Real-time Volume-based Ambient Occlusion

Georgios Papaioannou, *Member, IEEE,* Maria Lida Menexi,Charilaos Papadopoulos

Abstract

Real-time rendering can benefit from global illumination methods to make the three-dimensional environments look more convincing and lifelike. On the other hand, the conventional global illumination algorithms for the estimation of the diffuse surface inter-reflection make heavy usage of intra- and inter-object visibility calculations, so they are time consuming, and using them in real-time graphics applications can be prohibitive for complex scenes. Modern illumination approximations, such as ambient occlusion variants, use pre-calculated or frame-dependent data to reduce the problem to a local shading one. This paper presents a fast real-time method for visibility sampling using volumetric data in order to produce accurate inter- and intra-object ambient occlusion. The proposed volume sampling technique disassociates surface representation data from the visibility calculations, and therefore makes the method suitable for both primitive-order or screen-order rendering, such as deferred rendering. The sampling mechanism can be used in any application that performs visibility queries or ray marching.

Index Terms

Shading, shadowing, raytracing, volume visualization, visibility, ambient occlusion, ray marching.

1 Introduction

Irradiance from diffuse inter-reflection on surfaces can significantly increase the credibility of a three-dimensional environment, especially in locations unreachable by direct illumination. Global illumination algorithms, such as radiosity methods and particle tracing, provide a reliable and sufficiently accurate modeling of the physics of diffuse light transport in an effort to capture these effects [1]. For real-time applications with large dynamic environments though, global illumination estimation has to be accommodated in the limited duration of one twentieth of a second or less, which is also shared by other important rendering tasks. Furthermore, GPU-accelerated particle tracing approaches, such as [2], while achieving interactive frame rates for certain scenes, they are not compatible with the rendering pipeline of typical real-time applications. With these restrictions in mind, it is hard to provide efficient and generic implementations of the above genres of algorithms, although promising interactive but approximate results have been presented in the recent years, including – but not limited to – the work in [3], [4] and [5]. To this end, more localized solutions were sought, such as ambient occlusion.

Ambient occlusion is defined as the attenuation of incident light due to the occlusion of nearby geometry [6]. Unlike global illumination methods, which simulate the diffuse inter-reflection of light in a scene using approximate solutions

• *G. Papaioannou, M. L. Menexi and C. Papadopoulos are with the Department of Informatics, Athens University of Economics and Business, Athens, Greece.*
E-mail: gepap@aueb.gr

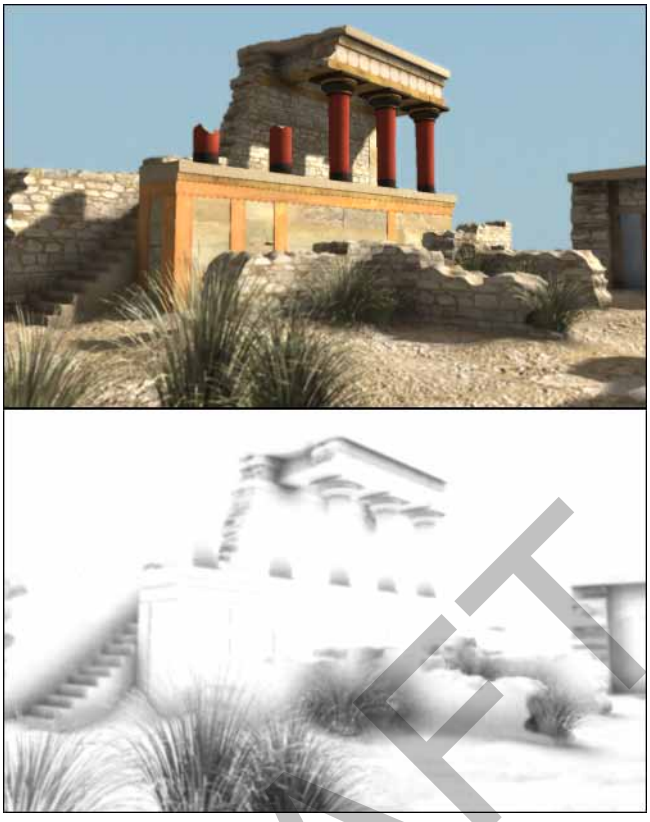

Fig. 1. An example of using our ambient occlusion method.

of the rendering equation, ambient occlusion calculates a local light attenuation factor based on the "openness" of the geometry above an arbitrary surface point:

$$A(\mathbf{p}) = \frac{1}{\pi} \int_{\Omega} \rho(d(\mathbf{p}, \omega)) cos(\theta_i) d\omega \tag{1}$$

where Ω is the hemisphere centered at the normal vector of the receiving point $\mathbf{p}$. $\rho(d(\mathbf{p}, \omega))$ is an obscurance attenuation function of the distance $d(\mathbf{p}, \omega)$ of the closest point to $\mathbf{p}$ in the direction $\omega = (\theta_i, \phi_i)$. For non-real-time lighting calculations, ambient occlusion can be estimated using Monte Carlo ray casting from the shaded point to the entire scene over Ω and up to a maximum distance r_{max}.

This paper presents a real-time method for visibility sampling using volumetric data in order to produce accurate inter- and intra-object ambient occlusion (Fig. 1). The proposed volume sampling technique decouples the surface representation data from the visibility calculations and proposes a sampling mechanism that avoids self-occlusion artifacts. This allows ray marching to operate on arbitrary environments represented as volumes, without the need to discriminate between occluders and receivers. The method does not depend on any screen-space calculations and therefore, it is stable and suitable for both primitive-order or screen-order rendering, such as deferred rendering. The sampling mechanism that is presented can be used in any application that performs visibility queries or ray marching. The volume data can be either pre-calculated for rigidly animated and static objects, or dynamically generated for arbitrarily animated entities and environments, using real-time voxelization.

2 Related Work

A number of solutions have been proposed for the efficient incorporation of ambient occlusion shading in dynamic three-dimensional environments. Real-time methods for the calculation of ambient occlusion either exploit pre-calculated visibility information and perform a fast per-frame sampling on these data using a fragment shader, or exploit view-dependent information stored in specially constructed auxiliary buffers to extract obscurance measurements and therefore, an approximate ambient occlusion value. Various specialized techniques that address the demands of specific types of data such as trees [7] and animated characters [8] have also been developed, but a brief overview of the more generic approaches will be attempted below.

Bunnell [9] uses a surfel representation of polygonal surfaces and constructs a hierarchical scheme for evaluating the ambient occlusion in real-time. His method requires a dense tessellation of surfaces and heavy pre-processing. Hoberock and Jia [10] proposed some technical improvements on Bunnell's method, which increase the result quality without demanding excessive mesh tessellation.

In the Ambient Occlusion Fields method [11], the approximate solid angle and the average occlusion direction are stored as fields around the occluder. At run time, for each shaded point and each occluder, the contributed ambient occlusion is reconstructed from the stored parameters of the ambient occlusion fields and subsequently it is blended with previous passes. The method handles differently ambient occlusion received from the inside than that received from the outside of the occluder's convex hull and is suitable for inter-object shading only.

A generic ray casting acceleration method is proposed by Gaitatzes et al. [12] for inter-object ambient occlusion calculation using stochastic sampling. The authors parameterize the distance from the occluder's surface to its bounding sphere for a discrete set of incident directions and store the result in a displacement field, which is later used to transform ray-object intersection tests into simple indexing operations. Their method produces accurate results at reasonable frame rates for non-interlocking objects, but the memory requirements and pre-processing time are high for environments containing many objects. Furthermore, the method cannot handle ambient occlusion inside object cavities or enclosed environments.

Malmer et al. [13] suggest a simple solution for inter-object ambient occlusion only; the ambient occlusion on a uniform grid containing an occluder is pre-calculated. At run time, for each occluder, the occlusion values are applied as a three-dimensional texture to the receiving surface, which intersects the ambient occlusion volume. The method requires that the ambient occlusion is calculated for every grid location at a pre-processing step. The run-time part of the algorithm is a simple multi-pass volume texturing of the receiving geometry, using the pre-calculated and rigidly transformed ambient occlusion grids of the occluders.

