Accelerated Ambient Occlusion Using Spatial Subdivision Structures

Chris Wassenius

Abstract

Ambient Occlusion is a relatively new method that gives global illumination like results. This paper presents a method to accelerate ambient occlusion using the form factor method in Bunnel [2005] as well as utilizing the fact that ambient occlusion is a function of nearby geometry only. The proposed method combines existing ideas in order to accelerate ambient occlusion calculations by using a spatial subdivision structure (an octree) and selective sampling.

Keywords: ambient occlusion, ambient obscurance, global illumination, shadowing techniques, spatial data structures

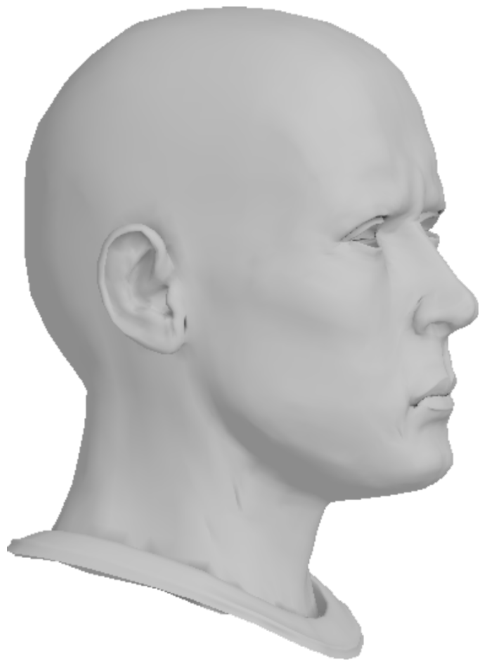

Figure One: Ambient occlusion and diffuse lighting.

1 Introduction

Global illumination in computer graphics is a key factor in producing realistic scenes. However, the complexity of a true radiosity approach is still too time consuming for modern technology. Ambient Occlusion is a technique that generates global illumination effects (soft shadowing, color bleeding) without taking into account an actual light source. Instead, the ambient term of the standard lighting equation is modified as a function of the local geometry of a scene. That is, the ambient term for a point on a surface is determined by how occluded that point is from other local surfaces in the scene [Iones et al. 2003]. Ambient Occlusion simulates global illumination fairly well considering its relative simplicity. Ambient Occlusion calculations for the most part have remained a non-real time process however. The technique presented accelerates the occlusion calculations by using a hierarchical spatial subdivision system. Hierarchical spatial subdivision systems have been used to speedup radiosity, ray tracing, occlusion culling, and many other graphics-based problems.

2 Related Work

Goral et al. [1985] first introduced the concept of radiosity, and brought global illumination effects to computer graphics. Due to radiosity's complexity various acceleration methods were later introduced. One such method is to store patches in a hierarchy [Hanrahan et al. 1991]. The idea, for example, is that instead of calculating the amount of light coming from a distant wall in a room by finding the form factor for each patch that is far way, it is possible and nearly equivalent to group those distant patches together (average their radiosities) and compute the form factor between a large patch (the entire far wall) and the receiving patch in question. This significantly reduces the amount of form factor calculations between patches, yielding a linear solution, rather than a quadratic solution, to the problem. However, the radiosity method is still exceptionally time consuming, and thus other methods that attempt to mimic global illumination have been introduced.

The idea of ambient occlusion was introduced by Zhukov et al. [1998] and as mentioned earlier gives global illumination-like effects at a much lesser computational cost. A patch's ambient term is modified based on how open that patch is with respect to other nearby patches. If a patch is fully open its ambient term is multiplied by 1, otherwise its ambient term is multiplied by a value between 0 and 1. Later, Iones et al. [2003] proposed storing the resulting ambient term computations in light maps rather than patch vertex attributes. This allows a scene to be rendered with only the original polygons, independent of the patches that were used in the occlusion calculation. Also, Méndez et al. [2003] added color bleeding support to ambient occlusion with no extra computational cost.

