart from the described in this work. A GPU implementation of those algorithms will mainly be bounded by the calculations performed per-fragment and the memory intensive media traversals. Both elements are directly addressed by us.

As future work, we would like to test new approaches to include multiple scattering transmission. With the existing methods, this can only be achieved at the cost of considerable performance drops or using precomputation. Even while creating visually acceptable images, assuming an isotropic phase function makes our method less accurate. This can be alleviated using already known methods [1]. Finally, different optimization techniques based on alternative space classification structures and compression algorithms can alleviate the memory overhead imposed by the voxelized representation of both the data and the distance maps.

Acknowledgements The authors would like to express their gratitude to Tim Hawkins, Per Einarsson and Paul Debevec from the USC Institute for Creative Technologies, for the captured smoke data and hdr images used in this paper. The horse data set has been made available by Lin Shi and Yizhou Yu from the University of Illinois at Urbana-Champaign. Julián Flores from the University of Santiago de Compostela for his support in the initial steps of this work. We would also thank the anonymous reviewers for their keen insight and meaningful suggestions. This research has been funded by the projects UZ2007-TEC06 (University of Zaragoza) and TIN2007-63025 (Spanish Ministry of Science and Technology). Diego Gutierrez was additionally supported by a mobility grant by the Gobierno de Aragón (Ref: MI019/2007).

