Historygrams: Enabling Interactive Global Illumination in Direct Volume Rendering using Photon Mapping

Daniel Jönsson, Joel Kronander, Timo Ropinski, and Anders Ynnerman

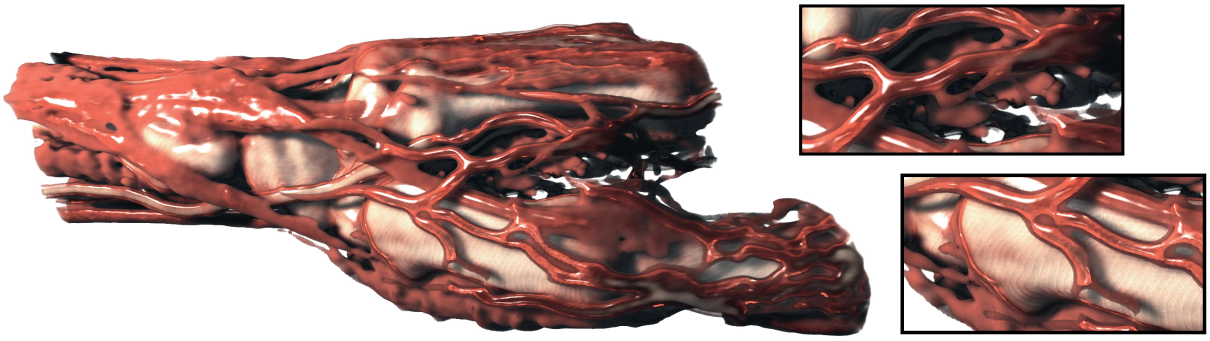


Fig. 1. A CT scan of a hand rendered using the proposed historygram-based volumetric photon mapper. The insets show the high dynamic range achievable in shadowed areas resulting from multiple scattering, as well as the usage of different materials. When using our method only $\sim 30\%$ of the light transport solutions need to be recomputed when changing the material of the bone.

Abstract—In this paper, we enable interactive volumetric global illumination by extending photon mapping techniques to handle interactive transfer function (TF) and material editing in the context of volume rendering. We propose novel algorithms and data structures for finding and evaluating parts of a scene affected by these parameter changes, and thus support efficient updates of the photon map. In direct volume rendering (DVR) the ability to explore volume data using parameter changes, such as editable TFs, is of key importance. Advanced global illumination techniques are in most cases computationally too expensive, as they prevent the desired interactivity. Our technique decreases the amount of computation caused by parameter changes, by introducing *Historygrams* which allow us to efficiently reuse previously computed photon media interactions. Along the viewing rays, we utilize properties of the light transport equations to subdivide a view-ray into segments and independently update them when invalid. Unlike segments of a view-ray, photon scattering events within the volumetric medium needs to be sequentially updated. Using our *Historygram* approach, we can identify the first invalid photon interaction caused by a property change, and thus reuse all valid photon interactions. Combining these two novel concepts, supports interactive editing of parameters when using volumetric photon mapping in the context of DVR. As a consequence, we can handle arbitrarily shaped and positioned light sources, arbitrary phase functions, bidirectional reflectance distribution functions and multiple scattering which has previously not been possible in interactive DVR.

Index Terms—Volume rendering, photon mapping, global illumination, participating media.

1 Introduction

Interactive Direct Volume Rendering (DVR) is an increasingly important method for exploration and presentation of data in a wide range of application domains. Rendering speeds for volume data have improved greatly over the past decades due to development of both hardware and algorithms. However, current methods for volumetric illumination in DVR are still severely limited, often only handling single-scattering, fixed illumination or not even taking volumetric self-shadowing into account [16]. This is in sharp contrast to the more complex models that have been deployed in off-line rendering of realistic looking images of both surface and volumetric objects [4]. Within DVR tradeoffs have usually been made in order to support interactive editing of the transfer function (TF), which is an integral part of any DVR pipeline.

The driving force behind the desire to apply more realistic illumination is the fact that light is the fundamental carrier of visual information. The lighting used in the rendering of an image has an enormous impact on how it is interpreted by a human observer. By carefully designing the lighting setup, illumination can reveal strong visual cues [21, 23] describing global structures, local details and curvature of surfaces and volumetric structures that would otherwise be hidden from the observer. It has, furthermore, been shown that a perceptually important part of the light interaction in the scene is light that has scattered more than one time in the medium [8, 17].

In this work we enable highly realistic volume renderings at interactive rates, based on photon mapping. We achieve interactive frame rates, by introducing a novel approach where only light interactions that have been affected by the parameter changes are recomputed. Photon mapping (PM) is extensively used in computer graphics for rendering participating media [14, 15], as it is an efficient method for producing physically plausible global illumination effects. Even though photon mapping can produce a wide range of illumination effects, it has so far been too expensive to be used in interactive volumetric illumination. This is due to the fact that all view-rays and all photons need to be recomputed as soon as the scattering parameters change, which prevents interactive TF editing. In this paper we introduce interactive photon mapping for DVR by proposing a novel concept, which we refer to as *Historygrams*. A Historygram is a histogram of the parameters used to compute a photon, or a viewing ray. Thus,

• *Daniel Jönsson, Joel Kronander, Timo Ropinski, and Anders Ynnerman are with Linköping University, Sweden. E-mail: {daniel.jonsson, joel.kronander, timo.ropinski, anders.ynnerman}@liu.se.*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Historygrams enable us to detect which photons and viewing rays are invalid based on a parameter change, for example, editing the TF. We then only recompute these invalid parts, and can thus achieve interactive rendering times while producing highly realistic images (see figure 1). The main contributions of this paper are:

- Enabling volumetric photon mapping for use in DVR, allowing interactive editing of the TF and, a range of perceptually important global illumination effects, such as multiple-scattering and specular materials.
- Introduction of *Historygrams*, a novel concept for efficiently storing and evaluating the parameter history upon which a particular computed object (viewing ray segment or photon path) is dependent on.
- Methods and data structures for efficient queries of individual *Historygrams* for large numbers of objects.

