Special Section on CEIG

Transient instant radiosity for efficient time-resolved global illumination

Xian Pan*, Victor Arellano, Adrian Jarabo

Universidad de Zaragoza, I3A, Zaragoza, 50018, Spain

ARTICLE INFO

Article history:
Received 1 May 2019
Revised 16 July 2019
Accepted 22 July 2019
Available online 1 August 2019

Keywords:
Transient rendering
Transient instant radiosity

ABSTRACT

Over the last decade, transient imaging has had a major impact in the area of computer graphics and computer vision. The ability of analyzing light propagation at picosecond resolution has enabled a variety of applications such as non-line of sight imaging, vision through turbid media, or visualization of light in motion. However, despite the improvements in capture at such temporal resolution, existing rendering methods are still very time-consuming, requiring a large number of samples to converge to noise-free solutions, therefore limiting the applicability of such simulations. In this work, we generalize instant radiosity, which is very suitable for parallelism in the GPU, to transient state. First, we derive it from the transient path integral, including propagation and scattering delays. Then, we propose an efficient implementation on the GPU, and demonstrate interactive transient rendering with hundreds of thousands of samples per pixel to produce noiseless time-resolved renders.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

In 1964, Prof. Harold Edgerton captured the famous photo *Bullet Through Apple* in the MIT laboratory, where he was able to visualize the movement of a bullet traveling at a speed of approximately 850 m/s, or an obturation time of 4–10 μs. Fifty years later, Velten et al. introduced *femto-photography* [1], which allowed to capture videos at an effective exposure time of 1.85 ps per frame (roughly a trillion frames per second), allowing to capture the propagation of light. The development of this technology has led to a new area in computational imaging, termed *transient imaging* [2]: this field breaks the classical assumption in computer vision and computer graphics of infinite speed of light, allowing the use of such information in numerous applications such as capturing and recognizing materials [3], reconstructing non-line-of-sight (NLOS) geometry [4–6], or imaging through turbid media [7].

Influenced by remarkable advances in the field of *ultra-fast imaging*, and due to the need of reproducing the observed phenomena in order to perform analysis by synthesis, generation of ground truth data in reconstruction techniques, machine learning, etc., simulation of this type of light transport has become more and more important. In 2008, Smith et al. [8] proposed the first framework within the context of the traditional render equation [9], which was later formalized by Jarabo et al. [10], extending the classic *path integral* [11] by adding the temporal domain in the form of propagation and scattering delays.

The use of path integral in rendering techniques such as path tracing has led to realistic results using stochastic integration. However, in order to obtain high quality images, a large number of samples must be processed to reduce stochastic error. Consequently, this results in a high computation time cost. This downside effect is even more critical in the case of transient rendering, where several images need to be generated to visualize how light travels through the scene. Moreover, the spatio-temporal structure of the transient light transport signal makes the traditional sampling and reconstruction techniques [12] unsuitable for this new domain, since most of them generate paths incrementally without any control of the total path length; this lack of control makes very difficult to reconstruct specific frames of an animation, which was alleviated by using different density estimation techniques along the temporal domain [10]. Unfortunately, this still results in the need of generating several thousands of samples to obtain a low-error output. This limits the use of the transient render in, for example, inverse rendering [13] or machine learning problems [14], where hundreds or thousands of renders may be necessary to achieve a solution or to have a suitable training dataset.

Therefore, the use of approximate global illumination is needed, which does not aim for exact results, but robust approximations of them in order to reduce computation time. Additionally, since rendering is a very parallelizable process, it is especially suitable to be implemented in the GPU. In this work, we focus on develop-
ing a realistic efficient transient renderer, that leverages the GPU computing capacities. We adopt the instant radiosity method [15], that transforms the rendering equation into a many-lights problem, and generalize it to transient state. Then, we demonstrate a GPU-friendly implementation, that allows very efficient renderings of indirect time-resolved global illumination. We are confident that our work will serve in many applications using transient imaging, to serve as a powerful tool for generating datasets for benchmark and training, as well as a forward model for inversion techniques.

