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

# Cost Prediction Maps for Global Illumination

Richard Gillibrand†, Kurt Debattista and Alan Chalmers

Computer Graphics Group, Department of Computer Science, University of Bristol, UK

**Abstract**

*There is a growing demand from the media industry, including computer games, virtual reality and simulation, for increasing realism in real-time for their computer generated images. Despite considerable advances in processing power and graphics hardware, increasing scene complexity means that it is still not possible to achieve high fidelity computer graphics in a reasonable, let alone real, time on a single computer. Cost prediction is a technique which acquires knowledge of computational complexity within the rendering pipeline as the computation progresses and then uses this to best allocate the available resources to achieve the highest perceptual quality of an image in a time constrained system. In this paper we describe a method of acquiring computational cost complexity knowledge within a high fidelity graphics environment. This cost map may be used in combination with other perceptually derived maps to control a selective renderer in order to achieve the best perceptual quality results for a user specified frame-rate.*

Categories and Subject Descriptors (according to ACM CCS):
I.3.7 [Computer Graphics]: Raytracing

## 1. Introduction

Recently the proliferation of high quality computer graphics within the media industry, both as a tool for presentation of information to a wide, and not necessarily expert, audience, and as a entertainment medium in its own right has been increasing rapidly. High quality archaeological reconstructions, for example, are now frequently used both to explore new ideas and to present those ideas in a user friendly format. This increase in the exposure and usage of these techniques raises expectations of quality and usability and not all of these scenarios are suitable for the usual pre-rendering or detailed pre-processing of data. Similarly, gamers also expect increasing visual immersivity and quality. In both these situations the need to be able to guarantee a target frame rate is critical to the success of the final result.

In applications where offline rendering can be used, the need to maintain a fixed frame rate does not arise. However, it is still useful to know how long a rendering will take, both to ensure the most efficient use of computer resources and, from a commercial point of view, to be able to guarantee deadlines and provide accurate quotations for the work about to be undertaken.

Within a perceptual selective rendering system, individual pixels are rendered at different qualities. The human visual system is such that these quality differences can go unnoticed, enabling images of equal perceptual high quality to be (selectively) rendered in a fraction of the time required by a full rendering solution [CCW03].

In a predictive rendering framework, any knowledge of the computational cost associated with rendering pixels enables the computational resources to be best utilised in the time allowed. This is especially true in a parallel system, in which this computational knowledge can be used to schedule the parallel tasks most efficiently amongst the processors.

Whilst the need for timing prediction is clearly important, the difficulty is that the cost of rendering the scene with global illumination is not often known with total certainty until the image has been completed due to the variation of complexity across the scene. Consequently, the most efficient allocation of resources, and indeed whether the scene can be rendered in a reasonable time at all, is not known initially. Even progressive techniques, which start at a low quality and keep improving for the duration of available time,

† Email: gillibrand@cs.bris.ac.uk

may become tied up with highly complex yet perceptually relatively unimportant areas of the scene to the general detriment in perceived quality of the image as a whole.

In this paper we describe how we can derive computational cost information for each pixel from knowledge of the spatial subdivision of the environment. We establish the relationship between rendering time and octree subdivision. This information enables us to determine the most appropriate subdivision strategy to adopt for complex scenes in order to achieve the best computational performance. In addition, our system utilises a small number of profiling rays to establish an accurate cost map for the whole image for a given view.

## 2. Previous Work

Most of the work done in the field of cost prediction has been concentrated in three areas: to provide the most efficient parameterisation of spatial subdivision methods, for efficient load balancing when rendering across a parallel network, and to guarantee a fixed frame rate.

### 2.1. Cost Prediction for Parameterisation of Spatial Subdivision Methods

Accelerating raytracing by dividing the space occupied by a scene to reduce intersection tests is a well established technique with several variations. Whether the division strategy is in the form of an octree, k-d tree, BSP-tree or other, the subsequent efficiency of many of these algorithms during the rendering stage can depend on the decisions made at tree construction time. There is always a cost trade-off between traversal of many layers of sub-division and ray-object intersection testing. Optimising this trade-off has been investigated by several researchers, for example [ABCC03], to find a cost function that balances and minimises the combined cost of traversing the tree and the cost of ray-object intersections.