2 Related Work

Interactive rendering of global illumination effects has been a long standing challenge in computer graphics. Here we focus on photon mapping techniques as well as those techniques proposed for interactive DVR. Ritschel et al. provide a more comprehensive overview of interactive global illumination techniques in a recent survey [27].

Photon Mapping. Photon mapping is a biased two-pass global illumination algorithm that often produces images with less noise than other Monte Carlo algorithms [14]. In the first pass, a photon map, representing the incident illumination in the scene, is generated by tracing photons from light sources through the scene. The second pass estimates the final rendering using ray-tracing, whereby the radiance in the scene is estimated by computing an approximation of the photon density given by the photon map computed in the first pass. However, for finite sample sets, this approximation smooths the true radiance and causes bias, i. e., a nonzero expected error. Still, PM is generally a consistent Monte Carlo method, meaning that, in the limit of using an infinite number of photons, the approximation converges to the true solution. Recent work on Progressive Photon Mapping (PPM) [9, 5], provides a way to reduce bias by using several rendering passes where photons are progressively traced and discarded, updating the radiance estimates after each photon tracing pass in such a way that the approximation converges to the correct solution in the limit. PM for volumetric media was first introduced by Jensen and Christensen [15]. Jarosz et al. [13] improved on the original formulation by using a more efficient beam radiance query, instead of using redundant photon density estimations during ray marching. Recently, Jarosz et al. [12] also showed how to approximate the photon density as a set of photon beams, improving the performance further. In general our method is orthogonal to these concepts, however some minor modifications are necessary which we discuss in Section 7. None of the discussed methods effectively handle the effects of local parameter changes and, thus, need to recompute the entire photon map (all photon beams) and all view-rays even for small parameter changes. In contrast, Dmitriev et al. [6], propose to reuse unaffected photons across frames in dynamic scenes. Using Quasi-Monte Carlo photon tracing they first randomly retrace a small set of photons to find regions where photons need to be recomputed. They then show how photons in affected regions can be found from the set of sparse samples with a high probability by using the periodicity of the generated quasi-random Halton sequence to find photons with similar paths. Although, the idea of reusing valid photons across frames is not new in computer graphics, to the authors knowledge these methods have not been applied in the context of volume rendering before and no previous work has considered adaptive photon retracing based on TF edits.

Interactive Illumination for DVR. Traditionally, full global illumination effects have been too costly to be used in interactive DVR applications, and instead different approximations have been made. We address a few of these approaches, and refer the reader to a more complete survey by Jönsson et. al [16].

Several methods exploit texture slicing to estimate forward scattering, shadows and color bleeding [18, 32, 34]. While these approaches, can only be used in slice-based renderers, several ambient occlusion approaches are renderer independent. Hernell et al. perform a ray-casting pass per voxel over a local spherical neighborhood [11], while Ropinski et al. [30] compute local data histograms to obtain ambient occlusion effects. However, these methods are limited to local effects and do not, in contrast to our approach, incorporate physically based global illumination effects. To simulate global shadow effects, methods computing shadow volumes [3, 29] and deep shadow maps [10] have been presented. These methods sample the volumetric occlusion from the light sources utilizing efficient pre-computed data structures. More recently, Sundén et al. [33] have proposed exploiting the plane sweep paradigm to enable dynamic illumination with a low memory overhead at interactive frame rates. Kronander et al. [20] uses an efficient multi-resolution grid to store visibility information using Spherical Harmonics. Lindemann and Ropinski [22] have exploited spherical harmonics in a similar manner for enabling advanced material properties in the context of DVR. To achieve physically more plausible results, unbiased Monte Carlo methods have also been adopted for real-time DVR [31, 19]. However, interactivity is limited as TF updates are expensive since the complete volume transport has to be recomputed for all, even minor or local, TF changes. Alternatively, for real-time rendering of participating media the full global illumination solution can be approximated by focusing only on single scattering [7].

3 Background

In this section we provide the theoretical background for volumetric photon mapping. Subsection 3.1 describes the details of light transport in a participating medium and establises notations, while we show how to numerically approximate the light transport using volumetric PM in Subsection 3.2.

3.1 Volumetric Illumination Model

In participating media, photons are affected by emission, in-scattering, absorption and out-scattering [25].

We start our derivation by considering a sample x along a viewing ray. Radiance $L_s(x,\vec{\omega}_o)$ emitted and scattered from x into the direction $\vec{\omega}_o$ can be defined as:

$$L_s(x,\vec{\omega}_o) = \sigma_s L_i(x,\vec{\omega}_o) + \sigma_a L_e(x,\vec{\omega}_o), \tag{1}$$

where σ_s is the scattering coefficient and σ_a is the absorption coefficient. The radiance reaching a point x_c along the viewing ray with direction $\vec{\omega}_o$ is the sum of the background radiance $L_0(\vec{x_0},\vec{\omega}_o)$, at the boundary of the volume and the accumulated emitted and in-scattered radiance from the medium.

$$L(x_c,\vec{\omega}_o) = T(x_0,x_c)L_0(x_0,\vec{\omega}_o) + \int_{x_o}^{x_c} T(x,x_c)(\sigma_s L_i(x,\vec{\omega}_o) + \sigma_a L_e(x,\vec{\omega}_o))dx, \tag{2}$$

where $T(x_i,x_j)$ is the transmittance between points x_i and x_j, describing how light is attenuated when traveling through the volume

$$T(x_i,x_j) = e^{-\int_{x_i}^{x_j}\tau(x')dx'}, \tag{3}$$

where $\tau(x')$ specifies the extinction at x'. One property of the transmittance, which we utilize in this work, is that the transmittance is multiplicative, i. e. that it can be divided into segments and multipled together:

$$T(\vec{x_i},\vec{x_j}) = T(\vec{x_i},\vec{x_k})\cdot T(\vec{x_k},\vec{x_j}). \tag{4}$$

The in-scattered radiance, $L_i(x,\vec{\omega})$, depends on radiance arriving at x from all directions, $\vec{\omega}$, over the sphere of directions ,$\Omega_{4\pi}$,

$$L_i(x,\vec{\omega}) = \int_{\Omega_{4\pi}} s(x,\vec{\omega'},\vec{\omega})L(x,\vec{\omega'})d\vec{\omega'} \tag{5}$$

where $s(x,\vec{\omega}_i,\vec{\omega}_o)$ represents the scattering function at the point x. In this work, we enable $s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ to shift between phase function and BRDF depending on the material given by the TF. This allows the user to create surface-like materials within the volume.