2. Related work

The field of physically-based rendering is an extensive domain, with a large number of papers addressing the subject. Here we focus on those most related to our work, and refer the reader to other sources for more details on the field in general [16,17].

Transient rendering. The first to introduce the time of flight of light into the render equation [9] were Smith et al. [8]. Their model provide a solid theory basis for transient rendering, but do not propose a practical framework for the estimation of transient global illumination. Jarabo et al. [10] presented a generalized formulation for light propagation based on the path integral, including scattering and propagation delays in bidirectional path tracing (BDPT). They also proposed sampling techniques adapted to the temporal domain to reduce the resulting variance. Pitts et al. [18] and Adam et al. [19] followed a similar approach, but with a slower convergence rate. Later, Jarabo and Arellano [20] showed that transient light transport can be modeled as a vector-based version of the path integral. Iseringhausen and Hullin [21] proposed an efficient GPU implementation of a two-bounces forward path tracing, in the context of NLOS reconstruction via non-linear optimization. Alternatively, Meister et al. [22,23] used a transient version of photon mapping, resulting in a robust estimation of light transport, and allowing caustic rendering in the transient domain. Ament et al. [24] also used transient photon mapping to solve the refractive radioactive transport equation, where they propose a solution for rendering participating media with different refractive indices. Marco et al. [25] proposed a transient version of the photon beams algorithm for efficient rendering of participating media.

Both BDPT and photon tracing provide robust results when a sufficient number of samples are sampled at the cost of hours and hours of GPU computation. The time-consuming generation of sequential transient videos dramatically limits their use. Several algorithms already exist that try to solve the problem but within a different application domain, mainly in the area of non-line-of-sight imaging, which takes advantage of GPU parallelism to efficiently reconstruct the geometry of occluded objects using the transient state of light: Hullin [26] and Klein et al. [27] demonstrated that a transient image can be approximated using a transient version of the GPU radiosity methods. Our method follows the same lines, to obtain transient videos in an interactive time using GPU-based approximation techniques.

Interactive global illumination. Even though ray tracing techniques (e.g. path tracing) can be implemented on GPUs [28–30], complex scenes are generally not suitable for such methods, requiring heavy denoising for noise-less solutions. Instead, using methods based on virtual lights (VPLs)[31], are very suitable for GPU rasterization, do not have this type of limitation, making them widely used for different applications: Keller [15] introduced instant radiosity (IR), which approximates the indirect illumination using a set of virtual point lights placed at the reflecting indirect surfaces. Several extensions were later proposed to improve both the VPL generation process such as bidirectional IR [32], Metropolis IR [33], or to reduce singularities that can be caused by the use of VPLs [34–36]. In order to generate VPLs in real-time, Dachsacher et al. [37] proposed reflective shadow map (RSM) to use from the light source GPU rasterization to store at each pixel the position, normal and color of the visible object from the camera, so that each pixel represents a VPL. Later, they proposed to splat indirect illumination [38] from the VPLs. To accelerate visibility, Ritschel et al. [39] proposed to approximate the visibility from the shadow maps allowing indirect occlusion at real-time rates. We generalize instant radiosity to resolve transient light transport, using RSM to efficiently generate VPLs in GPUs.

3. Transient instant radiosity

Here we develop a time-resolved generalization for instant radiosity [15]. First we provide an overview of the original method. Then, we develop its time-resolved version by making use of the general path integral formulation.

3.1. Background: Instant Radiosity

The rendering equation [9] models the outgoing radiance \( L_o(\mathbf{x} \rightarrow \omega_o, t) \) at a point \( \mathbf{x} \) in a surface in direction \( \omega_o \), as

\[
L_o(\mathbf{x} \rightarrow \omega_o) = \int_{\Omega} L_i(\mathbf{x} \leftarrow \omega_i) \rho(\omega_i \rightarrow \mathbf{x} \rightarrow \omega_o) \cos \theta_i \, d\omega_i. \tag{1}
\]

