EG UK Theory and Practice of Computer Graphics (2005)
L. Lever, M. McDerby (Editors)

Selective Parallel Rendering for High-Fidelity Graphics

K. Debattista† , V. Sundstedt, F. Pereira and A. Chalmers

Department of Computer Science, University of Bristol, United Kingdom

Abstract

High-fidelity rendering of complex scenes is one of the primary goals of computer graphics. Unfortunately, high-fidelity rendering is notoriously computationally expensive. In this paper we present a framework for high-fidelity rendering in reasonable time through our Rendering on Demand system. We bring together two of the main acceleration methods for rendering: selective rendering and parallel rendering. We present a selective rendering system which incorporates selective guidance. Amongst other things, the selective guidance system takes advantage of limitations in the human visual system to concentrate rendering efforts on the most perceptually important features in an image. Parallel rendering helps reduce the costs further by distributing the workload amongst a number of computational nodes. We present an implementation of our framework as an extension of the lighting simulation system Radiance, adding a selective guidance system that can exploit visual perception. Furthermore, we parallelise Radiance and its primary acceleration data structure, the irradiance cache, and also use the selective guidance to improve load balancing of the distributed workload. Our results demonstrate the effectiveness of the implementation and thus the potential of the rendering framework.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

High-fidelity rendering is the process of computing physically accurate synthetic images of real scenes, see Figure 1. Such high-fidelity rendering is computationally expensive and typically can not be computed in a reasonable time on a single computer. Two approaches have been considered for rendering high-fidelity images within reasonable time. The first approach relies on using parallelism to improve rendering times [CDR02]. The other approach, is an alternative to the traditional brute-force method. By identifying which computations are more relevant to the final solution, selective rendering approaches [YPG01, CCW03] can drastically reduce rendering computation times without compromising the perceived visual quality of the resultant images.

In this paper we bring together the two approaches of selective rendering and parallel rendering within the same framework. We present what we believe is the first selective parallel rendering framework and demonstrate how it is possible to reduce rendering times by exploiting these two approaches to near real-time high-fidelity rendering for complex scenes. We also compare and contrast the two traditional parallel algorithms for the irradiance cache, a data structure for accelerating indirect diffuse reflections for global illumination. We implement the framework within the physically-based *Radiance* rendering system [LS98] and demonstrate how selective and parallel rendering can be combined to achieve significant performance improvements.

2. Previous work

Our work draws from previous research in visual attention, selective rendering and parallel rendering.

2.1. Visual attention

Human visual attention is directed by two major processes [Jam90]. These are labelled *bottom-up*, which is an automatic visual stimulus and *top-down*, which is voluntary and focuses on the observer's goal. The *bottom-up* process has been found to be influenced by contrast, size, shape,

† Kurt.Debattista@bristol.ac.uk

Figure 1: *High-fidelity rendering examples: (left) The temple of Kalabsha [SCM04], (middle) the corridor scene and (right) the art gallery [LS98].*

colour, brightness, orientation, edges and motion. Koch and Ullman [KU85] presented the notion of a saliency map, a two-dimensional map that encodes the saliency of objects in the environment. Itti et al. [IKN98] developed a computer model to predict the saliency of objects in the scene. Perceptual metrics such as Daly's Visible Differences Predictor [Dal93] were developed as image-space algorithms that measure the perceptual difference between two images. The contrasting *top-down* approach, highlighted by the visual psychologist Yarbus [Yar67], demonstrated the affinity of the eye movements to the task at hand. Cater *et al.* [CCW03] introduced the concept of task maps to exploit the *top-down* approach.

2.2. Selective rendering

Although techniques based on visual attention had been developed before, such as Mitchell's [Mit87] adaptive antialiasing sampling for ray tracing, they have become more popular recently. Prikryl and Purgathofer [PP99] provide a good overview of perceptually-driven rendering radiosity algorithms. Myszkowski [Mys98] and Bolin and Meyer [BM98] use visual difference predictors, both to direct the next set of samples within a stochastic ray tracing framework, and as a stopping condition. The main drawback of these approaches is the expense of computing the visual difference predictors many times within the calculation of a single image. Yee *et al.* [YPG01] exploited a saliency model termed the *Aleph Map* to adjust the search radius accuracy of the interpolation of irradiance cache values. Cater *et al.* [CCW03] selectively rendered the foveal angle around the task related objects of an image at higher quality. In [SCCD04] both task maps and saliency maps are used to vary the number of rays shot per pixel in a global illumination environment to produce high perceptual quality images at a reduced computational cost.