### 2.2. Cost Prediction for Load Balancing

As stated above, gaining knowledge of the cost distribution across the scene greatly aids the allocation of resources when using a parallel network for rendering. By determining the most heavily traversed or intersected voxels these can be duplicated in a demand driven system to reduce the number of data fetches. In a data parallel system the loading on the processors can also be most efficiently balanced by knowing the cost distribution of the scene, [CDR02].

Reinhard et al. developed a cost function based on the geometric complexity of a scene by defining the probability that a ray intersects an object in an octree [RKJ96]. The cost of traversal is derived from the weighted average depth of the octree $\overline{D}$, where

$$\overline{D} = \frac{\sum_{i=1}^{k} h_i 4^{-h_i}}{\sum_{i=1}^{k} 4^{-h_i}}$$

and a ray traversing the octree without intersections would encounter $2^{\overline{D}}$ cells. The probability of ray-object intersection is found from the ratio of object to cell surface areas. The final cost for a ray is then the sum of the traversal costs and the intersection test costs.

This approach was then developed to include a calculation for the number and distribution of generated rays per voxel based on geometric and material properties and the number and position of light sources [RKC98]. From this and the previous work they proposed a calculation of the cost of tracing the different parts of an octree in order to provide a more efficient load balancing for a scene rendered on a parallel network. However, their actual results of the number of rays per voxel differed too much from their predictions to be applied directly to determine the overall cost of ray tracing the scene.

### 2.3. Cost Prediction to Guarantee Frame Rate

Funkhouser and Séquin [FS93] used pre-processed level of detail cost and benefit for each object in a scene in order to allow their system to produce the best available benefit sum for the scene whilst not exceeding a maximum allowed cost. Their cost metric is a function of the number of vertices and polygons and the number of pixels rendered, each of which had an experimentally derived constant applied. The use of experimentally derived constants for the per polygon and per pixel costs gave an accurate result for their test scenes but, as stated previously however, this level of preprocessing is not always appropriate or possible in an interactive environment and the need to have a test calibration phase for each machine or environment in which the rendering would be undertaken limits the application of this technique.

Horvitz and Lengyel [HL97] presented a similar approach to real-time rendering by using flexible rendering of level of detail. Their system adapted resolution per object and temporal coherence within a rasterisation framework. These time-constrained frameworks used simple perceptual models as a selective criteria for the decisions made. Dumont et al. [DPF03] also used a decision theoretic framework but with more complex perceptual models for their system for rendering global illumination using hardware.

Wimmer and Wonka [WW03] define a rendering time estimation function for GPU rendering based on estimations for each of the major components of the rendering process:

- system tasks $(ET_{system})$
- CPU tasks $(ET^{CPU})$

- Idle time ($ET_{idle}^{CPU}$, $ET_{idle}^{GPU}$)
- GPU tasks ($ET^{GPU}$)

In the form:

$$RT = ET_{system} + max(ET^{CPU}, ET^{GPU})$$

where

$$ET^{CPU} = ET_{nr}^{CPU} + ET_{r}^{CPU} + ET_{mm}^{CPU} + ET_{idle}^{CPU}$$

and

$$ET^{GPU} = ET_{fs}^{GPU} + ET_{r}^{GPU} + ET_{mm}^{GPU} + ET_{idle}^{GPU}$$

nr denotes non-rendering code, fs frame setup, r rendering code, mm memory management and idle is idle time.

