Horizon-Based Indirect Lighting (HBIL)

The ideal companion for your far-field indirect lighting solution.

March 2018 – Benoît “Patapom” Mayaux.
benoit.mayaux@gmail.com

In the following document, vectors will be written in boldface characters (e.g. $\boldsymbol{x}$, $\boldsymbol{\omega_i}$), world-space vectors will be noted as a hat vector (e.g. $\hat{\boldsymbol{\omega}}_{\boldsymbol{o}}, \hat{\boldsymbol{x}}$) and scalar values will use regular characters (e.g. a, π, θ).
The spherical coordinates used throughout the paper have an elevation angle θ aligned with the vertical **Z** axis, and an azimuthal angle φ lying in the tangent plane measured from axis **X**:

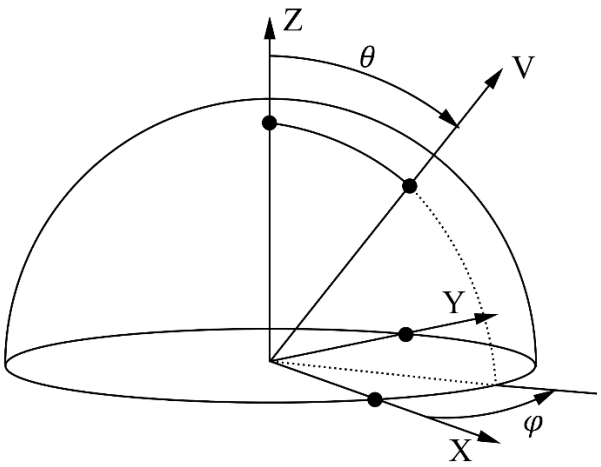

FIGURE 1. *THE FRAME FOR SPHERICAL COORDINATES USED IN THIS PAPER.*

1. Introduction

In this paper we are presenting a new real-time, screen-space technique that can be easily integrated into existing rendering pipelines and that drastically improves the quality of the lighting by adding an important near-field illumination term. It exploits a reprojection of the radiance from one frame to another to provide a theoretically infinite amount of light bounces and at the same time keeps a tight frame budget since it basically relies on the same foundation as the Horizon-Based Ambient Occlusion (HBAO) technique introduced by Bavoil et al. [1] except it uses the information gathered while computing the horizon to its maximum potential.

HBAO already improves upon the traditional Ambient Occlusion integral by skipping the computation of rays that wouldn't contribute to the visibility term because they are below the geometric horizon and would therefore intersect the nearby geometry:

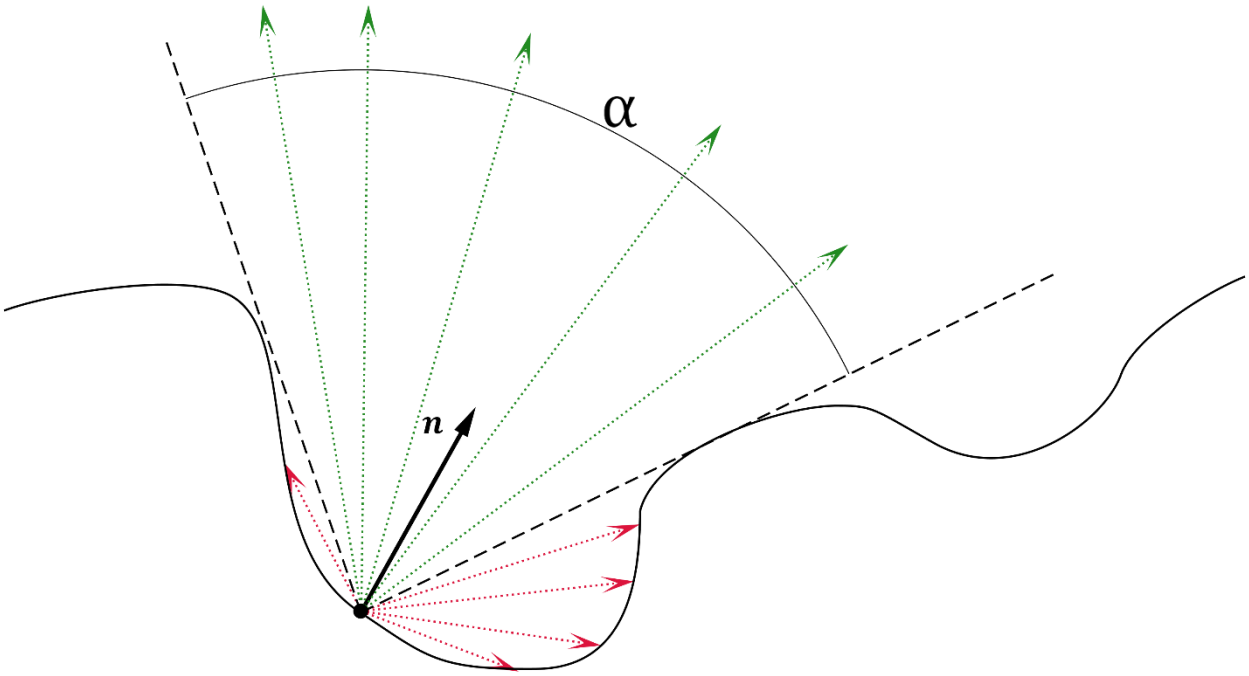

FIGURE 2. *THE HBAO ALGORITHM IS EXPLOITING THE FULL VISIBILITY OF THE HORIZON CONE OF ANGLE α: WE KNOW ALL THE GREEN RAYS INSIDE THE CONE ESCAPE THE SURFACE OF THE HEIGHTFIELD AND ARE THE ONLY ONES ACTUALLY CONTRIBUTING TO THE VISIBILITY TERM. THE RED RAYS, BELOW THE GEOMETRIC HORIZON HIT THE HEIGHT FIELD AND BRING NO ACTUAL CONTRIBUTION TO VISIBILITY.*

The ambient occlusion term is normally written:

$$AO(\boldsymbol{x}) = \frac{1}{2\pi}\int_{\Omega^+} \mathrm{V}(\boldsymbol{x}, \boldsymbol{\omega_i})\, \mathrm{d}\omega_i \qquad (\mathbf{1})$$

With:

- $\boldsymbol{x}$ the location of the pixel for which we are calculating the AO
- $\boldsymbol{\omega_i}$ the incoming ray direction
- Ω^+ the upper hemisphere of directions whose solid angle is 2π
- $\mathrm{d}\omega_i$ the portion of solid angle covered by the $\boldsymbol{\omega_i}$ vector
- $\mathrm{V}(\boldsymbol{x}, \boldsymbol{\omega_i})$ the visibility term returning 1 if the ray escapes to infinity, 0 if the ray intersects the heightfield

Thanks to HBAO, it gets simplified into:

$$AO(\boldsymbol{x}) = \frac{1}{2\pi}\int_0^{2\pi}\int_0^{\theta_h(\varphi)} \sin(\theta)\,\mathrm{d}\theta\,\mathrm{d}\varphi$$

Where $\theta_h(\varphi)$ is the horizon angle for each azimuthal direction φ.

This technique allows us to avoid tracing rays for the entire hemisphere, instead we simply need to determine the horizon angle $\theta_h(\varphi)$ for a particular azimuthal angle φ by sampling the height field in screen space:

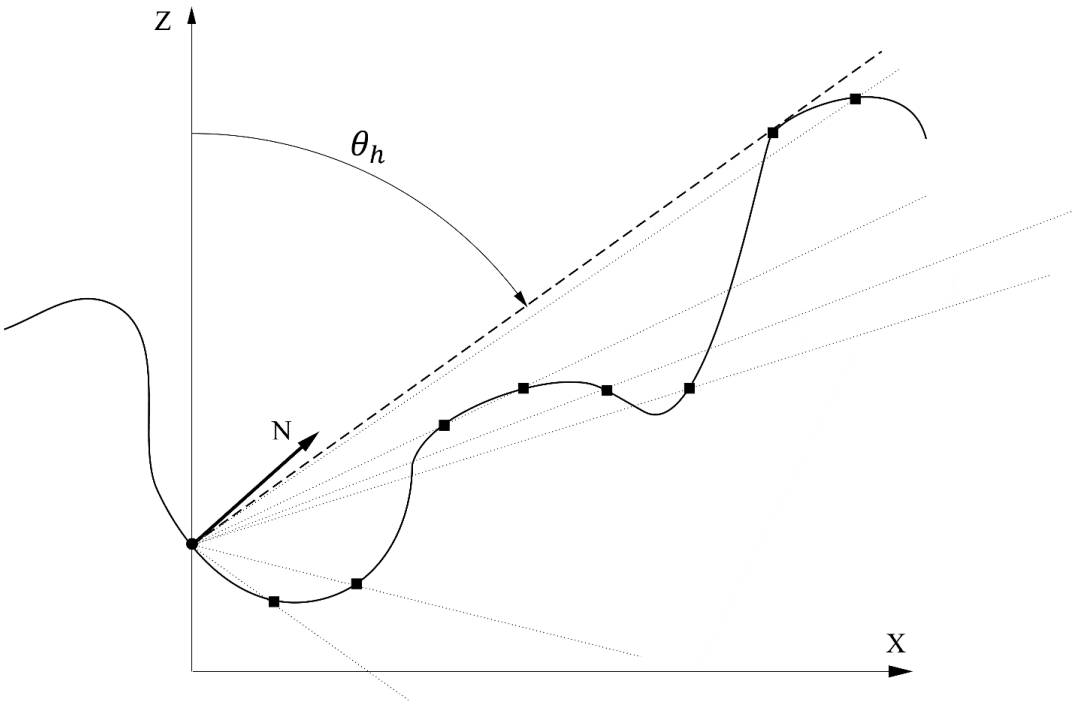

FIGURE 3. *THE HEIGHTFIELD IS SAMPLED ALONG THE X DIRECTION AND THE HORIZON IS UPDATED ALONG THE WAY BY REDUCING THE HORIZON ANGLE θ_h EACH TIME.*

The portion of AO computed by the inner integral is then:

$$\int_0^{\theta_h(\varphi)} \sin(\theta)\,\mathrm{d}\theta = 1 - \cos\big(\theta_h(\varphi)\big)$$

And so, the final expression for the AO becomes:

$$AO(\boldsymbol{x}) = 1 - \frac{1}{2\pi}\int_0^{2\pi} \cos\big(\theta_h(\varphi)\big)\,\mathrm{d}\varphi$$

At runtime, the integral is discretized into a series of S azimuthal "slices" for which we must find the horizon angle:

$$AO(\boldsymbol{x}) \approx 1 - \frac{1}{S}\sum^{S} \cos\big(\theta_h(\varphi)\big)$$

1.1. Camera Spaces

When computing HBAO, we are actually dealing with two different camera spaces:

- The global camera space given by our standard view transform where $\widehat{\boldsymbol{X}}, \widehat{\boldsymbol{Y}}, \widehat{\boldsymbol{Z}}$ are the world-space camera "right", "up" and "at" vectors respectively.

- The local camera space that is reconstructed from the view vector and the camera's up vector.