2.3. Parallel rendering

Parallel rendering algorithms have been used to alleviate the cost of rendering for a number of years. Chalmers *et al.* [CDR02] offer a comprehensive analysis of the standard approaches for static and dynamic load balancing, data and task management, and more advanced approaches. Parker *et al.*'s parallel ray tracer [PMS*99], through optimised code, ray traced simple scenes interactively on a shared memory parallel computer. Wald *et al.* [WBWS01] also obtained interactive rates for ray tracing, this time over a distributed cluster and by using cache-coherent techniques and SIMD instructions commonly found in modern architectures. Subsequently, in [WKB*02], they extended their distributed ray tracer to interactively render images using global illumination. Günther *et al.* [GWS04] extended this distributed framework further to support real-time caustics through photon mapping.

Ward's irradiance cache [WRC88], an acceleration data structure which caches indirect diffuse samples within the framework of a distributed ray tracing algorithm, lies at the heart of *Radiance*. The irradiance cache can improve rendering performance times by an order of magnitude. Since the irradiance cache is a shared data structure it is notoriously hard to parallelise. There have been a number of implementations of a parallel irradiance cache within *Radiance*. The standard *Radiance* distribution [LS98] supports *Radiance* in parallel over a distributed system using the Network File System (NFS) for concurrent access of the irradiance cache. This has been known to lead to contention and may result in poor performance when using inefficient file lock managers. Koholka *et al.* [KMG99] used the Message-Passing Interface (MPI) instead of NFS for their distributed *Radiance* implementation. The irradiance cache values are broadcast amongst processors after every 50 samples calculated at each slave. Robertson *et al.* [RCLL99] presented a centralised parallel version of *Radiance* whereby the calculated irradiance cache values are sent to a master process whenever a threshold is met. Each slave then collected the values deposited at the master by the other slaves.

