Honors Thesis

Perumaal Shanmugam
The University of Texas at Austin

under the supervision of

Dr. Okan Arikan
Assistant Professor
The University of Texas at Austin

Department of Computer Sciences

December 2006

“A research paper is akin to writing your own review for the film you directed”

Acknowledgements

I would like to sincerely thank Okan Arikan for strongly supporting my research endeavors and providing the excellent research opportunities and in particular the extensive creative freedom, stringent (often very high) quality standards and the environment I was able to work in.

I would also like to thank the Texas Advanced Computing Center, and in particular Gregory S Johnson for allowing me to use various resources and supporting my research in general.

Thanks to the RTGPS (Real-time Graphics and Parallel Systems) group for providing, dissecting and brain-storming graphics research in general.

A slightly modified version of the following content was submitted to the i3D ACM SIGGRAPH Symposium on Interactive 3D Graphics

Downloadable content related to this paper may be available at
http://www.cs.utexas.edu/users/perumaal/

To my parents
Dedicated in memory of my cousin Mathi Maran

Hardware Accelerated Ambient Occlusion Techniques on GPUs

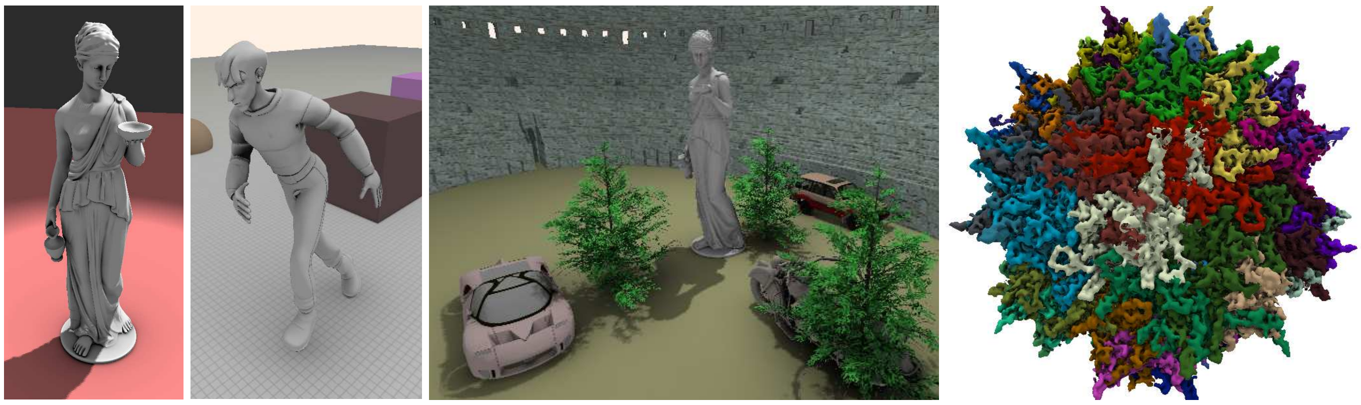


Figure 1: *These images illustrate our ambient occlusion approximation running on a modern GPU. Our method can be used for a number of applications including (a) Rendering high-detail models; this model has around 65K triangles. (b) Dynamic/deforming models (that undergo non-rigid deformations) (c) Enhancing natural objects such as trees, and in games where models such as cars are used. This scene currently renders at around 4 fps and has over a million triangles. (d) Realistic rendering of molecular data that provides similar/superior quality to specialized software such as [Bajaj et al. 2004; Tarini et al. 2006].*

Abstract

We introduce an approximate yet visually pleasant ambient occlusion approximation running on real-time graphics hardware. Our method is a multi-pass algorithm that separates the ambient occlusion problem into high-frequency, detailed ambient occlusion and low-frequency, distant ambient occlusion domains, both capable of running independently and in parallel. The high-frequency detailed approach uses an image-space approach to approximate the ambient occlusion due to nearby occluders due to high surface detail. The low-frequency approach uses the intrinsic properties of a modern GPU to greatly reduce the search area for large and distant occluders with the help of a low-detail approximated version of the occluder geometry. Our current results show a promising trend in utilizing the highly parallel, stream processors (GPUs) to perform real-time visually pleasant ambient occlusion. We show that our ambient occlusion solution works on a wide variety of applications such as molecular data visualization, dynamic deformable animated models, highly detailed geometry. Our algorithm demonstrates scalability using a multi-GPU environment, and is Direct3D 10-aware in many respects, allowing for a smooth adaptation to the anticipated fundamental changes in upcoming graphics hardware.

Keywords: gpu,ambient occlusion,soft shadows,real-time rendering

1 Introduction

Current real-time graphics applications such as games provide increasingly convincing realism, using a number of methods such as soft-shadows, high-dynamic range effects, post-processing effects and surreal lighting. Ambient occlusion is one other method that has recently attracted real-time graphics researchers aiming to increase the level of realism without performing a complete global-illumination step. Compared to other global illumination approximations, ambient occlusion gives a unique edge by providing a significant increase in quality despite a crude and a simple approximation of global illumination. We propose a real-time multi-pass approximation for ambient occlusion. We provide different approximations to the occlusion caused by nearby surface detail (often creating the high frequency detail in ambient occlusion) and the occlusion caused by distant surfaces (often creating a smooth change in ambient occlusion). We describe how these approximations can be computed on the graphics hardware to perform ambient occlusion in real-time. Our method has the advantage of being largely independent from the scene complexity. Our method approximates the ambient occlusion for each frame independently, making it suitable for deformable and dynamic objects.

2 Related Work

Ambient occlusion has been studied rigorously in the past few years both in ray-traced and hardware-accelerated (GPU) based approaches. It has found its way into films as an alternative to full-scale global illumination methods. One could compute ambient occlusion at each point, by determining the visibility along randomly selected directions. However, this method is computationally too expensive to be implemented in real-time yet.