In most production renderers, such as Renderman, ambient occlusion is now supported, and implemented by ray tracing [Christensen 2002]. As an extension the normal of a patch is bent based on the average direction that a patch is open or not occluded. This bent-normal is then used when looking up data from an environment map. This typically gives more interesting and realistic results.

Recently, real-time, or close to real-time, ambient occlusion results have been achieved with the help of the latest graphics hardware [Bunnel 2005]. The idea is to create surface elements out of a scene 's polygonal data, and to compute the shadowing of these surface elements onto each other using a typical form factor-like equation. A Surface element is defined as an oriented disk, with a surface position, normal, and area. Every vertex in the polygonal scene, has a corresponding surface element. Similar to radiosity's hierarchical clustering technique [Hanrahan 1991], surface elements are stored in a hierarchy such that when computing the shadowing from far away elements, one large surface element may be used rather than computing the shadowing from every far away surface elements. Fragment shaders are used by the GPU with this method.

The benefit of using the form factor approach, rather than ray tracing to determine the visibility of an element is that ray tracing requires many samples to sufficiently represent the total occlusion value. The form factor approach on the other hand mimics computing the visibility of an element by using the areas of the occluders. So while ray tracing could utilize a spatial structure for determining local visibility, the method suffers from the fact that many rays are required.

3 Method

The proposed method is to accelerate the ambient occlusion calculation for a scene, by utilizing the fact that the ambient occlusion for an element is only a function of the nearby geometry in the scene with respect to that element. The method introduced uses Bunnel's [2005] idea of surface element discs and form factor-like shadowing ideology as well as Iones et al. 's [2003] idea of neighboring geometry.

Ambient occlusion as first introduced by Zhukov [1998] was a function of just that, nearby geometry, however many ambient occlusion techniques, such as the real-time method introduced by Bunnel [2005], seem to neglect this fact to an extent. Shadows cast from distant elements on a receiving element are lessened considerably by how far the occlusion elements are from the receiving element in question. This is represented in the form-factor equation. Where the form-factor between one element and another is:

$$\frac{A \cos \Theta_E \cos \Theta_R}{\pi r^2 + A} \qquad (1)$$

Where A is the area of the occluder, Θ_E is the angle between the occluder's normal and the vector between the occluder and the element, Θ_R is the angle between the element's normal and the vector between the element and the occluder, and r is the distance between the occluder's center and the element's center. This is illustrated in figure two.

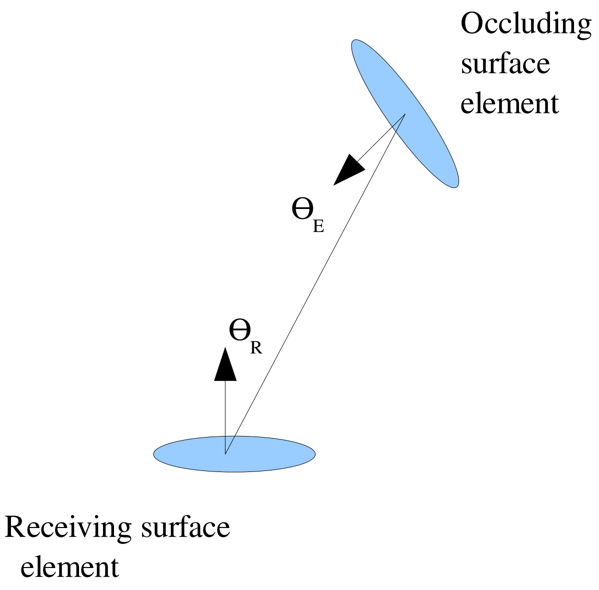

Figure Two

Based on this, while the occlusion is more so a function of nearby geometry, it is still represented as a function of the entire geometry in the scene. The occlusion from any distance from the element in question is still being computed. By enforcing a restriction on the distance in which an object can occlude or shadow another object, classic spatial subdivision techniques can be employed to accelerate the overall ambient occlusion calculation. Bunnel's [2005] method does use a hierarchical clustering technique which in effect does support the notion of locality as, occlusion is calculated only in full for a limited range, while the occlusion from further away elements is grouped and approximated. However the memory required to store multiple levels of surface elements is undesirable. The method presented in this paper focuses and examines the results of having a limited distance or cutoff for ambient occlusion, and the subsequent benefits.

