Cost Prediction in Ray Tracing

Erik Reinhard[1], Arjan J. F. Kok[2], Frederik W. Jansen[1]

[1]Faculty of Technical Mathematics and Informatics,
Delft University of Technology

[2]TNO/FEL
The Hague
The Netherlands

Abstract: Although it is generally known that ray tracing is 'time consuming', yet rewarding with respect to image quality, there are few attempts to predict the rendering time for a given model in advance. This paper focusses on the development of such a technique.
The cost of ray tracing using adaptive spatial subdivisions has been studied by analysing the probability that a ray intersects an object. Per spatial subdivision cell the surface area relative to the cell size provides a measure for this probability. This cost function is refined by taking into account possible overlap when multiple objects inhabit the same cell. A further refinement is applied by computing the average tree depth of the spatial subdivision and by assuming that each ray will on average traverse the spatial subdivision at this depth. To evaluate and validate our method we applied it to some complex models and compared the results with the actual rendering cost.

1 Introduction

Computer generated photo-realistic imaging of complex scenes is an important tool for applications such as lighting design of offices, theatres, and conference rooms. However, the demands on processing power, memory size and specific lighting engineering expertise that are required for such simulations may easily exceed the local capacities of small architectural practices and firms. For this reason one may expect in the coming years a market for commercial Internet services that will offer render services on demand[1]. The service provider will be able to organise access to pools of workstations and parallel processors on a larger scale than individual customers.

However, there are many practical aspects involved with such a kind of remote service, for instance to guarantee that models to be rendered and supplied by the customers are geometrically correct and contain all the required material specifications for the shading calculation. An other important aspect is an accurate and a priori estimate of the total rendering time and cost for each image. Some model parameters have a clear and direct impact on the rendering time, but other factors may be much more difficult to assess.

In the following, we will briefly discuss some of these parameters and then focus on the cost estimate for ray tracing using a spatial subdivision method.

[1] This is a basic assumption of the 3DOME project (3D Object MEdiator), a pre-competitive research project in the Netherlands

For our discussion we assume the rendering package to be an extended ray tracer such as Radiance [1] that, in addition to the standard ray tracing functionality, also takes into account the diffuse and specular inter-reflection. Major factors that effect the rendering time are the image size and the number and type of light sources in a scene. Each point light source will call for an extra shadow ray for each new intersection point. Area light sources may multiply this number.

A second important factor will be the material properties of the objects in the scene. They influence the number and direction of secondary rays. The number of secondary rays may be calculated by averaging the material properties over all surfaces in the scene. In that case we neglect the spatial distribution of the material properties over the scene, which could be of crucial importance, for instance if we compare looking right in a mirror with looking at a wall next to the mirror.

A third factor is the geometric complexity of the scene. Here also the view point may not be neglected: one object may block the vision on a complete scene, although in practice it will not be likely that such a viewpoint will be chosen.

An accurate cost estimate can be obtained with a low resolution ray tracing pass. A drawback of this approach, is that the whole procedure of data preparation and resource scheduling has to be performed. This way of cost estimation is therefore applicable only at the last step of the procedure, just before the final rendering task is issued.

In an earlier stage of the process, we would like to have a first estimate not strictly based on all the details, but only based on information derived from a first quick pass through the model data. In this stage we still want to abstract from viewing and shading parameters, and only have a general cost estimate as an average for any image generated with the given geometric model.

The next step would then be to include the shading parameters in the cost estimate, and only at the end a more accurate estimate could be made based on the actual viewing parameters. This third step could be done with a fully functional ray tracer. This leads to the following three-step approach for cost estimation:

– first estimate an average cost per ray on basis of the averaged geometrical properties of the scene
– then estimate the number of rays based on the shading parameters (light sources, material properties) averaged over the scene
– finally, do a full-functional low-resolution ray tracing pass using the real geometric model data, shading and viewing parameters

In this paper we will limit the discussion to the first step and only give some preliminary results for the other steps. The average cost for a ray is very much dependent on the parameters of the spatial subdivision used for reducing the number of ray object intersection tests. The creation of the spatial subdivision can be done in a pre-processing phase and can easily be combined with the initial checking and parsing of the model data. We will therefore use statistics obtained from building the spatial subdivision to derive a cost function.