[Arikan et al. 2005] is another method that uses near-field ambient occlusion for increasing realism in a ray-traced setting, but this method is not geared for real-time.

[Bunnell 2005] was one of the first real-time solutions that paved the way for further research in this area. This approach uses disc-based occluders and a per-vertex ambient-occlusion term; however, it requires a huge pre-computation step with little support for dynamic objects, and requiring high-tesellation of scene geometry.

[Kontkanen and Laine 2005] partially solves the dynamic object problem; however it is limited to rigid body motion and is not suitable for deformable objects. Moreover, for every occluder that we place within the view-frustum, we may have to switch textures

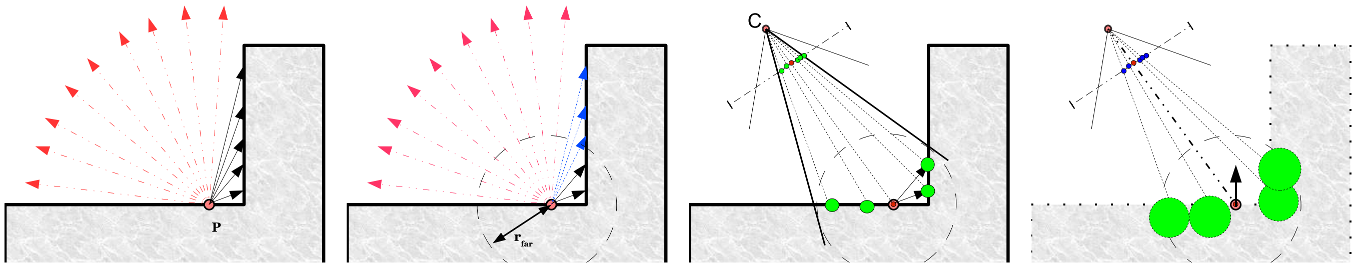


Figure 2: *Local ambient occlusion in image-space: (a) Rays emerging from a point P many of which hit no occluder (red arrows). (b) Rays being constrained within a hemisphere of radius r_far, with distant occluders being neglected (blue arrows). (c) Pixels as viewed by a camera C. The hemisphere is now approximated in image-space to obtain pixel samples within the constrained region (marked in green). (d) Now, we un-project these pixels back to world-space and approximate them using spherical occluders (green circles). These are the final approximated occluders around the point P that are sought in image-space. Note that the spheres are pushed back a little along the opposite direction of the normal ($-\hat{n}$) so as to prevent incorrect occlusion in flat surfaces.*

(cube-maps), hence leading to an explosion in the number of textures as well as expensive texture context-switches that is impractical for real-time applications especially games. However, this approach provides us insights into ways of approximating occluders and reducing over-occlusion problems.

[Ren et al. 2006] provided a different approach that has similar results. This approach uses spherical occluders and spherical harmonics to calculate approximate ambient occlusion (or low-frequency soft shadows, as described in their paper). This solution is unique in that it attacks the problem of over-occlusion and provides a simple pre-computation step while supporting deformable objects. However, this method suffers in performance when there are a large number of objects (as is the case with most real-time applications). As all of these approaches target an approximated version of the original geometry that has far less details, they tend to miss high-frequency ambient occlusion effects that would provide a huge increase in quality.

We borrow some ideas from the research mentioned above to build on our ambient occlusion solution.

There is other interesting research such as [Kontkanen and Aila 2006; Hegeman et al. 2006; Wassenius 2005] targetting specific applications such as animated characters, trees, etc. Solely precomputation based static methods such as PRT (Precomputed Radiance Transfer) [Sloan 2006; Sloan et al. 2002; James and Fatahalian 2003], baked-AO (such as [ShadeVis 2006]) are less flexible yet simpler solutions for static ambient occlusion. Ambient occlusion is also used in a wide variety of real-time applications, such as molecular rendering (QuteMol - [Tarini et al. 2006]) and nature-rendering (grass-rendering) mainly to increase realism.

3 Overview

Ambient Occlusion is a fast and a close approximation to global illumination. Ambient occlusion at a point on an object is the amount of occlusion it receives due to objects surrounding it. The ambient occlusion A at a point P with a surface normal $\hat{n}$ is given by:

$$A(p,\hat{n}) = \frac{1}{\pi}\int_{\Omega}(V(\hat{\omega},p))max(\hat{\omega}\cdot\hat{n},0)d\hat{\omega} \qquad (1)$$

where $\hat{\omega}$ represents the various directions at p in the hemisphere Ω; V is the visibility function for a given occluder along that direction at p [Kontkanen and Laine 2005]. The effect of ambient occlusion can roughly be described as the effect seen in a cloudy day; here we consider the sky to be a uniform area light source resulting in soft, low-frequency non-directional shadows [Ren et al. 2006].

4 Image Space Approach

Ambient occlusion on high surface-detail geometry tends to vary rapidly, due to the rapid appearance and disappearance of occluders along the surface. This is mainly because occluders near a point and along that point's surface normal contribute more to the ambient occlusion at the point than those that are not; this is particularly true in the case of high surface-detail geometry.

We use this observation to present an image-space approach. Nearby occluders contribute the most to this rapidly changing ambient occlusion. We then approximate these neighboring occluders in image space.