Acceleration based on locality can be achieved by storing a scene in an octree. An octree was chosen due to its simplicity and applied application. Surface elements are composed of the scene's vertex data, and thus each surface element is only located at a specific position. Therefore when inserting surface elements into an octree, there is never a case where an element is located in more than one node in the tree, as would occur when inserting polygons into the tree. This is a common downside to the spatial uniformity of octree nodes when inserting polygons, as polygons must either be represented in more than one node, or must be dynamically subdivided. Although an octree risks the case of being unbalanced, this is not a concern when calculating ambient occlusion, as rejecting nodes is not an expensive operation with respect to the form-factor calculations that are required when computing ambient occlusion. However, an issue when using an octree, particularly if the ambient occlusion implementation is using the GPU, is that an unbalanced octree wastes valuable memory storage. In this case a more balanced structure, such as a kd-tree would be more suitably. On the other hand, for real-time purposes and with applications with dynamic environments, a kd-tree would be harder to maintain than a more general structure, such as an octree.

Surface elements should be stored in an octree such that there are relatively few surface elements per octree node, with respect to total amount of surface elements in the scene. The implementation used for this paper restricted a nodes capacity to 10 surface elements when handling scenes of 100,000 to 300,000 surface elements. This is to ensure that the ambient occlusion algorithm can reject as many surface elements as possible when calculating the occlusion for an element.

With the surface elements stored in an octree (see figure three), only the occlusion from surface elements at nearby nodes of the octree are calculated. When calculating the occlusion for an element, the tree is traversed from the top down. A leaf node's surface elements are only included in a receiving elements occlusion calculation if and only if the leaf node (a cube) they are in, is considered local to the receiving element. Likewise, a node is only traversed downward to its children if the node is near the receiving element. An exception to this, is if the receiving element is itself in one of the node's children. The range of what is considered nearby or local to a receiving surface element is flexible. For ideal results however, the range should be proportional to the size of objects in the scene. Iones et al. [2003] recommend the range cover 10 to 100 surface elements.

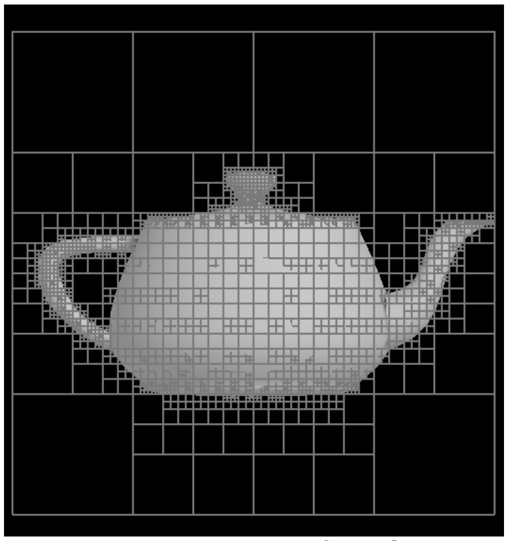
Figure 3: Scene's surface elements are stored in an octree.

Further use of the octree structure can accelerate the ambient occlusion calculation by examining the definition of ambient occlusion, and subsequently the form-factor equation. The occlusion for a surface element is a function of the hemisphere oriented above the surface normal of the element. Only the surface elements located in the hemisphere of the normal contribute to the receiving element's occlusion. Based on this, it is only necessary to traverse a node in an octree if it is close to the receiving element and also if the node is located in the hemisphere of the receiving element's normal. Testing if a node is in the normal oriented hemisphere can be done by simple dot product tests on the bounding cube of the node. For all eight corners of the bounding cube, equation two can be applied.

$$\text{if } E_N \bullet V < 0;\ \text{reject} \qquad (2)$$

If the test rejects for all eight corners, the bounding cube, and all of its subsequent children, are not in the hemisphere of the receiving element.