They compared several methods of cost prediction, ranging from measured to calculated, including per-object sampling, per view-point estimation and per view-cell estimation and also compared four mathematical heuristics for the $ET_r^{GPU}$; the triangle count, the actually transformed vertices count, Funkhouser's cost function and their own which is an adaptation of Funkhouser's using actually transformed vertices. Their conclusion was that while their method improved the accuracy of the estimation, the limitations of the timing accuracy of graphics hardware hampered the effectiveness so they proposed an extension to hardware to resolve their problem.

## 3. Ray Timing

As a ray passes through a scene there are two possible outcomes; it either intersects an object or it doesn't. There is clearly an associated cost associated with either outcome. A ray that intersects an object will have the times associated with the calculation of the intersection point, calculation of surface normal at the intersection point, the retrieval of material information and possible associated costs of procedural texturing and this is only for primary rays. For a global illumination solution the spawning of innumerable secondary rays each of which have the same associated costs adds to the time taken to trace that primary ray.

Even for rays that do not intersect an object, the cost is not zero. Depending on the use and efficiency of spatial subdivision the ray may even have had to undertake intersection checks with every object in the scene and consequently may in fact have a higher cost than a ray that makes an early intersection.

In order to ascertain these costs and thus use these to establish an efficient spatial subdivision, our method uses an adapted version of of the lighting visualisation system Radiance [War94]. This system uses the octree as its spacial subdivision structure and so our method concentrates on the octree but could be simply extended to other strategies. We shoot and time the cost of the different situations that rays encounter for a variety of scenes. For each ray timing experiment, one thousand rays were traced per pixel with each ray having directional jittering. The large number of rays shot within each pixel boundary provided the most accurate average timing for a single ray traced for that pixel.

### 3.1. Scenes

In order to assess the cost of the different parts of a primary ray's path a variety of scenes were constructed that provided the most likely scenarios that occur in a typical scene.

- BLANK - A totally empty scene was used first to determine the setup time required for tracing a ray outside the octree. Since the octree is a subdivision of the bounding cube of the scene objects it is possible to have a ray that that does not pass through the octree at all.
- UNSUB - A pair of cubes diagonally separated from each other to produce an undivided octree with large amounts of empty space in it. The rendered view contained no part of either cube and established the time for rays to be traced through an octree without hitting an object.
- CUBE - A single cube producing a undivided octree with the view looking directly at the cube. This established the time for a ray to hit and return the value from a single object.
- SUB - A pair of cubes diagonally separated as earlier, but with the octree forced into a subdivision. The scene is set with an oblique view, so that the times for both rays that hit a single object without traversing more than a single cell (the near cube) and for those that hit a single object after traversing a single empty cell (the far cube) are available.
- MULTI, MULTIROT - The final calibration scenes contained different size cubes, one set axis aligned, one rotated forcing the octree into further subdivisions and giving timings for rays with more occupied cell traversals and more intersection tests figure 1.
- The complex test scene is the reconstruction of the temple of Kalabsha [SCM04], figure 2, giving a more typical test scene.

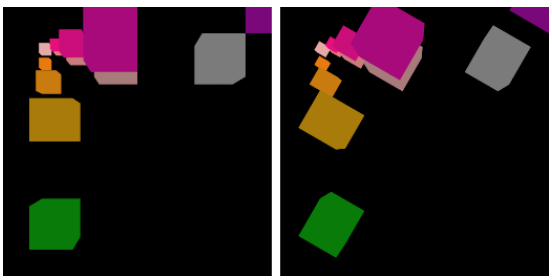

**Figure 1:** *Primary ray-traced images of the MULTI and MULTIROT scenes.*

| Path | Scene | Proportional Cost |
|---|---|---|
| No Octree Intersection | EMPTY | 1 |
| Traversal of Empty Octant | SUB | 1.1 |
| Traversal of Full Octant with Single Object | CUBE | 1.8 |
| Intersection with Single Object Scene | CUBE | 1.8 |
| Traversal of Full Octant with Two Objects | UNSUB | 2 |
| Empty Octant Traversal Followed by Intersection | SUB | 3 |
| Empty Octant Traversal Followed by Full Octant Traversal | UNSUB | 3.5 |
| Full Octant Traversal Followed by Intersection | MULTI | 5 |
| Full Octant Traversal Followed by Full Octant Traversal | MULTI | 6 |