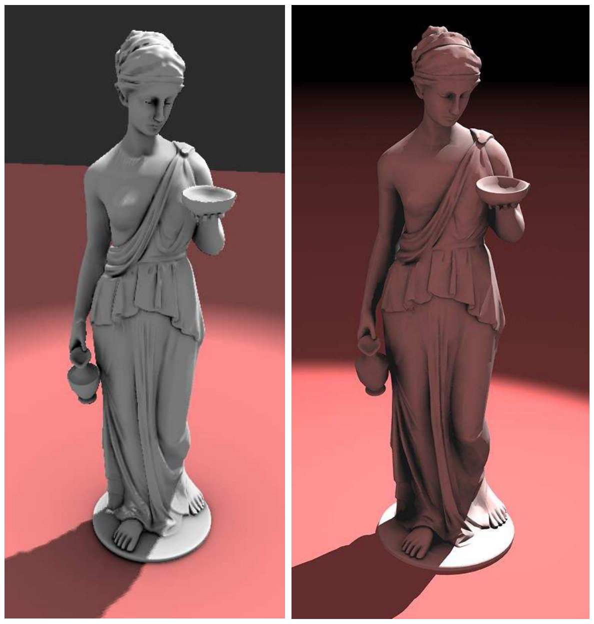


Figure 3: *Ambient occlusion generated real-time using our image-space approach (left) at about 7 fps compared with an offline version on the right ([Arikan et al. 2005]) computed in 5 minutes.*

4.1 Detailed view of the approach

We find the occlusion at a point P on a given surface by shooting rays outward from that point and finding those rays that intersect an occluder, weighted by the surface normal at P. This can be approximated further by constraining the length of the rays within a hemisphere centered at P facing $\hat{n}$ on the surface. Assume the radius of this hemisphere to be r_{far}, the far falloff distance. We can then approximate ambient occlusion at P due to the effect of those occluders that are within this hemisphere. This approximation works well for most cases, especially for high-detail geometry.

Although the idea of using local geometry as occluders is not new ([Arikan et al. 2005] and [Wassenius 2005]), there are some significant differences in the overall approach and the actual idea being implemented.

The overall idea is as follows. The *ND-buffer* is a 2D array containing the normal/depth values for every rendered pixel in the framebuffer, each pixel corresponding to a point (P) in world-coordinates. For every point P with normal $\hat{n}$ represented in the *ND-buffer*, we obtain the world coordinates, by un-projecting this point. Then, we obtain the neighboring pixels for this point P, and approximate each of those points as an occluder. For each of those occluders, we calculate the effect of ambient occlusion for the receiver point $< P, \hat{n} >$ against these approximated spherical occluders. The resulting accumulated value represents the local ambient occlusion value at P due to those approximated occluders.

Some important occluders to P may themselves be occluded in the framebuffer. Therefore, we use the first two layers of depth (as seen by the camera). We obtain these layers by rendering the front facing polygons and then the back facing polygons, and keeping the closest pixels (similar to depth-peeling [Everitt 2001]).

This algorithm is simple, efficient and highly-parallelizable. We have mapped this algorithm onto the GPU using the fragment shaders. We approximate world-space neighbors of P to its screen-space neighbors (using the two depth buffers - one for the front faces, the other for the back faces). We can search for these world-space neighbors in screen-space with the help of the depth-value of P from the camera, which provides us an accurate search area (in screen space) to find all p_i such that

$$|p_i - P| < r_{far} \tag{2}$$

where r_{far} is the far fall-off distance. This also provides us non-neighbors of P, but these non-neighbors have little noticeable effect on the occlusion performed at the pixel for P precisely for the reason that they are non-neighbors. Hence we can obtain this list of neighbors in screen space that are within the sphere of radius r_{far} from P in world-space. We can approximate the pixels in the depth-buffer as disk-occluders ([Bunnell 2005]), but we chose a non-directional occluder such as a sphere. This is because an non-directional occluder that does not have any direction has been shown to provide a better approximation and performance than an occluder approximation with an associated direction [Ren et al. 2006; Wang et al. 2006].

This approach is almost entirely constrained by the memory bandwidth available; for every point P in the depth-buffer, we access N_p points which are within the imaginary bounded hemisphere around P. This constraint is also the advantage of this approach; with faster memory/caching, we obtain faster results, shifting the majority of constraints to particular corner. N_p typically varies from 8 neighboring points (that are immediately adjacent to P) to 100s of neighboring points.

4.2 Fragment Shader

We use deferred shading and render a full-screen quad to invoke the fragment shader on every screen pixel. The fragment shader, at every pixel, finds the neighbors for that pixel and performs ambient occlusion between the occluder approximation for each of those neighbors and the given pixel (as a receiver point). These neighbors can be searched using a loop that iterates over those pixels whose world-space coordinates (P_i) are within a distance of r_{far} from P. We can calculate the screen-space coordinates of these neighbors in the GPU using 2, however we supply the coordinates of the search-pixels using a texture due to the huge cost in branching. This texture is a jittered sampling pattern provided as a Mx1 texture, each pixel of which contains two coordinates in its two components. These coordinates provide the positions of the samples around a given pixel. The motivation for the use of a texture to provide the sampling pattern is that arrays or temporary registers prove significantly expensive due to the nature of the current GPUs.

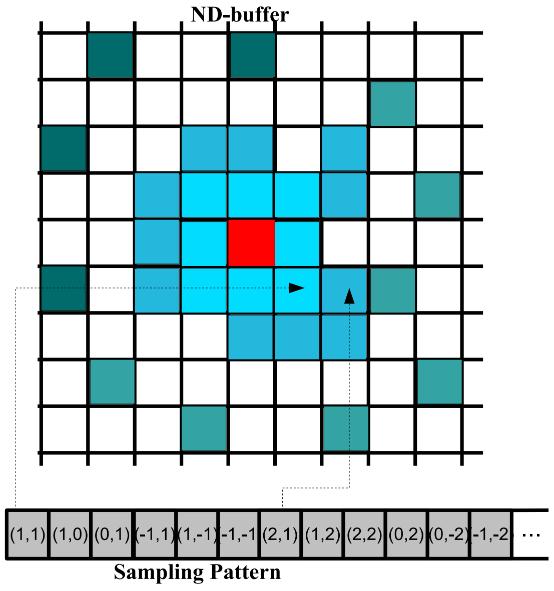


Figure 4: *This figure explains the image-space method where we sample neighboring pixels as occluders. The red pixel is the current pixel under consideration in the fragment shader. The surrounding pixels that are sampled from the ND-buffer are shown. The texture below this grid called the “sampling pattern” provides the locations for these sample pixels. The number of sample pixels, m, that need to be obtained is calculated using the Z-value of the current pixel (shown in red).*

Some of the salient points with regards to existing ambient occlusion solutions such as [Kontkanen and Laine 2005], [Bunnell 2005] are:

- No precomputation is necessary; fitting this algorithm for animation of complex models is simple and fast.
- Easily plugged into existing approaches, with the only necessity being the use of deferred shading.
- The constraints are reduced to a single factor, that being the memory bandwidth.
- Comparing with a baked-AO approach, our method offers real-time detailed ambient occlusion that may be difficult to

compute using a baked-AO approach due to differing resolutions in the baked-AO textures and the underlying geometry.