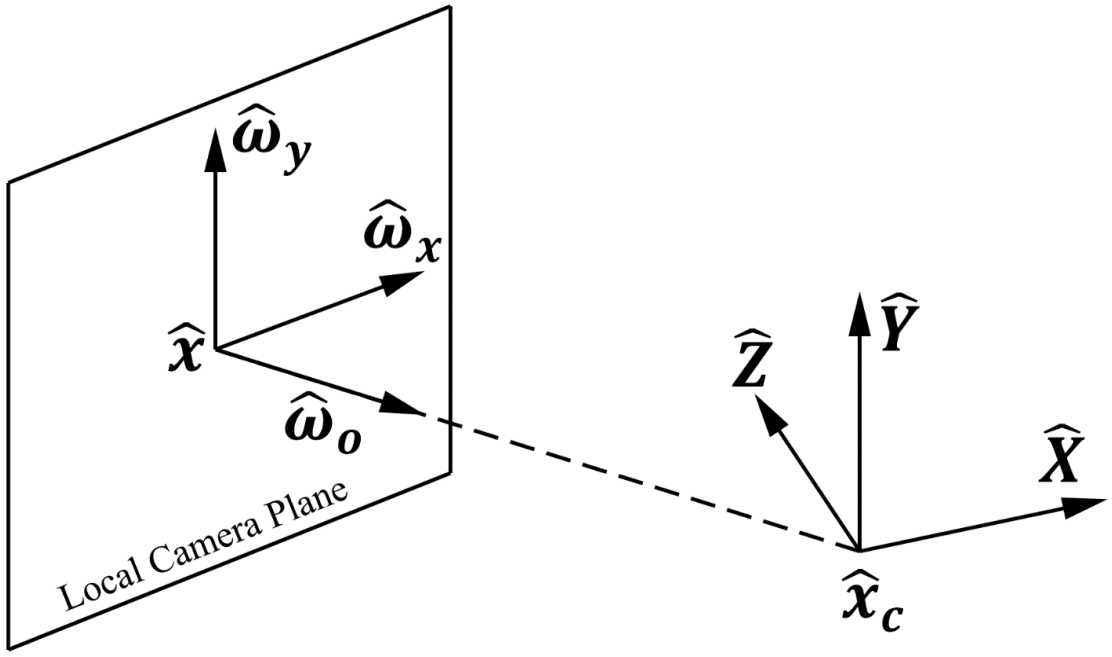

FIGURE 4. THE LOCAL AND GLOBAL CAMERA SPACES. IN ORDER TO AVOID CAMERA-SPACE NORMAL VECTORS WITH NEGATIVE Z COMPONENTS, A LOCAL CAMERA SPACE CENTERED ON THE SAMPLING LOCATION $\widehat{\boldsymbol{x}}$ IS RECONSTRUCTED FROM THE LOCAL VIEW VECTOR $\widehat{\boldsymbol{\omega}}_{\boldsymbol{o}}$.

In order to avoid getting a camera-space normal with a negative z component, we must construct a "local" camera space specifically for our sampling location $\widehat{\boldsymbol{x}}$:

$$\hat{\omega}_x = \frac{\hat{Y} \times \hat{\omega}_o}{\|\hat{Y} \times \hat{\omega}_o\|}$$
$$\hat{\omega}_y = \hat{\omega}_o \times \hat{\omega}_x$$

Where:

- $\hat{\omega}_o = \frac{\hat{x}_c - \hat{x}}{\|\hat{x}_c - \hat{x}\|}$ is the normalized local view vector pointing from our world-space location $\hat{x}$ toward the world-space camera position $\hat{x}_c$
- $\hat{\omega}_x, \hat{\omega}_y$ are the reconstructed local "right" and "up" vectors respectively

1.2. Slice Space

The local camera space being setup, this will serve as our working reference frame where we can start subdividing the hemisphere of the HBAO algorithm into multiple slices, as shown in the figure below:

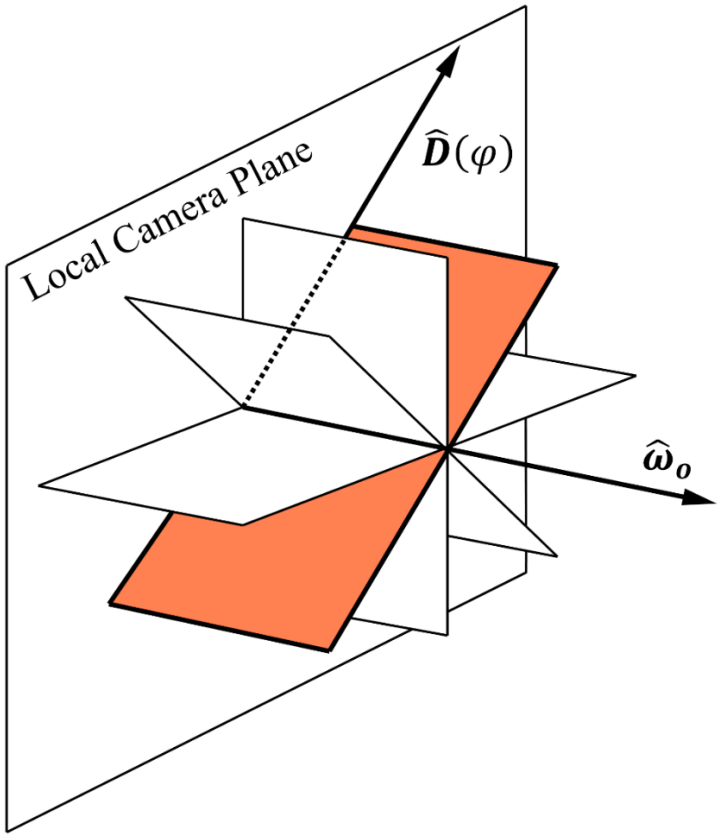

FIGURE 5. THE HEMISPHERE OF DIRECTIONS FROM EQUATION (1) IS SUBDIVIDED INTO MULTIPLE SLICES. EACH SLICE'S TANGENT DIRECTION IS GIVEN BY THE WORLD VECTOR $\hat{D}(\varphi)$.

We will subdivide the hemisphere Ω^+ into multiple azimuthal slices whose tangent directions are given by:

$$\widehat{\boldsymbol{D}}(\varphi) = \cos(\varphi)\,\widehat{\boldsymbol{\omega}}_{\boldsymbol{x}} + \sin(\varphi)\,\widehat{\boldsymbol{\omega}}_{\boldsymbol{y}}$$

Where φ is the azimuthal angle for the slice.
In local camera space, the slice's direction and outgoing view vector are simply given by:

$$\boldsymbol{D}(\varphi) = \begin{cases} \cos(\varphi) \\ \sin(\varphi) \\ 0 \end{cases}$$

$$\boldsymbol{\omega}_{\boldsymbol{o}} = \begin{cases} 0 \\ 0 \\ 1 \end{cases}$$

We will further need an additional space called the "slice space" which is the 2D plane of a specific slice that is defined by the canonical directions:

$$\boldsymbol{D}' = \begin{cases} 1 \\ 0 \end{cases}$$

$$\boldsymbol{\omega}'_{\boldsymbol{o}} = \begin{cases} 0 \\ 1 \end{cases}$$

In slice space, the central location of our sampling point is simply:

$$\boldsymbol{x}' = \begin{cases} 0 \\ 0 \end{cases}$$

And the projection of the world-space normal vector $\widehat{\boldsymbol{n}}$ into the slice, shown in figure 6, is given by:

$$\boldsymbol{n}' = \begin{cases} \widehat{\boldsymbol{n}} \cdot \widehat{\boldsymbol{D}}(\varphi) \\ \widehat{\boldsymbol{n}} \cdot \widehat{\boldsymbol{\omega}}_{\boldsymbol{o}} \end{cases} = \begin{cases} n_x \\ n_y \end{cases} \qquad (\mathbf{2})$$

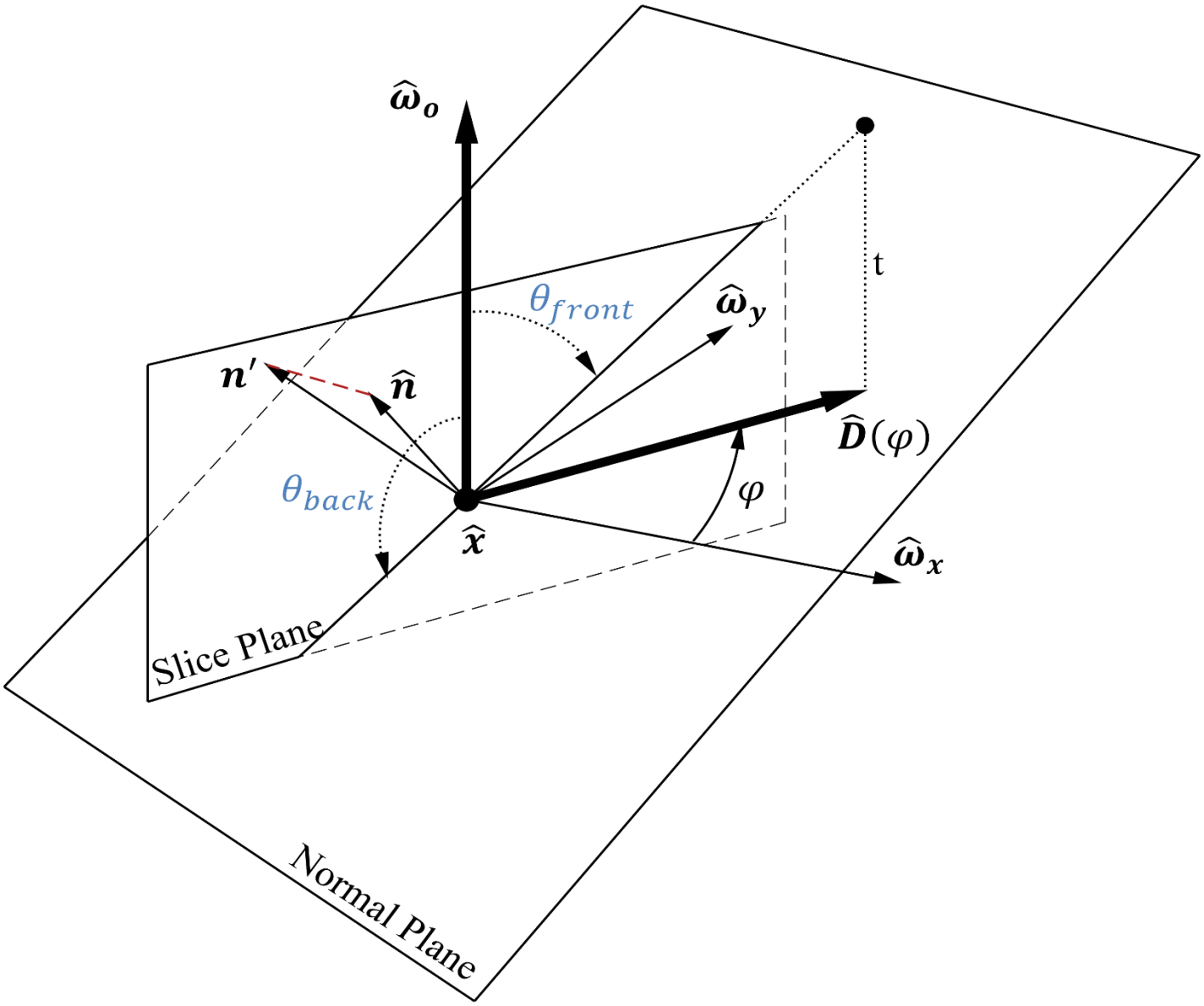

FIGURE 6. *A SAMPLING SLICE ROTATING BY AN ANGLE* φ *ABOUT THE* $\hat{\omega}_0$ *AXIS.* $\hat{D}(\varphi)$ *IS THE TANGENT DIRECTION OF THE SLICE.* *WE POINT OUT THE IMPORTANT FACT THAT THE WORLD-SPACE NORMAL* $\hat{n}$ *IS NOT NECESSARILY LYING IN THE PLANE OF THE SLICE AND NEEDS TO BE PROJECTED ONTO THE SLICE, YIELDING THE* n' *VECTOR. WE NOTICE THAT PROJECTING* $\hat{D}(\varphi)$ *ONTO THE NORMAL PLANE GIVES US INITIAL VALUES FOR THE HORIZON ANGLES* θ_{front} *AND* θ_{back} *(THIS IS DISCUSSED IN SECTION 2.1).*