**Table 1:** *Proportional costs for different ray-path outcomes*

**Figure 2:** *The temple of Kalabsha model.*

## 4. Timing Results

The timings from the initial basic scenes show the costs of tracing different parts of the scene and the values are scaled to be relative to the cost of a ray that passes outside the octree completely, table 1. Hence we can see that, for example, it takes nearly twice as long to trace a ray with an intersection test as it takes to trace a ray that misses the scene completely.

From this we can see that the cost associated with a ray-object or ray-object bounding box intersection test outweighs the cost of an octree traversal by around a factor of 2. This result is important as it shows that the octree subdivision level parameter has a large impact on the resulting rendering cost, if there are many intersection tests performed in each octant the cost for the scene can rise rapidly. This aspect is explored in more detail later in this section.

The results on the multiple cube test scene show the clear correlation between the shape of the octree and the cost of traversing the octree, figure 3. The differing costs for different subdivision parameter settings can be seen in figures 4 and 5. In these images and those following the costs are presented as an image false-coloured for printing clarity with a proportional timing scale where the shortest time taken to trace a ray in the scene is always 1 and the other times are scaled accordingly. On comparative sets of images the same false-colour scale is used for all the images in the set, as in figure 8, for example, where the most costly rays took 100 times as long to trace as the quickest.

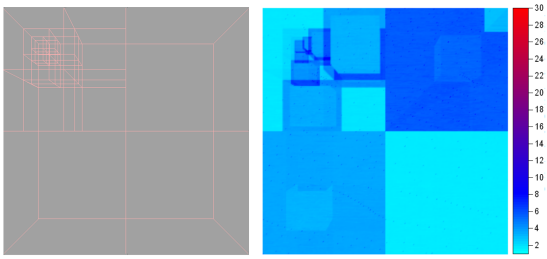


**Figure 3:** *Subdivided octree of the MULTI test scene and resulting times, as viewed from the camera.*

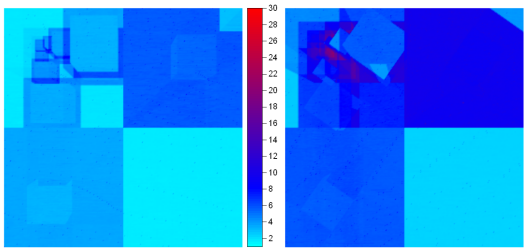


**Figure 4:** *Proportional timing maps from MULTI and MULTIROT for octree with total subdivision.*

Having performed the initial tests using simplistic scenes we looked at the effects of octree subdivision on ray cost within a more realistic environment, the temple of Kalabsha model, figures 6 and 7.

We chose different levels of octree subdivision for the scene then measured the total time taken to trace the scene and the total size of the octree file produced. We also produced maps showing the rendering cost for each pixel in the image for each of the subdivision levels, figure 8.