Mittring [14] proposed a depth-buffer-based method, where point samples are chosen from within a sphere centered at the shaded fragment and depth-tested against the depth buffer. The ambient occlusion approximation is calculated using the weighted result of the depth tests. The resulting effect of the above approach is similar to a curvature estimator on a view-oriented height field. Luft et al. [15] use this observation to approximate ambient occlusion with a high pass filter over the depth buffer. Their method is fast but only resembles the intended effect. Shanmugam and Arikan [16] use a view-dependent multi-pass approach, where an ambient occlusion-like post-processing effect is produced based on the pre-calculated depth and normal information (ND-buffer) in the neighborhood of each fragment. Distant objects are treated as simplified occluders and contribute to the inter-object shadowing effect. These

view-dependent approaches are fast, require minimal or no pre-calculation, but cannot model ambient occlusion correctly, because depth discontinuities, such as object edges and buffer boundaries, interfere with the change of slope and distant geometry is not adequately sampled.

The Horizon-Split ambient occlusion method by Dimitrov et al. [17] is a different two-part approach to screen-space occlusion algorithms. First, for each shaded point, the surface elevation horizon is discovered by iteratively sampling the depth buffer in a number of directions around the point up to an r_{max} range. The solid angle subtended by the horizon is considered to contribute to ambient occlusion. To discover occlusion on discontinuous surfaces visible in the buffer, ray marching on the depth buffer is performed, from the horizon and up to the surface normal direction. The method produces convincing and smooth results, which in many cases match those of the standard ambient occlusion algorithm, provided there are no view-dependent discontinuities or hidden surfaces within the r_{max} range (see Fig. 9). The first part of the algorithm is fast and does not require many samples in order to converge to a convincing result, but the ray marching stage introduces noticeable artifacts and therefore requires many rays and post-filtering. The Image-Space Horizon-Based AO algorithm by Bavoil et al. [18] is a screen-space-only version of the Horizon-Split technique, which reduces calculations by not having to transform vectors from image space to eye coordinates and vise versa.

A brief overview of some of the techniques listed above as well as a general description of ambient occlusion and its implementation can also be found in [19].

Finally, a hybrid method for enhancing the appearance of ambient occlusion results with color bleeding and approximate diffuse global illumination that is targeted for highly interactive applications with dynamic environments, was proposed by Ritschel et al. [20]. This is also a screen-space algorithm, which takes advantage of directional information on the depth and normal buffers to bias the sampling of the light sources for the direct illumination and also re-use illumination recorded in a previous frame as a single-bounce global illumination approximation.

3 Method Overview

The method proposed in this paper essentially replaces the ray casting of the stochastic hemisphere sampling of the ambient occlusion calculation in (1) or any other visibility and distance estimator, with a modified ray marching mechanism over a volume representation of the geometry, which is stored as a three-dimensional texture. It is a view-independent and robust approach that provides stable and accurate results, while it can be integrated within any rendering environment, from off-line image generation to screen-space deferred rendering. Furthermore, as will be explained in Section 5.2, our method can be used to calculate real-time visibility and ambient occlusion on scalar fields as well. In contrast to many other ambient occlusion methods presented in Section 2, where the sampling domain and pattern as well as the choice of attenuation function $\rho(x)$ depend on the specific algorithm used, our method can adapt to the requirements and parameterization of each application.

Previous volume-based methods, such as [13], used pre-calculated ambient occlusion data on a volume grid for each occluder and could not account for self occlusion, as conventional volume sampling would always yield high density/occlusion near the shaded surface. To avoid self-intersection on the surface of ray origin, special sampling of the volumes is required. For instance, for pre-calculated ambient occlusion and irradiance gathering on surface models, Kontkanen and Laine [21] proposed a sample rejection algorithm that instead of shooting rays from a regular volume grid, it iteratively generated random ray samples inside the portion of the grid cell that was not occupied

by the model interior. For each grid cell, ray-polygon intersection points from rejected rays driven from within the model interior were maintained and all other ray samples were incrementally tested and validated against them. Unfortunately, for dynamic scenes and occlusion conditions, such a pre-calculation technique is not applicable at run-time. The proposed sampling technique, which is a central part of this work, is explained in detail in Section 5. In particular, three volume sampling variations are presented, one for the most common case of surface data, including voxelized polygonal models, one for transparent surfaces and iso-surface rendering in scalar fields, and one for ambient occlusion in direct volume rendering.

For the calculation of ambient occlusion for rigid body animation, each independently animated object is voxelized – if not already in volume representation – and the respective volume is stored as a 3D texture (see Section 4). Note that, as the method can indiscriminately handle both inter- and intra-object occlusion, there is no need to partition the environment into occluders and receiving surfaces. Therefore, static environments can be represented by a single volume. For rigid body animation or limited deformation of soft bodies and characters, the volume for each object is generated once and reused in every consecutive execution of the method. To apply our method to arbitrarily deformed models, a real-time GPU-based voxelization algorithm is required to either prepare only the deformed geometry or perform a full-scene volume generation at every frame, regardless of the content displayed (see Section 4.2).

In a real-time screen-space implementation of the method (e.g. in deferred rendering), the ambient occlusion for the visible fragments is calculated in a fragment shader, using the normal and depth buffers along with a three-dimensional texture of the object as input. For N_{obj} objects (or just one in the case of real-time full-scene voxelization), the volume of each one of them is used as an input texture to the shader in N_{obj} consecutive passes and the results are multiplicatively blended, as suggested in [11]. As the sampling domain for each object is bounded by the voxelized volume ($\pm r_{max}$ in each direction), each pass does not cover the full screen but rather the area occupied by the object. Fragments mapped outside the volume bounds are rejected and rendering complexity is very unlikely to reach the upper bound of O($N_{obj}wh$) with respect to the screen coverage of each $w \times h$ frame. An outline of our method applied to real-time screen-space polygonal rendering is listed below. The subscripts WCS, CSS and VCS for points and vectors refer to world, canonical screen space and volume coordinate systems respectively.

1) **Preprocessing**. For each individual object use existing volume. If volume does not exist or requires update, voxelize object and store volume (Section 4).
2) **Buffer preparation**. Render geometry into the normal and depth buffers. Render other buffers as required by the application.
3) **Ambient occlusion**. For each fragment $\mathbf{p}_{(CSS)}$:
   a) **Fragment Rejection**. Transform $\mathbf{p}_{(CSS)}$ to volume reference frame: $\mathbf{p}_{(VCS)}$.
   If $\mathbf{p}_{(VCS)}$ is outside the volume range $\pm r_{max(VCS)}$, then reject fragment, else:
   b) **Stochastic sampling**. Initialize fragment ambient occlusion: $A(\mathbf{p}) = 0$. Cast N_{dir} random rays $\mathbf{r}_{j(WCS)}$ from the WCS fragment position $\mathbf{p}_{(WCS)}$ in a hemisphere around the normal $\mathbf{n}_{(WCS)}$. For each ray:
      - Sample volume using ray marching with at most $N_{samples}$ steps (Section 5). Return the intersection distance $d(\mathbf{p}_{(WCS)}, \omega_j)$ in the direction ω_j on the ray $\mathbf{r}_{j(WCS)}$.
      - Calculate the distance attenuation function $\rho(d(\mathbf{p}_{(WCS)}, \omega_j))$.
      - Calculate the directional occlusion contribution $A_j(\mathbf{p}) = \mathbf{n}_{(WCS)} \cdot \mathbf{r}_{j(WCS)} \cdot \rho(d(\mathbf{p}_{(WCS)}, \omega_j))$.

c) **Monte Carlo integration**. Sum the ambient occlusion contribution of each ray $A_j(\mathbf{p})$: $A(\mathbf{p}) = \frac{1}{N_{dir}} \sum_{j=1}^{N_{dir}} A_j(\mathbf{p})$

4) Repeat Step 3 for all object volumes.

4 VOLUME GENERATION

The method requires that a binary or scalar volume representation exists for each model. All polygonal models are voxelized and the resulting volume is used as a three-dimensional texture. For scalar fields, it is not necessary to threshold the volume data in order to obtain a binary representation, as the sampling method can be extended to operate on continuous fields and transparent voxels (see Section 5.2).