References

1. Blasi, P., Saĕc, B.L., Schlick, C.: A rendering algorithm for discrete volume density objects. Comput. Graph. Forum (Eurographics '93) **12**(3), 201–210 (1993)
2. Blinn, J.F.: Light reflection functions for simulation of clouds and dusty surfaces. SIGGRAPH Comput. Graph. **16**(3), 21–29 (1982)
3. Cerezo, E., Pérez, F., Pueyo, X., Serón, F.J., Sillion, F.X.: A survey on participating media rendering techniques. Vis. Comput. **21**(5), 303–328 (2005)
4. Chandrasekhar, S.: Radiative Transfer. Clarendon, Oxford (1950)
5. Cohen, J., Debevec, P.: The LightGen HDR shop plugin. http://www.hdrshop.com/main-pages/plugins.html (2001)
6. Cuntz, N., Kolb, A.: Fast hierarchical 3d distance transforms on the GPU. In: Eurographics 07 (Short Presentations) (2007)
7. Donnelly, W.: Per-pixel displacement mapping with distance functions. In: GPU Gems 2, Programming Techniques for High-Performance Graphics and General-Purpose Computation, pp. 123–136. Addison–Wesley, Reading (2005). Chap. 8
8. Engel, K., Ertl, T.: Interactive high-quality vollume rendering with flexible consumer graphics hardware. In: Eurographics State of the Art of Report (2002)
9. Entezari, A., Scoggins, R., Möller, T., Machiraju, R.: Shading for Fourier volume rendering. In: VVS'02: Proceedings of the 2002 IEEE Symposium on Volume Visualization and Graphics, pp. 131–138. IEEE Press, New York (2002)
10. Goodnight, N., Wang, R., Woolley, C., Humphreys, G.: Interactive time-dependent tone mapping using programmable graphics hardware. In: EGRW'03: Proceedings of the 14th Eurographics Workshop on Rendering, pp. 26–37. Eurographics Association, Aire-la-Ville (2003)
11. Grzeszczuk, R., Henn, C., Yagel, R.: Advanced geometry techniques for ray casting volumes. In: ACM SIGGRAPH 1998 Course Notes, Course 4 (1998)
12. Guthe, S., Wand, M., Gonser, J., Straßer, W.: Interactive rendering of large volume data sets. In: IEEE Visualization (2002)
13. Harris, M.J., Lastra, A.: Real-time cloud rendering. In: Chalmers, A., Rhyne, T.M. (eds.) EG 2001 Proceedings, vol. 20(3), pp. 76–84. Blackwell, Oxford (2001)
14. Hawkins, T., Einarsson, P., Debevec, P.: Acquisition of time-varying participating media. In: SIGGRAPH'05: ACM SIGGRAPH 2005 Papers, pp. 812–815. Assoc. Comput. Mach., New York (2005)
15. Hegeman, K., Ashikhmin, M., Premoze, S.: A lighting model for general participating media. In: SI3D, pp. 117–124 (2005)
16. Ishimaru, A.: Wave Propagation and Scattering in Random Media. Academic Press, New York (1978)
17. Jensen, H.W., Christensen, P.H.: Efficient simulation of light transport in scenes with participating media using photon maps. In: SIGGRAPH'98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pp. 311–320. Assoc. Comput. Mach., New York (1998). DOI: 10.1145/280814.280925, ISBN 0-89791-999-8
18. Kajiya, J.T., Herzen, B.P.V.: Ray tracing volume densities. In: SIGGRAPH'84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, pp. 165–174. Assoc. Comput. Mach., New York (1984)
19. Kalos, M.H., Whitlock, P.A.: Monte Carlo Methods, vol. 1: basics. Wiley, New York (1986)
20. Kay, T.L., Kajiya, J.T.: Ray tracing complex scenes. In: SIGGRAPH'86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, pp. 269–278. Assoc. Comput. Mach., New York (1986)
21. Klein, T., Strengert, M., Stegmaier, S., Ertl, T.: Exploiting frame-to-frame coherence for accelerating high-quality volume raycasting on graphics hardware. In: IEEE Visualization, p. 29 (2005)
22. Kollig, T., Keller, A.: Efficient illumination by high dynamic range images. In: EGRW'03: Proceedings of the 14th Eurographics Workshop on Rendering, pp. 45–50 (2003)
23. Kruger, J., Westermann, R.: Acceleration techniques for GPU-based volume rendering. In: VIS'03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03), p. 38. IEEE Comput. Soc., Los Alamitos (2003)
24. Lafortune, E.P., Willems, Y.D.: Rendering participating media with bidirectional path tracing. In: Proceedings of the Eurographics workshop on Rendering techniques'96, pp. 91–100. Springer, Berlin (1996)
25. LaMar, E., Hamann, B., Joy, K.I.: Multiresolution techniques for interactive texture-based volume visualization. In: VIS'99: Proceedings of the Conference on Visualization'99, pp. 355–361. IEEE Comput. Soc., Los Alamitos (1999)
26. Li, W., Mueller, K., Kaufman, A.: Empty space skipping and occlusion clipping for texture-based volume rendering. In: VIS'03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03), p. 42. IEEE Comput. Soc., Los Alamitos (2003)
27. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: LeCam, L.M., Neyman, J. (eds.) Proceedings of Fifth Berkeley Symposium on Math. Stat. and Prob., vol. 1, pp. 281–297. Univ. California Press, Berkeley (1967)
28. Max, N.L.: Atmospheric illumination and shadows. SIGGRAPH Comput. Graph. **20**(4), 117–124 (1986)
29. Narasimhan, S.G., Nayar, S.K.: Shedding Light on the Weather. IEEE Comput. Soc., Los Alamitos (2003), p. 665
30. Ng, R., Ramamoorthi, R., Hanrahan, P.: All-frequency shadows using non-linear wavelet lighting approximation. ACM Trans. Graph. **22**(3), 376–381 (2003)
31. NVidiaCorporation: Cg language specification. http://developer.download.nvidia.com/cg/Cg_1.5/1.5.0/0019/Cg_Specification.pdf
32. NVidiaCorporation: Framebuffer object extension. http://oss.sgi.com/projects/ogl-sample/registry/EXT/framebuffer_object.txt
33. Ostromoukhov, V., Donohue, C., Jodoin, P.M.: Fast hierarchical importance sampling with blue noise properties. ACM Trans. Graph. **23**(3), 488–495 (2004)
34. Ramamoorthi, R., Hanrahan, P.: An efficient representation for irradiance environment maps. In: SIGGRAPH'01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 497–500. Assoc. Comput. Mach., New York (2001)
35. Reinhard, E., Stark, M., Shirley, P., Ferwerda, J.: Photographic tone reproduction for digital images. In: SIGGRAPH'02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 267–276. Assoc. Comput. Mach., New York (2002)