Ray tracing with a spatial subdivision is generally assumed to be $O(\log N)$, where N is the number of objects. There is some theoretical support for this assumption, but in practice the performance is often much better. For instance, from our own empirical tests we did not found a strong correlation between the rendering time and the number of objects. By running the objects of the SPD data base [2] for different model sizes, we only found a slight (linear) increase in time as a function of the number of objects, in addition to a rather high initial basic cost for a small number of objects. This was true for both the (two-level) grid, bintree and octree [3]. One could therefore argue that ray tracing with spatial subdivision is either constant or linear, purely on how much one values the small increase in time. Different models, however, show significant differences in rendering time. The rendering time is therefore more a function of the geometric and material properties of the model than of the number of objects.

Several authors have analysed the time complexity of ray tracing in order to derive optimal parameter settings for spatial subdivision methods. They derive a cost function to optimise the depth/resolution of the spatial subdivision.

Cleary and Wyvill [4] derive an expression that confirms that the time complexity is less dependent on the number of objects and more on the size of the objects. They calculate the probability that the ray intersects an object as a function of the total area of the subdivision cells that (partly) contain the object.

MacDonald and Booth [5] use a similar strategy but refine the method to avoid double intersection tests of the same ray with the same object. They determine the probability that a ray intersects at least one leaf cell from the set of leaves within which a particular object resides. In addition they propose a scaling factor to account for the fact that a ray may early terminate at an other object. Empirical evidence suggests that this factor is proportional to the density of the objects in the scene. They use this cost function to find the optimal cutting planes for a k-d tree construction. A similar method was implemented by Whang *et. al.* [6].

Subramanian and Fussell [7] present yet a more accurate model. They estimate the average number of cells traversed by a ray and the average probability that a ray finds an intersection in a cell. This intersection probability is estimated using the projected area of the box enclosing the objects in a cell.

In this paper we will extend this approach in the sense that the average ray cost is expressed as a function of the average cell size of the spatial subdivision in combination with the average of the blocking (intersection) factors for each cell. The emphasis will not be on finding an optimal spatial subdivision, although this would be possible using our method as well, but on an accurate cost estimation for rendering using a given spatial subdivision.

In section 2 an expression is derived that estimates the number of cells each ray will traverse on average. To evaluate and validate our method we have applied it to some complex models and compared the results with the actual rendering cost, see section 3.

Some other factors that influence rendering costs are assessed in section 4. These include the number of light sources, the eye point and material properties. Conclusions regarding our cost per ray estimate and the above mentioned factors are drawn in the final section.

2 Cost per ray

The cost per ray generally depends on ray traversal costs and the number of ray-object intersections. Both issues are addressed in this section.

If no spatial subdivision is used, then obviously the number of ray-object intersection is linear in the number of objects, while there is no ray traversal cost. For spatial subdivisions such as the grid, the bintree and the octree, the number of objects intersected is greatly reduced, but ray traversal becomes more expensive. Assume for simplicity that the number of cells in a spatial subdivision equals the number of objects N in the scene (which in practice is usually close to optimal). Further assume that the objects do not block any rays and that any tree structures are completely balanced. Finally, the size of the objects is assumed to be small with respect to the cells they occupy. Under these simplifying circumstances, an upper bound for the following spatial subdivisions may be constructed (T_{cell} is the cost for traversing a single cell and T_{int} is the cost associated with a ray-object intersection test):

grid The number of grid cells is N, so that in each direction x, y and z the number of cells is $\sqrt[3]{N}$. A ray will travel linearly through the structure, visiting the same number of cells on average:

$$T = \sqrt[3]{N}(T_{cell} + T_{int}) = O(\sqrt[3]{N}) \tag{1}$$

bintree For a bintree with N leaf cells, the height of the tree will be h, where $2^h = N$. The number of cells traversed by a single ray is then $O(2^{\frac{h}{3}})$, giving:

$$T = 2^{\frac{h}{3}}(T_{cell} + T_{int}) = \sqrt[3]{N}(T_{cell} + T_{int}) = O(\sqrt[3]{N}) \tag{2}$$

octree Finally, in a balanced octree with N leaf cells, the height of the tree is h, where $8^h = N$. A linear traversal of a ray with such an octree will intersect $O(2^h)$ cells:

$$T = 2^h(T_{cell} + T_{int}) = \sqrt[3]{N}(T_{cell} + T_{int}) = O(\sqrt[3]{N}) \tag{3}$$

In practice, such an upper bound almost never occurs. First of all, objects have a positive surface area, which means objects can block rays. Second, objects are generally not homogeneously distributed over space. An octree or bintree spatial subdivision will therefore not be balanced. Both assumptions account an increase in performance over $O(\sqrt[3]{N})$. For this reason, a cost estimation based on a simple object count will not be very accurate.