4.1 Volume Pre-computation

For the voxelisation of polygonal models, the XOR slicing method by Chen and Fang [22] and the depth-buffer-based approach by Karabassi et al. [23] were first investigated, due to their simplicity and relatively good performance. The first one proved to be very sensitive to intersecting geometry and non-watertight models, while the second one could only work for objects with no cavities. Passalis et al. [24] improved the depth-buffer-based approach by introducing a depth peeling modification, but did not completely remove the problem. The solution that was finally adopted for pre-processed volume generation was the accelerated cumulative three-way polygon rasterization directly in the voxel grid, proposed in [25], since only voxel shells are required and the method works for arbitrary surfaces and nested cavities. The scene is rendered into volume-slice-thin orthogonal frusta aligned with a primary axis. Each rendering pass updates the corresponding slice of an arbitrarily sized volume texture. When the slope of a triangle relative to the projection plane exceeds $\pi/4$, its surface is undersampled, leaving holes in the resulting volume. To rectify this, three volumes are built, one for each primary axis and the results are combined with an OR operation.

4.2 Real-time Voxelization

There are several cases, where the pre-calculation of the volume data does not lead to an acceptable performance or rendering quality. One major concern is that for scenes with many animated objects, the ambient occlusion rendering frame rate for the accumulation of the contributions of individual objects can become non-realtime (see scalability plot of Figure 12). Furthermore, deformable or breakable objects may invalidate their initially estimated binary volume representation (e.g. animated characters or physically animated geometry). Finally, the multiplicative blending required to merge the ambient occlusion results for each individual object, can lead to over-shadowing; when two occluding surfaces (and therefore, the respective voxels) overlap at the same distance from the shaded point, obscurance is overestimated in this direction.

Real-time voxelization can address all the above issues, especially when it is performed on the entire scene and not in a per-object basis. A volume covering the extents of the scene is created and updated at every frame or whenever the environment changes. If the scene boundaries are too large to achieve an acceptable voxelization detail, the extents of the volume can be reduced to a box centered at the view point. The fast, single-pass voxelization approach of Eisemann and Décoret [26], [27] has been used for the real-time volume generation. Their method performs a binary encoding of the depth values of the rasterized polygons into the 32 bits of a frame buffer (a 32-voxel-deep *slicemap*). The desired depth resolution is achieved by stacking the results of additional slicemaps, which, as in the implementation used here, can be rendered in a single pass, using multiple rendering targets. As the method

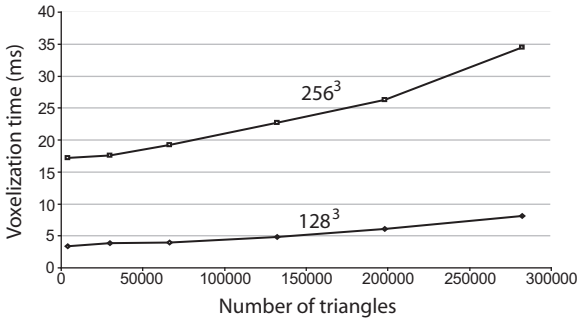

Fig. 2. Three-way real-time voxelization results for two different volume resolutions on an NVIDIA GTX285 card.

creates a view-dependent voxelization of the scene, leaving unoccupied voxels in general, we adapted the three-way voxelization approach described above by replacing each axis-aligned slicing pass with the single-pass slicemap method. Fig. 2 presents timings for the three-way real-time voxelization for various scenes at two different volume resolutions.

5 Volume Sampling

Given a surface position $\mathbf{p}_{(WCS)}$ with corresponding normal vector $\mathbf{n}_{(WCS)}$ and a visibility sampling ray $\mathbf{r}$ in a direction $\omega = (\theta, \phi)$ relative to the normal, we seek the intersection distance $d(\mathbf{p}_{(WCS)}, \omega)$ of the ray with the closest point to $\mathbf{p}_{(WCS)}$. More precisely, as we sample a density function, we seek a location $\mathbf{x} = \mathbf{p}_{(WCS)} + t \cdot \mathbf{r}, t \in (0, r_{max}]$ corresponding to the nearest volume sample with density $V(\mathbf{x})$ greater than a predefined threshold V_{thres}. Typically, in real-time graphics, $V(\mathbf{x})$ is the scalar result in the range $[0, 1]$ of sampling a three-dimensional texture, after transforming $\mathbf{x}$ to the texture space coordinate system and performing tri-linear interpolation on the nearest 8 texels. The sampling of the volume is performed with ray marching, i.e. the density measurement at $N_{samples}$ equidistant consecutive samples along the ray. For voxelized surfaces, as the ray marching begins on the surface fragments, it is certain that the nearest high density voxel contains the fragment and therefore, a straightforward incremental search in the direction ω would result in total self-occlusion. Fig. 3a demonstrates this problem. In order to avoid immediate self intersection, a small offset has been added along each ray for the initial sample. However, adding a constant offset from $\mathbf{p}_{(WCS)}$ for the first sample along a ray only migrates the problem to rays with larger deviation from the normal vector (Fig. 3b,c). To overcome this problem, a more careful sampling is required. Section 5.1 presents the modification of the typical ray marching to handle self-intersections correctly (see Fig. 3d) and Section 5.2 extends the sampling technique to occlusion for transparency and continuous scalar fields.

5.1 Binary Volume Sampling

Assuming cubic voxels with an s_v edge length, we seek the distance d_0 of the first sample x_0 to the surface fragment along the ray $\mathbf{r}$ so that x_0 is outside the voxel containing the fragment (starting voxel) but also outside any other neighboring voxel, potentially generated for the same polygon. To this end, a voxel boundary, locally aligned with the polygonal surface plane, is considered (guard limit in Fig. 4). Points within this guard limit are potentially contained

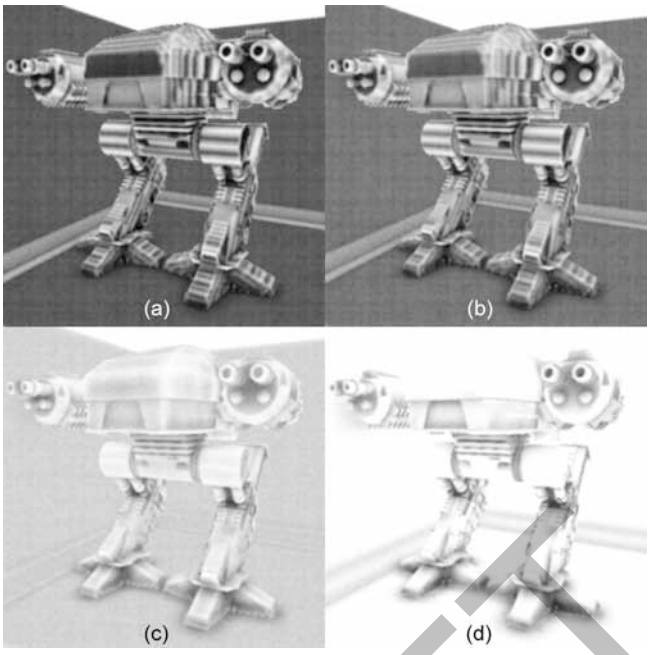

Fig. 3. Ambient occlusion sampling results for a robot model using ray marching on a $135 \times 131 \times 121$ volume representation of the model with: (a) no guard limit, (b) constant guard limit of 1 voxel (diagonal), (c) constant guard limit of 7 voxels, (d) our variable-length guard limit.

within the starting voxel, or the voxels produced for the same polygon, and are therefore excluded from visibility sampling. If h is the distance of the fragment location to the guard limit, d_0 can be calculated as follows:

$$d_0 = h/\cos\theta = h/(\mathbf{r}\cdot\mathbf{n}) \tag{2}$$

For cubic voxels, the worst case is $h = h_{max} = s_v\sqrt{3}$ (voxel diagonal - fragment resides on the lower/innermost corner of the voxel). In practice though, for linearly interpolated 3D textures, there is no need to start the sampling at $d_{0(max)} = s_v\sqrt{3}/(\mathbf{r}\cdot\mathbf{n})$. The starting voxel is always occupied, as guaranteed by the voxelization stage and therefore, density within this voxel may either decrease toward the voxel boundaries or remain equal to one. Furthermore, proper voxelization ensures that voxel centers lie at most $s_v\sqrt{3}/2$ units away from the surface. Consequently, a discriminating value with regard to whether the next voxel is occupied, which is unaffected by the current voxel, can be encountered from $s_v\sqrt{3}/2$ and beyond. Using $h = s_v\sqrt{3}/2$ in Eq. 2, the ray encounters a valid intersection point, if the current ray density sample exceeds a high threshold V_{thres} for a sample point beyond $\mathbf{p}+\mathbf{r}\cdot s_v\frac{\sqrt{3}}{2}/(\mathbf{r}\cdot\mathbf{n})$.