- The performance of this approach does not depend on the polygon count to a large extent; instead, it is directly related to the number of pixels shaded in the *ND-buffers*. This is a significant advantage over existing approaches.

4.3 Limitations

The following are some of the limitations of this approach:

- The obvious limitation is that the image-space ignores the effect of distant yet significant occluders. We provide a different method to take these distant occluders into account (see Section 5)
- The memory bandwidth poses a problem with older hardware, as the quality of the approach directly depends on the bandwidth. One can try to reduce this by using fewer samples per pixel albeit at the cost of less ambient occlusion effect.
- Artifacts may appear on surfaces where the occluders are nearly parallel to the direction of view since the sampling pattern may miss such planes. This can be compensated by distributing more samples along the direction of such an occluder, at increased branching costs.
- For the algorithm to be visually pleasing we need high-polygon count geometry. This may also be complimented by the current trend of modern GPUs that scale well to high-detail geometry.

The main limitation in this approach is our inability to account for large and distant occluders. We can use the methods described in [Kontkanen and Laine 2005], [Ren et al. 2006], [Kontkanen and Aila 2006] or [Bunnell 2005] to overcome this limitation. However we try to solve this problem using a different and a faster approach.

5 Distant-occluder Approach

Surface geometry can be approximated using a variety of approaches for the purpose of performing ambient occlusion. For example [Bunnell 2005] use a disc-based approach; [Ren et al. 2006] and [Hegeman et al. 2006] use a sphere-based approach; [Kontkanen and Laine 2005] use a cube map-based approach.

We propose a new solution using the modern GPU architecture that will perform fast ambient occlusion by approximating distant occluders as spheres. Our solution depends on the fact that for a given primitive, the amount of occlusion it receives due to an occluder decreases gradually with the distance from that occluder. More precisely, the effect of ambient occlusion at a particular point is inversely proportional to the distance from the primitive. Thus beyond a particular distance, the effect of ambient occlusion due to this primitive is a very small ε that is too small for visual perception (by Weber's law). We picked spherical occluders for the following reasons:

- Very little precomputation - usually a combination of Lloyd Clustering and an optimizing step ([Wang et al. 2006]).
- We need to store only 4 degrees of freedom ($<C,r>$) per sphere, where C is the center $<C_x,C_y,C_z>$ of the sphere and r is its radius.
- The general analytical, geometrical properties of a sphere including its symmetry present a number of advantages over other methods.

The following are the cases that might lead to a noticeable artifact due to ε. First, a number of occluders might be situated such that a particular point may be receiving ε occlusion from all these occluders. Fortunately, due to the choice of ε, we require a significantly large number of occluders in such a setup so as to make this effect noticeable. In many situations, ambient occlusion is performed in a closed area such as in a room or inside a box. Theoretically, ambient occlusion performed using the traditional approach should produce full occlusion for every point in such a scenario. However, this can be avoided by constraining the effect of occlusion by one object onto another, usually beyond a particular distance. Hence, using this logic we can assume that the artifact introduced by this case can be ignored to a great extent to be visually pleasing. The other case is that a point receiving ε occlusion from an occluder might be situated so as to be occluded significantly by other occluders. In this case, the ε occlusion is visually unperceivable because the quantity $(a+\varepsilon)$ is visually the same as (a), due to our choice of ε. By this argument, we can see that this method becomes plausible.

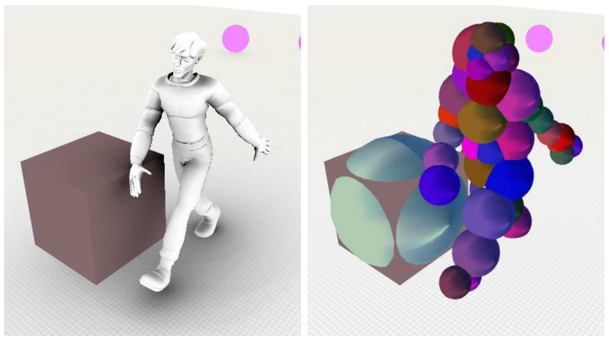


Figure 5: *The left image shows the distant occluder stage in isolation. Shadows and the image-space stage have been turned off to illustrate the nature of the distant occluder approach. The image on the right shows the spheres for the model that are being recalculated per frame. The rest pose provides us the initial position of these spheres due to Lloyd clustering, and we use this to reposition the spheres based on the new vertex positions. This recomputation is simple and fast.*

For a given spherical occluder $S_{occl}(<C,r>)$, we obtain a distance d_{far} from C where the occlusion is ε. This results in a sphere $S_{far}(<C,d_{far}>)$ that represents the influence of occlusion due to the spherical occluder. Beyond the volume bounded by the sphere $S_{far}(<C,d_{far}>)$, the occlusion is low enough to be ignored. We call this sphere the *sphere of influence*. Due to the symmetric nature of this *sphere of influence*, we can approximate the screen-space area of this sphere using a billboard. By confining our ambient occlusion calculation within this billboard in screen-space, we again transform this larger problem in world-space to a simplified version in screen-space.