In the remainder of this section an expression is derived which incorporates both the blocking capabilities of objects and the unequal distribution of objects over space. The expression only deals with the octree. A similar expression will be easy to derive for the bintree. As the grid spatial subdivision does not adapt to the object distribution over space, it should be possible to derive a cost estimate based on blocking properties of objects alone.

2.1 Average tree depth

A key notion in the remainder of this section is the fact that an unequal distribution of objects leads to an unbalanced octree. During ray traversal, each ray will traverse through differently sized cells with different numbers of objects in each cell (but with a given upper bound). For an accurate cost estimation of the ray traversal, we propose to actually build the octree and derive a cost function based on the average depth of the leaf cells (we'll call this average tree depth).

In the average tree depth computation, large cells contribute more to the depth than small cells, because large cells stand a higher probability of being traversed than small cells. The weights are computed according to the area of a face of the cells in that level, because the surface area of a cell determines the probability that a ray will enter this cell. The weighted average tree depth $\bar{D}$ thus becomes for the octree:

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

in which k is the number of leaf cells in the tree and h_i is the level of the i^{th} leaf cell. The total number of cells at this depth of the octree would be $8^{\bar{D}}$. A ray linearly traversing through the octree would therefore on average intersect $2^{\bar{D}}$ cells, if it wasn't blocked by any object. Because rays tend intersect objects, this expression should be refined, which is the topic of the following sections.

2.2 Blocking factor

In addition to computing the average tree depth, the estimation is refined by computing an average blocking factor, which accounts for the fact that once a ray enters a leaf cell, it may intersect an object with some probability p. This probability is expressed by the ratio of the area of each object and the area of the cell it is in. To speed up this computation, the surface areas of the bounding boxes of each object in a cell may be taken as an approximation. A two-dimensional example is given in figure 1 (left). The blocking factor p_i for leaf cell i containing n_i objects in the three-dimensional case is given by:

$$p_i = \sum_{j=1}^{n_i} \frac{A_{b_j}^x + A_{b_j}^y + A_{b_j}^z}{A_{h_i}^x + A_{h_i}^y + A_{h_i}^z} \tag{5}$$

If more than one object is present in a cell, the projections of their bounding boxes on the faces of the octree cell, may overlap (figure 1 (right)). The total area covered by multiple objects may be less than the sum of the bounding box areas. For this reason, the overlap is compensated for with the following algorithm.

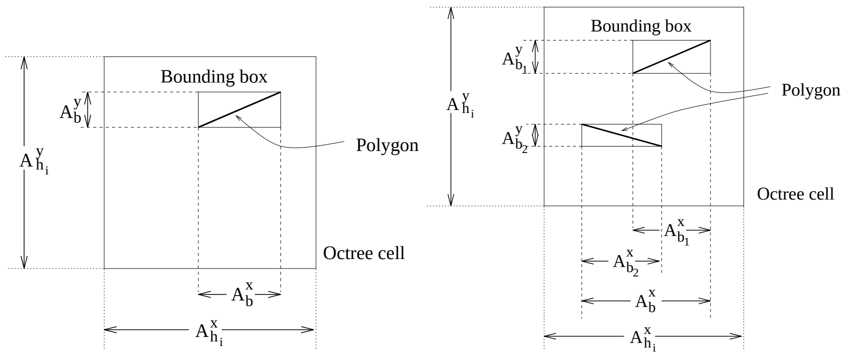

Fig. 1. 2D example of fast computation of blocking factor for one object p_i: $\frac{A_b^y + A_b^x}{A_{h_i}^y + A_{h_i}^x}$ (left). The overlap of bounding box projections in x direction is compensated for (right).

For each X, Y and Z direction, the bounding box of each object is computed. Each bounding box is in turn tested. During this operation, a list of bounding boxes is maintained as follows. First, the area of the bounding box is added to the total surface area. Then the bounding box of the current object is tested against all bounding boxes in the list. The overlaps computed this way are added to the list provided their area is larger than zero. In that case the overlap area is added to or subtracted from the total surface area. Finally, the bounding box of the current object is added to the list.