1.3. Computing Horizon Angles

Special care must be observed when taking screen-space horizon angles –computed by sampling the depth buffer, whose Z values are expressed along the global camera $\hat{\boldsymbol{Z}}$ axis– into local camera-space horizon angles θ expressed from the local view axis $\hat{\boldsymbol{\omega}}_{\boldsymbol{o}}$:

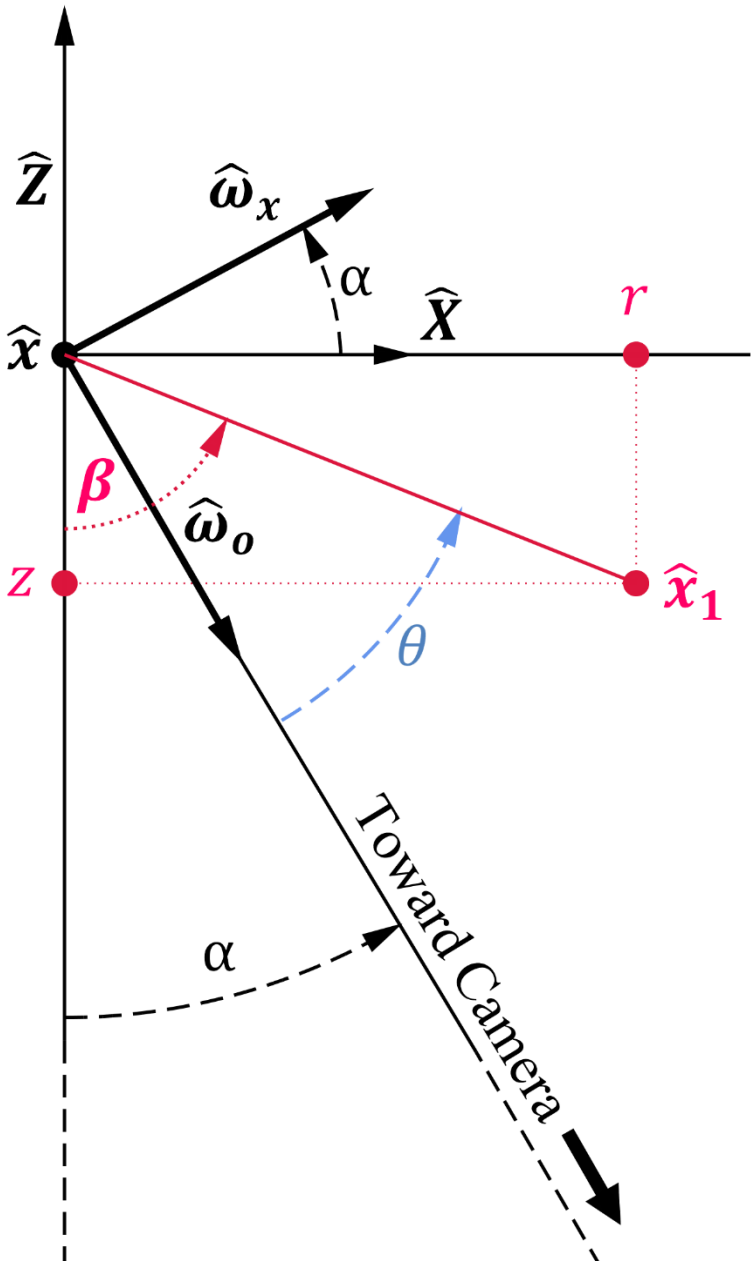

FIGURE 7. *CORRECTION SETUP TO TRANSFORM CAMERA-SPACE HORIZON ANGLES INTO ACTUAL SLICE-SPACE HORIZON ANGLES.*

Let's imagine we are sampling a new location $\hat{\boldsymbol{x}}_1$ away from our central location $\hat{\boldsymbol{x}}$ while walking along a slice.

In screen space (i.e. in the camera plane defined by the $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$ tangent axes), we moved r units away from our central location where we sample our new depth value (relative to our central location's depth) z and we obtain the angle β.

Unfortunately, we are rather interested by the horizon angle θ that the red axis $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}_1)$ forms with our view direction $\hat{\boldsymbol{\omega}}_{\boldsymbol{o}}$, than the horizon angle β it forms with the camera axis $\hat{\boldsymbol{Z}}$.

We know that the global camera $\hat{\boldsymbol{Z}}$ axis is making an angle α with our local view direction $\hat{\boldsymbol{\omega}}_{\boldsymbol{o}}$.

We easily notice that $\theta = \beta - \alpha$ and thus we can write:

$$\cos(\theta) = \cos(\beta - \alpha)$$

Trigonometric identities give us:

$$\cos(\theta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)$$

We know:

$$\cos(\beta) = \frac{z}{\sqrt{r^2 + z^2}}$$

$$\sin(\beta) = \frac{r}{\sqrt{r^2 + z^2}}$$

And finally, our horizon angle is simply given by the expression:

$$\cos(\theta) = \frac{\cos(\alpha)\, z + \sin(\alpha)\, r}{\sqrt{r^2 + z^2}}$$

We notice that $\cos(\theta) = \cos(\beta)$ when the camera axis and view directions are aligned (i.e. at the center of the screen), while $\cos(\theta) \to \sin(\beta)$ when we reach the borders of the screen and the deviation from the camera axis is maximal.

2. Improvements over HBAO

In this paper, we will discuss several improvements to make the most out of the information gathered while performing the HBAO algorithm, these improvements will give us what I called the Horizon-Based Indirect Lighting (HBIL) algorithm:

- Section 2.1. will discuss how to use the camera-space normal to make the result more robust and reveal more details.

- Section 2.2 will discuss how to obtain the bent-cone from our samples that will yield a better estimate of the far-field irradiance.

- Finally, section 2.3 will explain how to eventually re-use the indirect diffuse lighting from the previously rendered frame to compute a very important near-field indirect lighting term.

2.1. Using the Normal

We can first improve a little upon the quality of the HBAO algorithm by using the normal to initialize the horizon angles.

The normal vector is often readily available from the G-Buffer produced by deferred renderers. For each slice, we can initialize the front and back horizon angles by projecting the slice's direction vector $\widehat{\boldsymbol{D}}(\varphi)$ onto the normal plane by following the camera view vector $\widehat{\boldsymbol{\omega}}_{\boldsymbol{o}}$, as shown in figure 6:

$$t = -\frac{\widehat{\boldsymbol{D}}(\varphi) \cdot \widehat{\boldsymbol{n}}}{\widehat{\boldsymbol{\omega}}_{\boldsymbol{o}} \cdot \widehat{\boldsymbol{n}}}$$

$$\cos(\theta_{front}) = \frac{t}{\sqrt{1+t^2}}$$
$$\cos(\theta_{back}) = -\frac{t}{\sqrt{1+t^2}}$$

We thus obtain θ_{front} and θ_{back}, the initial front and back horizon angles represented in figure 6.

Figure 8. *Left: Horizon angles are initialized by depth alone. Right: Horizon angles are initialized with the G-Buffer's normal. We can see the resulting images are much more detailed and convincing.*

2.2. Bent Cones

We follow the methodology of the GTAO computation described by Jimenez et al. [3] to compute the Ambient Occlusion but we are also interested in computing what I call a "bent cone", which is a combination of a bent normal used as the central axis of a cone whose aperture depends on the ambient occlusion:

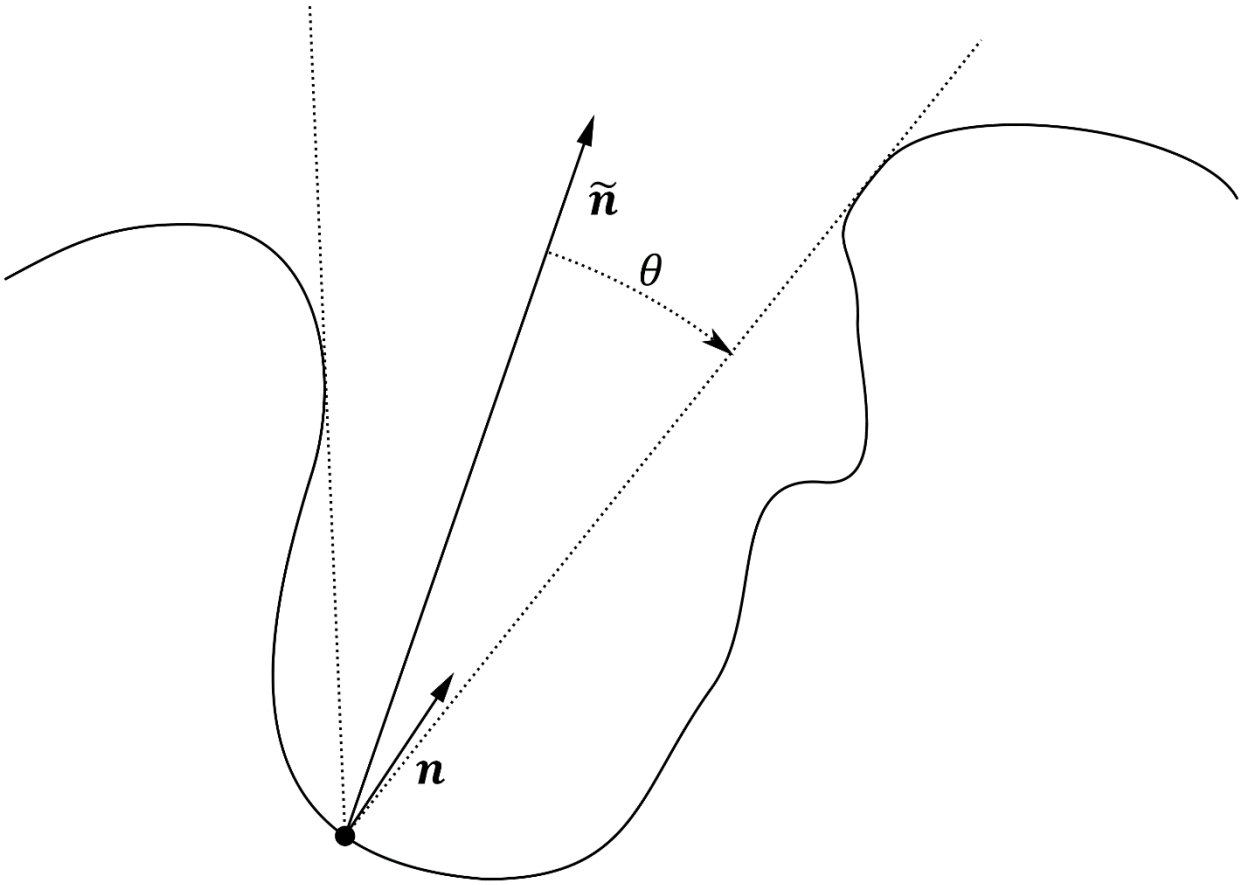

FIGURE 9. THE "BENT-CONE", A BENT NORMAL WITH AN ANGLE.