3.2 Volumetric Photon Mapping

In this work we adopt the standard photon mapping framework for volumetric media, originally proposed by Jensen and Christensen [15].
Photon Tracing. In a first pass, a Markov random-walk approach is used to trace photons from light sources into and through the volume. For each photon, importance sampling is used to select from which light source the photon should be traced, and to generate a position and direction on the chosen light source. Photons traced through the volume may either pass unaffected or interact (be absorbed, or scattered) at each point. To decide if and where the photon interacts with the media, we draw a sample from the probability distribution of a scattering or absorption event, by using the inversion method. The cumulative probability distribution for an interaction event is given by

$$C(x) = 1 - T(x_s, x) = 1 - e^{-\int_{x_s}^{x} \tau(x')dx'}, \tag{6}$$

where x_s is the point at which the photon enters the volume and the transparency, $T(x_s, x)$, is computed using standard ray-marching. At each point where the photon interacts with the volume a copy of the photon flux and incoming direction of the photon is stored in the photon map. Russian roulette is used to decide if the photon is absorbed or scattered further in the medium. The probability that a photon is scattered is given by

$$p(x) = \frac{\sigma_s(x)}{\tau(x)}. \tag{7}$$

If the photon is scattered, importance sampling of the phase function or BRDF is used to determine its new direction.
Rendering. To render the final image, viewing rays are generated and ray-marching is used to evaluate equation (2). For each sampled point along the ray, the photon map is used to form an estimate of the in-scattered radiance. By using a spherical kernel of radius r, the n closest photons are used to estimate the in-scattered radiance

$$L_i(x, \vec{\omega}) \approx \frac{1}{\sigma_s(x)} \sum_{p=1}^{n} s(x, \vec{\omega}_p, \vec{\omega}) \frac{\Delta\Phi_p(x, \vec{\omega}_p)}{\frac{4}{3}\pi r^3}, \tag{8}$$

where Φ_p is the flux of the photon, p, in direction $\vec{\omega}_p$.

4 Method

In this section we describe in detail how volumetric photon mapping can be extended to handle interactive TF and other rendering parameter changes, by using the Historygram concept in the context of DVR.

4.1 Overview

An important part of any DVR pipeline is the exploration of volume data by performing interactive TF changes. This affects light transport as the TF maps data values to material properties, such as transmittance, $\tau(x)$, scattering coefficient, $\sigma_s(x)$, absorption coefficient, $\sigma_a(x)$, scattering function, $s(x, \vec{\omega}_i, \vec{\omega}_o)$. Thus, any pre-computation which relies on volume material properties must be recomputed as soon as the user changes the TF. Furthermore, interactive data exploration using the TF means that the volumetric medium may change in every rendered frame. However, typically, the user does not make global changes to the TF, but, instead, focuses on exploring different objects in the volume, one at a time, or on tuning localized material properties. Consider, for example adjusting the diffuse color of the bone in figure 1. This enables us to only recompute the parts of the light transport simulation which are affected by the changes, and reuse the ones that do not change. To accomplish this we introduce an efficient partition of the light transport calculations where we are using photons to partition the incident illumination in the volume and view-ray segments for partitioning the ray-marching computations, as illustrated in figure 2. Each partitioning of the light transport solution stores its own Historygram specifying which parameters have been used during the computation of this object (view-ray segment, photon). It is important to note that the parameters in our case consist of volume data intensities since this makes the created Historygram independent of current TFs, see section 4.3. Historygrams are a binary data structure defined to provide a minimal memory footprint, while at the same time, being efficient to query. When the user changes the TF we use the changed parameter values to form a query over all stored Historygrams and, thus, isolate the parts of the light transport solution which need to be recomputed. To recompute the light transport in the volume, we first update all invalid photons, i. e., retracing them from the last unaffected intersection. Afterwards, affected viewing-ray segments are updated.

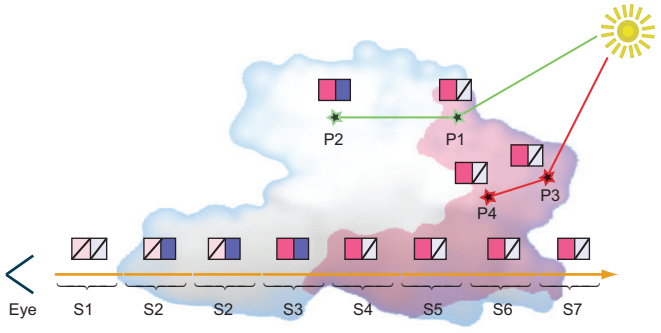


Fig. 2. To localize the effect of parameter changes on the light transport solution, we partition both the incident illumination using photons and view-rays using ray-segments. To encode the parameters used for the computation of each partition object, we store a separate Historygram for each object. In this scheme, using only two material parameters, depicted by light blue and light red, a simple Historygram using only two bits can be used to store the parameter history of each computational object (photon, ray-segment). When tracing photons from the light source, each photon inherits its ancestors Historygram due to scattering, while for ray-segments each Historygram is independent of the other segments Historygrams.

In section 4.2 we discuss how a general Historygram is constructed, stored and effectively queried. We then, in section 4.3, explain how we can efficiently store commonly used parameters in DVR using the Historygram data structure. Compared to a standard photon map, we store an additional Historygram for each photon in the photon map, encoding the history of the light path the photon has traversed. The exact details on how each photons Historygram is computed, are discussed in section 4.3.1. We additionally store a separate Historygram for each view-ray segment. To only update parts of the viewing rays at a time, the multiplicative property of the volume rendering integral is used which states that compositing transparency can be performed in any order. In section 4.3.2, we show how we can recompute only those segments of viewing rays, which are affected by a TF update.