where \( L_i(\mathbf{x} \rightarrow \omega_i) \) is the emitted radiance, \( L_i(\mathbf{x} \rightarrow \omega_i) \) is the incoming radiance from direction \( \omega_i \), \( \rho(\omega_i \rightarrow \mathbf{x} \rightarrow \omega_o) \) is the BRDF at \( \mathbf{x} \), and \( \theta_i = \angle \mathbf{n}_o \mathbf{n} \) is the angle between the normal at \( \mathbf{x} \) and \( \omega_i \). Instant radiosity [15] solves Eq. (1) by using a two-pass algorithm, where light particles are traced from the light, and then used to compute the illumination reflected from the surfaces visible from the camera. Such particles model the reflected radiance at their interaction point, acting as virtual point lights (VPL). Then, in the rendering pass, the rendering equation at point \( \mathbf{x} \) is solved by just computing the contribution of each VPL into \( \mathbf{x} \) as

\[
L_o(\mathbf{x} \rightarrow \omega_o) \approx L_o(\mathbf{x} \rightarrow \omega_o) + \sum_{i=1}^{N} L_i^t(\mathbf{x} \rightarrow \omega_o), \tag{2}
\]

with

\[
L_i^t(\mathbf{x} \rightarrow \omega_o) = \frac{L_i^t(\mathbf{x}_i \rightarrow \mathbf{x}) V(x \leftarrow x)}{|x - x_i|} \rho(\omega_i \rightarrow \mathbf{x} \rightarrow \omega_o) \cos \theta_i. \tag{3}
\]

where \( N \) is the number of VPLs in the scene, \( L_i^t(\mathbf{x}_i \rightarrow \mathbf{x}) \) is the radiance emitted by the VPL \( i \) placed at \( \mathbf{x}_i \) towards \( \mathbf{x} \), and \( V(x \leftarrow x) \) is the binary visibility function between \( \mathbf{x} \) and \( \mathbf{x}_i \).

3.2. Transient instant radiosity

In order to define our time-resolved generalization of VPL-based global illumination, we will first generalize the rendering equation [Eq. (1)] to transient state, by including the temporal dependence as

\[
L_o(\mathbf{x} \rightarrow \omega_o, t) = L_c(\mathbf{x} \rightarrow \omega_o, t) + L_r(\mathbf{x} \rightarrow \omega_o, t), \tag{4}
\]

\[
L_r(\mathbf{x} \rightarrow \omega_o, t) = \int_{\Omega} \int_{\Omega} L_i(\mathbf{x} \leftarrow \omega_i, t') \rho(\omega_i \rightarrow \mathbf{x} \rightarrow \omega_o, t - t') \cos \theta_i \, dt' \, d\omega_i. \tag{5}
\]

where we include the temporal emission profile in the emission \( L_o(\mathbf{x} \rightarrow \omega_o, t) \), as well as the scattering delays encoded in \( \rho(\omega_i \rightarrow \mathbf{x} \rightarrow \omega_o, t - t') \), which can be due to e.g. multiple internal reflections within micro-geometry [40,41] or inelastic scattering effects.
such as fluorescence [42]. Last, \( t_0 \) is an initial time usually set to \( t_0 = -\infty \). To solve Eq. (4), we can define our transient instant radiosity by accounting for the temporal domain in Eqs. (2) and (3) as

\[
L_t(x \rightarrow \omega_0, t) \approx L_t(x \rightarrow \omega_0, t) + \sum_{i=1}^{N} L_i^t(x \rightarrow \omega_0, t),
\]

(6)

\[
L_i^t(x \rightarrow \omega_0, t) = \frac{\cos \theta V(x_i \rightarrow x)}{|x - x_i|^2}
\times \int_{t_i}^{t_i + \Delta t_i} l_i^t(x \rightarrow x, t_i(t) - t') \rho(\omega_i \rightarrow x \rightarrow \omega_0, t - t') \, dt',
\]

(7)

where \( t_i(t) = t - t(x \rightarrow x_i) \) accounts for the propagation delay due to light travelling from the light source to \( x \) as

\[
l(x \rightarrow x_i) = \int_{s_x} \eta(x_\xi) \, ds_x.
\]

(8)

where \( r \) parametrizes the path of light between the two points, \( s_x \) and \( s_x \), are the parameters of the path at \( x \) and \( x_i \), respectively, \( c \) is the speed of light in vacuum and \( \eta(x_\xi) \) represents the index of refraction of the medium at \( x \). In order to solve Eqs. (6) and (7) we therefore need to define the time-resolved emitting function \( L_i^t(x \rightarrow \omega_0, t) \). To do so, we will make use of the path integral view of instant radiosity.