The main bottleneck in [Ren et al. 2006] and [Bunnell 2005], two state-of-the-art algorithms for ambient occlusion approximation algorithms, is the use of traversal of occluders in either the vertex processor or the pixel processor. Unfortunately, these operations are not easily parallelizable; [Bunnell 2005] makes use of a vertex-shader to traverse a linked-list type of data structure to find the cumulative occlusion. [Ren et al. 2006] makes use of a pixel-shader to essentially do the same operation. We propose a new solution where we well-utilize the rasterization hardware of GPUs as well as the vertex/pixel shading processes.

We use the screen-space rectangle (billboard) covering this *sphere of influence* for the rasterizer to produce the fragments inside this billboard to be tested against. The pixel-shader then takes the given point in the *ND-buffer* and calculates the ambient occlusion due to the incoming spherical occluder (as represented by the billboard). This approach is highly parallelizable, and uses the vertex/pixel processors for performing operations they were designed for. Also, our approach can be directly applied to any other approximation such as a disc or a cube, after making minor changes to the calculation of a *sphere of influence* for the corresponding approximation.

The motivation for such an approach is that it is difficult to perform gathering in the GPU for a large number of occluders, where we "gather" the effect of occlusion from a number of occluders. We convert this gather problem to a "scatter" problem that disperses the ambient-occlusion effect on to surfaces.

5.1 Implementation Details

For an object, we obtain the set of spherical occluders approximating this object. Each element in this set consists of four values: three for the center of the sphere and one for the radius. For the vertex-processor, we construct an unit quad along with the sphere's $(< C, d_{far} >)$ as texture-coordinates (i.e. the four components of `texcoord0`). The vertex-shader then finds the corresponding quad (billboard) necessary for fitting the given *sphere of influence*.

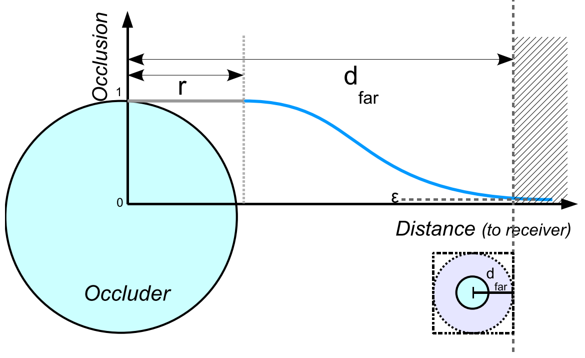


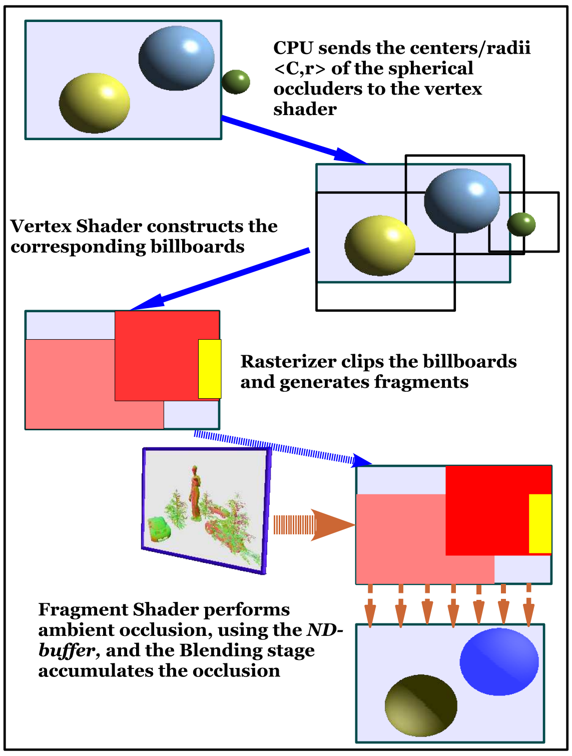
Figure 6: *This 2D visualization of occlusion explains the fall-off parameter ε as the receiver's distance (x-axis) from a spherical occluder varies. The receiver receives maximum occlusion from that occluder, by making its normal point toward the occluding sphere's center.*

Once computed, the rasterizer generates the fragments inside this quad. We make use of the non-programmable yet fast nature of the rasterizer such as its ability to perform view frustum clipping and generate of a large number of fragments using very few and fixed number of vertices.

The fragment shader takes the *ND-buffer* generated for a deferred shading approach as described in Section 4 to perform local ambient occlusion. We also make use of the blending units to additively combine the ambient occlusion contributions across a number of spherical-occluders.

5.2 Issues

We discuss the main issues with this approach and some solutions for those issues.



CPU sends the centers/radii <C,r> of the spherical occluders to the vertex shader
Vertex Shader constructs the corresponding billboards
Rasterizer clips the billboards and generates fragments
Fragment Shader performs ambient occlusion, using the *ND-buffer*, and the Blending stage accumulates the occlusion

Figure 7: Here the spherical occluders that are used to approximate the geometry are shown. The billboards representing the extent of the far fall-off term from each of these spheres are sent to the vertex processor for rasterization. The inset illustrates the sphere with its sphere of influence along with the billboard corresponding to the distance d_{far} from the center of the sphere.

We use additive ambient-occlusion resulting in possible multiple occluder artifacts, as discussed in [Bunnell 2005] and [Kontkanen and Laine 2005]. An occluder fully invisible at a point due to another nearer yet larger occluder can cause incorrect occlusion at that point. [Kontkanen and Laine 2005] and [Bunnell 2005] both suggest different approaches to handle this *over-occlusion*; both provide a visually pleasing yet physically incorrect solution. This works in most cases. In reality, the benefit of performing ambient occlusion even with over-occlusion is far greater than when not performing ambient occlusion at all. We discuss our solution to over-occlusion in Section 8.2.