4.2 Historygrams

A Historygram represents a set of parameters, which has been used during the computation of a particular object, e. g., a view-ray segment or a photon. Before we discuss the application of Historygrams, we will detail the notation of a Historygram and show how a Historygram is incrementally constructed. We then describe an efficient way of querying Historygrams, and how we can represent Historygrams using an efficient binary data structure.
Notation. The Historygram, H, is a set of representable parameter sets Z_i. A specific Z_i represents a group of parameters represented as a single entry in the Historygram. To map an arbitrary parameter, x, to a representable set Z_i we define a mapping, $\delta : x \to \{Z_i, i = 1...n\}$. This mapping thus describes how arbitrary parameters are represented in the Historygram H.
Constructing Historygrams. At first, the Historygrams of all objects are set to the empty set $H_0 = \emptyset$. During the computation of an object with Historygram H_i, the parameters used , x_j, $j = 1...n$, are added to

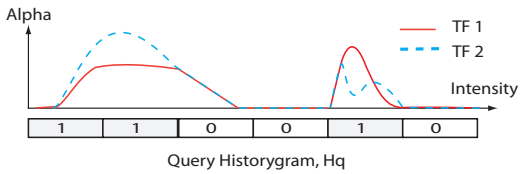


Fig. 3. Transfer function changes are mapped to a query Historygram, H_q, in the form of a finite sequence of binary values. The current TF setting is represented by a dashed blue line, and the new TF setting by a solid red line. Given the difference in the TF setting, the changes are mapped to affected data value intervals. A query Historygram with a 6 bit representation is shown in the bottom. By intersecting the intervals with the support of the bins in the discrete Historygram representation, affected bins are set to 1, and unaffected bins are set to 0.

the set H_{i+1}

$$H_{i+1} = H_i \bigcup_{j=1}^{n} \delta(x_j), \tag{9}$$

where H_i is the previous Historygram and H_{i+1} is the new Historygram containing both the previous set of used parameters, and x. When the computation has completed, the Historygram will thus contain information about all parameters used during its lifetime.

Historygram Queries. When the parameters specifying the global light transport solution change, we would like to evaluate which objects need to be updated. Therefore, equation 9 is used to map the changed parameters into a Query-Historygram H_q. Then, for all objects we evaluate which subsets of parameters in the objects Historygram, H, is affected by the change, by intersecting the sets

$$H_u = H \cap H_q, \tag{10}$$

where H_u represents the set of affected parameter sets. If the resulting H_u is non empty we recompute the object with Historygram H.

Historygram Representation. We propose to use a binary representation of the Historygram H. This saves storage as only one bit needs to be used to represent if a subset of parameters Z_i has been used. Furthermore, for Historygram queries, the union $\cup$, and intersection $\cap$, can then be evaluated using the OR and AND operator respectively. As these operations can be translated directly to hardware instructions, they are extremely fast to evaluate. Also, since we use a binary representation, we know that if H_u is not equal to zero at least one parameter subset that was used during the computation has changed . All together, this allows us to efficiently answer the question if a computation has used a certain set of parameters or not. Thus, computations which are not affected by a set of parameter changes may be reused.

4.3 Historygrams for Photon Mapping in DVR

In a standard DVR pipeline, the TF is used to map scalar data values to optical properties. Using the fact that material properties in the volume directly correspond to data values through the TF mapping, we can use a mapping δ, to transform data values to a representable set of parameters Z_i. This enables the Historygram computation to be independent of the current TF as only data values from the volume are used. An example of a $\delta(x)$ function, which maps data value $x \in [0,1]$ to a binary representable set in H is

$$\delta(x) = 1 << floor(x * (p-1)), \tag{11}$$

where $<<$ denotes the left bit shift operator and p is the number of bits used for the Historygram. Typically, p is 32 or 64 such that the native bit shift operator can be used. This corresponds to uniformly discretizing the TF domain into a set of p bins. Each bin corresponds to the parameters specified through the TF for the range of data values represented by the bin. Query Historygrams are constructed in a similar manner. First, we map the TF change to the affected data values, and then we construct the query using $\delta(x)$ as in equation (11) (see figure 3). In the following two subsections, we explain how we facilitate the Historygram concept for photons and viewing-rays in the context of DVR. Based on the Historygram representation, it can be determined efficiently through binary processing, i. e., bitwise operations, if photons or viewing-rays are invalid. For storage, a single bit represents a change of a parameter, which means that for instance eight parameters can be represented using a single byte.

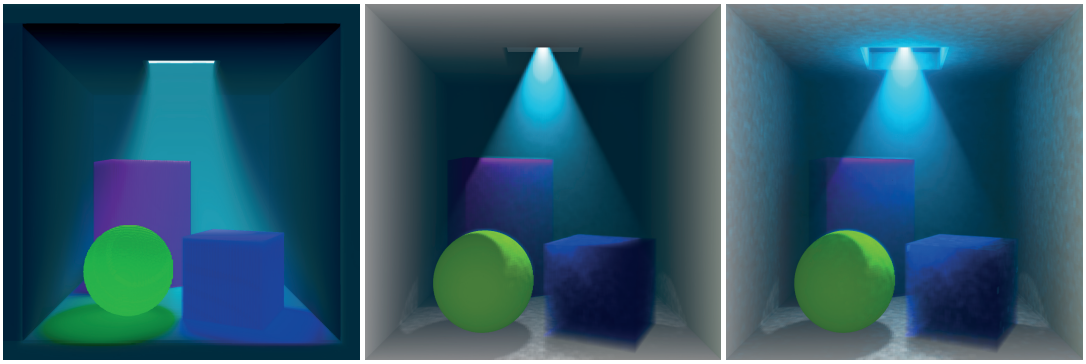
4.3.1 Photon Historygrams

As a photon traverses the medium it may be absorbed, or scattered in different directions depending on optical properties of the medium. This means that photons are dependent on previous interactions with the volumetric medium. Therefore we must take this into account when updating photon interactions. To do so, we create Historygrams based on the volume data intensity as discussed in the previous section. As soon as a photon interacts with the medium (scatters or absorbs, see Section 3.2), we store the Historygram, position, direction and power of the photon interaction. A scattered photon continues to use the Historygram, which means that the last photon interaction will know about all data that was used for all scattering events of that photon. Then, when the parameters change and a query Historygram has been created, we start by evaluating the last photon interaction. If the last photon interaction was not affected by the parameter change, we know that none of the previous ones where not either, and thus we do not need to update any of them. However, if the photon interaction was affected by the parameter change, we continue backwards until we find a photon interaction which was not affected. Given the position of the last valid photon interaction, and the material properties at that location, we can apply importance sampling to determine a new direction and restart photon tracing.