If $d_0 \geq r_{max}$, the particular ray cannot leave the guard limit and is discarded from the ambient occlusion calculation (Fig. 4). The lower V_{thres} is, the lower the probability is to miss an intersection. On the other hand, as will be discussed in more detail in Section 6.5, low V_{thres} may result in early termination of the ray marching and therefore erroneous distance estimation. The modified ray marching can be summarized as follows:

1) Calculate $d_0 = s_v\frac{\sqrt{3}}{2}/(\mathbf{r}\cdot\mathbf{n})$.
2) Set current distance d to d_0

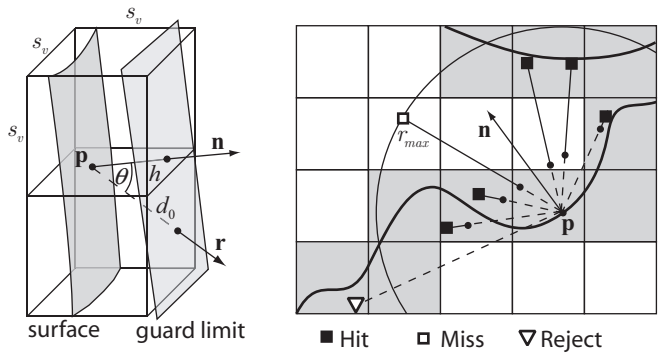

Fig. 4. Volume ray marching for voxelized polygons. Sampling begins outside the guard limit of the voxel occupied by the current fragment to avoid surface patch self-intersection.

3) If $d_0 \geq r_{max}$, discard the ray
4) Set ray increment step dr to $r_{max}/N_{samples}$
5) Calculate current sample position: $\mathbf{x} = \mathbf{p} + \mathbf{r} \cdot d_0$
6) While $d < r_{max}$ and $V(\mathbf{x}) < V_{thres}$
   $\mathbf{x} += dr \cdot \mathbf{r}$, $d += dr$

5.2 Sampling of Scalar Fields and Participating Media

When the data to be visualized are already in volume representation, the sampling method and obscurance calculations need to be adapted, according to the rendering method required.

5.2.1 Isosurface Rendering

For the calculation of directional obscurance $A_j(\mathbf{p})$ on an isosurface of density d_{iso} in a scalar field using ray marching, the range r_{max} is again split into $N_{samples}$ equally-sized intervals. If occlusion calculation accounts only for density values near d_{iso} (occluding isosurfaces), ray marching stops when $|V(\mathbf{x}) - \mathbf{d_{iso}}| < d_{tol}$, d_{tol} being a volume-dependent isosurface tolerance, but otherwise proceeds as in Section 5.1.

5.2.2 Direct Rendering of Volume Sets

An important effect that contributes to the realistic rendering of volumes representing continuous scalar fields, such as tissue of variable density or a distribution of particles in a volume, is the attenuation of incident light combined with the in-scattering of transmitted photons. These effects have been studied and properly modeled for interactive rendering (see for instance the work of Kniss et al. [28]). Volumetric ambient occlusion can approximate the attenuation of incoming light to a point in space from paths converging to it from every direction, in direct analogy to the manner that surface ambient occlusion regards the incident light attenuation on a surface patch. When casting rays in an unsegmented scalar field, or calculating ambient occlusion for ray transmission through participating media, the contribution of density values less than V_{thres} must be also taken into account. Furthermore, the obscurance attenuation function $\rho(d(\mathbf{p}, \omega))$ that is well defined for discrete obstacles along a ray, is not applicable to transmissive

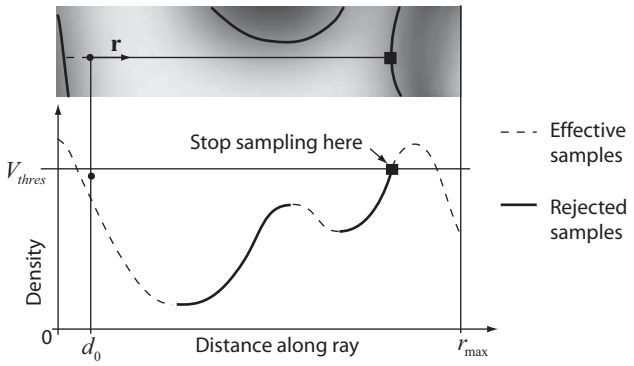

Fig. 5. Scalar field rejection sampling for directional ambient occlusion contribution along a ray **r**.

media. For this reason, we need to replace $\rho(d(\mathbf{p},\omega))$ with a directional attenuation function (obscurance along a ray) $\gamma(\mathbf{p},\omega)$, which does not depend on the distance to a specific intersection point:

$$A(\mathbf{p}) = \frac{1}{\pi}\int_{\Omega}\gamma(\mathbf{p},\omega)cos(\theta_i)d\omega \tag{3}$$

A derivation of an attenuation function, suitable for our calculations is presented below.

If a complete volume scattering process is simulated during rendering, ambient occlusion is treated as out-scattering from multiple illumination directions. However, as the role of ambient occlusion shading in real-time rendering is to replace the more computationally expensive illumination calculations, it is usually applied as a stand-alone local shading technique, without accounting for in-scattering or multiple-scattering. In this case, points within large, dense regions would always yield high ambient occlusion, producing an effect similar to self-occlusion discussed earlier in this section. To counter the excessive darkening of self-occlusion, the following heuristic sample rejection criterion is proposed. Decreasing density values along the ray $\mathbf{r}(\omega)$ indicate that $\mathbf{r}(\omega)$ exits a high density region and the volume should not contribute to the obscurance in direction ω. On the other hand, increasing density corresponds to samples that march toward a high density pocket, so obscurance contributes to the result (Fig. 5). To facilitate calculations, it is useful to represent the rejection result $R(\mathbf{x}_k)$ at sample $\mathbf{x}_k$ as a function:

$$R(\mathbf{x}_k) = \begin{cases} 0, & V(\mathbf{x}_k) < V(\mathbf{x}_{k-1}) \ \text{ (sample } \mathbf{x}_k \text{ is rejected)} \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

When V_{thres} is exceeded at a certain sampling point along $\mathbf{r}(\omega)$, or the obscurance measured is near 1, ray marching is terminated. Apart from handling self-occlusion due to high density values in the vicinity of $\mathbf{p}$, this rejection sampling method can also correctly handle semitransparent surface geometry, provided the voxelization method can produce non-binary density volumes (see example in Figure 6). If a full volume scattering model is used instead of only ambient occlusion shading, $R(\mathbf{x}_k)$ should always equal 1.

5.2.3 Attenuation Function

As the scalar field (or participating medium) is not homogeneous, $\gamma(\mathbf{p},\omega)$ cannot be evaluated directly for the entire $[0, r_{max}]$ range. Let us examine the case when $V(\mathbf{x})$ is directly mapped to a density value and therefore to an

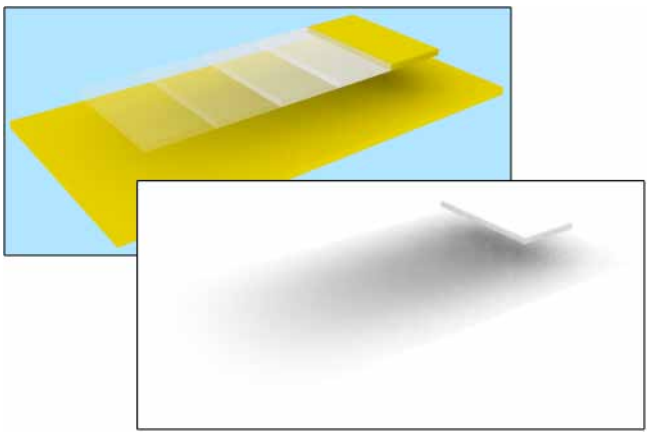

Fig. 6. Ambient occlusion from transparent geometry. The volume of the voxelized objects stores the maximum alpha value of the contributing polygons at each voxel. For clarity of presentation, the transparent geometry is not shown in the ambient occlusion inset.