The hemisphere test contributes to the acceleration process the most when large ambient occlusion cutoff ranges are used. Since the test can eliminate more surface elements which the locality test does not eliminate.

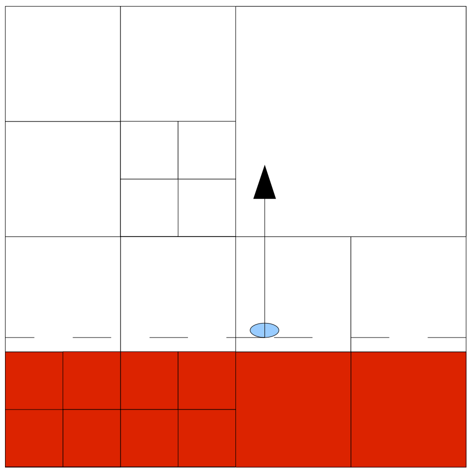

Figure 4: Two-dimensional depiction of hemisphere test. A receiving element's occlusion value is only affected by the elements of nodes in the hemisphere oriented around its surface normal. Nodes colored red are not traversed due to this.

Each calculated occlusion value can be stored directly as a vertex attribute. This works nicely since every surface element corresponds to a vertex. However, since the number of surface elements is no greater than the number of verticies the quality of the results is greatly reduced in areas of little geometry.

Due to time constraints and simplicity, the implementation presented in this paper was done in software on the CPU, rather than hardware. However, despite this the acceleration gained is equally applicable if the GPU was used.

4 Results

By limiting the range of occlusion, the acceleration gained was significant, as seen in Figures five and six. Also, by comparing the following pictures, it can be seen that occlusion can be accurately represented even with a tight limit on its range. Indeed, as by definition, the ambient occlusion of a point is most greatly impacted by the local geometry around that point. The limit on the range of ambient occlusion is also not obvious and distracting, as the form-factor equation (equation 1) is a quickly falling, but smooth function. It is also important to note that scenes with no limit on the occlusion range, appear darker, since the likelihood of over shadowing increases. Multiple passes of the algorithm would need to be run to remedy this issue [Bunnel 2005].

The hemisphere test in itself caused moderate acceleration. As stated earlier, the acceleration gained directly from the hemisphere test is related to the range used for occlusion. This is the case since, in theory, the hemisphere test rejects (does not traverse) at a maximum, half the nodes that the optimization gained by limiting the range of occlusion would accept.

All tests were done on a Pentium 4, 2.8 ghz. processor.

Scene	***Total Elements***	***Occlusion Range***	***With hemi-sphere test***	***time (secs)***
teapot	19,200	none	no	531.21
teapot	19,200	none	yes	178.8
head model	94,860	none	no	8521
head model	94,860	none	yes	4030

Figure Five: Shows the results of having no limit on the occlusion range, and with the hemisphere test optimization on and off.

Accelerated Ambient Occlusion Using Spatial Subdivision Structures

Chris Wassenius

Scene	*Total Elements*	*Occlusion Range*	*With hemi-sphere test*	*time (secs)*
teapot	19,200	1/12 of scene length	no	8.328
teapot	19,200	1/12 of scene length	yes	7.832
head model	94,860	1/12 of scene length	no	130.9
head model	94,860	1/12 of scene length	yes	113.53

Figure Six: Shows the results of having a limit on the occlusion range, and with the hemisphere test optimization on and off.

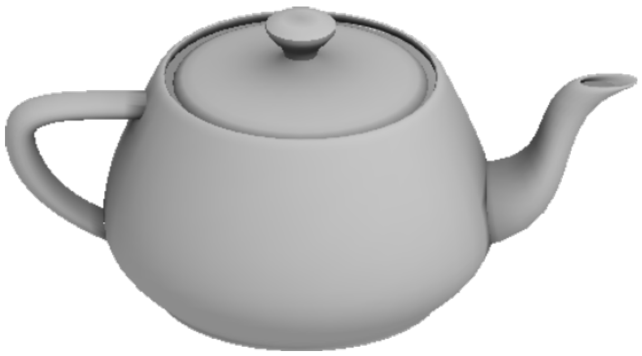

Figure Eight: Teapot with ambient occlusion with no limit on occlusion.