36. Ritsche, N.: Real-time shell space rendering of volumetric geometry. In: GRAPHITE'06: Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, pp. 265–274. Assoc. Comput. Mach., New York (2006)
37. Rushmeier, H.E., Torrance, K.E.: The zonal method for calculating light intensities in the presence of a participating medium. In: SIGGRAPH'87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 293–302. Assoc. Comput. Mach., New York (1987)
38. Shi, L., Yu, Y.: Controllable smoke animation with guiding objects. ACM Trans. Graph. **24**(1), 140–164 (2005)
39. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In: SIGGRAPH'02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 527–536. Assoc. Comput. Mach., New York (2002)
40. Stam, J.: Stochastic rendering of density fields. In: Proceedings of Graphics Interface'94, pp. 51–58 (1994)
41. Stam, J.: Multiple scattering as a diffusion process. In: Hanrahan, P.M., Purgathofer, W. (eds.) Rendering Techniques'95 (Proceedings of the Sixth Eurographics Workshop on Rendering), pp. 41–50. Springer, Berlin (1995)
42. Sun, B., Ramamoorthi, R., Narasimhan, S.G., Nayar, S.K.: A practical analytic single scattering model for real time rendering. In: SIGGRAPH'05: ACM SIGGRAPH 2005 Papers, pp. 1040–1049. Assoc. Comput. Mach., New York (2005)
43. Szirmay-Kalos, L., Sbert, M., Umenhoffer, T.: Real-time multiple scattering in participating media with illumination networks. In: Eurographics Symposium on Rendering, pp. 277–282 (2005)
44. Tost, D., Grau, S., Ferre, M., Puig, A.: Ray-casting time-varying volume data sets with frame-to-frame coherence. In: Proceedings of SPIE (2006)
45. Vollrath, J., Weiskopf, D., Ertl, T.: A generic software framework for the gpu volume rendering pipeline. In: VMV'05: Proceedings of Vision, Modeling, and Visualization Conference (2005)
46. Westermann, R., Sevenich, B.: Accelerated volume ray-casting using texture mapping. In: VIS'01: Proceedings of the Conference on Visualization'01, pp. 271–278, IEEE Comput. Soc., Los Alamitos, 2001
47. Zhou, K., Hou, Q., Gong, M., Snyder, J., Guo, B., Shum, H.Y.: Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. Technical Report, Microsoft Research (2007)
48. Zhou, K., Ren, Z., Lin, S., Bao, H., Guo, B., Shum, H.Y.: Real-time smoke rendering using compensated ray marching. Technical Report, Microsoft Research (2007)

Fernando Navarro has an MsC from the University of Zaragoza and since 1997 has been leading different R&D departments in computer graphics, visual effects and feature film companies. From 2006, he has been working for Lionhead Studios (Microsoft Games Studios). He is also following a Ph.D. in Computer Science, focused on advanced rendering algorithms and their real time implementations. He is a member of several committees including several ACM conferences. His interests include advanced rendering algorithms, global illumination and HDRI lighting.

Diego Gutierrez is a tenured Associate Professor at the University of Zaragoza, Spain, where he received his Ph.D. in Computer Science, earning the Ph.D. Extraordinary Award. He's Chair of SIGGRAPH Asia 2008 Sketches and Posters. He was also co-chair of ACM GRAPHITE 2006 and has served on several other committees, including Eurographics, the SIGGRAPH sketches program, Pacific Graphics and other ACM or EG-related conferences. His areas of interest include rendering, computational photography, global illumination, high dynamic range and perception. He has published more than 60 papers in international conferences and journals.

Francisco J. Serón is a Professor of Computer Science at the Technical School of Engineering at the University of Zaragoza. He received a Physical Science Degree from the University of Zaragoza in 1977 and a Ph.D. from the same University in 1984. At present he is the head of the Advanced Computer Graphics Group within the Computer Science Department. His research interests include simulation of natural phenomena, illumination engineering, and virtual reality.