2.2.1. Bent Normal

Vidiger et al. [9] showed the importance of the bent normal in the production of realistic images in their SSRTGI technique, we will show here how to obtain the bent normal without the need for ray-tracing, once again by extending the HBAO algorithm.

In order to get the bent normal $\widetilde{\boldsymbol{n}}$, we need to compute the average direction of a vector unoccluded by the environment:

$$\widetilde{\boldsymbol{n}} = \int_{\Omega^+} V(\boldsymbol{x}, \boldsymbol{\omega_i})\, \boldsymbol{\omega_i}\, \mathrm{d}\omega_i \qquad (\mathbf{3})$$

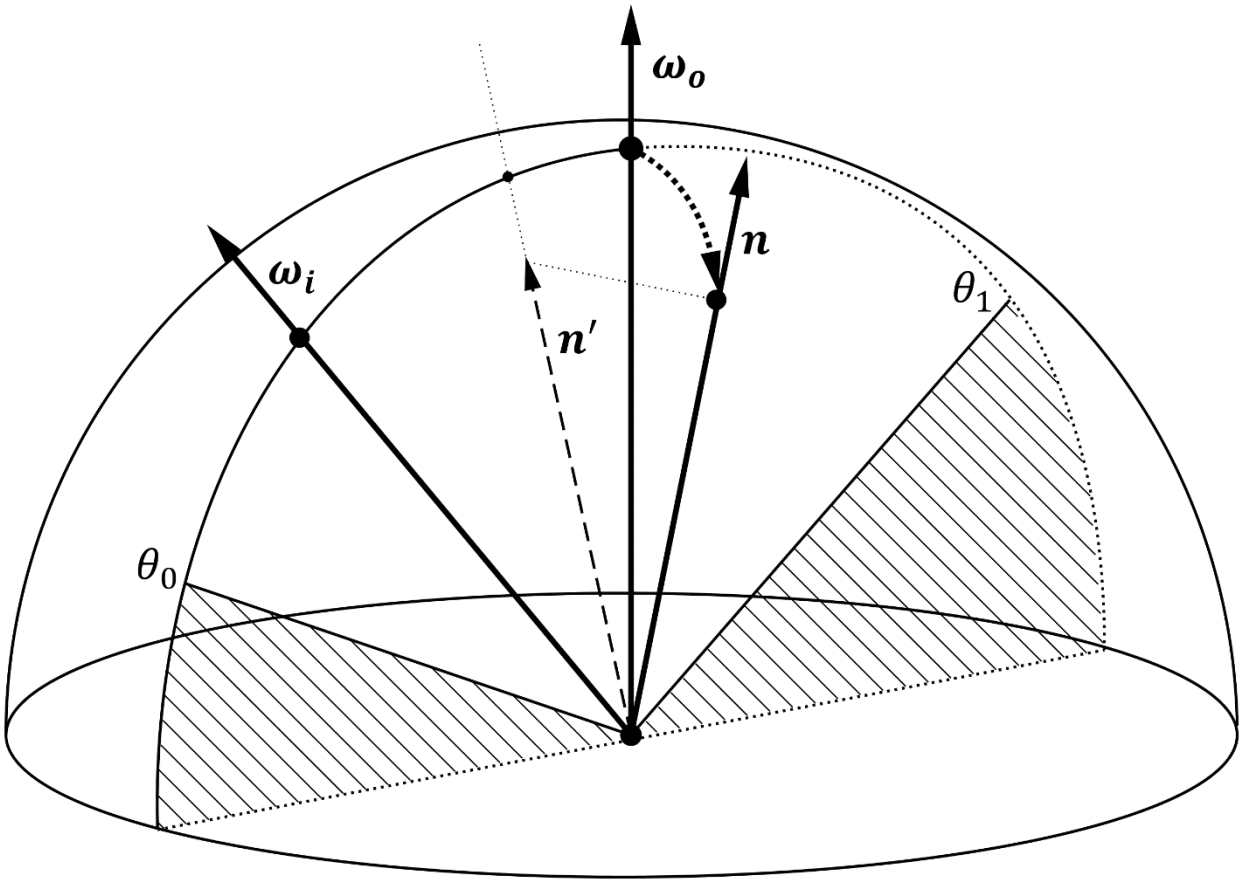

FIGURE 10. *INTEGRATION OF THE VECTOR $\boldsymbol{\omega_i}$ BETWEEN ANGLES θ_0 AND θ_1, RESTRICTED TO A SINGLE SLICE. NOTICE THAT THE NORMAL VECTOR $\boldsymbol{n}$ IS NOT NECESSARILY LYING IN THE SLICE'S PLANE.*

We can get an approximation of the integral for equation 3 by computing a series of integrals restricted to 2D slices rotating about our view axis $\boldsymbol{\omega_o}$ as shown in the figure 10 above:

$$\boldsymbol{\omega_i}(\theta) = \sin(\theta)\,.\,\boldsymbol{D}(\varphi) + \cos(\theta)\,.\,\boldsymbol{\omega_o}$$

$$\tilde{\boldsymbol{n}} \approx \frac{\pi}{S}\sum^{S}\int_{\theta_0}^{\theta_1} \boldsymbol{\omega_i}(\theta)\,|\sin(\theta)|\,d\theta \qquad (\mathbf{4})$$

Where:

- S is the amount of slices
- θ_0 is the "back" horizon angle for the slice. $\theta_0 \in [-\pi, 0]$
- θ_1 is the "front" horizon angle for the slice. $\theta_1 \in [0, \pi]$

Focusing on the inner integral, we rewrite its expression in slice-space as:

$$\boldsymbol{\omega_i'}(\theta) = \begin{cases}\sin(\theta)\\ \cos(\theta)\end{cases}$$

$$\tilde{\boldsymbol{n}}' = \int_{\theta_0}^{\theta_1} \boldsymbol{\omega_i'}(\theta)\,|\sin(\theta)|\,d\theta$$

Where:

- $\boldsymbol{\omega_i'}(\theta)$ is the slice-space version of $\boldsymbol{\omega_i}(\theta)$
- $\tilde{\boldsymbol{n}}'$ is the resulting normal vector for the slice

And we get:

$$\tilde{\boldsymbol{n}}' = \begin{cases}\tilde{n}_x\\ \tilde{n}_y\end{cases}$$

$$\tilde{n}_x = \frac{1}{2}(\theta_1 - \theta_0 + \sin(\theta_0)\cos(\theta_0) - \sin(\theta_1)\cos(\theta_1)) \qquad (\mathbf{5})$$

$$\tilde{n}_y = \frac{1}{2}(2 - \cos^2(\theta_0) - \cos^2(\theta_1)) \qquad (\mathbf{6})$$

From which can finally rebuild a unit camera-space normal:

$$\tilde{\boldsymbol{n}} = \frac{\sum_i^S \boldsymbol{D}(\varphi_i).\,\tilde{n}_x(i) + \boldsymbol{\omega_o}\,.\,\tilde{n}_y(i)}{\left\|\sum_i^S \boldsymbol{D}(\varphi_i).\,\tilde{n}_x(i) + \boldsymbol{\omega_o}\,.\,\tilde{n}_y(i)\right\|}$$

Where:

- $\tilde{n}_x(i)$ and $\tilde{n}_y(i)$ are the $\tilde{n}_x$ and $\tilde{n}_y$ values computed by equations 5 and 6 for slice i
- $\boldsymbol{D}(\varphi_i)$ is the slice direction for slice i

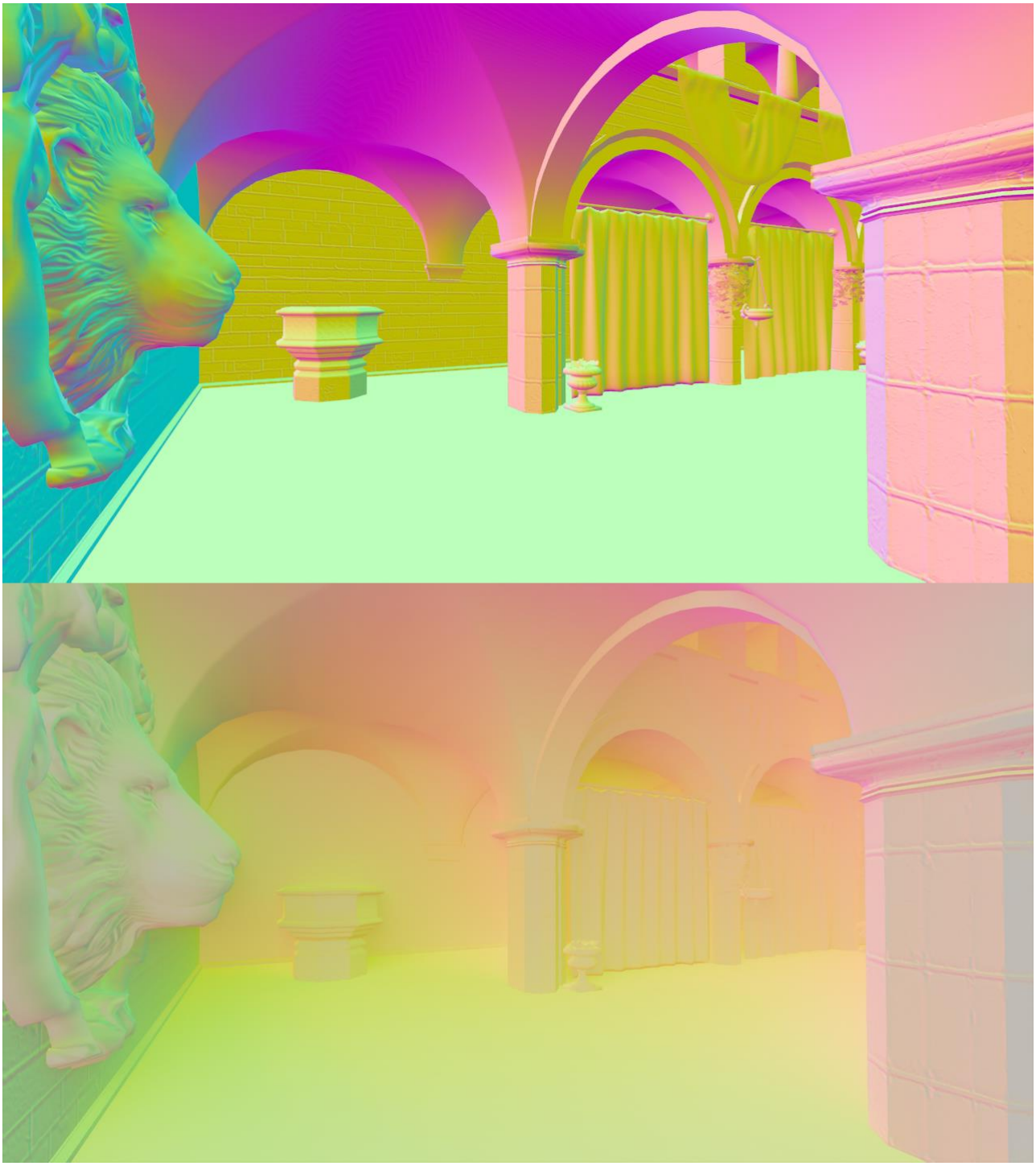

Figure 11. *Top: World-space normal. Bottom: World-space Bent Normal. The direction of the normal is influenced to bend toward the direction of least-occlusion. This is the direction that is the most suited to sample the far-field irradiance.*