The resulting area-sum is divided by the sum of the areas of the cell faces to give the blocking factor p_i for cell i. The blocking factor computed this way is always overestimated because a bounding box usually has a larger surface area than the projection of the relevant object onto the bounding box. The average blocking factor $\bar{p}$ can be computed by weighing the contributions of each cell according to the surface of the leaf cells:

$$\bar{p} = \frac{\sum_{i=1}^{k} p_i 8^{-h_i}}{\sum_{i=1}^{k} 8^{-h_i}} \tag{6}$$

2.3 Ray traversal cost

The traversal cost per ray C_t is affected by the weighted average blocking factor $\bar{p}$ and the weighted average tree depth $\bar{D}$ in the following way:

$$C_t \cong \sum_{i=1}^{\lfloor 2^{\bar{D}} \rfloor} ip(1-p)^{i-1} + 2^{\bar{D}}(2^{\bar{D}} - \lfloor 2^{\bar{D}} \rfloor)p(1-p)^{2^{\bar{D}} - \lfloor 2^{\bar{D}} \rfloor} + 2^{\bar{D}}(1-p)^{2^{\bar{D}}} \tag{7}$$

In plain words, the cost per ray C_t is congruent with the sum of the probabilities that this ray is blocked in the i^{th} cell plus the probability that the ray isn't blocked by any object at all (last term). Each ray may encounter at most $2^{\bar{D}}$ leaf cells, as it traverses linearly through the octree. The middle term in equation 7 accounts for the fact that $2^{\bar{D}}$ may be non-integer.

2.4 Ray-object intersections

The number of intersection computations per leaf cell should be fairly constant, as during octree creation, a cell is split whenever a certain threshold is reached. The average number of objects per cell may be taken as an estimate for the number of intersection tests per cell.

If $\bar{m}$ is the average number of objects per leaf cell, the number of intersection tests can be estimated with $\bar{m}\,C_t$. The cost per ray is now estimated to be the sum of the traversal cost and the number of intersections (α and β are algorithm and machine dependent constants):

$$C_{ray} = \alpha C_t + \beta\,\bar{m}\,C_t \tag{8}$$

3 Measurements

Tests have been performed to verify our hypotheses. We compare our estimates based on the octree with a low resolution rendering. The models used in the comparison are realistic and fairly large, see figure 2. The implementation is based on the Radiance lighting simulation package.

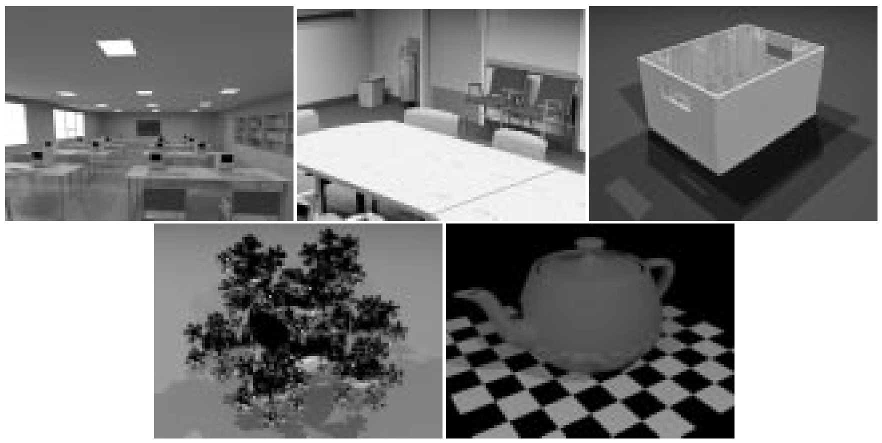

Fig. 2. Models of computer room (10,929 objects), conference room (23,177), crate (12,668), balls (7,385) and teapot (6,420).

For each model, several octrees were generated with different parameter settings. The maximum number of objects per cell parameter was varied, which affects the average tree depth and the average number of objects per cell, and thus our cost estimate should vary accordingly. On the other hand, as the octree is built differently, the ray traversal cost measured during rendering will vary as well. For the estimates to be reliable, they should correlate with the actual traversal costs. The same holds for the number of intersection tests.