**Transient Instant Radiosity as a Path Integral Estimator.** In order to define our time-resolved generalization of VPL-based global illumination, we will work within the path integral [11] framework. As shown by Walter et al. [43] and Krivanek et al. [44], instant radiosity can be understood as a bidirectional path tracer where the light paths are reused along all eye paths. While this generates some artifacts due to correlation, it ensures relatively small variance in smooth areas. Let us start with the transient path integral proposed by Jarabo et al. [10], which defines the light contributing in a (time-resolved) pixel \( I \) as the integral over the all light transport paths \( \Omega \) arriving to this pixel as

\[
I = \int \int \int_{\Delta T} f(x, \Delta t) \, d\mu(\Delta t) \, d\mu(x),
\]

(9)

where \( \Delta T \) is the space of temporal delays of all paths, \( x = x_0 \ldots x_k \) is the full light path of length \( k \), consisting of \( k + 1 \) vertices, where vertices \( x_0 \) and \( x_k \) lie on a light source and camera sensor, respectively, and \( x_1 \ldots x_{k-1} \) are intermediate scattering vertices. The differential measure \( d\mu(x) \) is the area integration for surfaces vertices. Finally, \( \Delta T = \Delta t_0 \ldots \Delta t_k \) represents the sequence of scattering delays in \( \rho(\omega_0 \rightarrow x \rightarrow \omega_0, t - t') \) along the path, and \( d\mu(\Delta t) \) is the temporal integration measure at each path vertex. The path contribution \( f(x, \Delta t) \) is defined as

\[
f(x, \Delta t) = L_{\omega_0} (x_0 \rightarrow x_1, \Delta t_0) f(x, \Delta t) W_{\omega_0}(x_{k-1} \rightarrow x_k, \Delta t_k). \]

(10)

where \( L_{\omega_0} \), \( W_{\omega_0} \) and \( f \) are the time-dependent emission, sensor importance, and path throughput, respectively. The sensor importance \( W_{\omega_0} \) defines the spatial and angular sensitivity as in the steady-state path integral, but also the region of time we are interested in evaluating (e.g. in a streak-camera or SPAD sensor, the temporal dependence of \( W_{\omega_0} \) for a pixel-frame would be the temporal bucket of that particular frame). The transient path throughput is defined as:

\[
\Delta T = \left[ \prod_{i=1}^{k-1} \rho(x_i, \Delta t_i) \right] \left[ \prod_{i=0}^{k-1} G(x_i, x_{i+1}) V(x_i, x_{i+1}) \right].
\]

(11)

where \( \rho(x_i, \Delta t_i) \) is the scattering event at \( x_i \), \( G(x_i, x_{i+1}) = \cos \theta_i \cos \theta_{i+1} |x_i - x_{i+1}|^{-2} \) is the geometric term, and \( V(x_i, x_{i+1}) \) is the binary visibility between \( x_i \) and \( x_{i+1} \). Note that Eq. (11) is essentially the same as in the classic path integral, with the exception that the scattering operator \( \rho(x_i, \Delta t_i) \) includes a temporal dependence modeling the scattering delay described before.

In order to fully define a transient light path, we need to account for the spatial, and temporal coordinates. The temporal coordinate at each path vertex \( x_i \) is \( t_i \), computed as

\[
t_i = \sum_{j=0}^{i-1} (f(x_j \rightarrow x_{j+1}) + \Delta t_j), \quad t_i = t_i + \Delta t_i,
\]

(12)

where \( t_0 \) and \( t_k \) are the emission and detection times of a light path, and \( f(x_j \rightarrow x_{j+1}) \) is the propagation delay between vertices \( x_j \) and \( x_{j+1} \) defined following Eq. (8).

