Instant Global Illumination on the GPU using OptiX

Ricardo Marques and Luís Paulo Santos

Universidade do Minho, Braga, Portugal,
ricjmarques@gmail.com, psantos@di.uminho.pt

Abstract. OptiX, a programmable ray tracing engine, has been recently made available by NVidia, relieving rendering researchers from the idiosyncrasies of efficient ray tracing programming and allowing them to concentrate on higher level algorithms, such as interactive global illumination. This paper evaluates the performance of the Instant Global Illumination algorithm on OptiX as well as the impact of three different optimization techniques: imperfect visibility, downsampling and interleaved sampling. Results show that interactive frame rates are indeed achievable, although the combination of all optimization techniques leads to the appearance of artifacts that compromise image quality. Suggestions are presented on possible ways to overcome these limitations.

Keywords: instant global illumination, ray tracing, graphics processors

1 Introduction

Interactive ray tracing became possible along the last decade on both CPU and GPU based platforms. However, this has been achieved through extensive optimization of code and data structures, thus developing such a ray tracer is a complex and time consuming task. In September 2009 Nvidia launched a programmable ray tracing engine for their GPUs, OptiX [1], which allows researchers to concentrate on higher level algorithms while still being able to trace rays efficiently.

The goal of this paper is to assess the performance of an interactive global illumination (GI) algorithm on OptiX. This algorithm, referred to as Instant Global Illumination [2], computes indirect diffuse interreflections by generating a particle based approximation of this illumination component, resulting in a three-dimensional distribution of secondary virtual point light (VPL) sources. The algorithm in its original form is barely interactive due to the high number of VPLs. Optimizations have been proposed under the form of imperfect visibility [3], downsampling the indirect diffuse evaluation rate [4] and interleaving VPL sampling patterns [5]. This paper assesses the performance achieved with these three optimizations on a last generation NV 480 GTX GPU using OptiX and proposes a few hypothesis for further performance gains.

The next section presents related work and details on the optimization techniques. Section 3 briefly introduces OptiX, while the used algorithm is detailed

INForum 2010 - II Simpósio de Informática, Luís S. Barbosa, Miguel P. Correia (eds), 9-10 Setembro, 2010, pp. 329–340

in section 4. Results are analyzed in section 5. The paper concludes with some suggestions for future work.

2 Related Work

2.1 Interactive Ray Tracing and Global Illumination

Interactive Whitted-style [6] ray tracing (iRT) became possible along the first decade of the XXIst century, for both static and dynamic scenes, through clever exploitation of advances in available computing power and careful optimization of both code and used data structures [7, 8]. The performance of such ray tracers arises mainly from fine tuning data structures such that the memory hierarchy performs to its maximum and by exploiting ray coherence: rays are grouped into packets or frusta and SIMD instructions are used to trace them through the scene. Ray coherence exploitation has been shown to be effective for primary and shadow rays. Whether it can be used to speedup tracing of secondary rays is still unclear, since these often do not share the same origin and exhibit less directional coherence [9].

Ray tracing is an embarrassingly parallel algorithm which naturally leads to the exploitation of parallel systems. Most the above cited approaches exploit parallelism at several levels: SIMD, multicore and clusters of distributed memory machines. With the computation power and core count of GPUs increasing at a higher rate than those of CPUs, ray tracing solutions that harness the GPUs processing capabilities begun to emerge [10, 11]. However, even with the appearance of flexible, C-like, GPU programming languages such as CUDA [12], efficient programming of these devices is still not straightforward due to their SIMT (Single Instruction Multiple Threads) computing model. NVIDIA has recently made available a GPU ray tracing engine, OptiX [1], which relieves researchers from the idiosyncrasies of efficient ray tracing programming and allows them to concentrate on higher level algorithms, such as global illumination.