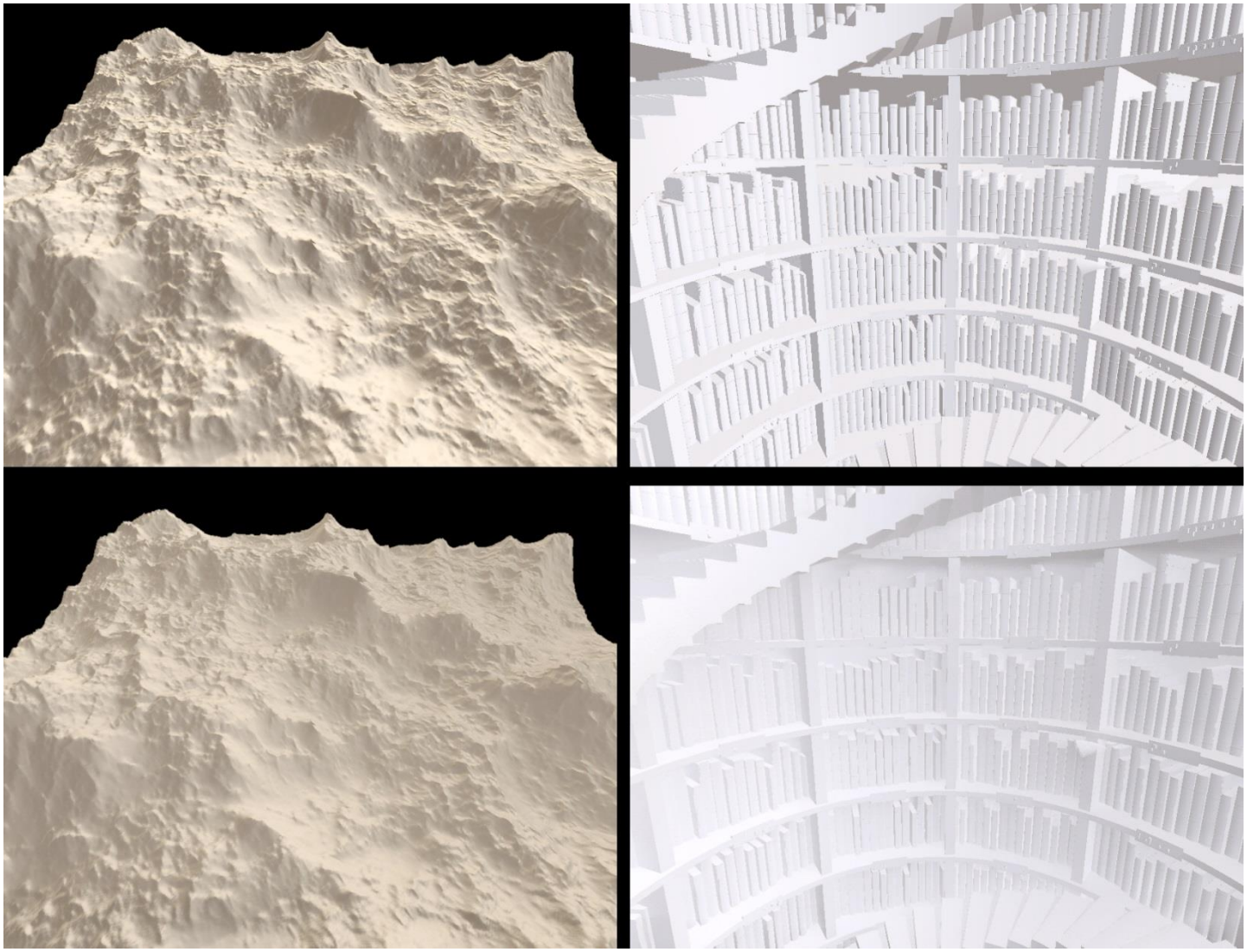

Figure 12. *Top: Far-Field irradiance sampling from terrain normal. Bottom: Sampling in the direction of the computed bent normal. The results are much smoother. (Library scene courtesy of Léon Denise)*

2.2.2. Cone Aperture and Ambient Occlusion

Now, for the angle part of the bent cone, we need the solid angle covered by the cone to be significant when we use it to sample the far-field environment:

$$E_{far}(\boldsymbol{x},\boldsymbol{n}) = \int_{\Omega^+} V(\boldsymbol{x},\boldsymbol{\omega_i}) L_i(\boldsymbol{x},\boldsymbol{\omega_i})(\boldsymbol{n}.\boldsymbol{\omega_i})\, \mathrm{d}\omega_i \tag{7}$$

Where:

- $E_{far}(\boldsymbol{x},\boldsymbol{n})$ is the irradiance at $\boldsymbol{x}$ for surface normal $\boldsymbol{n}$ from the far-field environment
- $L_i(\boldsymbol{x},\boldsymbol{\omega_i})$ is the incoming radiance at $\boldsymbol{x}$ from direction $\boldsymbol{\omega_i}$
- Ω^+ is the set of all directions covering the upper hemisphere
- $d\omega_i$ is the solid angle covered by the surface perceived along direction $\boldsymbol{\omega_i}$
- $V(\boldsymbol{x},\boldsymbol{\omega_i})$ is the visibility term from equation 1

I wrote in [4] that equation 7 is often simplified into:

$$\tilde{E}_{far}(\boldsymbol{x},\boldsymbol{n}) = \left[\int_{\Omega^+} L_i(\boldsymbol{x},\boldsymbol{\omega_i})(\boldsymbol{n}.\boldsymbol{\omega_i})\, \mathrm{d}\omega_i\right].\left[\frac{1}{2\pi}\int_{\Omega^+} \mathrm{V}(\boldsymbol{x},\boldsymbol{\omega_i})\, \mathrm{d}\omega_i\right] = E_0(\boldsymbol{x},\boldsymbol{n}).AO(\boldsymbol{x}) \tag{8}$$

Where $E_0(\boldsymbol{x},\boldsymbol{n})$ is the unoccluded irradiance from a diffuse cube map, or some SH representation.

Jimenez et al. in their "Ground-Truth AO" (GTAO) [3] use instead an AO term they call "radiometrically correct", meaning they account for the dot product with the normal in the visibility integral as well, and so the AO integral becomes:

$$AO(\boldsymbol{x}) = \frac{1}{\pi}\int_{\Omega^+} \mathrm{V}(\boldsymbol{x},\boldsymbol{\omega_i})(\boldsymbol{n}.\boldsymbol{\omega_i})\, \mathrm{d}\omega_i \tag{9}$$

And indeed, this is actually the term we need because we will be using the complement of that term in section 2.3 when we will compute the near-field irradiance term.

Also notice the normalization factor is now $\frac{1}{\pi}$ instead of $\frac{1}{2\pi}$.

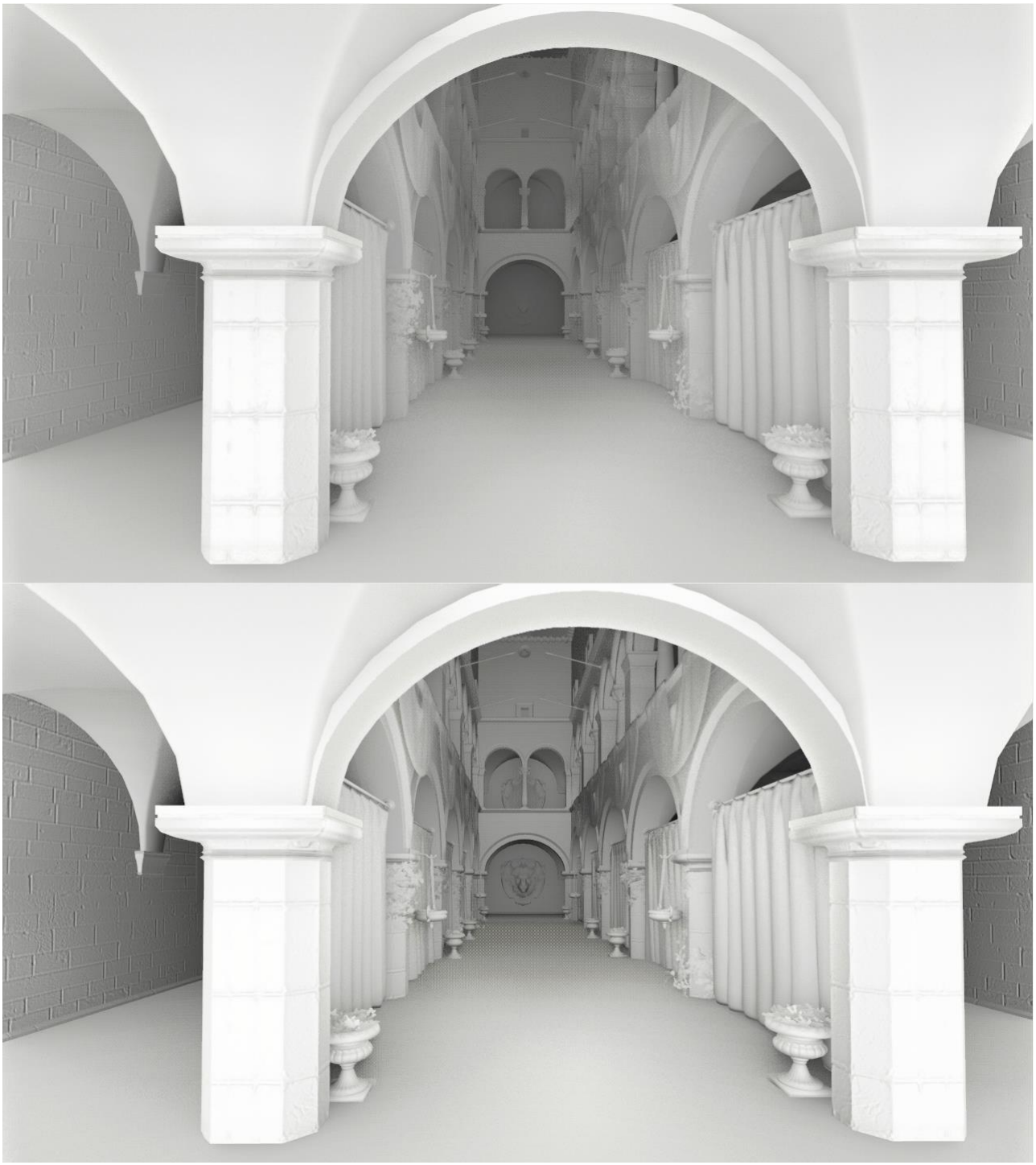

Figure 13. *Influence of the dot product with the normal on the computation of the Ambient Occlusion term. Top: without influence of the normal (equation 7). Bottom: with influence of the normal (equation 8). The bottom version is preferred as it is physically correct. We also notice it helps revealing more subtle details.*

Identically to equation 4 used to compute the bent normal, we can write:

$$\boldsymbol{\omega_i}(\theta) = \sin(\theta)\,.\,\boldsymbol{D}(\varphi) + \cos(\theta)\,.\,\boldsymbol{\omega_o}$$

$$AO(\boldsymbol{x}) \approx \frac{1}{S}\sum^{S}\int_{\theta_0}^{\theta_1}\left(\boldsymbol{n}.\,\boldsymbol{\omega_i}(\theta)\right)|\sin(\theta)|\,d\theta$$

Where i is the slice index and S the amount of slices.