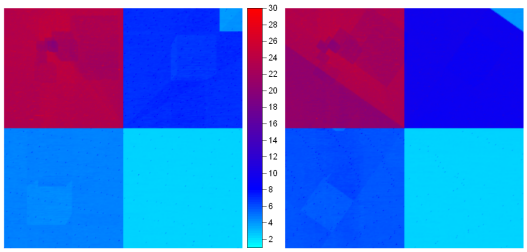

**Figure 5:** *Proportional timing maps from MULTI and MULTIROT for octree with single level of subdivision.*

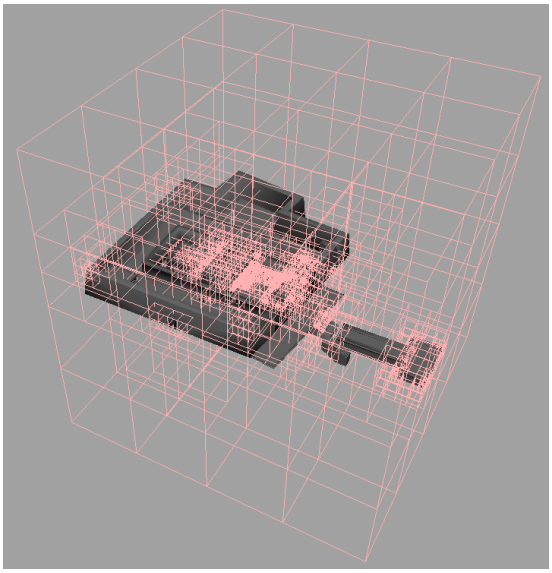

**Figure 6:** *The Kalabsha octree at a subdivision criteria set at <100 polygons per octant.*

As the rendering cost maps in figure 8 show, the subdivision strategy can have a dramatic effect on the cost of rendering an area of the scene. In detailed view "a" there is a section of sky that should, at first impression, take little time to render. Due to the subdivision strategy however, rays passing through this octant have to perform an intersection test with the top left corner of the temple structure. This adds a cost to the whole octant, as is shown by the dark blue cube protruding into the sky area. In detail "b" the reduction in the maximum allowable number of objects per octant before subdivision from 16 to 8 has triggered subdivision. Consequently, the cost of intersections with the corner is now confined to a series of much smaller octants and the cost of rendering the sky in that area has reduced.

Similarly in details "c" and "d" the reduction from 32 (d) to 4 (c) maximum objects per octant has meant that the octants containing parts of the pillars have reduced in size and

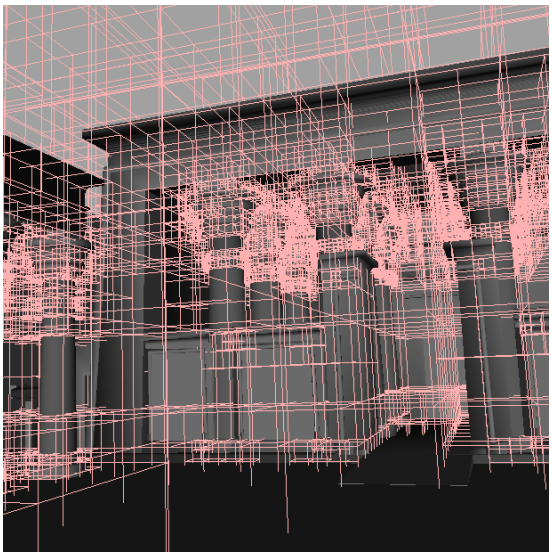

**Figure 7:** *The Kalabsha octree seen from the camera.*

increased in number. The number of rays performing failed intersection tests on the nearest visible pillars before intersecting the further pillars or back wall is therefore reduced to those that are very close to the near pillars, improving the cost balance across the area.

From these results it can clearly be seen that, as stated above, the costs associated with having to perform intersection tests with many objects in an octant greatly increases the cost of rendering the scene. However, performing endless subdivision is not the answer as the cost increases when traversing the huge number of octants produced as the 2 objects per octant maximum results shows. A level of around 6 objects per subdivision (the standard Radiance default setting) would be the best balance between cost and practical size for this scene, figure 9.

## 5. Cost Map

We can produce a cost map for the scene at a particular octree subdivision level by sampling a small number of pixel regions and then using the resulting timing map to produce an estimate for a higher resolution full image. For the sake of presentation clarity in this paper, we have used a $32\times32$ regularly spaced grid to sample the timings for the Kalabsha model octree at subdivision levels of 100 and 6 objects per octant maximum and compared these with the results from a $256\times256$ timing map of the view, figures 10 and 11.