4.3.2 View-Ray Historygrams

For rays cast from the camera, we utilize Equation (4), which states that the transmittance between two points can be divided into smaller segments and multiplied together. Thus, each segment can be treated independently. This means, that, if the user changes a parameter which affects the inner parts of the volume, segments before and after do not need to be recomputed as they are not affected by the change. Using the same approach as for photon interactions, each segment forms its Historygram using equation (9) and equation (11). Therefore, we store the Historygram of each segment together with the color. When the user changes the material parameters we analyze if the segment has been affected, using the query Historygram, and recompute it if necessary. Generally we uniformly divide each view-ray into a user defined number of segments, V_n. For a standard DVR pipeline the Historygram only needs to be computed the first time a segment is computed, as the underlying data does not change. The creation of Historygrams for view-rays is therefore performed as soon as the camera stops moving. Then, when performing incremental updates, the Historygram does not need to be updated.

In some cases ray segments need to be recomputed even though the used volume data intensity is not affected by the parameter change. For instance, this occurs when one structure within the data casts a shadow on a another structure consisting of different intensity values. If the occluding structure is removed, then the remaining structure should be updated to take new photons into account. If this is not done the view segments including the remaining object will not be updated, and will appear to be in shadow even though they are not. It is therefore not enough to know the data intensity based Historygram alone. Instead, the spatial location of the changes is important. This issue can be handled in various ways, but we propose the use of Historygrams for this as well. One Historygram is allocated for each of the three dimensions of the data set. As an example, for a volume of dimensions 64^3 we can recognize spatial changes at voxel resolution using only $3 \cdot 64$ bits, i.e. 64 bits per dimension. When a photon interaction is removed or added due to a parameter change, a mapping function is used for each dimension to update a single, global, spatial Historygram. When all photons have been updated, this global spatial Historygram will contain information about where all illumination



(a) Shadow ray-caster [28] (b) Our method using single scattering (c) Our method using multiple scattering

Fig. 4. (a) Synthetic Cornell Box ($256 \times 256 \times 256$ voxels) shaded using shadow volume propagation which simulates multiple-scattering with a diffusion process [28], (b) our method using 4 million photons and single scattering, (c) our method using 4 million photons and multiple scattering.

changes have been made. Then, when updating view-ray segments, in addition to the Historygram based on data intensity, the segment also queries the spatial Historygram for changes within the photon radius. In this paper, most changes made have been local, or affected the same data intensities. Therefore, the spatial Historygram evaluation has not been performed. It should, however, be considered for applications where illumination accuracy is of importance.

5 Implementation

To realize the interactive application of the proposed Historygram method, we have implemented the described concepts using OpenCL.

Data Structures. When the user changes the TF, it is neessecary to search for invalid photons and viewing-ray segments. Additionally, it is necessary to determine the order of scattering events for photons. For this purpose, we use an implicitly linked list where the photon interaction id determines the index into the list. In our implementation we store the Historygram for each photon interaction in addition to its position, direction and power. We apply the same implicitly linked list concept to view-ray segments where the Historygram is stored together with the color. Thus, when the TF changes, we can start at the last stored scattering event and move backwards in the list until a valid photon is found.

In order to evaluate the in-scattered radiance during rendering, we need to be able to find nearby photons for a location $\vec{x}$. To do so, we have adopted a chaining hash table as described by Alcantara et al. [1], which allows us to find nearby photons in $O(1)$ time. We use a hash function which maps the location $\vec{x}$ into uniformly distributed bins with the bin size equal to the photon radius. During photon tracing, the index of the bin is stored together with the index of the photon interaction in a list. The list can then be efficiently sorted on the GPU, based on bin index using radix sort [26]. The start and end of each bin are then found and used to query how many, and which, photon interactions are near a location. Since the photon radius is used for the bin width, all photons within the given radius can be found by querying the 27 surrounding bins of a location. When the user changes the TF, and a photon is determined to be invalid, we move it from the current location in the hash table to the new location.

Algorithm. During the photon tracing process we create a Historygram for each photon interaction using equation 11, with p set to 64, which allows us to use native instructions on the GPU when evaluating Historygrams. As soon as the camera stops moving, we start a ray-casting pass which generates and stores the Historygram for each view-ray segment. Then, when the user starts to change parameters, we create the Historygram H_q by comparing the new parameter to the old one and add it if changed. When a photon interaction is found invalid, we set the occupied location in the hash to the maximum integer value. To avoid the use of atomic operations, we reuse locations of the removed photon interactions in the hash table. Then, when the hash table has been sorted, we can determine the last index of the hash by evaluating the number of existing, added and removed photon interactions. Once the photons have been updated, we can start to evaluate view-ray segments. In order to increase the work-group load distribution we update all segments independently and then composite them in a separate step.

To incorporate realistic material shading effects, we support different material models. We primarily use isotropic phase functions, although we also support Henyey-Greenstein and the Schlick approximation. For BRDF specification we use the microfacet distribution as given by Ashikimin [2] or the recently published ABC model [24].

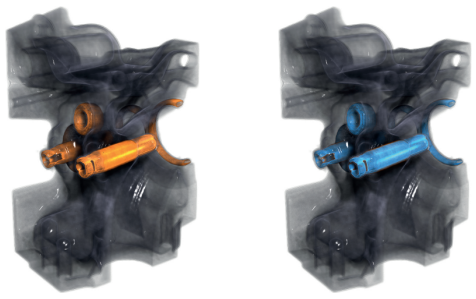


Fig. 5. A CT scan of an engine [$256 \times 256 \times 256$] rendered with multiple scattering (2 light bounces) and 2.1 million photons. Changing material properties from the left image to the right image can be performed 7.7 times faster using our Historygram method compared to a GPU *complete discard and rerender all* photon mapping implementation.