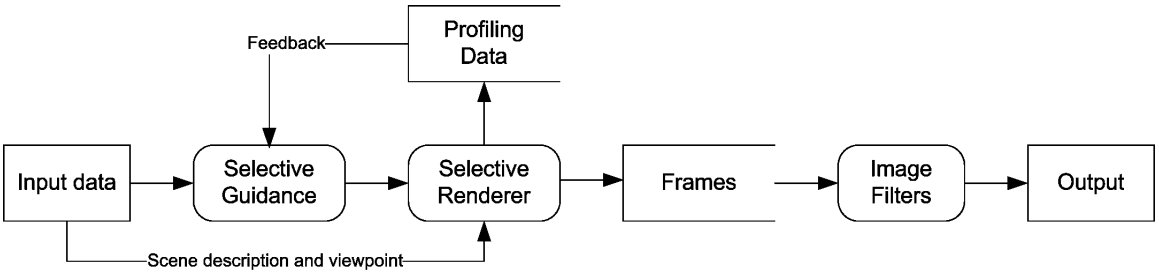

Figure 2: *Overview of the Rendering on Demand Framework.*

3. Rendering on Demand framework

Our rendering framework which we term Rendering on Demand (RoD) is designed for rendering physically-based high-fidelity still images, animations, and interactive systems for complex scenes in reasonable time. Our hypothesis is that we can obtain the quickest results while maintaining high-fidelity by combining selective rendering with parallelism. Our RoD framework can be represented in three main processing blocks (Figure 2) entitled *selective guidance*, *selective rendering* and *image filters*. Input is in the form of scene data files, viewpoint, lighting, configuration data and settings for the main rendering stage. Selective guidance is responsible for generating directives to drive the selective rendering. Bottom-up and top-down visual attention models, and other metrics such as motion and predicted complexity are handled within this process. Feedback from previous results could also influence the selective guidance. Selective guidance produces a set of rules or maps which are used to direct the rendering. The selective renderer uses the selective guidance results to parameterise the rendering quality, potentially for each pixel. The selective renderer within our framework is designed to take advantage of parallelism at a number of levels. At the highest level, the renderer functions over a distributed system. Subsequently, a per-node configuration determines the hardware mappings for the most efficient use of the hardware at that node (GPU, CPUs *etc.*...). Finally, image filters operate in the 2D space to adapt and remove unwanted artifacts from the animation and may include operations such as filtering, tone mapping and frame interpolation. This framework is designed to be modular and portable.

4. Implementation

While our framework is applicable to any form of high-fidelity rendering, we present an implementation of the framework based on the physically-based lighting simulation system *Radiance* [LS98]. Our implementation is an extension of the *Radiance* renderer for still images and animations, `rpict`. We have adapted *Radiance* to our rendering framework, by introducing a selective guidance system, and have extended *Radiance* to support selective rendering and parallelism. The image filtering tools already existing within *Radiance* were sufficient to satisfy our filtering, tone mapping and interpolation needs.

4.1. Selective guidance

After loading geometry, lighting, view points and general rendering parameters in the same manner as standard `rpict`, our first step corresponds to the selective guidance process. Within our implementation the directives take the form of an *importance map* [SCCD04, SDL*05] which is a two-dimensional map representing image space dictating were computational resources are best spent to obtain the highest perceptual result. The *importance map* is visualised as a grey-scale image, whereby the brightest pixels will result in preferential rendering. Within our system the *importance map* is a combination of a *task map* to account for the effect of top-down visual attention and a *saliency map* for bottom-up visual attention. Other potential maps not present within our implementation, such as a complexity and motion maps, would be included in this stage.

The process begins with a rapid image estimate of the scene. The image estimate helps to locate areas where an observer will be most likely to look and can be quickly generated with a low-level ray tracing pass or a quick rasterisation pass in hardware [LDC05] which can be used in two ways. Firstly, for generating the *saliency map* by using it as input to a saliency generator. We use the method from Itti *et al.* [IKN98] to compute our *saliency map*. Figure 3 demonstrates the saliency map (middle) of the rendered Kalabsha scene (left). Secondly, the quick estimate, can also be used to generate a task map by identifying user-selected task-related objects and applying a foveal-angle gradient around the objects. Figure 4 demonstrates how a *task map* (middle) with added foveal-angle gradient (right) is generated for one of the frames (left) for the animations described in Section 5.3. The *task map* and *saliency map* are combined using a user-defined weighting into the *importance map*. The *importance map* is then fed into the selective renderer for the next phase of the rendering.

Figure 3: *The Kalabsha scene: (left) the full rendered image, (middle) the saliency map and (right) visualisation of the parallel sub-division of workload, in reality the sub-division is finer.*

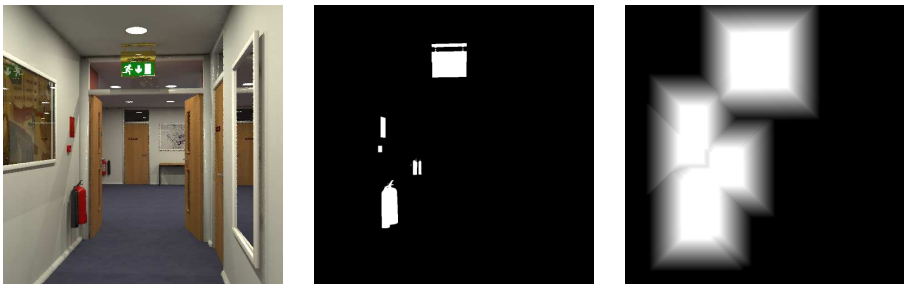

Figure 4: *The corridor scene (Frame 75): (left) image rendered with selective quality, (middle) the task objects and (right) the task map including foveal angle gradient. For this scene, the task objects are the fire extinguishers, fire alarms and emergency exit signs.*