While Whitted style ray tracing requires tracing a reasonable number of mostly coherent rays (specular rays may exhibit less coherence, specially over curved surfaces), GI entails simulating a huge number of incoherent light paths, thus making it hard to maintain interactive frame rates. GI light transport phenomena include diffuse interreflections, caustics and participating media. Indirect diffuse interreflections, the focus of this paper, are typically simulated using Monte Carlo path tracing [13], photon mapping [14], irradiance caching [15] or instant radiosity [16]. The later is frequently used for interactive rendering [2, 17]: it generates a particle approximation of the indirect diffuse radiant scene by performing quasi-random walks on a quasi-Monte Carlo integration framework. Photons are traced from the light sources into the scene and Virtual Point Light sources (VPLs) are placed at the intersections of the quasi-random photon paths with diffuse geometry. The image is then rendered by sampling the VPLs as point light sources. This algorithm exhibits ray coherence similar to direct lighting and is thus expected to perform similarly on vectorial processors such as the GPUs.

2.2 Accelerating Indirect Diffuse

The quality of the indirect diffuse estimate is dependent on the number of VPLs and the quality of their distribution throughout the scene. Unfortunately, rendering time is linearly proportional to the number of VPLs, which requires clever strategies to speedup indirect diffuse calculations.These strategies are based on two key observations: indirect diffuse lighting is mostly a low frequency signal that varies smoothly over the scene and accurate visibility is not required for indirect illumination.

Accurate visibility is traditionally used in light transport at the cost of tracing rays against the detailed scene description. However, indirect diffuse illumination varies smoothly across the scene, thus accurate visibility is perceptually unnecessary in this case, since visibility errors are masked by the low frequency nature of the signal [18]. This insight has been exploited by either performing accurate visibility queries only on the neighborhood of the shading point [19] or by testing visibility against a crude representation of the scene [3].

Instant radiosity, on its original form, requires evaluating VPLs visibility at each shading point. The number of shading points is thus linearly correlated with the number of pixels. By taking advantage of the low frequency nature of indirect diffuse reflections, the estimate can be computed at a lower image resolution and then upsampled to the target resolution [4]. Despite being mostly smooth, high frequencies are still present on the indirect diffuse signal, mostly due to geometric discontinuities. Upsampling must thus be done using some discontinuity preserving filter, such as the joint bilateral filter, which uses geometric information obtained at full resolution when computing direct illumination.

Sampling the whole set of VPLs for each pixel is expensive and might compromise performance. Interleaved sampling [5] has thus been proposed to accelerate instant radiosity resulting on the so-called Instant Global Illumination algorithm [2]. The set of VPLs is divided onto $m * n$ subsets and for each pixel within a $m * n$ tile of pixels only one of the subsets is sampled, spawning a much lower number of VPL shadow rays. Results are then integrated over each tile by using the discontinuity buffer.

3 OptiX

NVIDIA's OptiX engine is a programmable ray tracing pipeline for NVIDIA GPUs using the CUDA-C programming language [1, 20]. OptiX abstracts the execution to single rays, simplifying the application programmer's role, while internally exploiting the GPU architectural characteristics through deferred shading and built-in scheduling and load balancing. OptiX is tightly coupled with graphics APIs to allow combinations of raster and ray tracing approaches. The ray tracing pipeline is programmable through programmer supplied programs (CUDA kernels) that handle the various ray tracing events, such as intersections for procedurally accurate surface types, cameras for new composition potential, shading and scene graph traversal. OptiX includes support for parallelism across

multiple GPUs and building and traversal of acceleration structures (BVH and KD trees).

OptiX runs on most CUDA enabled GPUs although it is only fully functional and supported on the latest architectures (GT200 and GF100). Currently, there are two main drawbacks associated with OptiX: acceleration structure traversal is not programmable, which precludes their utilization on tasks other than ray traversal, and the CUDA context upon which OptiX runs is not visible to the programmer, prohibiting explicit access to shared and constant memory, which prevents the utilization of most common CUDA optimization techniques.

The availability of a programmable, high performance, ray tracing engine relieves application developers from the idiosyncrasies of efficient ray tracing programming, allowing them to concentrate on higher level algorithms such as global illumination. However, applications using OptiX must still be carefully designed and optimised if high performance is to be achieved.