The transient path integral (9) can be numerically approximated using a Monte Carlo estimator:

\[
\int I = \frac{1}{M} \sum_{j=1}^{M} \int_{\Omega} \int_{\Delta T} \frac{f(x_j, \Delta t_j)}{p(x_j, \Delta t_j)} \rho(x_j, \Delta t_j) G(x_j, x_{j+1}) \rho(x_{j+1}, \Delta t_{j+1}) V(x_j, x_{j+1}) \]

(13)

which averages \( M \) random paths \( x_j, \Delta t_j \), drawn from a spatio-temporal probability distribution (pdf) \( p(x_j, \Delta t_j) \) defined by the chosen path and time delay sampling strategy. Similar to bidirectional pathtracing, methods based on VPLs compute Eq. (13) by sampling a set of subpaths from the light source (called light subpaths) \( \Sigma x_j = x_0 \ldots x_k \), of length \( k \in [1, k-1] \), and connecting them with the subpaths generated from the camera (eye subpaths) \( \Sigma x_j = x_{k+1} \ldots x_l \), via deterministic shadow connections, to generate a full path \( \Sigma x = [\Sigma x_j, \Sigma x_0] = x_0 \ldots x_k x_{k+1} \ldots x_l \), where [·] is the concatenator operator. The main difference with respect to bidirectional path tracing is that the light subpaths are reused for all eye subpaths in the image, so that \( \int I \) becomes

\[
\int I = \frac{1}{MN} \sum_{j=1}^{M} \sum_{i=1}^{N} \int_{\Omega} \int_{\Delta T} \frac{f(x_j, \Delta t_j)}{p(x_j, \Delta t_j)} \rho(x_j, \Delta t_j) G(x_j, x_{j+1}) \rho(x_{j+1}, \Delta t_{j+1}) V(x_j, x_{j+1}) \]

(14)

where \( f(x_j, \Delta t_j) = \int_{\Omega} \int_{\Delta T} W_{\omega_0}(x_{k-1} \rightarrow x_k, \Delta t_k) \)

\[
\]

and \( f(x_j, \Delta t_j) = L_{\omega_0} (x_0 \rightarrow x_1, \Delta t_0) f(x, \Delta t) \)

\[
\]

are measurement of the eye and light subpaths, \( x_j \) and \( x_j \), respectively, \( M \) is the number of eye subpaths in a pixel, and \( N \) is the number of VPLs generated.

Eq. (14) implicitly relates with Eqs. (6) and (7) by defining the emission term \( L_i^t(x \rightarrow \omega_0, t) \) as

\[
L_i^t(x \rightarrow \omega_0, t) = \frac{1}{N} \delta(t - t_i) \int_{\Omega} \int_{\Delta T} \frac{f(x_j, \Delta t_j)}{p(x_j, \Delta t_j)} \rho(x_j, \Delta t_j) \cos \theta_{j}. \]

(15)

where \( \delta(t - t_i) \) is the Dirac delta function modeling the impulse emission of the VPL, where \( t_i \) is the emission time of the VPL computed using Eq. (12). Note the integration along the temporal domain in Eq. (7) is implicitly solved numerically by the sampled variable \( \Delta t \). In essence, in the first pass of the algorithm we generate a set of VPLs by sampling a set of light subpaths; each VPL would have a spatio-temporal emitting function defined by Eq. (15). Then, in render time we simply evaluate Eqs. (6) and (7) using Eq. (15).