4.2. Selective rendering

The phase corresponding to the selective rendering process within our framework is based on the distributed ray tracing used by *Radiance*. In our implementation we parallelise the rendering using a demand-driven approach, in the form of a master-slave model using the message passing system MPI. The *importance map* is used by the master to subdivide the workload and by the slaves to decide how many rays per pixel to cast. The master is responsible for subdividing the image plane into a number of tiles of a given granularity. Each image tile represents a job for a slave to compute. We use the *importance map* as a simple cost prediction map. Since, at the slave, the *importance map* dictates the number of rays shot per pixel, the master uses it to improve subdivision by ensuring that each tile contains an equal number of primary rays to be shot. This improves load balancing by ensuring a more even distribution of the workload. Although the computational requirements of each individual ray may differ, the demand driven approach together with our subdivision map alleviates the problem significantly. Figure 3 demonstrates how the *importance map* (middle), which in this case is just a saliency map, effects the subdivision of the workload (right). Each tile can be visualised as the area between two white lines. In this frame, the bottom part of the image is not that salient, therefore the image tiles for this part of the image are larger. Conversely, the middle part of the image is more salient, requiring more time to compute, so the tile sizes are smaller.

The master farms out the work to all the slaves in the form of the coordinates of the tile to be rendered. The slaves then render the image tile assigned to them using the *importance map* to direct the rendering of each pixel. A user-defined parameter indicates the maximum number of rays per pixel. This value is then modulated by the value in the *importance map* to calculate how many rays are shot for a given pixel. The higher the value within the importance map, the more rays per pixel are shot. When the slave finishes executing the job, it asks for more work from the master until the entire process is completed.

Ray tracing is traditionally easily extended into a parallel framework, however our approach follows the *Radiance* implementation. Although *Radiance* uses distributed ray tracing to render images, the irradiance cache [WRC88] is used to accelerate the calculation of the indirect diffuse component. As the irradiance cache is a shared data structure, it is non-trivial to parallelise.

4.3. Parallel irradiance cache

In order to investigate the speedup offered by our distributed selective renderer, we have parallelised the irradiance cache

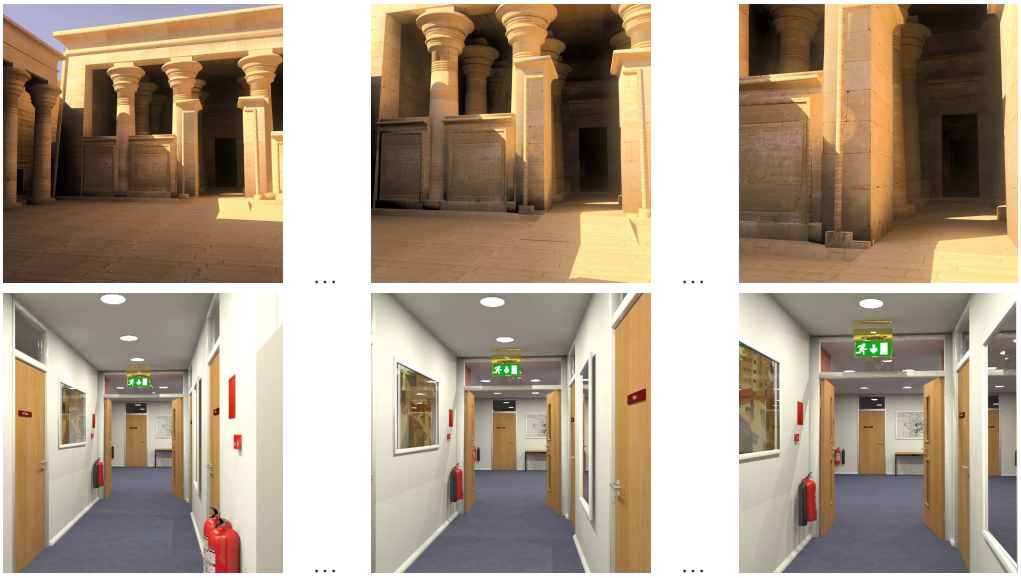

Figure 5: *The Kalabsha scene (top) and corridor scene (bottom) 90 frame animations. Frames: (left) the first frames, (middle) the 45th frames and (right) the final frames.*

using two methods. The first is the more traditional centralised approach inspired by that in [RCLL99]. The second approach is similar to that of [KMG99] which we call the distributed approach.