4 The Algorithm

The algorithm proposed on this paper simulates direct illumination, specular reflections and indirect diffuse interreflections using the Instant Global Illumination approach [2]. Figure 1 illustrates the fundamental stages of the proposed pipeline; the upper arrow depicts the temporal order of the different stages, while the arrows connecting the boxes illustrate data dependencies.

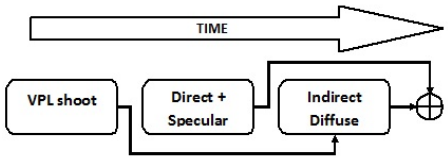


Fig. 1. Rendering pipeline for the canonical version of the algorithm

Particles are shot from the light sources following a quasi-random Halton sequence. VPLs are then created at each intersection of the particles path with geometric primitives whose material has a diffuse component. The number of bounces along each path is a user supplied parameter. The OptiX context launched to shoot the VPLs consists on as many threads as the number of paths, each thread processing a whole path. The quasi-random numbers used to build the particle paths are generated on the CPU and passed to the GPU as OptiX buffers.

Direct plus specular illumination entails shooting one primary ray per pixel, spawning a Whitted-style tree of rays. Only point light sources are supported. Indirect diffuse is only separated from direct plus specular at the primary ray hit point. Hit points further down the rays' tree, resulting from tracing specular rays,

have their indirect diffuse component calculated within the direct plus specular stage of the pipeline. This approach was selected to exploit OptiX recursion capabilities. Explicit separation of components could be implemented by storing the hitpoints on a buffer, but this would increase access to global memory thus increasing rendering time (shared memory can not be explicitly accessed from within the OptiX context).

Indirect diffuse radiance, $L_{indirect}(x)$, is evaluated by shooting, at each shading point x, one ray towards each VPL to assess its visibility:

$$L_{indirect}(x) = \sum_{k=1}^{N} \rho(x) L_{e,k} V(x, y_k) G(x, y_k) \tag{1}$$

where N is the number of VPLs, $V(.,.)$ is the visibility function between two points, $L_{e,k}$ is the emitted radiance for the k-th VPL, y_k is the position of the k-th VPL, ρ is the diffuse reflectance coefficient and G is the bounded geometry term, defined as

$$G(x, y_k) = \frac{cos\theta_x cos\theta_{y_k}}{\|x - y_k\|^2} f(0.8min_d, 1.2min_d, \|x - y_k\|)$$

where θ_x and θ_{y_k} are the angles between the normal at x, respectively y_k, and the direction $x \rightarrow y_k$, min_d is the bounding distance (to avoid singularities in G) and $f(a, b, d)$ is a smoothing function returning 0 if $d < a$, 1 if $d > b$ and a quadratic value in the interval [0..1] if $a <= d <= b$.

Equation 1 is quite expensive to compute since visibility has to be evaluated for all the VPLs and shading points. This motivates the acceleration strategies proposed on the next subsections.

4.1 Imperfect Visibility

It has been shown that due to the low frequency nature of the indirect diffuse illumination accurate visibility is not perceptually important [18]. Within our instant radiosity inspired approach this means that assessing the term $V(.,.)$ on equation 1 can be relaxed in an attempt to reduce rendering times. To validate this hypothesis VPL shadow rays are tested against the triangles bounding boxes rather than testing them for intersection against the triangles themselves.

The application supplied intersection program (or kernel) used by OptiX is associated with the geometric primitive type and does not depend on the ray type. In practice, this means that it is not possible to have OptiX calling a given intersection program for all ray types and then another program, that would test the ray against the triangle bounding box rather than the triangle itself, for VPL shadow ray types. In order to use a different intersection algorithm there are two alternatives: either a conditional statement is included on the general intersection program or the scene graph is duplicated within the OptiX context and the new intersection program is associated with the second scene graph. The former has the disadvantage of implying evaluating the conditional statement for all

intersection tests and, worst, can lead to execution divergence due to the GPU's SIMT computation paradigm. The latter has the disadvantage of consuming more memory (although only the bounding box coordinates are stored, not the respective triangle vertices) and requires building a second acceleration structure - on dynamic scenes this might compromise interactivity. Since dynamic scenes are currently not supported, the second approach was selected and a second scene graph is built with the triangles' bounding boxes rather than the triangles themselves.