6 Results and Evaluation

In this section, we show the quality and performance of our method on a number of both real world and synthetic examples. All tests were performed on a computer with an Intel Xeon 2.67 GHz processor, 6 GB random access memory and an Nvidia Geforce 570 graphics card.

6.1 Results

Using a full global illumination simulation, perceptually important effects, such as multiple-scattering, color bleeding and realistic material functions can be incorporated in the DVR pipeline. Figure 4 shows a synthetic Cornell Box ($256 \times 256 \times 256$ voxels) shaded using shadow volume propagation as proposed by Ropinski et. al. [28], which uses a diffusion process for approximating multiple-scattering, our method using single scattering, and our method using multiple scattering (2 light bounces). As can be seen in the renderings, the diffusion approximation fails to account for multiple scattering in these media, and works best for dense objects. Using single scattering, our method is able to accurately render the thin media in the interior of the room, the transparent box as well as the solid objects in the scene. Multiple scattering effects provide a visually pleasing result, and color-bleeding

(a) Golden lady, $512 \times 512 \times 625$ voxels

(b) Cenovix, $361 \times 331 \times 361$ voxels

Fig. 6. Results achieved with our method. a) displays a CT scan of a woman, rendered using 2.1 million photons, 9 view segments, single scattering, and three light sources. A microfacet BRDF function (Ashikimin-Shirley [2]) was used to add specular reflections to the blood vessels, and an isotropic phase function was used for the bone structures. Varying the material properties of the outer skin layer requires 74% of the photons to be retraced and 24% of the view segments to be recomputed. b) displays a CT scan rendered using 1.6 million photons and 7 view segments. Removing the kidney content requires 35% of the photons and 9% of the view segments to be recomputed. Changing the color of the vessels requires 24% of the photons and 4% of the view segments to be recomputed, thereby enabling speedups of approximately 7 times.

Table 1. Performance measurements for the data sets used in this paper. As can be seen in the photon tracing and gathering columns with and without Historygrams, interactive editing is enabled in almost all cases with up to 15 times speedup.

Data set	P_n	Samples /voxel	Without Historygrams		With Historygrams				
			Photon tracing	Photon gathering	V_n	Initialization overhead	Photon tracing / Recomputed	Photon gathering / Recomputed	Total speedup
Hand, fig 1	2.1 M	8.0	0.22 s	1.97 s	10	16%	0.10 s / 19 %	1.63 s / 29 %	1.3x
Cornell, fig 4(b)	4.2 M	2.0	0.32 s	9.11 s	7	26%	0.04 s / 3 %	0.57 s / 18 %	15.2x
Engine, fig 5	2.1 M	4.0	0.37 s	5.41 s	10	5%	0.10 s / 5 %	0.64 s / 19 %	7.7x
Golden lady, fig 6(a)(top)	2.1 M	1.5	0.37 s	1.31 s	9	25%	0.25 s / 74 %	0.43 s / 24 %	2.4x
Golden lady, fig 6(a)(bottom)	2.1 M	1.5	0.23 s	1.32 s	9	3%	0.15 s / 71 %	1.11 s / 100 %	1.2x
Cenovix, fig 6(b)(top)	1.6 M	4.0	0.11 s	1.25 s	7	8%	0.03 s / 24 %	0.16 s / 4 %	6.8x
Cenovix, fig 6(b)(bottom)	1.6 M	4.0	0.11 s	0.95 s	7	11%	0.07 s / 35 %	0.25 s / 9 %	3.2x

from the blue medium in the interior of the room can be seen on the walls, the transparent box in the lower right also shows the effect of sub-surface scattering.

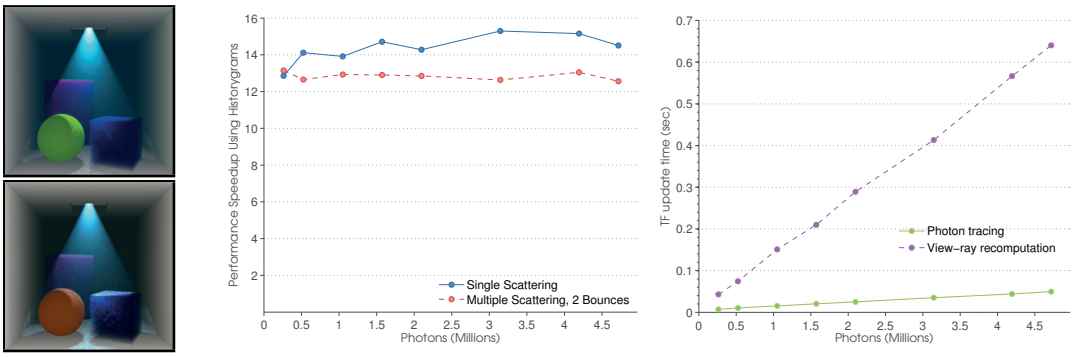
Figure 6(a) shows a rendering of a CT scan of a woman (resolution $512 \times 512 \times 625$). The images were rendered using single scattering, 2 million photons, 9 view-ray segments, and three light sources. Specular blood vessels and the diffuse bone structure eases the perception of the vessel structure. Changing the transfer function for values affecting the outer skin layer requires recomputing 74% of the photons and 24% of the view segments (see figure 6(a)). As can be seen in the photon tracing and gathering columns in table 1, enabling Historygrams reduces the computation time being non-interactive to interactive in almost all cases. Thus, interactive updates which could not otherwise be performed are now possible. In general our method also performs well on larger data sets, as the photon representation is independent of the resolution of the data set. However, one could also argue that a higher resolution data set requires more photons to display high-frequency shadows.