The centralised approach relies on each of the slaves computing irradiance cache values and storing them in an outgoing buffer. Whenever the buffer reaches a user-defined threshold the buffer is transmitted to the master. The master receives the buffer and transmits back to the slave the rest of the samples that have been computed since its last transmission. The master maintains a list of all the samples sent to it and also a running status of which samples each slave has been given so far. This approach is simple to implement since only one process is needed for every slave which can handle both communication and computation. Unfortunately, this approach suffers from contention at the master node and the latency for each slave to update its own cache.

For the distributed approach, each slave maintains an outgoing buffer as in the centralised approach. However, whenever the user-defined threshold is met the buffer is broadcast to all other slaves. This approach is more complex than the previous method, since each slave is allowed to broadcast to every other slave. In order to maximise computation, each slave has a separate communicator process which listens for incoming irradiance cache samples. Whenever a set of samples is received, the communicator process stores the data in a shared memory area, where the computation process can collect it and insert it onto the irradiance cache. This approach removes the contention on the centralised node but can still run into scalability problems.

5. Results

We used two complex scenes for evaluating our system, the Kalabsha scene [SCM04] and the corridor scene. The system we used for our results, is a cluster of eight dual Intel Xeon processors running at 2.4 GHz with 3 GB of memory under Linux and a single workstation with a single processor 2.53 GHz Intel with 1 GB of memory acting as the frontend for the parallel implementations. All the nodes were connected by a Fast Ethernet N-way switch (100 Mbit).

5.1. Building the irradiance cache

In this section we compare the timings for rendering a still image without the use of a precomputed irradiance cache. This is equivalent to computing an irradiance cache for a particular view. We rendered the first image from the animations of the Kalabsha scene, with two indirect diffuse bounces, and the corridor scene, with one indirect diffuse bounce, as shown in Figure 5 (left). Results for both the scenes, up to sixteen processors, can be seen in Figure 6. We plot the results obtained when the irradiance cache samples are not shared (no sharing), one for the centralised system (centralised) and one for the distributed system (distributed). For the corridor scene, when not sharing samples, the results are slightly worse than the centralised and distributed approaches. For the Kalabsha scene, when not sharing any values, the results were substantially worse than for the other two versions. It can also be seen that the distributed version outperformed both other systems in all cases.

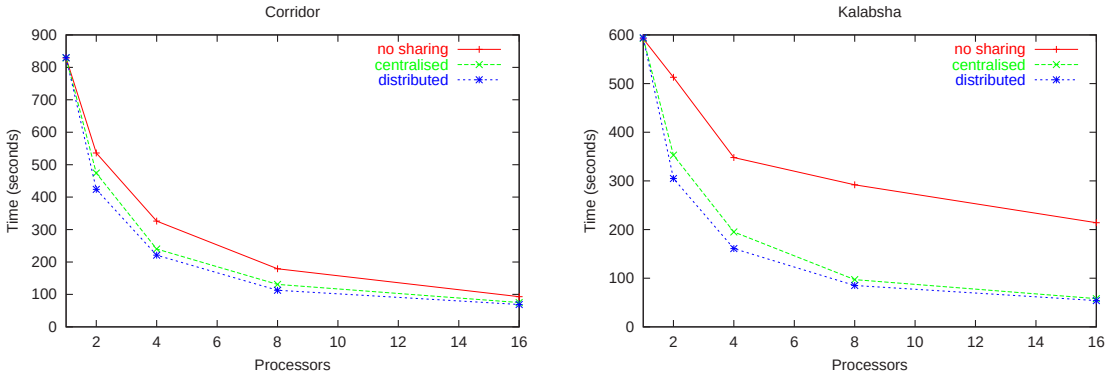

Figure 6: *Building the irradiance cache (Frame 1): (left) the corridor scene, with one indirect diffuse bounce, and (right) the Kalabsha scene, with two indirect diffuse bounces.*