4.2 Downsampling

Typically, indirect diffuse illumination is computed for all shading points. Image resolution, however, continues to grow every year with advances in available computing power, storage space and display capabilities. Since rendering complexity is linear in time with the number of pixels, computing indirect illumination at such high resolution prevents interactivity. The fact that the indirect diffuse component is mostly a low frequency signal can be exploited by rendering it at a lower resolution and then upsampling to the target final resolution [4].

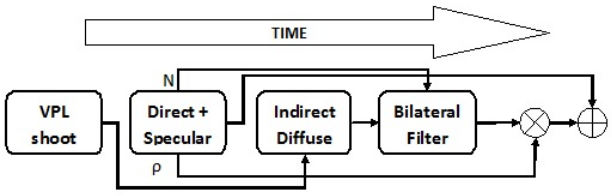


Fig. 2. Rendering pipeline for indirect upsampling - the direct stage contributes with full resolution normal and ρ maps for filtering and composition

The indirect diffuse signal, however, still has high frequencies, mostly due to geometric discontinuities. Upsampling can not be performed by convolving the signal with some low-pass kernel, since sharp edges would be unacceptably blurred. Since a high resolution pass is still required to compute direct plus specular illumination, this can be used to gather geometric information about each pixel in the target image (see figure 2). This information, the normal at the intersection point, can be used during upsampling to properly weight the contribution of each neighbor to the final value. The reasoning is that pixels in the neighborhood which have similar orientations to the center pixel will contribute more to its final result. We use the joint bilateral filter to perform this task: a spatial filter is applied to the low resolution image I and a range filter is applied to the full resolution image $\tilde{I}$. Let $\tilde{p}$ and $\tilde{q}$ denote the coordinates of two pixels in $\tilde{I}$, and p and q denote the corresponding coordinates in the low

resolution solution I. The upsampled solution $\tilde{S}$ is

$$\tilde{S}_{\tilde{p}} = \frac{1}{k_{\tilde{p}}} \sum_{q \in \Omega} I_q f(|p - q|)(\boldsymbol{N}_{\tilde{p}} \cdot \boldsymbol{N}_{\tilde{q}}) \tag{2}$$

where f is the spatial Gaussian kernel centered over p, Ω is the spatial support of f and $k_{\tilde{p}}$ is a normalizing constant. The range filter is the cosine of the angle of the normals at $\tilde{p}$ and $\tilde{q}$.

The indirect diffuse stage computes incident indirect radiance, rather than reflected, such that the upsampling filter does not blur details due to local material properties (e.g., mapped textures). Multiplication by the diffuse reflection coefficient, ρ, is done just before composition.

The indirect diffuse stage computes indirect illumination only for the shading points determined by the primary rays. Thus, upsampling is only applied to these points. Shading points further down the tree of rays have their indirect illumination calculated by the direct stage, which operates at full resolution.

4.3 Interleaved Sampling

In order to further reduce the number of VPLs visibility queries interleaved sampling is applied [2, 5]. The VPLs are divided into 9 subsets and within each pixel of a $3 * 3$ tile a different subset is used to compute indirect illumination. The contributions of the different VPL subsets are then integrated using the discontinuity buffer. The difference of depths and the dot product of the normals of a pixel are compared to those of each of its 8 neighbors. If both these values are below some given thresholds, then geometry is considered locally continuous and incident irradiance from that neighbor is added to the center pixel. The final indirect incident value is evaluated by dividing by the number of neighbors that contributed (including the center pixel itself).