It can clearly be seen that there is an accurate correlation between the timings from the lower resolution cost map and the results from the final timings on the larger image. The errors per pixel between the value for predicted cost and the actual cost of that pixel in the final image are shown on the

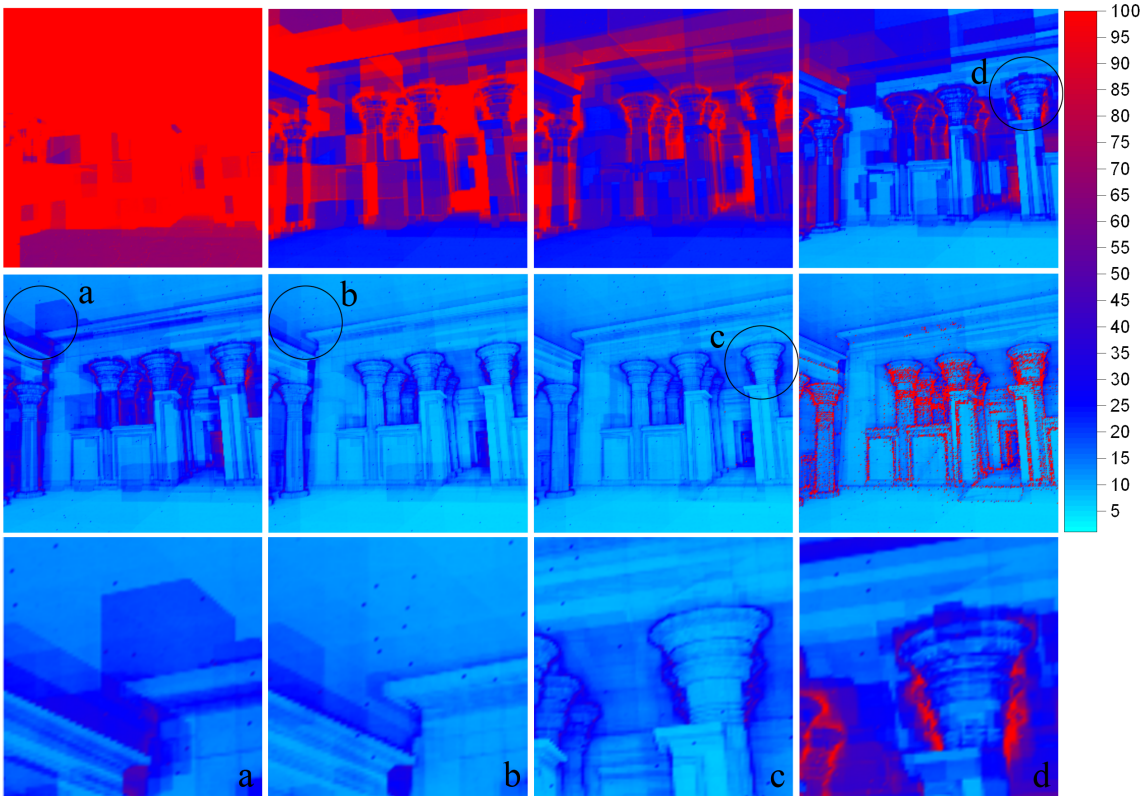


**Figure 8:** *The Kalabsha timing maps for different octree subdivisions (256, 128, 64, 32, 16, 8, 4, 2 objects per octant maximum) with enlarged sections for detailed comparison a-d.*

right hand side of both figures 10 and 11. Positive error values represent pixels that took longer to render than predicted, negative error values represent pixels that took less time to render than predicted.

The least accurate areas of prediction are, as expected, those where the cost map grid cell covers a wide change in costs, this demonstrates the trade off between the low cost of sparse sampling and the potential for inaccurate results.

These results for the errors in timing also show that while the proportional error remains the same for both levels of subdivision, the reduction in the maximum number of objects per octant reduces the absolute error by a factor of 10. This result gives another strong argument towards an optimum level of octree subdivision.

## 6. Conclusion and Future Work

In this paper we have investigated the cost of ray tracing differing parts of a variety of scene octrees. We have established the link between the relative costs of different levels of octree subdivision and for a complex scene demonstrated the most practical balance between size and cost.