We rewrite once again the inner integral in slice-space between our 2 horizon angles θ_0 and θ_1:

$$\boldsymbol{\omega_i'}(\theta) = \begin{cases}\sin(\theta)\\ \cos(\theta)\end{cases}$$

$$AO_i = \int_{\theta_0}^{\theta_1}\left(\boldsymbol{n}'\cdot\boldsymbol{\omega_i'}(\theta)\right)|\sin(\theta)|\,d\theta \qquad (\mathbf{10})$$

With AO_i the AO value for slice i and $\boldsymbol{n}'$ the normal vector $\hat{\boldsymbol{n}}$ projected into slice-space (cf. equation 2 and figure 6).

We finally get:

$$AO_i = \frac{n_x}{2}[\theta_1 - \theta_0 + \sin(\theta_0)\cos(\theta_0) - \sin(\theta_1)\cos(\theta_1)] + \frac{n_y}{2}[2 - \cos^2(\theta_0) - \cos^2(\theta_1)] \qquad (\mathbf{11})$$

And finally, AO is given by:

$$AO(\boldsymbol{x}) \approx \frac{1}{S}\sum_{i}^{S} AO_i$$

To finalize our bent-cone, we are interested in another formulation for the AO, which would be the solid angle covered by the cone with aperture angle α :

$$\Omega = 2\pi(1 - \cos(\alpha))$$

The true integral of the visibility term in equation 8 actually yields exactly this solid angle:

$$\Omega = \int_{\Omega^+} \mathrm{V}(\boldsymbol{x}, \boldsymbol{\omega_i})\,\mathrm{d}\omega_i = 2\pi\, AO(\boldsymbol{x})$$

From this we can deduce that:

$$AO(\boldsymbol{x}) = 1 - \cos(\alpha)$$

And finally, the aperture angle of our bent-cone is given by:

$$\alpha = \cos^{-1}\big(1 - AO(\boldsymbol{x})\big) \qquad (\mathbf{12})$$

We can also conclude that the cosine of the aperture angle of our bent-cone is the *geometric equivalent* of the classical AO formulation, but we will prefer it to the AO term since the cosine of an angle lends itself better to a comparison to the $N \cdot L$ dot product generally found in lighting equations.

The effect of accounting for the cone aperture angle is most important, as can be seen in the images below:

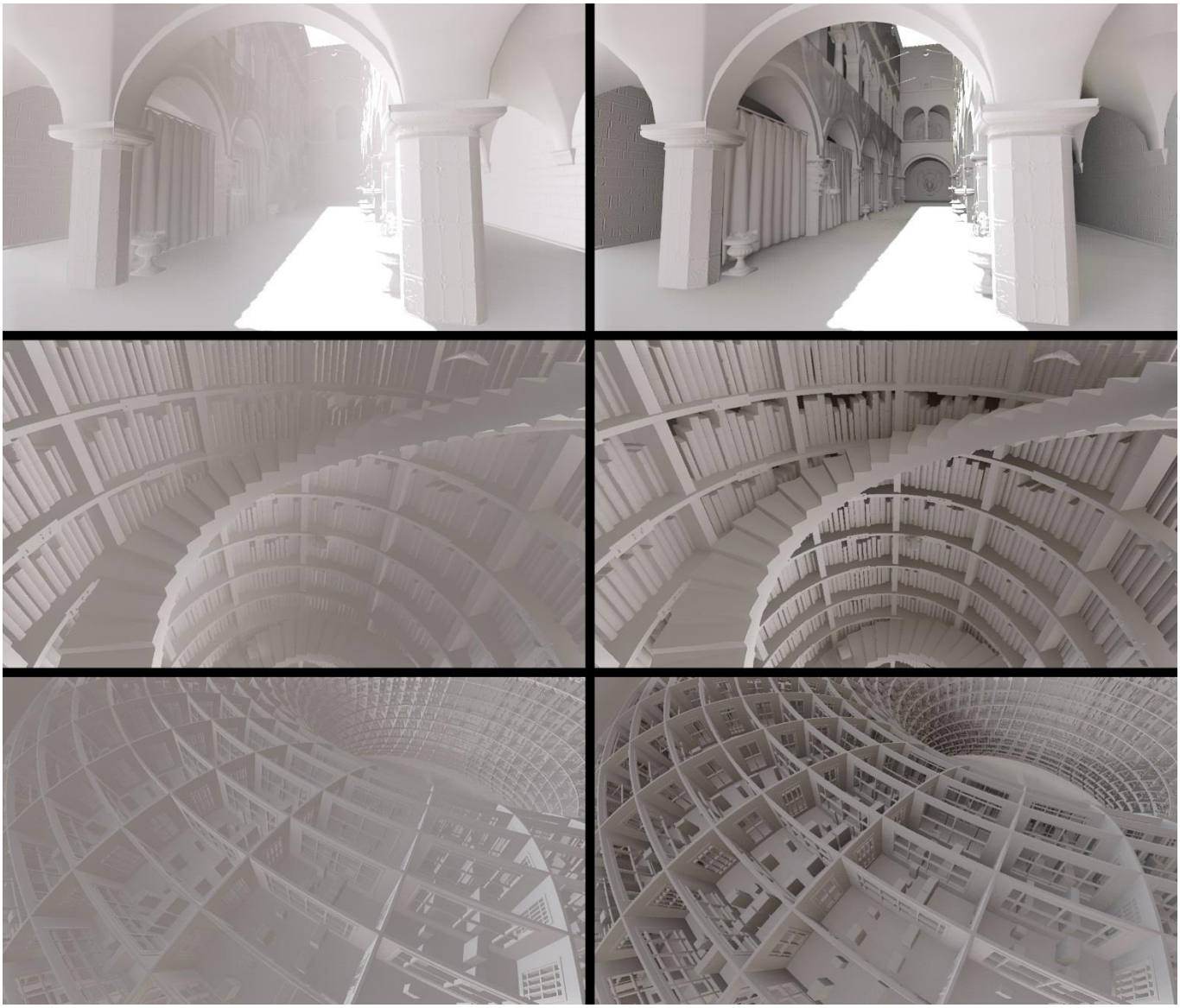

Figure 14. *Influence of the Ambient Occlusion / Cone Angle term. Left: No Ambient Occlusion is used. Right: With Ambient Occlusion term. Note that the bent-normal is enabled in all images. (Torus Rooms scene courtesy of Léon Denise)*

2.2.3. Using the bent-cone for far-field indirect illumination (Cube Map)

In [4] we noted that the simplification induced by equation 8 suffers from a loss in energy that should be compensated by applying a factor given by $\mathcal{F}_0(\alpha)$ so that:

$$\tilde{E}_{far}(\boldsymbol{x},\boldsymbol{n}) = E_0(\boldsymbol{x},\boldsymbol{n}).\mathcal{F}_0\big(AO(\boldsymbol{x})\big)$$

With:

$$\mathcal{F}_0(\alpha) = \alpha\,.\left(1+\frac{(1-\alpha)^{0.75}}{2}\right) = \alpha\,.\left(1+\frac{1-\alpha}{2\sqrt[4]{1-\alpha}}\right)$$

Here, $E_0(\boldsymbol{x},\boldsymbol{n})$ is sampled from a pre-convolved diffuse cube map, or obtained from any other way where the angular aperture is predefined to be 90° and cannot be changed.

2.2.4. Using the bent-cone for far-field indirect illumination (Spherical Harmonics)

If the distant environment is available as spherical harmonics (SH) instead, then we should use the modified irradiance estimate from [5] that allows to reduce the aperture angle used in Ramamoorthi's calculations [7] using the simple free parameter θ_{max} as replacement for the upper bound to his integral (16) instead of the default $\frac{\pi}{2}$ value:

$$A_n = 2\pi \int_0^{\theta_{max}} Y_{n,0}(\theta_i')\cos(\theta_i')\ \sin(\theta_i')\,d\theta_i'$$

I will only give the procedure for an order-2 SH representation (i.e. 9 coefficients), details can be found on my wiki page for a possible extension to higher orders:

$$t = \cos(\theta_{max}) = 1 - AO(\boldsymbol{x})$$

$$c_0 = \frac{1}{2}\sqrt{\pi}\,(1-t^2)$$

$$c_1 = \frac{1}{3}\sqrt{3\,\pi}\,(1-t^3)$$

$$c_2 = \frac{1}{4}\sqrt{5\,\pi}\left[\frac{3(1-t^4)-2(1-t^2)}{4}\right]$$

Then the resulting irradiance estimate for a cone of aperture θ_{max} is given by:

$$\boldsymbol{n} = \begin{cases} x \\ y \\ z \end{cases}$$

$$\begin{aligned}\tilde{E}_{far}(\boldsymbol{n}) &= c_0.L_0^0 \\ &+c_1.(y.L_1^{-1} + z.L_1^0 + x.L_1^1) \\ &+c_2.L_2^0.(3z^2 - 1) \\ &+c_2\sqrt{3}.L_2^2(x^2 - y^2) \\ &+c_2 2\sqrt{3}.(x.y.L_2^{-2} + y.z.L_2^{-1} + z.x.L_2^1)\end{aligned}$$

The L_l^m represent the encoded SH terms for the distant environment's radiance.

Note: It is also totally possible to estimate the irradiance by simply using the classical hemisphere SH estimate from [7] and multiply the result by $\mathcal{F}_0\big(AO(\boldsymbol{x})\big)$ as seen in section 2.2.3. but I find the results more accurate using the reduced-cone formulation.