attenuation coefficient $\sigma(\mathbf{x})$. Considering a piecewise constant density for each one of the total $N_{samples}$ marching steps, the transmittance $\tau(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k)$, i.e. the fraction of photons transmitted from a sample point $\mathbf{x}_{k-1}$ to the next, is $\tau(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k) = e^{-V(\mathbf{x})r_{max}/N_{samples}}$ (Beer's law [29]). Therefore, by taking into account the sample rejection, the total obscurance perceived in an ω sampling direction $\gamma(\mathbf{p}, \omega)$ becomes:

$$\gamma(\mathbf{p},\omega) = 1 - \prod_{k=1}^{N_{samples}} \tau(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k)^{R(\mathbf{x}_k)} = 1 - \prod_{k=1}^{N_{samples}} e^{-R(\mathbf{x}_k)(V(\mathbf{x}_k)r_{max})/N_{samples}} \tag{5}$$

On the other hand, when the volume data $V(\mathbf{x})$ represent opacity (alpha), an estimate $\hat{\sigma}(\mathbf{x})$ of the attenuation coefficient $\sigma(\mathbf{x})$, has to be indirectly extracted from $V(\mathbf{x})$. 1-$V(\mathbf{x})$ can be regarded as the transmittance τ through one voxel spanning v_s units. Considering a constant density for the v_s interval: $1 - V(\mathbf{x}) = e^{-\hat{\sigma}(\mathbf{x})v_s}$ and therefore, $\hat{\sigma}(\mathbf{x})$ at a sample point $\mathbf{x}$ is: $\hat{\sigma}(\mathbf{x}) = \frac{-ln(1-V(\mathbf{x}))}{v_s}$.

Let us now consider a (small) arbitrary number of samples $\mathbf{x}_k$ along the ray, leading to sampling intervals greater than v_s, in general. Assuming that density varies slowly with respect to the distance along the ray, the most efficient sampling strategy is to use a stratified sampling pattern with constant jittering for all samples $\mathbf{x}_k$ along each ray [30]. To avoid undersampling steep density transitions, a mip-map of lower resolution of the 3D texture can be used, according to the ratio of the stratum length to v_s. For $N_{samples}$ points, the length of each stratum is $r_{max}/N_{samples}$ and the resulting transmittance is:

$$\tau(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k) = e^{-\hat{\sigma}(\mathbf{x}_k)r_{max}/N_{samples}} = \left(1 - V(\mathbf{x}_k)\right)^{\frac{r_{max}}{v_s N_{samples}}} \tag{6}$$

Using the transmittance of Equation 6 in the attenuation function of Equation 5, we finally obtain an expression

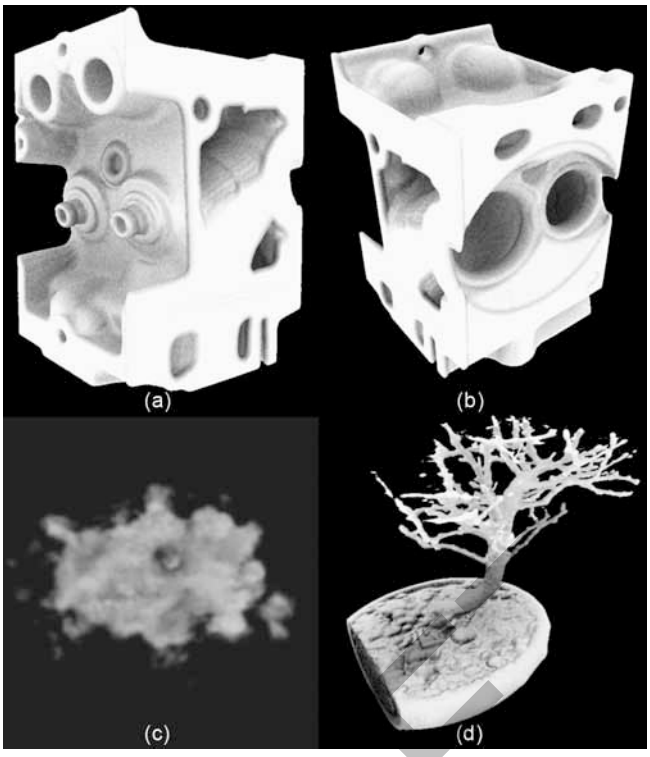

Fig. 7. Ambient occlusion ray marching estimation in volume rendering. (a),(b) The engine block data set - 28 fps. (c) Perlin noise smoke density cloud ($200 \times 128 \times 200$ voxels) - 14 fps. (d) The Bonsai tree volume data set - 31 fps.

for the obscurance along a ray cast through a dense three-dimensional texture:

$$\gamma(\mathbf{p},\omega) = 1 - \left[\prod_{k=1}^{N_{samples}} \Big(1 - V(\mathbf{x}_k)\Big)^{R(\mathbf{x}_k)} \right]^{\frac{r_{max}}{v_S N_{samples}}} \tag{7}$$

Equation 7 is GPU-friendly as it replaces the rejection sampling branching operation in the ray marching loop with a low-cost exponentiation and performs only one expensive exponentiation operation per ray.

Results of the rejection sampling technique for the estimation of ambient occlusion on a volume isosurface are presented in Fig. 7a, b and d. In these examples, the GPU-based volume ray casting method of Stegmaier et al. [31] has been used along with our method for shading the volume samples.

5.2.4 Positional Ambient Occlusion

In contrast to the ambient occlusion formulas of Equation 1 or 3, which are used for surface ambient occlusion, a slightly different formulation should be used in participating media, where a particle in space is shaded instead of an infinitesimal oriented surface patch. We define *positional ambient occlusion* $A_P(\mathbf{x})$ as a location-based feature, in order to decouple the obscurance calculations from a directional dependence. Considering a spherical particle with an infinitesimal diameter centered at an arbitrary location in space $\mathbf{x}$, positional ambient occlusion $A_P(\mathbf{x})$ corresponds to the fraction of incident light on it that is blocked by the surrounding geometry:

$$A_P(\mathbf{x}) = \frac{1}{4\pi} \int_S \gamma(\mathbf{p}, \omega) d\xi \tag{8}$$

where $\gamma(\mathbf{p}, \omega)$ is the directional attenuation function (obscurance) and S is the unit sphere around $\mathbf{x}$.

Fig. 7c demonstrates the positional ambient occlusion shading of a cloud of smoke. In this example, Equation 7 is used for the estimation of directional obscurance for 20 rays per volume sample along the view direction through each fragment.

6 Implementation and Results

To test our algorithm in realistic conditions and be able to compare it with the most accurate screen-space ambient occlusion algorithm presented so far (see results in Fig. 8), we implemented an extended version of the Horizon-Split ambient occlusion algorithm (HSAO) [17]. Our implementation incorporates the quality enhancements of the Image-Space Horizon-Based Ambient Occlusion (ISHBAO) [18]. We have introduced the distance attenuation of occlusion samples, which was also proposed in the ISHBAO method and extended its clamping mechanism to account for abrupt depth changes using sample rejection for distant depth samples (likely to belong to a different view depth layer).

A fully-featured common test platform was developed that could support both the modified HSAO method and our own. We built a deferred rendering engine, where all primitive-intensive stages (albedo with transparent textures, normal, depth and specular coefficient buffers) were merged into one multiple render target pass. Ambient occlusion is rendered in a separate buffer using the normal and depth buffers along with the volume textures. The ambient occlusion buffer can be of different resolution than the final frame buffer, although in most test cases presented here, the relative scale is 1.

Before combining the occlusion buffer with the deferred direct illumination, both the HSAO and our stochastic ray marching methods require a post-processing pass, which performs selective smoothing and up-scaling, the latter in the case where ambient occlusion is rendered at a lower resolution than the frame buffer. Smoothing and up-scaling is performed with a joint bilateral filter [32], which is implemented as a 17-sample Gaussian disk kernel with a domain of support guided by the magnitude of the depth buffer gradient at each fragment.

6.1 Quality and Performance

Fig. 8 shows some of the test results of our method. The frame rate for the chosen viewpoint is presented for both our method and HSAO. A general observation was that for the same order of texture sampling operations for both methods (maximum 400 per pixel in the case of Fig. 8), HSAO was typically slower, but produced smoother results. However, the resulting ambient occlusion from the screen-space method exhibited view-dependent artifacts and errors, which in some cases were severe (see Fig. 9, right column). Volume-based ambient occlusion results are inherently stable, compatible with the results of off-line renderers, and view-independent. Furthermore, the method can handle arbitrarily large r_{max} values with no ghosting or false positive values. Increasing the number of rays significantly improves the visual quality, as expected in any Monte Carlo sampling algorithm.