5.2. Exploiting bottom-up visual attention

We demonstrate bottom-up visual attention within our framework by showing timings of rendering a 90 frame animation in the Kalabsha scene, see Figure 5 (top), with four different settings. All frames were rendered at a resolution of 500×500 with a maximum of five rays per pixel. The first rendered sequence used the plain uniprocessor version, representing the traditional rendering method within *Radiance*. In this sequence all the pixels in each frame were rendered with five rays per pixel. The second sequence exploited visual attention by rendering only the salient parts of the scene at a higher quality, based on a saliency map of that scene. The saliency map was generated using the Itti *et al.* [IKN98] method (this takes about one second, not included in timings). In this case the *importance map* was based only on the saliency map. The *importance map* dictated how many rays per pixel were shot up to a maximum of five. The third rendered sequence used the parallel version on sixteen processors to render the sequence without visual attention. The final sequence used both parallelism on sixteen processors and the saliency maps for visual attention. We used the distributed version of the parallel irradiance cache for both of the sequences that were rendered in parallel.

Results, presented in Figure 7, clearly demonstrate the effectiveness of our approach. For all the rendered sequences, the first few frames of the animations were initially quite expensive. This was due to the irradiance cache being empty. In subsequent frames as the irradiance cache became more populated the timings became more homogenous. The uniprocessor (uni SM) saliency version gained a 3 time speedup over the standard uniprocessor version (uni). The parallel (16) version was around 13 times faster than the traditional version. The combined saliency and parallelism (16 SM) approach was around 37 times faster than the traditional rendering.

5.3. Exploiting top-down visual attention

We use the corridor scene to demonstrate how we exploit the top-down approach of visual attention. The corridor scene was designed to investigate the potential of selectively rendering an animation, where the user is asked to perform a fire safety task within the virtual scene [SDL*05]. The viewer is asked to verify the position and number of the fire safety objects placed within the corridor. We used this knowledge to render the task objects and the foveal angle around these objects at a higher quality than the rest of the image, see Figure 4. We used the *Snapshot* [LDC05] for the opening rendering pass to generate the task map ($\sim 10ms$). As with the Kalabsha scene we rendered a 90 frame animation for four sequences, see Figure 5 (bottom). All sequences were rendered at a resolution of 500×500 and a maximum of five rays per pixel. The distributed version of the irradiance cache was used for all sequences. We calculated timings for rendering an animation using the traditional method, using task maps to render the animation on uniprocessor, traditional method in parallel and using task maps for rendering on sixteen processors. For these animations the *importance map* was the task map as no saliency map was used.

Results, are presented in Figure 8. The uniprocessor version using task maps (uni TM) was about 2.5 times faster than the traditional version (uni). The parallel version (16) running on sixteen processors gained a 13 times speedup. The parallel version on sixteen processors using task maps obtained a speedup of about 31 times on the traditional version (16 TM).

6. Conclusions and future work

We have presented a framework for accelerating high-fidelity rendering by means of selective rendering using parallel rendering and selective guidance. We have demonstrated the approach through our implementation, based on