2.3. Indirect Lighting

The classical lighting equation to compute the outgoing radiance from a pixel at $\boldsymbol{x}$ in viewing direction $\boldsymbol{\omega_o}$ is essentially given by:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega_o}) = \int_{\Omega^+} L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \rho(\boldsymbol{x}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) (\boldsymbol{n}.\boldsymbol{\omega_i}) \, \mathrm{d}\omega_i \qquad (\mathbf{13})$$

Where:

- $L_i(\boldsymbol{x}, \boldsymbol{\omega_i})$ is the incoming radiance at $\boldsymbol{x}$ from direction $\boldsymbol{\omega_i}$
- $\rho(\boldsymbol{x}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ is the surface's BRDF
- $\boldsymbol{n}$ is the surface normal
- Ω^+ is the set of all directions covering the upper hemisphere
- $d\omega_i$ is the solid angle covered by the surface perceived along direction $\boldsymbol{\omega_i}$

Focusing only on diffuse Lambertian reflection, we can rewrite equation 13 as:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega_o}) = \frac{\rho(\boldsymbol{x})}{\pi} \int_{\Omega^+} L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) (\boldsymbol{n}.\boldsymbol{\omega_i}) \, \mathrm{d}\omega_i = \frac{\rho(\boldsymbol{x})}{\pi} E(\boldsymbol{x}, \boldsymbol{n}) \qquad (\mathbf{14})$$

Where:

- $E(\boldsymbol{x}, \boldsymbol{n})$ is the irradiance collected at surface location $\boldsymbol{x}$ and normal $\boldsymbol{n}$.

- $\frac{\rho(x)}{\pi}$ represents the diffuse BRDF for a surface with albedo $\rho(\boldsymbol{x}) \in [\mathbf{0}, \mathbf{1}]$.
  The division by π is here to guarantee energy conservation since $\int_{\Omega^+}(\boldsymbol{n}.\boldsymbol{\omega_i})\,\mathrm{d}\omega_i = \pi$.

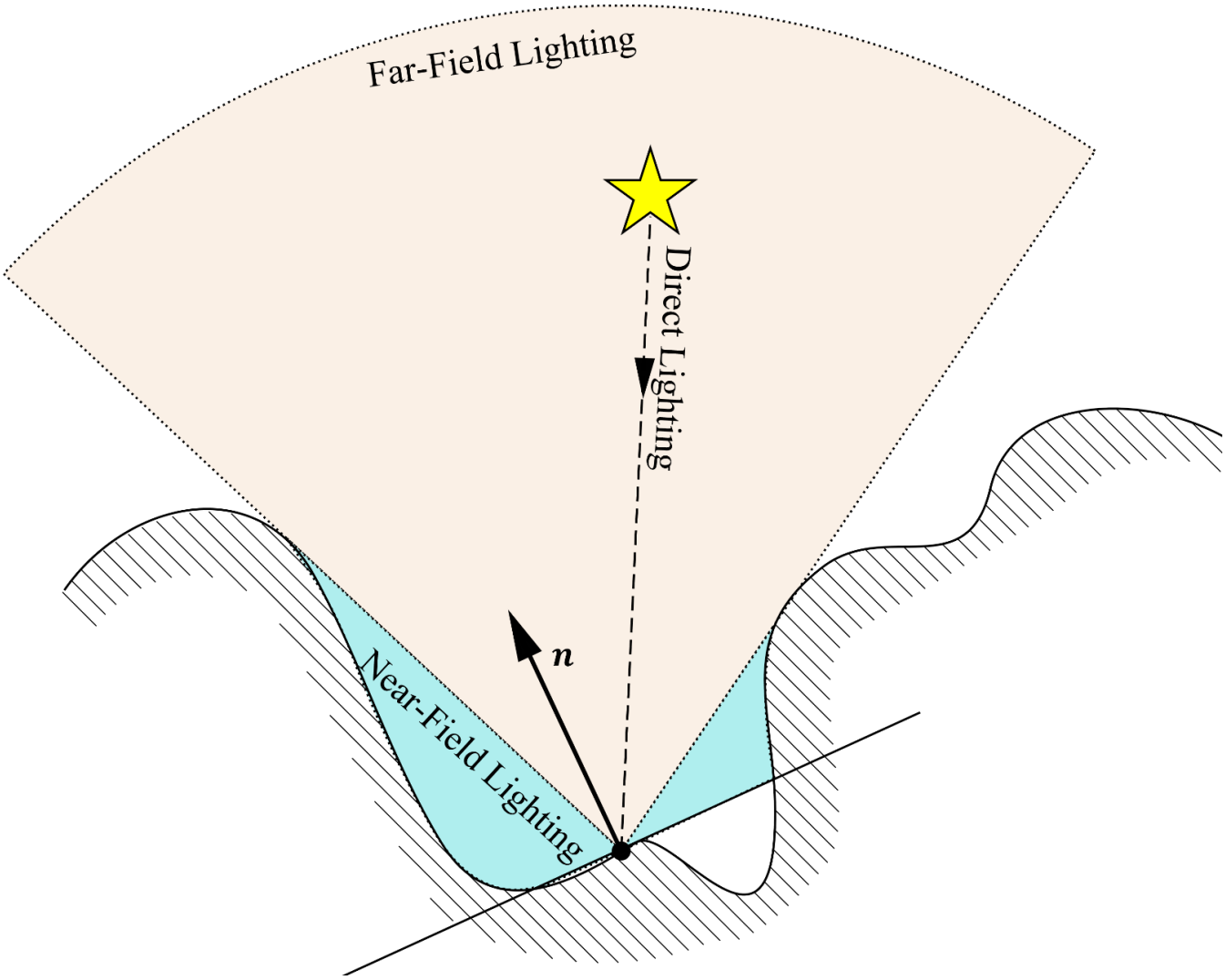

FIGURE 15. *REPRESENTATION OF THE 3 POSSIBLE SOURCES OF IRRADIANCE: DIRECT LIGHTING, FAR-FIELD LIGHTING FROM THE UNOCCLUDED ENVIRONMENT AND NEAR-FIELD LIGHTING WHICH REPRESENTS THE IRRADIANCE THAT BOUNCED OFF THE NEIGHBOR ENVIRONMENT. WE NOTICE THAT THE NEAR-FIELD ONLY ACCOUNTS FOR THE OCCLUDED ENVIRONMENT ABOVE THE NORMAL PLANE AND IS THE* ***EXACT COMPLEMENT*** *OF THE FAR-FIELD.*

We saw in equation 8 how to sample the far-field environment but as shown in figure 15, the *complete* formulation for the irradiance is really given by:

$$E(\boldsymbol{x}, \boldsymbol{n}) = E_{direct}(\boldsymbol{x}, \boldsymbol{n}) + E_{far}(\boldsymbol{x}, \boldsymbol{n}) + E_{near}(\boldsymbol{x}, \boldsymbol{n})$$

Where:

- $E_{direct}(\boldsymbol{x}, \boldsymbol{n})$ is the direct diffuse lighting irradiance computed using all the light sources of the scene and has already been extensively covered throughout the CG literature
- $E_{far}(\boldsymbol{x}, \boldsymbol{n})$ is the far-field distant environment term that can be computed using the bent-cone, as shown in sections 2.2.3. and 2.2.4.
- $E_{near}(\boldsymbol{x}, \boldsymbol{n})$ represents the missing near-field environment term, the part of the energy that bounced off the close environment and perceived indirectly by our sampling point.
  This part is often ignored, simplified or statically pre-computed as it is quite tricky and expensive to evaluate.

$E_{near}(\boldsymbol{x}, \boldsymbol{n})$ can be written as:

$$E_{near}(\boldsymbol{x}, \boldsymbol{n}) = \int_{\Omega^+} \big(1 - V(\boldsymbol{x}, \boldsymbol{\omega_i})\big) L_i(\boldsymbol{x}, \boldsymbol{\omega_i})(\boldsymbol{n}.\boldsymbol{\omega_i}) \, \mathrm{d}\omega_i \qquad (\mathbf{15})$$

We immediately notice this term is using the opposite of the visibility function and is thus the *exact complement* of the far-field environment term from equation 7 since we only gather radiance that bounced off of the *occluded* environment

Contrary to [4] where the near-field bounces were approximated via manually-fitted and ad-hoc terms, in this paper we will see how to actually compute these bounces.

2.3.1. Recursive Irradiance Bounces

We begin by noticing that, since we are only considering the diffuse Lambertian case, the incoming radiance term $L_i(\boldsymbol{x}, \boldsymbol{\omega_i})$ from equation 15 has to be a diffuse reflection of the light off the neighbor environment:

$$L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) = \frac{\rho(\boldsymbol{x}')}{\pi} E(\boldsymbol{x}', \boldsymbol{n}')$$

Where $\boldsymbol{x}'$ and $\boldsymbol{n}'$ are the location and normal of the neighbor environment perceived in direction $\boldsymbol{\omega_i}$.

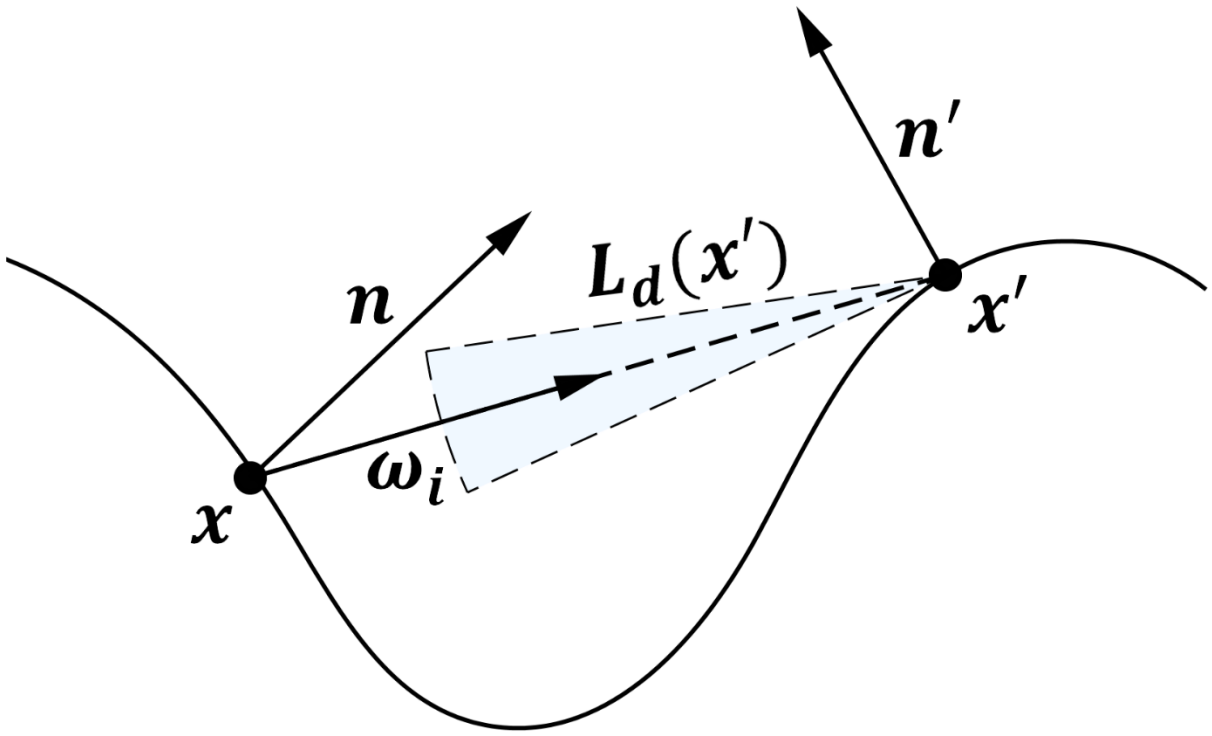

FIGURE 16. WE ARE ONLY INTERESTED IN THE DIFFUSE REFLECTION OF LIGHT AT x' WHEN SAMPLING THE ENVIRONMENT FROM x IN DIRECTION ω_i. WE CALL THIS NORMAL-AGNOSTIC, DIFFUSELY REFLECTED RADIANCE, $L_d(x')$.