The performance of this approach depends mainly on the size of the billboards we generate in the vertex-shader; thus a large number of large occluders can cause a performance decrease. In those cases, it is more efficient to process such large occluders in a separate pass. Comparing with the existing methods [Bunnell 2005; Kontkanen and Laine 2005; Ren et al. 2006] our method is advantegous, because we use the rasterizer to limit the influence of the approximated occluders for a given point to a large extent.

Our approach offers an efficient GPU-based ambient occlusion solution for approximated-occluders at the cost of a simple precomputation step. Moreover we target the fastest possible operations for the various units of a GPU for which they were designed for. This approach when combined with the image-space approach detailed above provides some very realistic occlusion effects that can be rivaled by ray-traced solutions, while surpassing existing methods in both quality as well as speed (Figure 3).

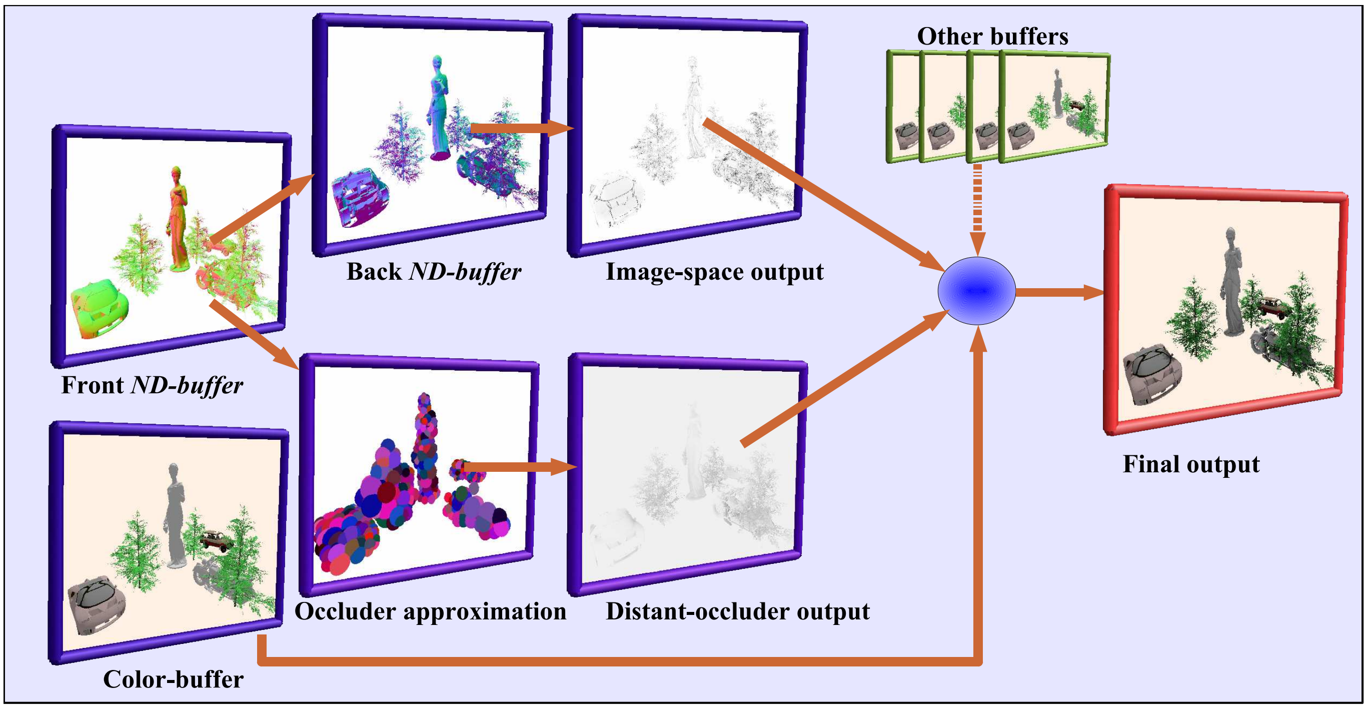


Figure 8: *Pipeline: This flow-diagram explains the various stages in our combined ambient occlusion algorithm. We start with the ND-buffer for the front-faces, along with the color-buffer. The color-buffer is required for deferred shading. The back-faces' ND-buffer is then created and supplied to the image-space approach. Similarly, we prepare the spheres' billboards for the distant-occluder approach. Finally, we combine these stages including shadow-buffers, color-buffers, and other buffers.*

6 Combined Ambient Occlusion

The image-space approach provides a non-precomputed and fast way to enhance high-surface detail geometry, while performing local ambient occlusion approximation. The distant-occluder approach provides softer occlusion effects due to larger and more distant occluders. Combining these two stages provides some very realistic ambient occlusion effects. With the addition of soft-shadows, the ambient occlusion effect provides further quality as opposed to just generating soft-shadows ([Ren et al. 2006]) (Figure 1 c). In the case of the soft-shadowed regions, ambient occlusion adds more visual realism and provides visual cues in brighter, non-shadowed regions. [Ren et al. 2006] provides a way to generate low-frequency soft shadows that tend to be closer to ambient occlusion. However, we can see that the soft-shadow approach compliments ambient occlusion and is not an approximation of ambient occlusion by itself. Soft-shadows can be generated using many algorithms, including the more recent and efficient approach ([Guennebaud et al. 2006]).

6.1 Overall Multi-Pass Algorithm

Both of these ambient occlusion approaches share a pass, namely the generation of the first *ND-buffer* for the front-facing polygons. Beyond this pass, the two algorithms can continue completely independently of each other resulting in a scalable, efficient and parallelizable combined algorithm. For the final deferred shading stage, we combine the ambient occlusion from these two stages using the two distinct textures and accumulate it using the color-texture. This final stage also combines various other elements such as HDR effects, shadow-maps, post-processing effects (depth blur, transition effects, etc.). Moreover the generation of the color-buffer is another stage that is demanded by deferred shading but is independent of the ambient occlusion stages.