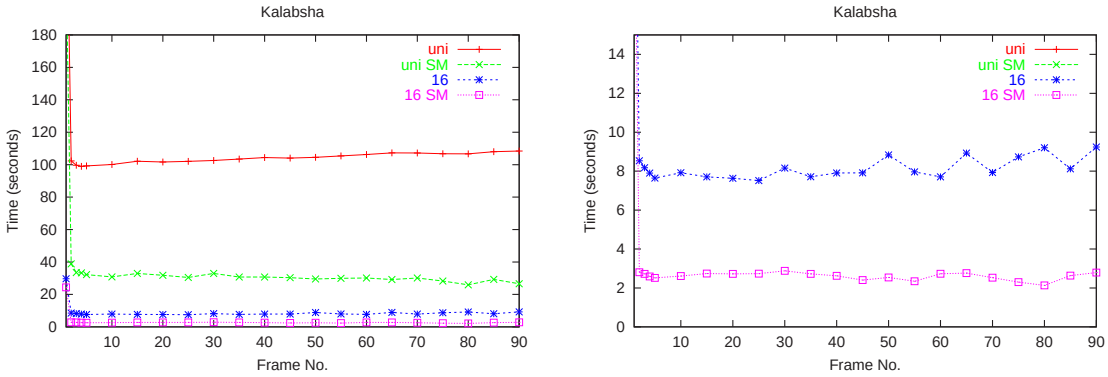

Figure 7: *The Kalabsha scene exploiting bottom-up visual attention: (left) comparison of all results and (right) zoomed in on the parallel results only*

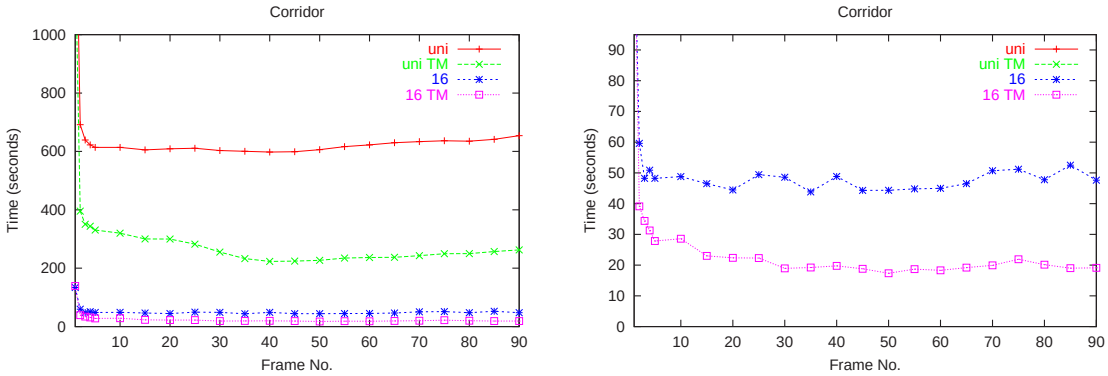

Figure 8: *The corridor scene exploiting top-down visual attention: (left) comparison of all results and (right) zoomed in on the parallel results only.*

the lighting simulation system *Radiance*. Our results demonstrate that selective guidance, particularly through the exploitation of the human visual system for both bottom-up and top-down approaches, reduces rendering times by directing the rendering resources to the more important parts of an image with no significant perceived loss in quality. Furthermore, we have described methods of combining the selective rendering approach to that of parallelism, used the *importance map* as a simple load balancing mechanism and parallelised the irradiance cache to take further advantage of temporal coherence within animations.

Further work on our implementation would be required to fulfill our ambitions of rendering high-fidelity in real-time. System-specific optimisations [WPS*03] could speed up performance at the cost of portability. The selective guidance could be extended further to provide better directions for the selective renderer, such as predicting rendering costs, direct light rendering for primary rays (through a quick GPU-based rasterisation render) and edge-detection. Another challenge would be determining how to combine the selective guidance into a set of directives for the selective renderer. For now we have equated quality with number of rays per pixel in the selective renderer. Other possible approaches, for example component-based rendering, that provide further flexibility will be investigated.

7. Acknowledgements

The work reported in this paper has formed part of the Rendering on Demand (RoD) project within the 3C Research programme whose funding and support is gratefully acknowledged.

References

[BM98] BOLIN M. R., MEYER G. W.: A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM Press, pp. 299–309.

[CCW03] CATER K., CHALMERS A., WARD G.: Detail to Attention: Exploiting Visual Tasks for Selective Rendering. In *Proceedings of the Eurographics Symposium on Rendering* (2003), pp. 270–280.

[CDR02] CHALMERS A., DAVIS T., REINHARD E.:. AK Peters Ltd, July 2002.

[Dal93] DALY S.: The visible differences predictor: an algorithm for the assessment of image fidelity. 179–206.

[GWS04] GUENTHER J., WALD I., SLUSALLEK P.: Realtime Caustics using Distributed Photon Mapping. In *Proceedings of the Eurographics Symposium on Rendering* (2004).