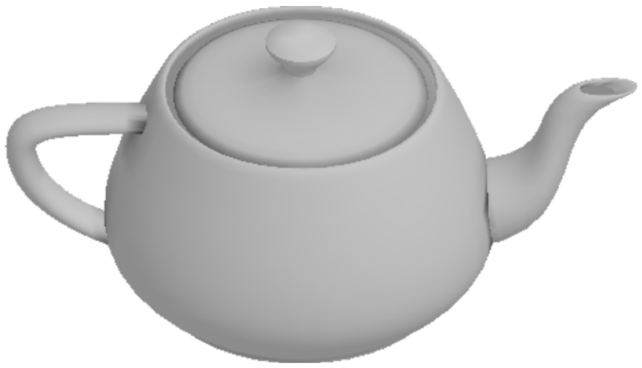

Figure Nine: Teapot with ambient occlusion with a limit on the range of occlusion.

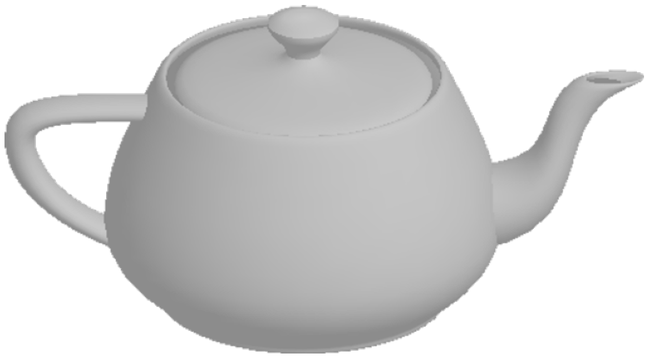

Figure Seven: Teapot with diffuse lighting only.

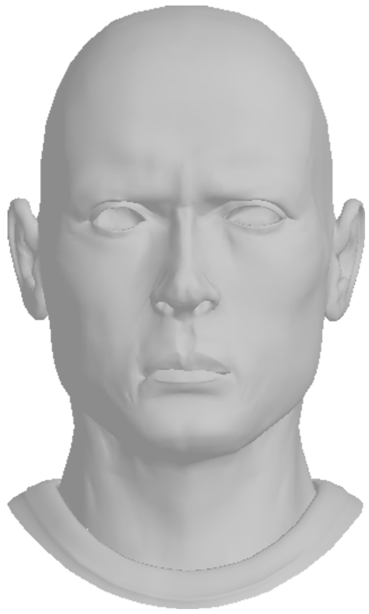

Figure Ten: Head model with diffuse lighting only.

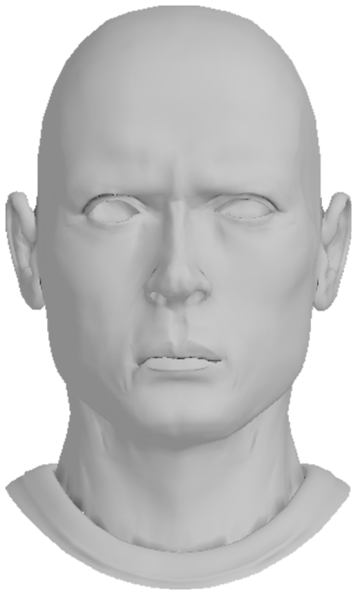

Figure Twelve: Head model with ambient occlusion with a limit on the range of occlusion.

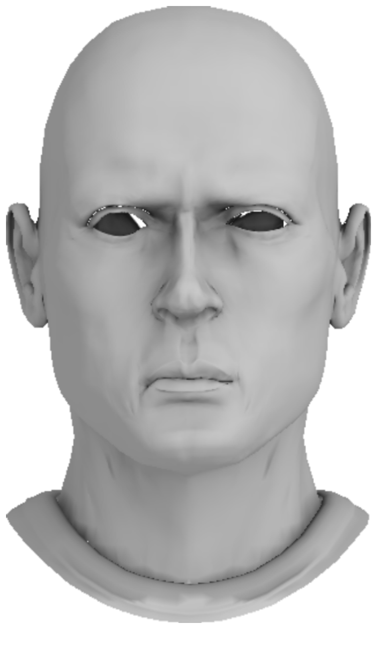

Figure Eleven: Head model with ambient occlusion with no limit on occlusion.