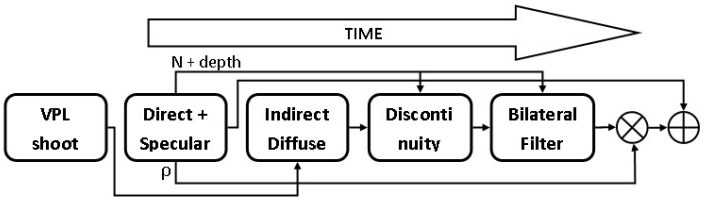


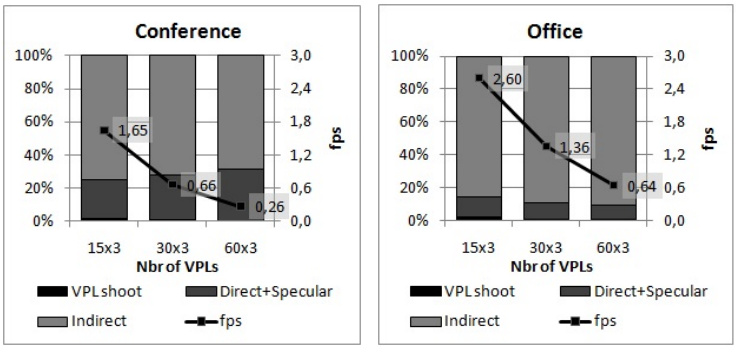
Fig. 3. Rendering pipeline for upsampling and interleaving - the direct stage contributes with full resolution normal, depth and ρ maps for filtering and composition

5 Results

5.1 Experimental Setup

All experiments and measurements were performed using OptiX 2 Beta 5 and Visual Studio 2008 on a dual core Intel Xeon 3.20 Ghz machine with 2 GB of memory and the new Nvidia 480 GTX GPU with 4 GB of RAM. Two scenes were used: the Conference room (190K triangles, 4 point light sources) and Office (21K triangles, 2 point light sources). Images were rendered at a resolution of 800x600 pixels, no anti-aliasing, using OptiX built-in BVH as the acceleration structure. Time measurements were performed with 30, 90 and 180 VPLs. Where appropriate the downsampling window was 4x4 and interleaving was 3x3.

5.2 Results Analysis



(a) Conference (b) Office

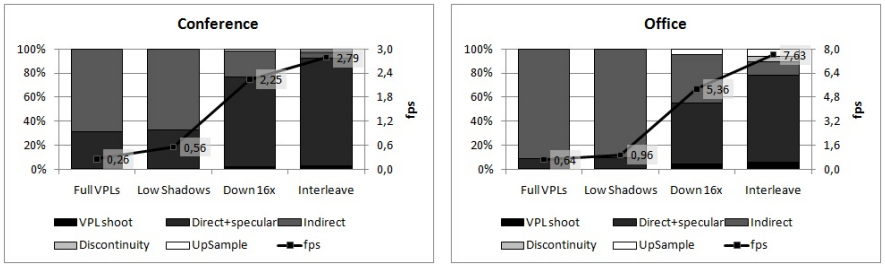
Fig. 4. fps and time percentage spent on each illumination component

Figure 4 shows evaluation of the indirect diffuse component dominates rendering time and that this aggravates with the number of VPLs. Although not obvious for the conference scene (figure 4(a)), because it contains many specular objects, the dependance on the number of VPLs for the office scene is quite obvious. Thus, according to Amdahl's law this component is the one that is worth optimizing.

Figure 5 shows the frame rate and relative execution time for each of the rendering and filtering kernels (see also figures 6 and 7).

Imperfect visibility achieves a speedup between 1.5 and 2.0 without any perceptually significant impact on the rendered image. This technique accelerates all indirect diffuse calculations, including those triggered by secondary rays - it has thus a most significant impact on the conference scene.