First, the ray traversal cost estimator C_t is compared with the actual ray traversal cost in low resolution renderings. Results for the three models are given in figure 3 (left). For each model separately, the least squares method is applied to find a best fit, indicated in the figure with a straight line.

These tests reveal that, with the exception of the balls model, the ray traversal cost is in general accurately estimated with our cost model. A probable cause for the miss-behaviour of the balls model is the ground plane in this model, which generates an excessively large bounding box and destroys the effectiveness of the octree.

The octrees that were deeper subdivided provide a better approximation of the blocking factor which accounts for the increased accuracy of the entire estimate towards the lower left hand corner of figure 3 (left).

Second, we compared the estimated number of object-intersection tests with a low-resolution rendering. The results are given in figure 3 (right). From this figure we may deduce that $\bar{m}\, C_t$ (see section 2.4) provides a reasonable estimate for the number of intersection tests that may be expected per ray.

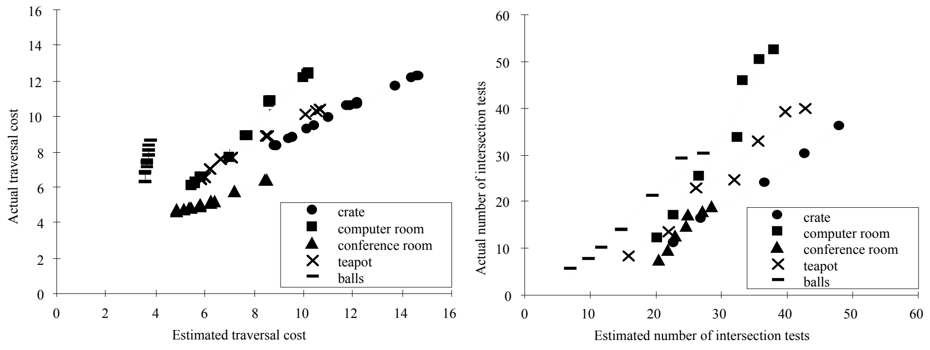

Fig. 3. Estimated traversal vs. the actual traversal cost measured in cells (left) and estimated number of intersection tests vs. the actual number of intersection tests per ray (right).

4 Other parameters

As mentioned in the introduction, the geometric complexity is only a first factor in estimating the cost of ray tracing. Other parameters are the number of light sources, shading properties and the view point. In this section we assess the (un)importance of these factors.

As expected, the number of rays depends linearly on the number of light sources, see figure 4 (left). Therefore, this factor can be determined by simply counting the number of light sources in the scene.

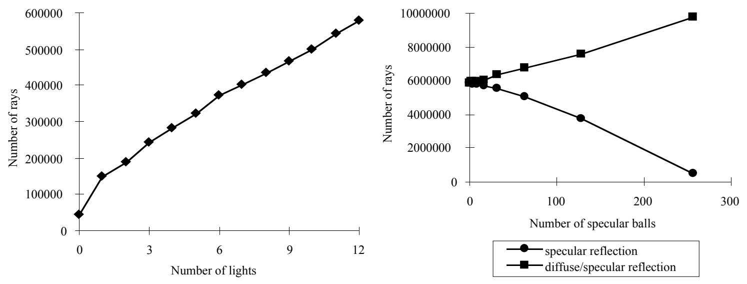

Fig. 4. Number of rays as function of the number of light sources (computer room model, left) and number of rays as function of material used (right).

Surface properties also play an important role in the number of rays spawned. In figure 4 (right) the number of rays is given for a model consisting entirely of spheres. To the left of the figure, all spheres are pure diffuse reflectors. Towards the right, an increasing number of spheres is replaced by either specular reflecting spheres or mixed diffuse and specular reflecting spheres. Dependent on the new material, the total number of rays either increases or decreases dramatically. These results imply that a measure for determining the influence of material properties (and their distribution over space) on rendering time needs to be developed. This aspect is currently under investigation.

The impact of changing the eye point on the number of cell traversals is assessed. For the crate and computer room models, an animation was made in which the eye point and viewing direction were varied. Because the same octree was used for each computation, the average number of objects per leaf cell remains constant. The number of intersection tests performed is therefore largely dependent on the number of leaf cells traversed.