7 Results

We obtain a framerate of about 6-15fps for a 150,000-triangle scene on a nVidia GeForce 7800 GT for our complete ambient occlusion solution. The image-space approach uses a maximum of 36 samples per pixel. There is also a 60 ms overhead for deferred shading, due to the current difficulties in FBO context-switches, texture-copies etc. For the image-space approach alone with a 40% screen coverage at 1024x768, we obtain frame-rates of about 15-25 fps. For the distant-occluder approach, we obtain consistent frame-rates of over 100 fps with a worst-case of 10 fps (if there are a number of occluders covering a single point). The main advantage with our approach is that the distant-occluder approach is made to perform at a very fast rate - capable of handling 2500 or more spheres per frame at about 70 fps - the combination provides a visually pleasant image at a fast rate.

We expect future hardware to provide better memory bandwidth, which is currently the major bottleneck. One issue that was mentioned previously is the fact that we do not account for all the occluders in the scene at a given point; this means that rooms and other closed areas that should appear dark (black) when ambient occlusion is performed, do not appear so. This provides better yet physically incorrect ambient occlusion approximation, which is also imitated by many ray-tracing systems limiting the length of the occluder rays.

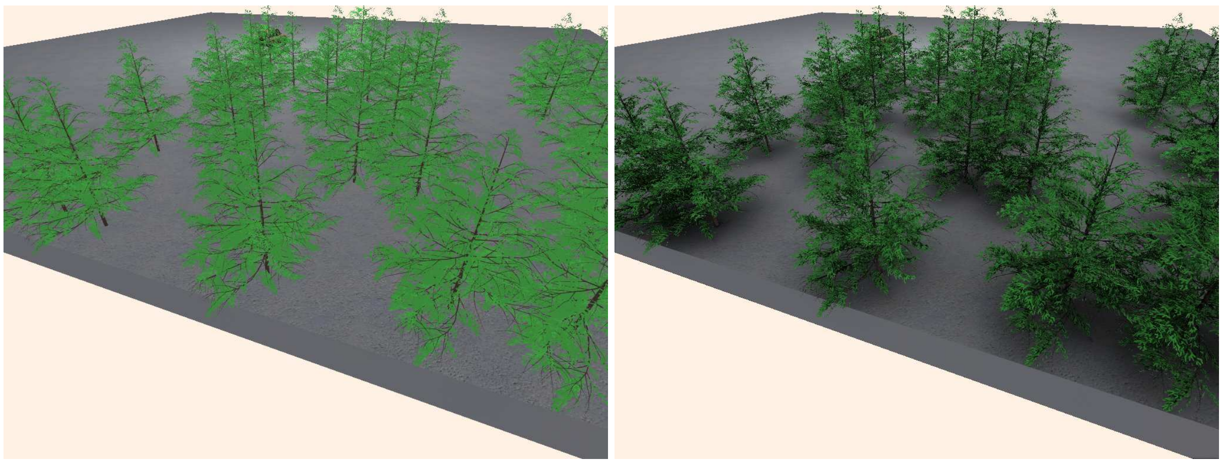


Figure 9: *(a) shows the effect without any ambient occlusion calculation. (b) shows the effect when ambient occlusion using our method is applied to this scene. This performs at around 8 fps when using the combined ambient occlusion effect and at around 25-50 fps using the distant-occluder stage alone.*

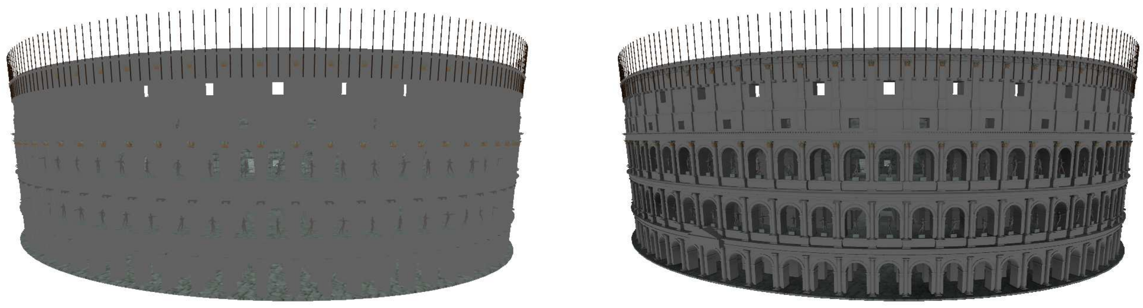


Figure 10: *(a) and (b) show the effect of without/with ambient occlusion using our approach. This model has over a million triangles and renders at around 4 fps on current hardware for our combined approach.*

8 Future Work

We can extend our distant-occluder based approach in a number of ways. For example, we can choose an ambient occlusion fall-off curve to suit our needs so that the occlusion at d_{far} is zero, to avoid artifacts previously discussed. We can also choose a different occluder depending on our needs. For our image-space approach, we can extend it to perform shading only in those regions that were covered by the quads that were generated by the distant-occluder based approach. We can also perform some simple rejection tests to improve speed: for example, if the lighting is too bright or too dark at a point, then image-space ambient occlusion can be avoided in those regions.

We can also rid the separate stage for the back-faces' *ND-buffer* by using a 2WxH texture instead of a WxH texture. Here, we send each vertex to the corresponding region – either $[(0,0)-(W,H)]$ (when it's facing towards the camera) or $[(W,0)-(2W,H)]$ (when it's facing away from the camera direction) based on the normal, thus saving one pass. We have also tried exploiting temporal coherency for transporting ambient occlusion values from previous frames based on a probabilistic approach; however, due to the extreme branching costs associated with this method, we have not found a current increase in speed. For the image-space approach, mip-maps can also be used to aggregate occluders to save unnecessary texture accesses.