Downsampling 16 times provides a speedup of approximately 5 times without significantly impacting on the quality of the rendered images. Artifacts due to



(a) Conference (b) Office

Fig. 5. fps and time percentage spent on each illumination component for the 4 different approaches and 180 VPLs

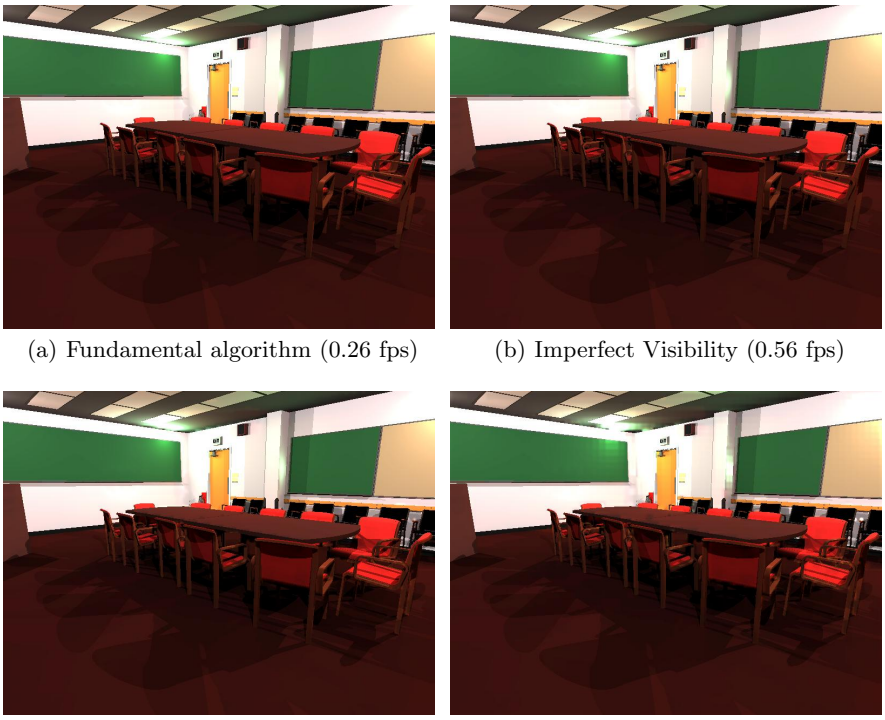
incorrect geometric continuity assumptions are however visible along edges. See for example the junction of the two halves of the table top in figure 6(c). The indirect diffuse rendering time is so drastically reduced that it is no longer the bottleneck: direct plus specular now takes more than 50% of the total rendering time. Note, however, that indirect calculations triggered by secondary rays are included in the direct stage and are not optimized by either downsampling or interleaving. In scenes with reflective materials, such as the conference room, this contributes to increase this stage relative weight on the total rendering time.

Interleaving VPLs sampling over a 3x3 window further accelerates indirect diffuse evaluation. However, artifacts are now perceivable, most noticeably when the indirect component has a strong contribution, such as on the Conference ceiling and under the desk in the Office scene (figures 6(d) and 7(d)). These artifacts are due to the regular interleaved sampling pattern over the image plane and are further enhanced by the upsampling step. Minimization of such artifacts might be possible by reducing the interleaving window size (e.g., 2x2) or by using an irregular interleaving pattern [21]. The gains obtained with interleaving are not enough to compensate for the added artifacts; furthermore, the direct plus specular component together with secondary indirect diffuse calculations now dominate rendering times, thus optimizing these is probably more important than applying interleaved sampling.

6 Conclusions

This paper discusses an implementation of Instant Global Illumination over OptiX and then evaluates three acceleration techniques: imperfect visibility, downsampling and interleaved sampling. Results show that these techniques combined allow for interactive rendering of relatively complex scenes (e.g., conference room, 190 K triangles, 180 VPLs), achieving up to 2.8 fps.