The number of cells traversed on average was 12.46 and 7.73 for the computer room and crate models respectively. The variance over 14 resp. 13 images was 0.012 and 0.019. These results show that the view point is insignificant when it comes to estimating the ray traversal cost. The number of intersection tests depends strongly on the number of cells traversed, and is therefore almost constant as well. A possible cause for the constant cost while varying the eye point, is that primary rays constitute only a small part of the total number of rays.

5 Conclusions

For a rendering service it is important to be able to estimate the cost of each job, both for scheduling purposes and for submitting a quotation to potential customers. In a parallel ray tracing application, predicting the cost of sub-parts of the octree may be used to find a good data-distribution over the available processors.

In order to estimate the rendering time required for a given model with reasonable accuracy, it is necessary to build the spatial subdivision that will be used for rendering. Our method derives an estimate based on statistics derived from the spatial subdivision. A weighted average tree depth is used to estimate the number of cells that will be traversed per ray if there are no objects that block the ray.

The average tree depth computation can be refined by taking into consideration the blocking abilities of surfaces. For each cell the blocking factor is approximated as the ratio of the surface areas of the bounding boxes of objects and the surface area of the cell itself. Because bounding boxes are used, instead of the surface areas of the objects themselves, in most cases the blocking factor is slightly over-estimated. The traversal cost is therefore slightly under-estimated. The computational cost of an improved blocking factor calculation, however, is expected to be high.

The combined effect of blocking factors and the weighted average tree depth leads to an expression which estimates the expected number of cells traversed by each ray reasonably well. The basic underlying assumption is that rays originate from random positions in the scene and also travel into random directions. The accuracy of our estimates and the fact that rendering with different viewing positions has no significant effect on traversal costs, show that this is a reasonable assumption. This is partly due to the fact that in Radiance at each intersection a hemisphere is sampled to account for diffuse inter-reflection.

Because the number of intersection tests can be estimated reasonably well too, C_{ray} is a sufficiently accurate estimator for the cost per ray. Only the algorithm and machine dependent constants α and β in expression 8 need to be specified, which may be accomplished by rendering a number of images using different models. The rendering times should then be tied to these parameters.

We also ran some preliminary tests to show which additional factors may be of importance to estimate the total rendering time. Light sources correlate linearly to the total rendering time, which is according to expectation.

The influence of the eye point is negligible in our experiments, but this is most probably due to inter-reflection calculations that constitute a far larger number of rays than there are primary rays. Moreover, diffuse reflection rays tend to travel into random directions, and thus adhere to our assumptions of randomness. Generally, the eye point can not be regarded as an insignificant factor.

Finally, for material properties and their distribution over space, a suitable estimator should be developed, because their influence on the total rendering time is evident.

Acknowledgements

This work has been carried out as part of the 3D Object Mediator (3DOME) project, which is being funded by the HPCN foundation and TNO/FEL.

We would like to thank Greg Ward and Anat Grynberg for modelling the conference room model. The crate model is courtesy BPO Delft. Joost van Lawick van Pabst and Wim de Leeuw contributed to this work with precious comments.

References

1. Ward, G. J.: ‘The radiance lighting simulation and rendering system’, *ACM Computer Graphics* pp. 459–472. (1994) SIGGRAPH ’94 Proceedings.
2. Haines, E. A.: Standard procedural database, v3.1. 3D/Eye. (1992)
3. Reinhard, E., Jansen, F. W.: ‘Pyramid clipping’, Ray Tracing News, volume 8, number 2. (1995)
4. Cleary, J. G., Wyvill, G.: ‘Analysis of an algorithm for fast ray tracing using uniform space subdivision’, *The Visual Computer* (4), 65–83. (1988)
5. MacDonald, J. D., Booth, K. S.: ‘Heuristics for ray tracing using space subdivision’, *The Visual Computer* (6), 153–166. (1990)
6. Whang, K.-Y., Song, J.-W., Chang, J.-W., Kim, J.-Y., Cho, W.-S., Park, C.-M., Song, I.-Y.: ‘Octree-r: An adaptive octree for efficient ray tracing’, *IEEE Transactions on Visualization and Computer Graphics* **1**(4), 343–349. (1995)
7. Subramanian, K. R., Fussell, D. S.: ‘Automatic termination criteria for ray tracing hierarchies’, *Graphics Interface ’91* pp. 93–100. (1991)