We have also demonstrated the accuracy of estimating the costs of rendering an image by first producing a low resolution map and that the costs are applicable to the large scale image.

Using profiling rays for timing prediction means that although these rays are never wasted since they form part of the final image, there is a cost associated with the profiling that could possibly be unnecessary. The next stage in our work is to derive the cost prediction map by means that do not require the shooting of profiling rays as this would produce all the demonstrated benefits without the risk of unnecessary cost. This could be done by using a fast rasterised preview of the image [LDC05] to produce a complexity map of the image.

The complexity map could also help to determine areas where the costs are expected to change rapidly and hence reduce the error in prediction caused by the uniform sampling strategy as described in the previous section.

Producing a cost map without profiling could also enable the use of an efficient subdivision strategy at octree creation to tune the costs and file size to the situation.

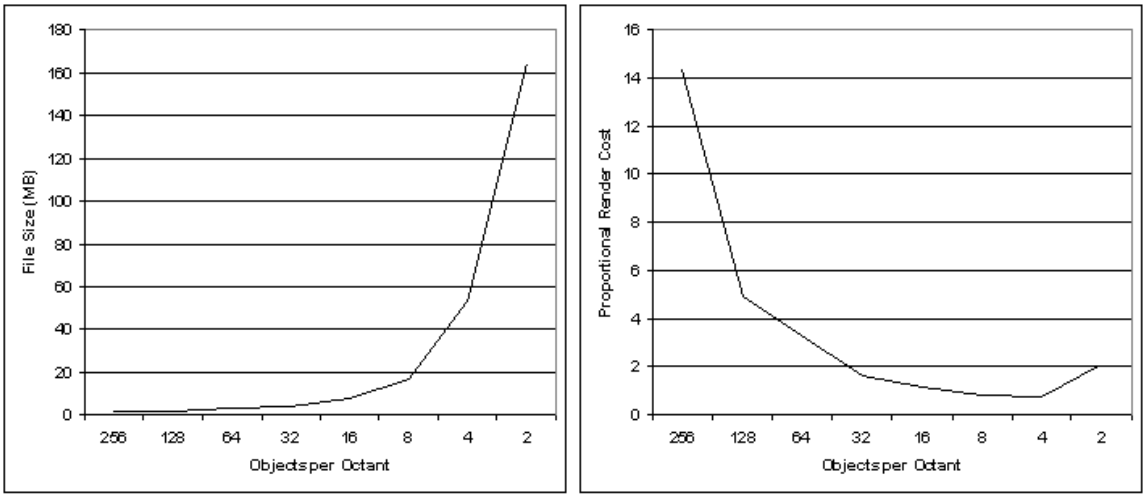


**Figure 9:** *The effect of subdivision parameter on octree file size and render cost.*

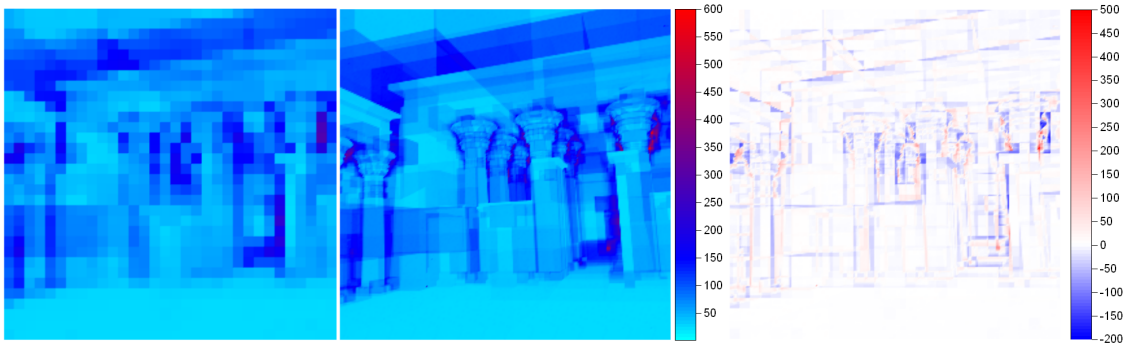


**Figure 10:** *The Kalabsha cost, timing and error maps for 100 objects per octant maximum.*

Currently our cost prediction method only calculates the cost for primary rays, this clearly needs to be extended to add the timing for secondary and further rays.