In the test case of Fig. 8e, the effect of voxelization (quantization) error, i.e. the offset of the voxel center from the actual voxelized surface, is demonstrated. A band of pixels, which is slightly brighter than expected, is present at

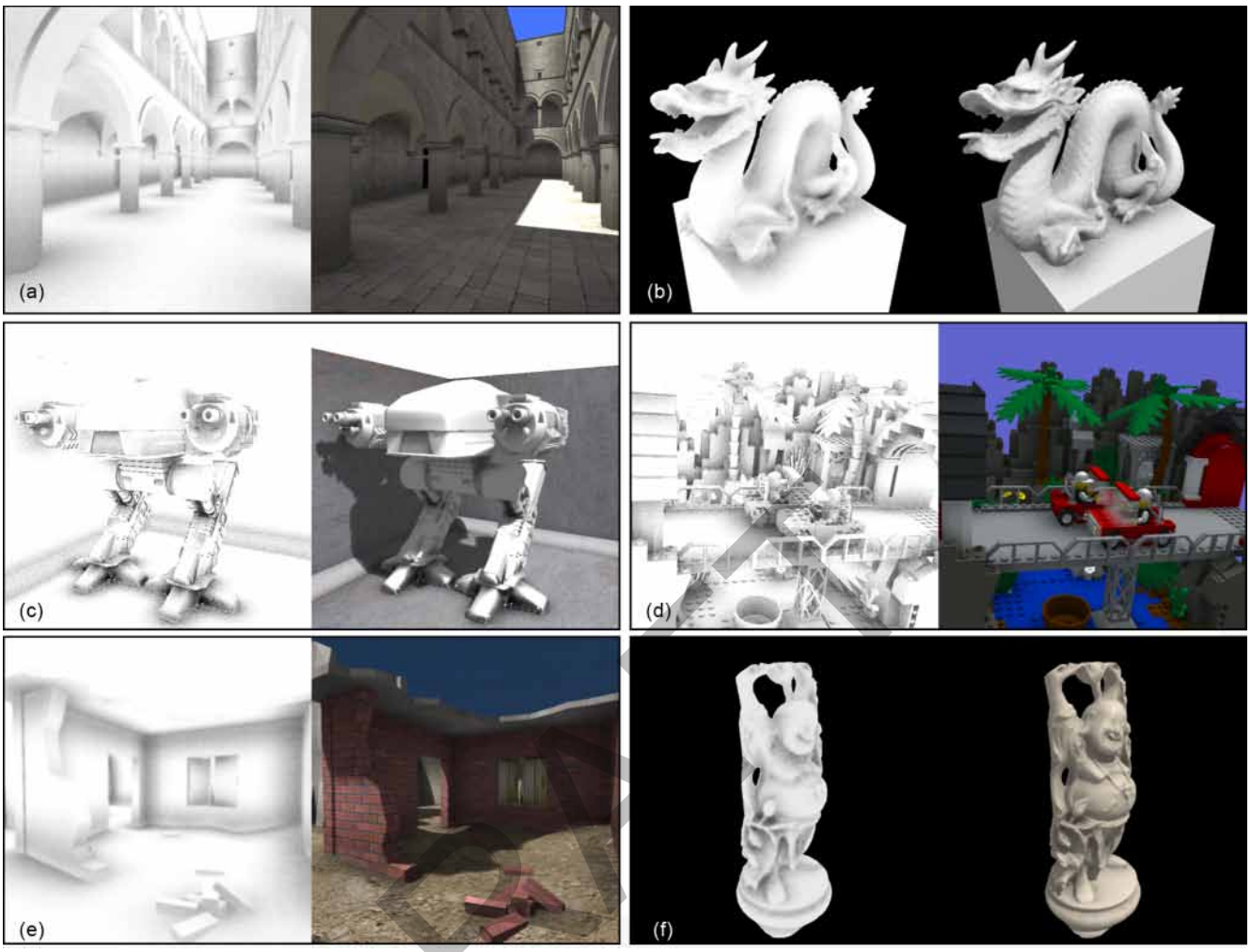

Fig. 8. Ambient occlusion and final image screen shots for various test cases. Image and ambient occlusion buffer resolution is 600×600 pixels and the maximum volume resolution is 160^3 voxels. Frame rate is provided in parentheses for our method and HSAO respectively for the same maximum number of samples. All frame rates were measured on an NVIDIA GTX285 graphics card with 1GB of on-board memory. (a) Sponza Atrium (89/27). (b) Dragon model (67/45). (c) Robot model in room (45/32). (d) Custom virtual LEGO® model built with MLCad (48/24). (e) Custom low-resolution game level (109/37). (f) The Happy Buddha model (231/65).

the base of the walls, due to d_0 skipping almost an entire voxel at the intersection of the ground and the building for many rays. If two surfaces occupy the same voxel, some rays emanating from one surface will inevitably miss the other entirely and will hit the next occupied voxel within the r_{max} range instead (Fig. 10). This effect can be especially noticeable, if V_{thres} is too high (see bottom right inset of Fig. 14) or the voxelization resolution is poor. However, this error is seldom noticeable in most cases and can be reduced either by introducing a small offset to the volume texture indexing, or by lowering V_{thres}.

Ambient occlusion for surface data may also be under-evaluated at the intersection of planar geometry. For points

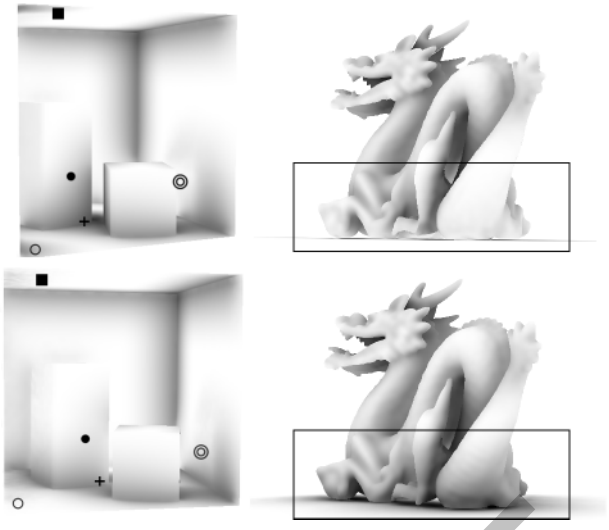

Fig. 9. Problems of pure screen-space algorithms. View-dependent depth changes can cause severe intensity variations, inaccurate occlusion and popping effects. The screenshots are taken from the NVIDIA image-space Horison-Split A.O. demo, using full-resolution A.O. buffer, 16 directions and 16 samples per direction.

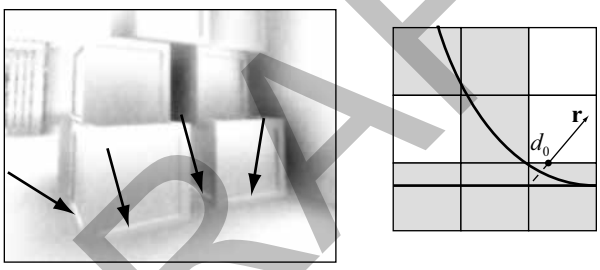

Fig. 10. Error due to inadequate resolution.

on one surface, d_0 can skip many voxels in directions $\mathbf{r}$ near the base of the sampling hemisphere, missing the voxels occupied by the intersecting surface. However, as the contribution of the directional obscurance to the ambient occlusion is weighted by the cosine term $\mathbf{n} \cdot \mathbf{r}$, the impact on the calculated occlusion is not significant.

6.2 Scalability

As in many screen-space fragment processing algorithms, the performance is linearly dependent on the screen coverage of the non-background portion of the scene. This is illustrated in Fig. 11, where an indoor scene is rendered at various screen resolutions up to 2.3 Mpixels (1920×1200 pixels). The ambient occlusion buffer is prepared at half the dimensions of the frame buffer and up-scaled to the proper image resolution. According to the draw times measured and presented in the graph, our method is linear with regard to the number of fragments for all three volume resolutions tested. The dependence of the draw time on the volume (3D texture) size is discussed in Section 6.4.