Imperfect visibility is straightforward to put in practice, but it is downsampling that results in the most impressive performance gains; artifacts are, however, slightly perceived throughout the image. Reducing the downsampling rate,



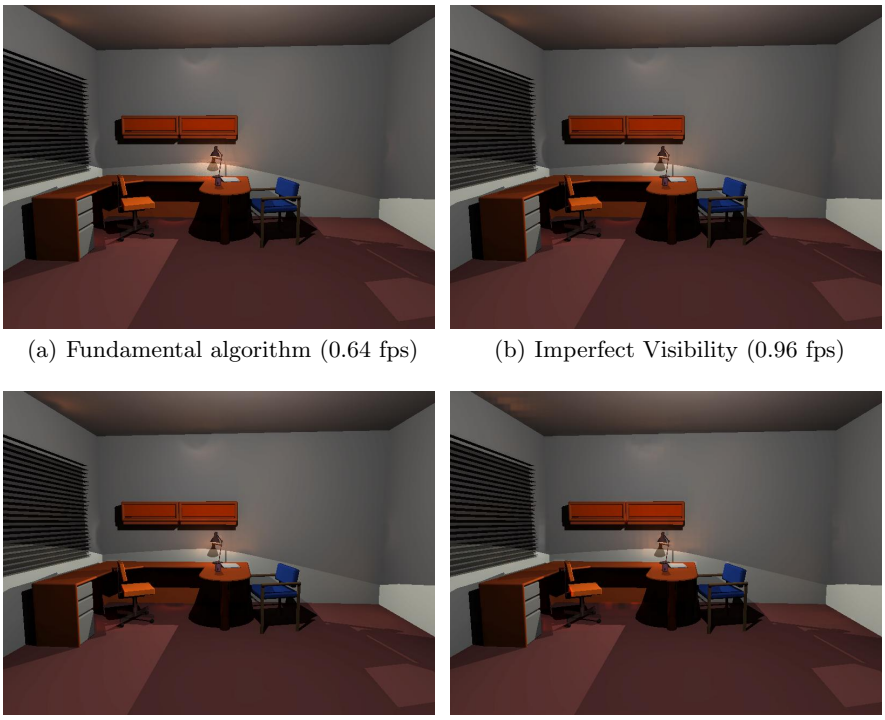
(a) Fundamental algorithm (0.26 fps) (b) Imperfect Visibility (0.56 fps)

(c) 4 x 4 downsampling (2.25 fps) (d) 3 x 3 interleaved (2.79 fps)

Fig. 6. Images for the conference scene - 180 VPLs

e.g. 3x3, will reduce such artifacts at some performance cost. Interleaved sampling further contributes to reduce rendering times, but, in combination with upsampling, artifacts become too obvious due to the regular interleaved sampling pattern. These two last techniques only operate on indirect diffuse calculations triggered by primary rays; those associated with specular secondary rays are neither downsampled nor interleaved. Improving such secondary irradiance calculations would significantly enhance performance in scenes with reflective materials and will be addressed as future work. We propose to adopt the instant caching approach [22], which, similarly to the irradiance cache [15], interpolates over scene space thus accelerating all indirect diffuse calculations.

Future work will include using irregular interleaved patterns and combining the discontinuity buffer and the bilateral filter in a single pass. Also combining rasterization with ray tracing, by resoting to shadow maps rather than shadow rays, might result in significant performance gains. Since OptiX and OpenGL interoperability is assured by NVidia, this should be straightforward to implement and evaluate. Finally, OptiX does not allow explicit access to CUDA shared memory, which results on implementation penalties, particularly on operations such as filtering. However, data can be passed between CUDA contexts and Op-



(a) Fundamental algorithm (0.64 fps) (b) Imperfect Visibility (0.96 fps)

(c) 4 x 4 downsampling (5.36 fps) (d) 3 x 3 interleaved (7.63 fps)

Fig. 7. Images for the office scene - 180 VPLs

tiX contexts through GL buffers; we intend to use this feature to optimize the discontinuity and bilateral filters, which might allow for the utilization of more sophisticated discontinuity detection techniques.