Due to the scalability of these two approaches, it is evident that future GPUs provide more processing power and memory bandwidth to improve both the quality and speed of these approaches:

8.1 Multi-GPU enhancement

The scalable nature of these two approaches allows for easily extending them to a multi-GPU environment such as an SLI (Scalable Link Interface) configuration. Such a configuration would involve dedicating most of the memory bandwidth in one GPU to the image-space approach while performing the other stages in a different GPU. This can be performed using Alternate Frame Rendering that divides the GPUs among the image-space approach and the rest. The advantage of the image-space approach is that, it needs just read-access to about 4 textures outputting the ambient occlusion on a separate texture. This means that we can avoid unnecessary bandwidth costs such as inter-GPU communication and copy-on-write broadcasts.

8.2 Direct3D 10

Direct3D 10 architecture provides the notion of generating vertices on-the-fly in the new Geometry Shader. This is very useful in the case of the distant-occluder approach, where we can batch many quads using a single draw-call per object. Moreover, we can also output a varying number of quads which means that we can reject suitable quads or combine quads into unified quads. Also, the CPU-GPU bandwidth is kept to a minimum since we can use point-sprite expansion to generate these quads.

Another extension is to solve the over-occlusion problem by using bit-vectors that is possible due to integer operations in these hardware. We then create a correspondence between each bit in this bit-vector (which is a 32-bit integer, or an 128-bit multiple render target) to an area on the surface ofthe hemisphere. This allows for us to use the logical OR-blending instead of additive blending to store these bit patterns at the cost of a decrease in the ambient occlusion precision.

9 Conclusion

We described a realistic yet an approximate ambient-occlusion solution. Considering that most games require the use of solid non-reflecting objects, ambient occlusion can approximate global illumination to a high degree. Moreover, it provides visual cues to help the user orient herself in the environment - relative distances among various objects become clearer using ambient occlusion (Figure 5).

Although we target real-time applications, we can compare it with other offline rendering solutions in many cases (Figure 3) due to the properties of ambient occlusion. We have shown that our ambient occlusion approximation could vastly improve the quality of a number of real-time applications.

10 Acknowledgments

This research was partially funded by the Undergraduate Research Oppurtunities Program (Summer 2006) and an Undergraduate Research Fellowship (Fall 2006). We would like to thank Gragory S Johnson and the Texas Advanced Computing Center for supporting our research and providing various resources. The molecular dataset was provided by Vinay K Siddavanahalli of Computational Visualization Center. A number of useful animation datasets for use in our distant occluder stage were provided by Kun Zhou of Microsoft Research (Asia). We would also like to thank Sony Computer Entertainment America for their motion capture datasets.

References

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Fast and Detailed Approximate Global Illumination by Irradiance Decomposition. In Proceedings of ACM SIGGRAPH 2005, ACM Press, Volume 24, Issue 3.

BAJAJ, C., DJEU, P., SIDDAVANAHALLI, V., AND THANE, A. 2004. TexMol: Interactive Visual Exploration of Large Flexible Multi-component Molecular Complexes. In Proceedings of IEEE Conference on Visualization, 243–250.

BLYTHE, D. 2006. The Direct3D 10 system. ACM Transactions on Graphics 25, 3, 724–734.

BUNNELL, M. 2005. Dynamic Ambient Occlusion and Indirect Lighting. GPU Gem2, NVidia Corporation, 223–234.

EISEMANN, E., AND DÉCORET, X. 2006. Fast scene voxelization and applications. In ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, ACM SIGGRAPH, 71–78.

EVERITT, C. 2001. Interactive order-independent transparency. White paper, NVidia 2, 6, 7.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2006. Real-time soft shadow mapping by backprojection. In Eurographics Symposium on Rendering, Nicosia, Cyprus, 26/06/06-28/06/06, Eurographics, http://www.eg.org/.

HALL, J., AND HART, J. 2005. GPU Acceleration of Iterative Clustering.

HEGEMAN, K., PREMOZE, S., ASHIKHMIN, M., AND DRETTAKIS, G. 2006. Approximate ambient occlusion for trees. In Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, C. Sequin and M. Olano, Eds., ACM SIGGRAPH.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. ACM Trans. Graph. 22, 3, 879–887.

KONTKANEN, J., AND AILA, T. 2006. Ambient occlusion for animated characters. In Rendering Techniques 2006 (Eurographics Symposium on Rendering), T. A.-M. Wolfgang Heidrich, Ed., Eurographics.

KONTKANEN, J., AND LAINE, S. 2005. Ambient Occlusion Fields. In Proceedings of ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games, ACM Press, 41–48.

REN, Z., WANG, R., SNYDER, J., ZHOU, K., LIU, X., SUN, B., SLOAN, P.-P., BAO, H., PENG, Q., AND GUO, B. 2006. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. ACM Trans. Graph. 25, 3, Volume 25, Issue 3, 977–986.

SHADEVIS. 2006. MeshLab http://meshlab.sourceforge.net.

SLOAN, P., KAUTZ, J., AND SNYDER, J., 2002. Precomputed Radiance Transfer for Real-time rendering in Dynamic, Low-frequency Lighting Environments.

SLOAN, P.-P. 2006. Normal Mapping for Precomputed Radiance Transfer. In Proceedings of ACM 2006 Symposium in Interactive 3D Graphics and Games.

TARINI, M., CIGNONI, P., AND MONTANI, C. 2006. Ambient occlusion and edge cueing to enhance real time molecular visualization. IEEE Transaction on Visualization and Computer Graphics 12, 6 (sep/oct).

WANG, R., ZHOU, K., SNYDER, J., LIU, X., BAO, H., PENG, Q., AND GUO, B. 2006. Variational sphere set approximation for solid objects. Submitted to Pacific Graphics.

WASSENIUS, C. 2005. Accelerated Ambient Occlusion Using Spatial Subdivision Structures.