Considering now a fixed screen coverage, frame rendering overhead is proportional to the number of separate objects present on screen, as demonstrated in Fig. 12. In this example, 10 crates are consecutively added to the scene

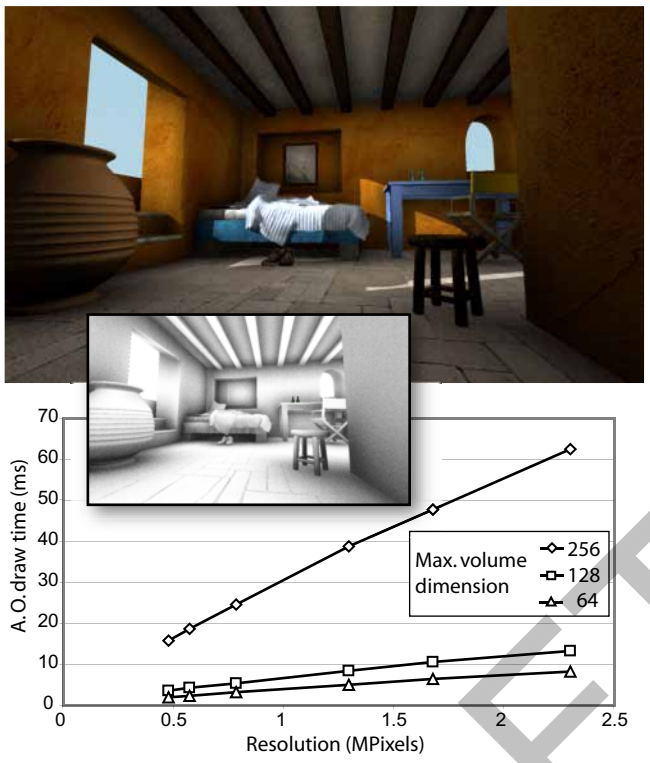

Fig. 11. Draw time versus number of pixels. An interior of a room was rendered with normal mapping, shadow maps, focal blurring and ambient occlusion in various resolutions from 800×600 to 1920×1200. The results in the image (above) were produced using a volume of $128 \times 104 \times 107$ voxels that enclosed the whole scene.

and the respective frame-rate is recorded. The same measurements are taken for a scene represented as a single object. As expected, in the second case the frame rate is almost constant (slight fluctuations occur as the growth of the stack affects the local behavior of the algorithm).

6.3 Sampling

For a given ambient occlusion radius r_{max}, and voxel size v_s, the minimum number of samples along each ray that ensures the closest approximation to the reconstructed surface can be estimated and used [33]. For real-time rendering though, it is imperative to keep the maximum number of iterations along each ray small, regardless of the sampling distance and volume dimensions, so that a minimum acceptable frame rate is maintained. Therefore, a constant sample density was preferred in all implemented tests (8-12 samples per ray, depending on the number of rays). The examples of Fig. 7 and Fig. 8 were rendered with 8 samples per ray. Due to the stochastic sampling of the input domain with variable starting distance d_0 (Equation 2), the effect on the reconstructed ambient occlusion is an increase in the high frequency noise, which is band-limited by the post-filtering, though.

To improve ray marching performance on (binary) volume data from surfaces, empty-space leaping could be employed [34], at the cost of additional texture memory storage and volume post processing time (mip-map generation). For scalar volume data, i.e. for transparency support and participating media, the volume processing cost is

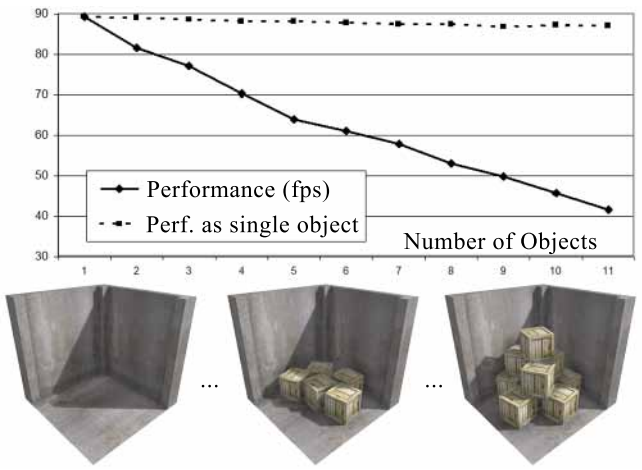

Fig. 12. Scalability study. Plot shows frame rate against number of separate objects on screen. The dashed graph shows the framerate when the same geometry is rendered as a single object.

further increased, as the hierarchical volume occupancy cannot be represented by the averaging pre-filter operation of automatic mip-mapping. Furthermore, empty-space leaping is not effective in crowded scenes with small contiguous regions of void space, as texture look-ups will outnumber those of the conventional ray-marching.

6.4 Volume Resolution

Poor volume resolution inevitably leads to loss of quantization precision during voxelization. Also recall from Section 5.1 that the distance d_0, at which the first ray sample is drawn, increases with v_s. These two factors result in the controllable yet potentially undesirable loss of high frequency occlusion detail, as in the example of Fig. 13. On the other hand, our tests have shown that a degradation in performance is expected when large volumes are used (Fig. 11) due to texture transfer time and less efficient caching. For a constant ambient occlusion radius r_{max}, sampling a smaller volume leads to more texture fetches from coherent memory locations and therefore, better cache utilization. Typical objects and small scenes, as those presented in the examples throughout the paper, can be adequately represented by volumes of 128 to 256 voxels in each dimension for the purpose of ambient occlusion calculation, which correspond to textures sizes of at most 2MB and 8MB respectively for 8 bits per pixel. When using the real-time voxelization method described in Section 4.2, the bit-encoded binary volume representation (slicemap stack) reduces the memory storage requirements by a factor of 8.

6.5 The Impact of Volume Threshold

In binary volume sampling for surface ambient occlusion, V_{thres} affects the sensitivity of the method, as demonstrated in Fig. 14. Due to the fact that three-dimensional texture sampling is performed via trilinear interpolation, off-center volume samples are generally in the range [0,1], depending on the values at the center of neighboring voxels. Therefore, V_{thres} may be exceeded early as the ray samples approach the voxel center $\mathbf{c}$. This fact has two consequences. First, for small values of V_{thres}, ray marching can frequently terminate even when the ray is not directed toward the center of the voxel. Second, for values below 0.2 and a coarse volume resolution, ray marching

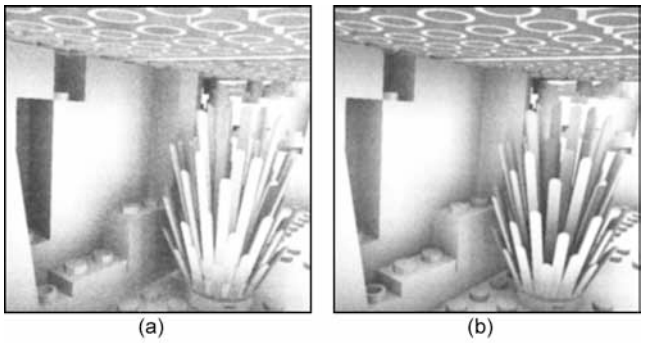

Fig. 13. Impact of volume resolution on high frequency ambient occlusion. (a) 128×86×106 voxels. (b) 256×172×212 voxels.

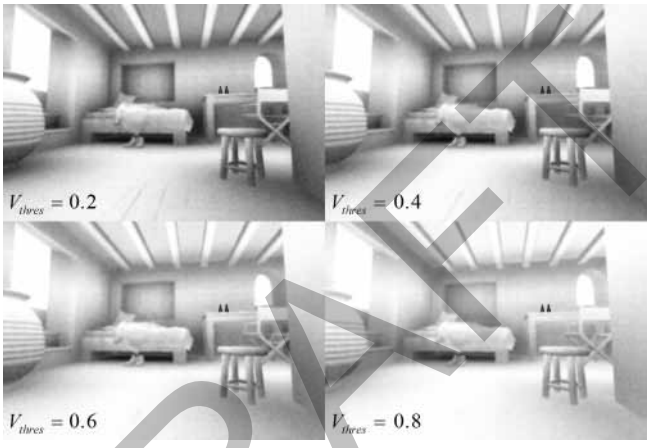

Fig. 14. Impact of different V_{thres} values on ambient occlusion sensitivity.

usually stops early inside a voxel, causing excessive shadowing, if ambient occlusion calculation depends on the intersection distance. In general, smaller V_{thres} values result in more rays exceeding the threshold, i.e. hitting an occluding voxel (over-sensitivity). On the other hand, for large values of V_{thres} (> 0.8), rays failing to cross a voxel near its center can entirely miss the occluding volume. Therefore, best practical results are achieved with V_{thres} in the range [0.2,0.8].