6.2 Evaluation and Performance

The performance of our method is affected by a number of parameters, the most important being:

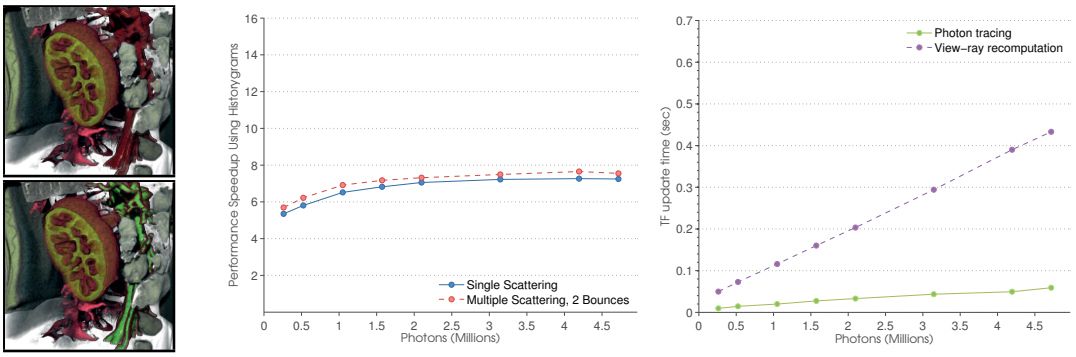
- P_n - The number of photons traced into the scene.
- V_n - The number of view-ray segments.
- S_n - The maximum order of scattering events used in photon tracing.

To evaluate the effect of these parameters on the TF update performance, we varied the material properties of one artificial and one medical data set. In the artificial Cornell box data set ($256 \times 256 \times 256$ voxels) we enabled nearest neighbor filtering to avoid partial volume effects and changed the color the sphere (see figure 7). For the medical data set we chose the Cenovix data set ($361 \times 331 \times 361$), where the vessel material was changed (see figure 8). Figure 7(b) and 8(b) shows how our method compare to a GPU volume photon mapper without our Historygrams. For both single, $S_n = 1$, and multiple scattering, $S_n = 2$, our method has a considerable speedup (≈ 15 times for the Cornell box and ≈ 8 times for the Cenovix data set). The constant speedup for varying number of photons occur since the relative amount of recomputation needed is approximately the same. For the Cornell box, the spatial locality of the data allows for better usage of the hardware, both in terms of cache coherency and work-group utilization which result in increased performance improvement. Figure 7(c) and 8(c) shows the update time for photon tracing and view-ray segments



(a) (b) (c)

Fig. 7. Performance evaluation for updating the material properties of the sphere in a synthetic Cornell Box data set ($256 \times 256 \times 256$ voxels) using 7 view-ray segments. (b) shows the relative speedup using our method compared to a standard GPU photon mapper. Consistent speedups of an order of magnitude are achieved for both single and multiple scattering. (c) shows the update time for retracing invalid photons and recomputing invalid view-ray segments using different number of photons in the scene (generated with single scattering).



(a) (b) (c)

Fig. 8. Performance evaluation for updating the diffuse color of the blood vessels in the Cenovix data set ($361 \times 331 \times 361$) using 7 view-ray segments. (b) shows the relative speedup using our method compared to a standard GPU photon mapper, speedups in the order of $6-8$ times are achieved for both single and multiple scattering when using more than 0.5 million photons. (c) shows the update time for retracing invalid photons and recomputing invalid view-ray segments using different number of photons in the scene (generated with single scattering).

for different number of photons traced into the scene, P_n. From these two plots, we can see that the photon gathering is considerably more expensive than the photon tracing. Figure 9 shows the effect of TF updates when varying the number of view-rays segments.

In table 1, the parameters used for each data set rendered in the paper is presented together with the update time for both the photon tracing and photon gathering stage. In the "Without Historygrams" column, the performance measurements are shown for the GPU photon mapper which recomputes everything for all changes. For the column "With Historygrams", the number of view-ray segments, the initialization overhead, and the amount of recomputation shown in addition to the performance timings. The initialization overhead measures the additional time it takes to create and store the Historygrams, which is only performed once when the camera stops moving. The speedup gained varies depending on the amount of recomputations that are needed and the spatial locality of the data. For changes affecting the outer parts of the volume, such as 6(a) almost all of the light transport needs to be recomputed. However, even for these visually large changes it is beneficial to use Historygrams.

6.2.1 Memory consumption

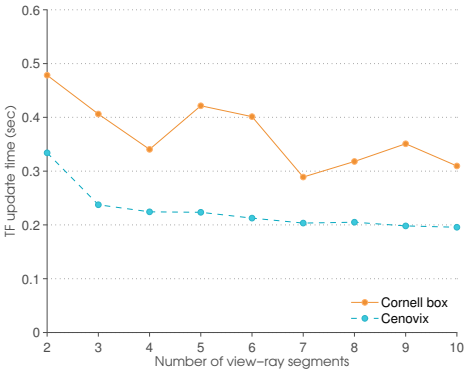
We have also evaluated the overhead of storing and querying the Historygrams used in our method. Each photon interaction requires one Historygram in addition to the regular data which consist of the position, direction and power. To trade speed for lower memory consumption one could also use a single Historygram per photon. Each view-ray segment requires one color in addition to the Historygram. Note that the memory requirement for view-ray segments is independent on the sampling rate. For the color, we use the half data type which means that 8 bytes are required for the red, green, blue, and alpha components.

Using 8 bytes (64 bits) for the Historygram and 8 bytes for the color, our method implies a 16 MB GPU memory overhead per view-ray segment for a viewport resolution of 1024×1024. Similarly for photons we generally use 8 bytes per photon Historygram, implying a minor memory overhead.

7 Conclusions and Future Work

In this paper we have enabled the use of photon mapping in a interactive DVR pipeline. To reach interactive frame rates during parameter changes, which is of crucial importance in explorative volume rendering, we have introduced and applied the novel concept of Historygrams. Historygrams encode the history of a photon or a view-ray, which can then be used to only recompute photons and view-rays affected by specific parameter changes such as TF edits. No approximations are introduced in the final solution since all invalid photons and view-ray segments are recomputed, but we can still accomplish an order of magnitude speedup. We have shown results using several examples of real-world data sets yielding realistic renderings, as well as synthetic data used for controlled performance measurements.

We would like to point out that the concept of Historygrams can also be used for other rendering techniques. For general path tracing methods, e. g., path tracing or bi-directional path tracing our concept is directly applicable. The only requirement is that each path needs to be stored and incrementally update its own Historygram based on the parameters affecting the path generation. Recent methods for volumetric



(a)

Fig. 9. Update times for changing the TF when varying the number of view-ray segments using 2.1 million photons and single scattering. The solid orange line corresponds to changing the material properties of the sphere in the Cornell box, figure 7(a), and the blue dashed line represents the performance for the Cenovix data set scenario, figure 8(a).

photon mapping use photon beams, both for photon queries and data representation [12]. Our method is also applicable for these methods, essentially a separate Historygram needs to be stored for each photon beam used for representing incident illumination in the scene.