### 4. Implementation

We implemented our transient instant radiosity renderer using OpenGL and Java. In order to fit it into the raster pipeline of the GPU, we perform a set of simplifications. First, we assume
that only one- and two-segment paths exist in the scene. Taking Eq. (9) and explicitly separating bounces, we get

$$l = \sum_{b=1}^{\infty} \int \int_{\Delta t^b} f(\mathbf{x}, \Delta t) \, d\mu(\Delta t) \, d\mu(\mathbf{x}).$$

where $\Omega^b$ and $\Delta t^b$ are the space of paths and temporal delays of paths of $b$ segments. By clamping the number of segments to $b = 3$ we end up having direct and one-indirect-bounce lighting, as well as light incoming directly from the light source to the camera. This results into an energy loss, but according to the study of Tabellion and Lamorlette [45], a single indirect light bounce is enough to obtain plausible results in many cases. Note however that this claim is based on steady-state images; in transient state it would mean that longer path lengths (i.e. light arriving at longer paths) will be lost. However, in most practical applications [2] most information is encoded in the first and second-order scattering. The second approximation consists into assuming that all scattering delays are negligible, and that can be approximated by a delta function. This allows us to trivially solve the integrand along the temporal domain. Finally, we assume all surfaces to be perfectly diffuse. These last two approximations do not introduce bias in the final result but limit the transient phenomena that our implementation can simulate.

For rendering, we use the two classical passes of instant radiosity (VPL generation, and rendering), and introduce a last step for reconstructing the temporal dimension of the scene. For this last step, following the terminology proposed by Jarabo et al. [10], we use the histogram density estimation for reconstruction, due to its simplicity to work with delta functions. More advanced reconstruction approaches are left as future work.

**Generating the VPLs.** In order to generate the VPLs on the GPU, we rely on reflective shadow maps (RSM) [37]: we render the scene from the point of view of the light source, storing the surface albedo $\alpha(p_i)$, normal $\mathbf{n}_i(p_i)$, and distances from the camera $d(p_i)$ for each pixel $p_i$ in the RSM. With these parameters, and leveraging the assumption of two-bounces light transport and diffuse reflectance, we can define Eq. (15) for each VPL as

$$l^t_i(\mathbf{x} \rightarrow \omega_2, t) \approx \delta(t - t_i) \frac{L(p_i)}{\Omega(p_i)} \frac{\alpha(p)}{\pi} \cos \theta_k,$$

where $t_i = d(p_i) n_i / c$, and $\Omega(p)$ is the solid angle of pixel $p$, which decreases quadratically with the number of VPLs $N$, and $L(p)$ is emitting function of the light source, which in our case is a spotlight with constant emission along the light frustum. After generating the RSM values in the first pass of the render, which are four independent textures, these are used in the final pass to calculate the global illumination in the rendering pass.

**Rendering.** In the second render pass, we compute Eq. (6) from the VPLs generated in the first pass, by using a deferred rendering approach. In order to do so, we need to compute the visibility function $V(\mathbf{x}, \mathbf{x}_i)$ between the shaded point $\mathbf{x}$ and the VPL. In order to compute it efficiently, we use an approximated visibility based on imperfect shadow maps (ISM) [39], where a large number of hemispherical shadow maps are generated for each VPL in parallel by using a coarse point-based approximation of the scene.

The use of VPLs provides good results for the calculation of global illumination, but introduces certain errors that are easily perceptible to the human eye when the number of VPLs used is low. One of them is the existence of lighting peaks in the scene due to the singularity produced by the inverse squared distance of the geometric term. To remove artifacts due to this singularity, we clamp the geometric term to a user specified value. We opt for this approach since for diffuse scenes the energy lost introduced by clamping does not result into significant appearance changes [46]. Note however that this results in an energy lost in the scene, generating dark areas especially in cavities and corners.

**Generating the space-time volumes.** Finally, once the space-time radiance samples are simulated, we need to use them to reconstruct the transient video. Unlike a stationary render, which uses a 2D texture to store the result (for each point, saves the sum of all radiance), in a transient render we save the data in a 3D texture. For that, we use an approach based on histogram density estima-
4.1. Discussion

Our algorithm is bounded by the memory requirements imposing by storing the results in a large 3D texture, as well as the auxiliary buffers needed for the VPLs computation. To store an HDR color video with resolution $X$ and $Y$ and duration $t$ (in frames) we need $16 \times X \times Y \times t$ bytes. For example, for a video with resolution $1024 \times 1024$ and 250 frames, we need 4GB approximately. The auxiliary buffers of the VPLs, mainly the RSM and the ISM additionally require a storage of order $O(N \times X_s \times Y_s)$, with $N$ the number of VPLs, and $X_s$, $Y_s$ the resolution of each shadow map.