5 Conclusion and Future Work

The method presented in this paper uses a spatial subdivision structure (an octree) to accelerate the ambient occlusion calculation for a scene. An octree is used to exploit two properties of ambient occlusion. The first one is that ambient occlusion is a function of nearby geometry. The second one is that the occlusion of a point is determined solely by the geometry that is located in the hemisphere oriented around the surface normal of that point. By using an octree, the algorithm can quickly compute the occlusion of nearby surface elements onto a receiving element by traversing down nodes accordingly. An octree also allows for fast hemisphere tests in order to determine if the occlusion caused by the elements in a node need be computed or not by using equation 2 on nodes.

Ambient Occlusion is a good method that mimics global illumination effects at a much lower cost than radiosity. It is therefore a worthwhile method to work with until graphics hardware can support such expensive methods. In addition to being used for regular lighting purposes ambient occlusion could be used in other areas. Ambient occlusion distinctly highlights the geometry of an object and could be considered for use in the field of non-photorealistic rendering.

Accelerated Ambient Occlusion Using Spatial Subdivision Structures

Chris Wassenius

Future work could be done in many areas. Octree levels can be used to incorporate Bunnel's [2005] surface element hierarchy, with each level also containing a representative surface element that is based on all the surface elements within that level. With this in place, the restriction on the range of occlusion could still remain intact, and the surface element hierarchy could be used within this range. This would benefit scenes that have a large occlusion range. Also, work could be done to decrease the dependency of the quality of ambient occlusion on the geometry of the scene. To remedy this a subdivision technique could be used, such as Iones [2003] lightmap method. Finer patches are created by projecting input polygons onto a uniform grid. The patches that are created from this can then be used in the ambient occlusion calculation. The calculated occlusion values can then be stored in a lightmap that is the size of the uniform grid. At run time, lightmaps are applied to the original polygon data, resulting in a better result, without the need to render extra geometry (although the time taken to calculate the ambient occlusion increases proportionally and this technique is suitable only for non real-time purposes. Another possibility for future work is to explore the use of different spatial subdivision structures.

References

Bunnel, Michael. 2005. Dynamic Ambient Occlusion and Indirect Lighting. In GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation, Matt Pharr, editor, chapter 14, pages 223–233.

Christensen, P. H. 2002. Ambient Occlusion, Image-Based Illumination, and Global Illumination. PhotoRealistic RenderMan Application Notes, Note #35.

Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. 1984. Modeling the interaction of light between diffuse surfaces. In Proceedings of the 11th Annual Conference on Computer Graphics and interactive Techniques H. Christiansen, Ed. SIGGRAPH '84. ACM Press, New York, NY, 213-222.

Hanrahan, P., Salzman, D., and Aupperle, L. 1991. A rapid hierarchical radiosity algorithm. In Proceedings of the 18th Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '91. ACM Press, New York, NY, 197-206.

Iones, A., Krupkin, A., Sbert, M., and Zhukov, S. 2003. Fast, Realistic Lighting for Video Games. IEEE Comput. Graph. Appl. 23, 3 (May. 2003), 54-64.

Méndez, A., Sbert, M., and Catá, J. 2003. Real-time obscurances with color bleeding. In Proceedings of the 19th Spring Conference on Computer Graphics (Budmerice, Slovakia, April 24 - 26, 2003). L. Szirmay-Kalos, Ed. SCCG '03. ACM Press, New York, NY, 171-176.

Zhukov, S., Iones, A., Kronin, G.: "An Ambient Light Illumination Model", Rendering Techniques'98, Springer-Wien, NewYork (Proc. of Eurographics Rendering Workshop'98 - Vienna, Austria), pp. 45—55.