[IKN98] ITTI L., KOCH C., NIEBUR E.: A model of Saliency-Based Visual Attention for Rapid Scene Analysis. In *Pattern Analysis and Machine Intelligence* (1998), vol. 20, pp. 1254–1259.

[Jam90] JAMES W.: A saliency-based search mechanism for overt and covert shifts of visual attention. In *Principles of Psychology* (1890).

[KMG99] KOHOLKA R., MAYER H., GOLLER A.: Mpi-parallelized radiance on sgi cow and smp. In *ParNum '99: Proceedings of the 4th International ACPC Conference Including Special Tracks on Parallel Numerics and Parallel Computing in Image Processing, Video Processing, and Multimedia* (1999), Springer-Verlag, pp. 549–558.

[KU85] KOCH C., ULLMAN S.: Shifts in selective visual attention: towards the underlying neural circuitry. In *Human Neurobiology* (1985), vol. 4, pp. 219–227.

[LDC05] LONGHURST P., DEBATTISTA K., CHALMERS A.: Snapshot: A rapid technique for driving a selective global illumination renderer. In *WSCG 2005 SHORT papers proceedings* (2005).

[LS98] LARSON G. W., SHAKESPEARE R.: *Rendering with radiance: the art and science of lighting visualization*. Morgan Kaufmann Publishers Inc., 1998.

[Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), ACM Press, pp. 65–72.

[Mys98] MYSZKOWSKI K.: The Visible Differences Predictor: Applications to global illumination problems. In *Eurographics Workshop on Rendering* (1998), pp. 223–236.

[PMS*99] PARKER S., MARTIN W., SLOAN P.-P. J., SHIRLEY P., SMITS B., HANSEN C.: Interactive ray tracing. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), ACM Press, pp. 119–126.

[PP99] PRIKRYL J., PURGATHOFER W.: *Overview of Perceptually-Driven Radiosity Methods*. Tech. Rep. TR-186-2-99-26, Vienna, Austria, 1999.

[RCLL99] ROBERTSON D., CAMPBELL K., LAU S., LIGOCKI T.: Parallelization of radiance for real time interactive lighting visualization walkthroughs. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)* (1999), ACM Press, p. 61.

[SCCD04] SUNDSTEDT V., CHALMERS A., CATER K., DEBATTISTA K.: Top-down visual attention for efficient rendering of task related scenes. In *Vision, Modeling and Visualization* (2004).

[SCM04] SUNDSTEDT V., CHALMERS A., MARTINEZ P.: High fidelity reconstruction of the ancient egyptian temple of kalabsha. In *AFRIGRAPH 2004* (November 2004), ACM SIGGRAPH.

[SDL*05] SUNDSTEDT V., DEBATTISTA K., LONGHURST P., CHALMERS A., TROSCIANKO T.: Visual attention for efficient high-fidelity graphics. In *Spring Conference on Computer Graphics (SCCG 2005)* (May 2005).

[WBWS01] WALD I., BENTHIN C., WAGNER M., SLUSALLEK P.: Interactive rendering with coherent ray tracing. In *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001* (2001), Chalmers A., Rhyne T.-M., (Eds.), vol. 20, Blackwell Publishers, Oxford, pp. 153–164.

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive global illumination using fast ray tracing. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (2002), Eurographics Association, pp. 15–24.

[WPS*03] WALD I., PURCELL T. J., SCHMITTLER J., BENTHIN C., SLUSALLEK P.: Realtime Ray Tracing and its use for Interactive Global Illumination. In *Eurographics State of the Art Reports* (2003).

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (1988), ACM Press, pp. 85–92.

[Yar67] YARBUS A.: Eye movements during perception of complex objects. In *Eye Movements and Vision* (1967), pp. 171–196.

[YPG01] YEE H., PATTANAIK S., GREENBERG D.: Spatiotemporal sensitivity and Visual Attention for efficient rendering of dynamic Environments. In *ACM Transactions on Computer Graphics* (2001), vol. 20, pp. 39–65.