As a result, it is not possible to store all the information in the GPU at the same time when GPU memory is insufficient. We address that by processing the VPLs in batches, so that a subset of VPLs are generated, processed, and removed from memory before the next one is, and then the next one is processed. We have made two segmentations: one for the ISM (batching the VPLs as they are processed) and the final transient volume (splitting the video along the spatial domain). Algorithm 1 shows a pseudocode of the implementation: We iterate on the different batches of the transient volume, calculate the corresponding direct and indirect lighting in GPU for each batch, and then transfer it back to the main memory, freeing space for the results of the next batch. We do not store any ISM on the main memory; while this means that we do not amortize the cost of computing them between batches, we found that the time to regenerate them in GPU is much faster than the memory transfer that would involve a CPU-to-GPU load.

5. Results

In the following, we present the results provided by our implementation, generated on workstation with a 2.8 GHz CPU and a NVIDIA GeForce 1060. The results are rendered with $1024 \times 1024$ camera resolution and 250 frames of duration (around 10 s). All the render times specified below refer to GPU render time plus GPU-to-CPU data transfer time. Videos of the shown results are included as supplementary material.
The selection of parameters for our methods has a big influence on both the final quality and the computation time. These parameters are: number of VPLs, number of points per VPL and ISM resolution (the shadow map resolution for each VPL). Figs. 1–3 compare the effect of these parameters and their costs. A logarithmic tone-mapping function has been used for visualization. Frames 100, 145 and 185 have been chosen for the comparison.

Fig. 4 shows the dependence of the render time on the temporal resolution. This parameter has no effect on the GPU render time given the histogram method used. However, it affects the transfer time of the GPU-to-CPU data due to texture size variation.

Fig. 5 shows the comparison of the algorithm performance when varying the different parameters. We can observe that as the dimension of the parameters increases, the fps decreases. The most significant change occurs with the number of points per VPLs used to generate the ISM, because this number together with the number of VPLs equals the total number of points that it is necessary to pass to GPU to compute the ISM.

The selection of the VPL count is quite important: A small number of VPLs can cause artifacts due to banding on the signal, while a large number of VPLs increases the computational cost of the algorithm.

Finally, setting the number of points per VPL and ISM resolution is closely related, and the quality also depends on the complexity of the scenes and the geometry of the objects. In simple scenes a low number of both parameters can be used, whereas when the complexity of the scene grows (number of objects, geometry complexity, scene size), more points are needed to correctly define the scene, increasing the rendering time as shown in Figs. 6 and 7.

6. Conclusions and future work

In this work we have presented a method to efficiently render scenes in transient state: we have reformulated the light transport in transient state using instant radiosity, so that it is possible to solve the calculation in interactive time. And we have implemented this calculation in GPU taking advantage of its parallelism, using techniques such as reflective shadow map and imperfect shadow map. We believe that this GPU technique will enable novel transient-based applications based on e.g. direct rendering for optimization or machine learning. In that context, our technique will be very useful to create novel databases using transient rendering (e.g. [48]).

In the future, we intend to reduce the number of VPLs used so that we can improve computational time, using more sophisticated and modern algorithms. In addition, our renderer offers a first approximation of transient light transport in GPU applying different assumptions, so it only shows limited effects. One of the future steps is to eliminate these assumptions by incorporating glossy materials, or participating media. Finally, efficiently computing the radiance derivatives to increase the applicability of the method in optimization and learning applications is an interesting avenue of future work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We want to thank the reviewers for their insightful comments. This project has been funded by DARPA (project REVEAL), the European Research Council (ERC) under the EU’s Horizon 2020 research and innovation programme (project CHAMELEON, Grant no. 682080), the Spanish Ministry of Economy and Competitiveness (project TIN2016-78753-P), and the BBVA Foundation through a Leonardo grant. Production baby: Noa Jarabo Martínez.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cag.2019.07.009

References