7 Conclusion

A volume sampling method for the calculation of surface visibility and distance queries and its application to the ambient occlusion estimation was presented in this paper. The proposed technique produces an accurate, stable, view-invariant result and can be applied to both screen-space rendering algorithms and individual primitive rendering or distance queries. The method overcomes self occlusion restrictions, so it does not require a discrimination between occluders and occluded surfaces.

The stochastic volume sampling method presented, can be used in a variety of diverse rendering environments, but also for other purposes. As an ambient occlusion method, apart from the rendering of polygonal scenes, it can

be easily integrated within any volume rendering system for the shading of any kind of volume data. Using a high number of samples, the method results are close to stochastic ray casting on polygons, while maintaining highly interactive rates, thus making the method ideal for preview or final image generation in off-line renderers. With high volume resolutions, it can be used as a fast alternative to ray casting for visibility calculations, including ambient occlusion or soft shadows.

Acknowledgements and Disclaimers

The work presented in this paper is funded by the Athens University of Economics and Business Research Grant (EP-1600-10/00-1). LEGO® is a registered trademark of LEGO Group. MLCad is developed by Michael Lachmann (http://www.lm-software.com/mlcad/). The Sponza Atrium model was created by Marko Dabrovic. The Happy Buddha and original Dragon models were downloaded from the online scanned object repository of the Stanford Computer Graphics Laboratory. The robot model was acquired from http://www.3dxtras.com and the engine block and bonsai data sets were obtained from http://www.volvis.org.

References

[1] P. Dutre, K. Bala, and P. Bekaert, *Advanced Global Illumination*. AK Peters, 2006.

[2] R. Wang, R. Wang, K. Zhou, M. Pan, and H. Bao, "An efficient gpu-based approach for interactive global illumination," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–8, 2009.

[3] M. Nijasure, S. N. Pattanaik, and V. Goel, "Real-time global illumination on gpus," *journal of graphics, gpu, and game tools*, vol. 10, no. 2, pp. 55–71, 2005.

[4] C. Dachsbacher and M. Stamminger, "Reflective shadow maps," in *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2005, pp. 203–231.

[5] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz, "Imperfect shadow maps for efficient computation of indirect illumination," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–8, 2008.

[6] S. Zhukov, A. Iones, and G. Kronin, "An ambient light illumination model," in *Rendering Techniques - Proc. Eurographics Rendering Workshop*, Eurographics. Springer-Wien, 1998, pp. 45–55.

[7] K. Hegeman, S. Premože, M. Ashikhmin, and G. Drettakis, "Approximate ambient occlusion for trees," in *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2006, pp. 87–92.

[8] A. G. Kirk and O. Arikan, "Real-time ambient occlusion for dynamic character skins," in *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2007, pp. 47–52.

[9] M. Bunnel, "Dynamic ambient occlusion and indirect lighting," in *GPU Gems 2*, M. Pharr, Ed. Addison-Wesley, 2005, pp. 223–233.

[10] J. Hoberock and Y. Jia, "High-quality ambient occlusion," in *GPU Gems 3*, H. Nguyen, Ed. Addison-Wesley, 2007, pp. 257–274.

[11] J. Kontkanen and S. Laine, "Ambient occlusion fields," in *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2005, pp. 41–48.

[12] A. Gaitatzes, Y. Chrysanthou, and G. Papaioannou, "Presampled visibility for ambient occlusion," in *WSCG '08: Proceedings of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2008, pp. 17–24.

[13] M. Malmer, F. Malmer, U. Assarsson, and N. Holzschuch, "Fast precomputed ambient occlusion for proximity shadows," *Journal of Graphics Tools*, vol. 12, no. 2, pp. 59–71, 2007.

[14] M. Mittring, "Finding next gen: Cryengine 2," in *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*. New York, NY, USA: ACM, 2007, pp. 97–121.

[15] T. Luft, C. Colditz, and O. Deussen, "Image enhancement by unsharp masking the depth buffer," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*. New York, NY, USA: ACM, 2006, pp. 1206–1213.

[16] P. Shanmugam and O. Arikan, "Hardware accelerated ambient occlusion techniques on gpus," in *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2007, pp. 73–80.

[17] R. Dimitrov, L. Bavoil, and M. Sainz, "Horizon-split ambient occlusion," in *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2008, pp. 1–1.

[18] L. Bavoil and M. Sainz, "Image-space horizon-based ambient occlusion," in *ShaderX7 - Advanced Rendering Techniques*. Delmar, 2009.

[19] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-Time Rendering, Third Edition*. A. K. Peters, 2008.

[20] T. Ritschel, T. Grosch, and H.-P. Seidel, "Approximating dynamic global illumination in image space," in *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2009, pp. 75–82.

[21] J. Kontkanen and S. Laine, "Sampling precomputed volumetric lighting," *journal of graphics, gpu, and game tools*, vol. 11, no. 3, pp. 1–16, 2006.

[22] H. Chen and S. Fang, "Fast voxelization of three-dimensional synthetic objects," *J. Graph. Tools*, vol. 3, no. 4, pp. 33–45, 1998.

[23] E.-A. Karabassi, G. Papaioannou, and T. Theoharis, "A fast depth-buffer-based voxelization algorithm," *J. Graph. Tools*, vol. 4, no. 4, pp. 5–10, 1999.

[24] G. Passalis, I. A. Kakadiaris, and T. Theoharis, "Efficient hardware voxelization," in *CGI '04: Proceedings of the Computer Graphics International*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 374–377.

[25] Z. Dong, W. Chen, H. Bao, H. Zhang, and Q. Peng, "Real-time voxelization for complex polygonal models," in *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 43–50.

[26] E. Eisemann and X. Décoret, "Single-pass gpu solid voxelization for real-time applications," in *GI '08: Proceedings of graphics interface 2008*. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2008, pp. 73–80.

[27] ——, "Fast scene voxelization and applications," in *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, 2006, pp. 71–78.

[28] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson, "A model for volume lighting and modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 150–162, 2003.

[29] M. Pharr and G. Humphreys, *Physically Based Rendering, From Theory to Implementation*. Morgan Kaufmann, 2004.

[30] M. Pauly, T. Kollig, and A. Keller, "Metropolis light transport for participating media," in *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*. Springer Wien, 2000, pp. 11–22.

[31] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl, "A simple and flexible volume rendering framework for graphics-hardware-based raycasting," in *Volume Graphics, 2005. Fourth International Workshop on*, June 2005, pp. 187–241.

[32] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Trans. Graph.*, vol. 26, no. 3, p. 96, 2007.

[33] T. Theußl, T. Möller, and M. E. Gröller, "Optimal regular volume sampling," in *VIS '01: Proceedings of the conference on Visualization '01*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 91–98.

[34] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf, *Real-Time Volume Graphics*. A K Peters, 2006.

Georgios Papaioannou received a BSc in Computer Science in 1996 and a PhD in Computer Graphics in 2001, both from the University of Athens, Greece. From 2002 till 2007 he has been a virtual reality systems engineer and developer at the Foundation of the Hellenic World. Dr. Papaioannou is currently a lecturer at the Department of Informatics of the Athens University of Economics and Business and his research is focused on real-time computer graphics algorithms, photorealistic rendering, virtual reality and three-dimensional shape analysis. He is a member of IEEE, ACM and SIGGRAPH.

Maria Lida Menexi received a BSc and an MSc in Computer Science in 2006 and 2008 respectively, both from the Department of Informatics of the Athens University of Economics and Business. She is currently occupied as a developer and a researcher and her research interests include global illumination and real-time graphics algorithms.

Charilaos Papadopoulos received a BSc in Informatics from the Athens University of Economics and Business and is currently a PhD student in Computer Science at the State University of New York at Stony Brook. His research and teaching interests focus around real-time photo-realistic rendering and visualization.