## 7. Acknowledgements

We would like to thank Veronica Sundstedt for allowing us to use the Kalabsha model. 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

[ABCC03] ARONOV B., BRONNIMANN H., CHANG A. Y., CHIANG Y.-J.: Cost-driven octree construction schemes: an experimental study. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry* (2003), ACM Press, pp. 227–236.

[CCW03] CATER K., CHALMERS A., WARD G.: Detail to attention: exploiting visual tasks for selective rendering. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (2003), Eurographics Association, pp. 270–280.

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

[DPF03] DUMONT R., PELLACINI F., FERWERDA J. A.: Perceptually-driven decision theory for interactive realistic rendering. *ACM Trans. Graph. 22*, 2 (2003), 152–181.

[FS93] FUNKHOUSER T. A., SÉQUIN C. H.: Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), ACM Press, pp. 247–254.

[HL97] HORVITZ E., LENGYEL J.: Perception, attention,

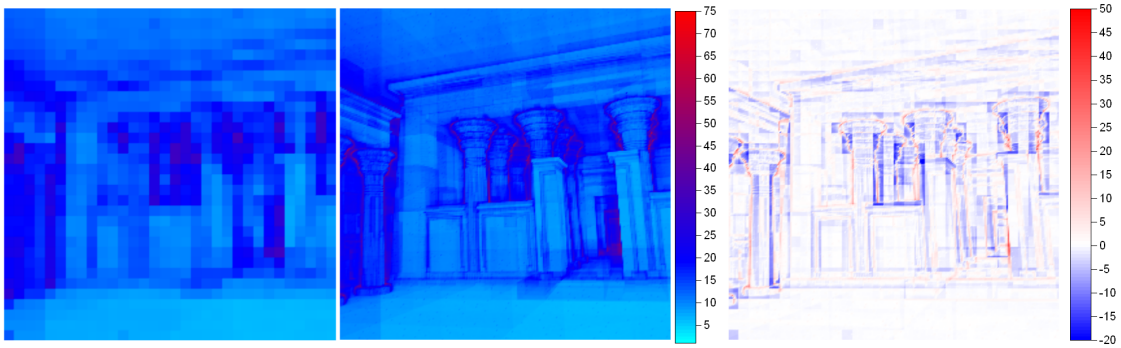


**Figure 11:** *The Kalabsha cost, timing and error maps for 6 objects per octant maximum.*

and resources: A decision-theoretic approach to graphics rendering, 1997.

[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), pp. 81–84.

[RKC98] REINHARD E., KOK A. J. F., CHALMERS A.: Cost distribution prediction for parallel ray tracing. In *Second Eurographics Workshop on Parallel Graphics and Visualisation* (September 1998), Eurographics, pp. 77–90.

[RKJ96] REINHARD E., KOK A. J. F., JANSEN F. W.: Cost prediction in ray tracing. In *Proceedings of the eurographics workshop on Rendering techniques '96* (1996), Springer-Verlag, pp. 41–50.

[SCM04] SUNDSTEDT V., CHALMERS A., MARTINEZ P.: High fidelity reconstruction of the ancient egyptian temple of kalabsha. In *AFRIGRAPH '04: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa* (2004), ACM Press, pp. 107–113.

[War94] WARD G. J.: The radiance lighting simulation and rendering system. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM Press, pp. 459–472.

[WW03] WIMMER M., WONKA P.: Rendering time estimation for real-time rendering. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (2003), Eurographics Association, pp. 118–129.