Acknowledgements This work was partially funded by PT-FCT grant PTDC/ EIA/ 65965/ 2006 (IGIDE project: Interactive Global Illumination within Dynamic Environments)

References

1. Steven Parker. Efficient ray tracing on nvidia gpus. SIGGRAPH ASIA 2009 Presentation, December 2009.
2. Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive global illumination using fast ray tracing. In *EGRW '02: Proceedings of the 13th Eurographics Workshop on Rendering*, 2002.
3. T. Ritschel, T. Grosch, M. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 27(5), 2008.

4. Johannes Kopf, Michael Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics*, 26(3), 2007.
5. Alexander Keller and Wolfgang Heidrich. Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 269–276, London, UK, 2001. Springer-Verlag.
6. Turner Whitted. An improved illumination model for shaded display. *Communicantions of the ACM*, 23(6):343–349, 1980.
7. Ingo Wald, Philipp Slusallek, Carsten Benthin, and Markus Wagner. Interactive rendering with coherent raytracing. In *Computer Graphics Forum/Proceedings of EUROGRAPHICS*, volume 20, pages 153–164, 2001.
8. I. Wald, W. R. Mark, J. Günther, S. Boulos, Ize T, Hunt W, S. Parker, and P. Shirley. State of the art in ray tracing animated scenes. In *STAR Proceedings of Eurographics 2007*, pages 89–116, September 2007.
9. S. Boulos, D. Edwards, J. Lacewell, J. Kniss, J. Kautz, I. Wald, and P. Shirley. Packet-based whitted and distribution ray tracing. In *Graphics Interface*, 2007.
10. Timo Aila and Samuli Laine. Understanding the efficiency of ray traversal on gpus. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, pages 145–149, New York, NY, USA, 2009. ACM.
11. Min Shih, Yung-Feng Chiu, Ying-Chieh Chen, and Chun-Fa Chang. *Algorithms and Architectures for Parallel Processing*, volume 5574 of *LN in Computer Science*, chapter Real Time Ray Tracing with CUDA, pages 327–337. 2009.
12. David Kirk and Wen mei Hwu. *Programming Massively Parallel Processors: a Hands-on Approach*. Korgan Kaufmann, 2010.
13. James T. Kajiya. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, New York, NY, USA, 1986. ACM.
14. Henrik Wann Jensen. Global illumination using photon maps. In X. Pueyo and P. Schröder, editors, *Rendering Techniques*, pages 21–30. Springer-Verlag, 1996.
15. G. Ward, F. Rubinstein, and R. Clear. A ray tracing solution for diffuse interreflection. *Computer Graphics*, 22(3), 1988.
16. Alexander Keller. Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
17. Rui. Wang, R. Wang, K. Zhou, M. Pan, and H. Bao. An efficient gpu-based approach for interactive global illumination. *ACM Trans. Graph.*, 28(3):1–8, 2009.
18. I. Yu, A. Cox, M. Kim, T. Ritschel, T. Grosch, C. Dachsbacher, and J. Kautz. Perceptual influence of approximate visibility in indirect illumination. In *ACM Transactions on Applied Perception*, volume 6, 2009.
19. Okan Arikan, David Forsyth, and James O'Brien. Fast and detailed approximate global illumination with irradiance decomposition. *ACM Transactions on Graphics (ACM SIGGRAPH 2005)*, pages 1108–1114, 2005.
20. Holger Ludvigsen and Anne Cathrine Elster. Real-time ray tracing using nvidia optix. In *Eurographics 2010 short papers*, 2010.
21. Solomon Boulos, Dave Edwards, Dylan Lacewell, Joe Kniss, Jan Kautz, Peter Shirley, and Ingo Wald. Interactive distribution ray tracing. Technical Report UUSCI-2006-022, SVCI Institute, University of Utah, 2006.
22. K. Debattista, P. Dubla, F. Banterle, L.P. Santos, and A. Chalmers. Instant caching for interactive global illumination. *Computer Graphics Forum*, 28(8):2216–2228, 2009.