Acknowledgments

We thank all reviewers for their valuable comments which helped to greatly improve this paper. We would also like to thank Erik Sundén for supplying transfer functions, and Matthew Cooper for proofreading the manuscript. This work was partly supported by grants from the Excellence Center at Linköping and Lund in Information Technology (ELLIIT) and the Swedish e-Science Research Centre (SeRC). The presented concepts have been realized using the Voreen open source visualization framework (www.voreen.org).

References

[1] D. A. Alcantara. *Efficient Hash Table on the GPU*. PhD thesis, University of California, Davis, California, USA, 2011.

[2] M. Ashikhmin and P. Shirley. An anisotropic phong brdf model. *Journal of Graphics Tools*, 5:25–32, 2000.

[3] U. Behrens and R. Ratering. Adding shadows to a texture-based volume renderer. In *IEEE Symposium on Volume Visualization*, pages 39–46, 1998.

[4] E. Cerezo, F. Perez, X. Pueyo, F. Seron, and F. Sillion. A survey on participating media rendering techniques. *The Visual Computer*, 21(5):303–328, 2005.

[5] M. Z. Claude Knaus. Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 20011)*, 30(3), 2011.

[6] K. Dmitriev, S. Brabec, K. Myszkowski, and H.-P. Seidel. Interactive global illumination using selective photon tracing. In *Proc. EGSR*, pages 25–36, 2002.

[7] T. Engelhardt and C. Dachsbacher. Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In *Symposium on Interactive 3D Graphics and Games*, I3D '10, pages 119–125. ACM, 2010.

[8] A. L. Gilchrist. The perception of surface blacks and whites. *Scientific American*, 240(3), 1979.

[9] T. Hachisuka, S. Ogaki, and H. W. Jensen. Progressive photon mapping. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2008)*, pages 130:1–130:8, 2008.

[10] M. Hadwiger, A. Kratz, C. Sigg, and K. Bühler. Gpu-accelerated deep shadow maps for direct volume rendering. In *Graphics hardware*, 2006.

[11] F. Hernell, P. Ljung, and A. Ynnerman. Local Ambient Occlusion in Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):548–559, 2010.

[12] W. Jarosz, D. Nowrouzezahrai, I. Sadeghi, and H. W. Jensen. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(1):5:1–5:19, Jan. 2011.

[13] W. Jarosz, M. Zwicker, and H. W. Jensen. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proceedings of Eurographics 2008)*, 27(2):557–566, Apr. 2008.

[14] H. W. Jensen. Global illumination using photon maps. In *Rendering Techniques*, pages 21–30. Springer-Verlag, 1996.

[15] H. W. Jensen and P. H. Christensen. Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of SIGGRAPH 1998*, SIGGRAPH '98, pages 311–320. ACM, 1998.

[16] D. Jönsson, E. Sundén, A. Ynnerman, and T. Ropinski. Interactive Volume Rendering with Volumetric Illumination. *Eurographics STAR 2012*, 31, 2012.

[17] D. Kersten and A. Hurlbert. Discounting the color of mutual illumination: A 3d shape-induced color phenomenon. *Investigative Ophthalmology and Visual Science*, 32(3), 1996.

[18] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson. A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):150–162, 2003.

[19] T. Kroes, F. H. Post, and C. P. Botha. Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE*, 2012. Accepted, to appear.

[20] J. Kronander, D. Jönsson, J. Löw, P. Ljung, A. Ynnerman, and J. Unger. Efficient visibility encoding for dynamic illumination in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):447–462, 2012.

[21] M. Langer and H. Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29:649–660, 2000.

[22] F. Lindemann and T. Ropinski. Advanced light material interaction for direct volume rendering. In *IEEE/EG Int. Symp. on Volume Graphics*, pages 101–108, 2010.

[23] F. Lindemann and T. Ropinski. About the Influence of Illumination Models on Image Comprehension in Direct Volume Rendering. *IEEE TVCG(Vis Proceedings)*, 17(12):1922–1931, 2011.

[24] J. Löw, J. Kronander, A. Ynnerman, and J. Unger. Brdf models for accurate and efficient rendering of glossy surfaces. *ACM Trans. Graph.*, 31(1):9:1–9:14, Feb. 2012.

[25] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[26] D. Merrill and A. Grimshaw. High Performance and Scalable Radix Sorting: A case study of implementing dynamic parallelism for GPU computing. *Parallel Processing Letters*, 21(02):245–272, 2011.

[27] T. Ritschel, T. Grosch, C. Dachsbacher, and J. Kautz. State of the art in interactive global illumination. *Computer Graphics Forum*, 31(31):160–188, 2012.

[28] T. Ropinski, C. Döring, and C. Rezk Salama. Advanced Volume Illumination with Unconstrained Light Source Positioning. *IEEE Computer Graphics and Applications*, 2010.

[29] T. Ropinski, C. Döring, and C. Salama. Interactive volumetric lighting simulating scattering and shadowing. In *PacificVis (IEEE Pacific Visualization)*, 2010.

[30] T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. Hinrichs. Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum (Proceedings of Eurographics 2008)*, 27(2):567–576, 2008.

[31] C. Salama. Gpu-based monte-carlo volume raycasting. In *Pacific Conference on Computer Graphics and Applications*, pages 411–414, 2007.

[32] M. Schott, V. Pegoraro, C. Hansen, K. Boulanger, and K. Bouatouch. A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum (Proceedings of EG/IEEE Symposium on Visualization 2009)*, 28(3):855–862, 2009.

[33] E. Sundén, A. Ynnerman, and T. Ropinski. Image plane sweep volume illumination. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2125–2134, 2011.

[34] V. Šoltészová, D. Patel, S. Bruckner, and I. Viola. A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum (Eurographics/IEEE VGTC Symp. on Visualization 2010)*, 29(3):883–891, 2010.