Assuming the neighbor is taking care of its own lighting (i.e. we are making an independence hypothesis) then we write:

$$L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) = L_d(\boldsymbol{x}')$$

Which is the diffusely-reflected radiance sampled at location $\boldsymbol{x}'$.

Note: Because we are going to sample a buffer where we stored the diffuse *radiance* $L_d(\boldsymbol{x}')$, the $\frac{\rho(x\prime)}{\pi}$ and $E(\boldsymbol{x}', \boldsymbol{n}')$ are already combined together and we cannot make the same assumption as in [3] that the neighbor reflectance is the same as our current reflectance $\rho(\boldsymbol{x}') = \rho(\boldsymbol{x})$, nor is it desired to make such an assumption any more since we are not computing an approximate term here**.**

We can then rewrite equation 15 as:

$$E_{near}(\boldsymbol{x}, \boldsymbol{n}) = \int_{\Omega^+} L_d(\boldsymbol{x}')(\boldsymbol{n}.\boldsymbol{\omega_i})\,\mathrm{d}\omega_i \;, \qquad V(\boldsymbol{x}, \boldsymbol{\omega_i}) = 0 \tag{16}$$

And interestingly, we notice that estimating the near-field irradiance simply involves sampling the diffusely-reflected radiance from neighbor surfaces.

So, when rendering frame N, assuming the diffuse radiance values computed for the *last frame N-1* are available then we could solve the entire lighting diffuse equation 14 by computing:

$$L_o{}^N(\boldsymbol{x}, \boldsymbol{\omega_o}) = \frac{\rho(\boldsymbol{x})}{\pi}\left[E_{direct}{}^N(\boldsymbol{x}, \boldsymbol{n}) + E_{far}{}^N(\boldsymbol{x}, \boldsymbol{n}) + \int_{\Omega^+} L_d{}^{N-1}(\boldsymbol{x'})\ (\boldsymbol{n}.\boldsymbol{\omega_i})\ \mathrm{d}\omega_i\right] \qquad (\mathbf{17})$$

With:

- $L_o{}^N(\boldsymbol{x}, \boldsymbol{\omega_o})$ the radiance computed at frame N that will be stored in the diffuse radiance buffer which will, in turn, be re-used as input at frame N+1
- $E_{direct}{}^N(\boldsymbol{x}, \boldsymbol{n})$ the direct lighting irradiance computed at frame N
- $E_{far}{}^N(\boldsymbol{x}, \boldsymbol{n})$ the far-field radiance computed at frame N
- $L_d{}^{N-1}(\boldsymbol{x'})$ the diffusely-reflected radiance computed at frame N-1 and stored at neighbor location $\boldsymbol{x'}$

Computing radiance in this fashion would theoretically give us an "infinite amount of light bounces" since we keep the light bouncing from one frame to the next.

The details on how to reproject radiance from frame N-1 are given later in section 3.1.

2.3.2. Solving the indirect lighting integral

All we are left to solve is equation 16. Coming back to the context of our integration slice, where we are slowly computing the horizon angles for our front and back directions:

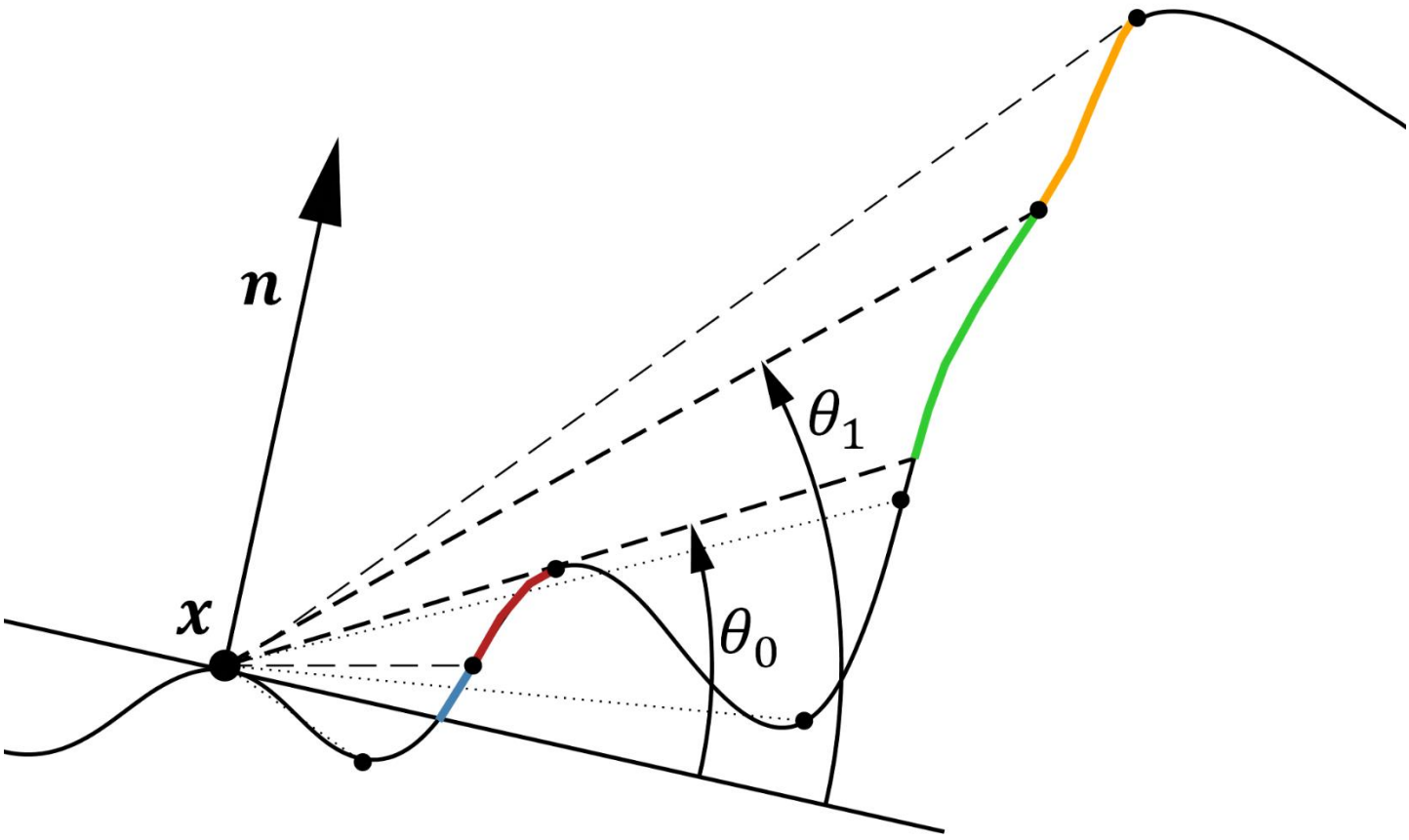

FIGURE 17. *SAMPLING THE GATHERED IRRADIANCE WHEN THE HORIZON IS RISING FROM θ_0 TO θ_1: WE SIMPLY NEED TO SAMPLE THE NEIGHBOR RADIANCE $L_d(\boldsymbol{x}')$ (SHOWN IN GREEN) AND COMPUTE ITS PERCEIVED INFLUENCE ON OUR CENTRAL LOCATION $\boldsymbol{x}$.*

In this figure, we can easily see that the only irradiance we perceive from $\boldsymbol{x}$ is the one from when the horizon jumps up a little: only the area represented in green contributes to the perceived irradiance when we rise from θ_0 to θ_1. This integrates very well into our existing ray-marching scheme and we see that we can get much more bang for our buck by gathering the indirect radiance as well!

So, for a small jump of the horizon angle from θ_0 to θ_1, we rewrite once again almost exactly the integral from equation 10 except we no longer samples a "unit far-field radiance" but the radiance reflected by our neighbor instead:

$$\boldsymbol{\omega_i'} = \begin{cases} \sin(\theta) \\ \cos(\theta) \end{cases}$$

$$\Delta E = \int_{\theta_0}^{\theta_1} L_d(\boldsymbol{x}') \, (\boldsymbol{n}' . \boldsymbol{\omega_i'}) \sin(\theta) \, d\theta$$

With $\boldsymbol{n}'$ the normal vector $\hat{\boldsymbol{n}}$ projected into slice-space (cf. equation 2 and figure 6).

Assuming $L_d(\boldsymbol{x}')$ is constant for the entire interval $[\theta_0, \theta_1]$ then ΔE becomes:

$$\Delta E \approx L_d(\boldsymbol{x}') \int_{\theta_0}^{\theta_1} (\boldsymbol{n}'.\boldsymbol{\omega_i'}) \sin(\theta)\, d\theta \qquad (\mathbf{18})$$

Solving the integral yields:

$$\int_{\theta_0}^{\theta_1} (\boldsymbol{n}'.\boldsymbol{\omega_i'}) \sin(\theta)\, d\theta = \frac{n_x}{2}[\theta_1 - \theta_0 + \sin(\theta_0)\cos(\theta_0) - \sin(\theta_1)\cos(\theta_1)] + \frac{n_y}{2}[\cos^2(\theta_0) - \cos^2(\theta_1)]$$

This new formulation is easy to integrate into the existing framework of HBAO computation and only requires to sample an additional radiance buffer, on top of our sampling of the depth buffer used for classical HBAO.

FIGURE 18. *LEFT: DIRECT LIGHTING ONLY. MIDDLE: IRRADIANCE GATHERED BY THE HBIL ALGORITHM. RIGHT: THE COMBINED RESULT. THE SCENE IS LIT BY A SINGLE LIGHT PLACED ON THE CEILING.*

2.3.3. Accounting for the Fresnel Term

When considering the full micro-facet BRDF model, we often find ourselves with an expression similar to:

$$\rho(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{\rho_d}{\pi}[1 - F(\boldsymbol{\omega_i}, \boldsymbol{\omega_h})] + F(\boldsymbol{\omega_i}, \boldsymbol{\omega_h}).\rho_s(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$$

Where $F(\boldsymbol{\omega_i}, \boldsymbol{\omega_h})$ is the Fresnel term that balances the distribution of the reflection between the specular part of the BRDF, namely $\rho_s(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$, and its diffuse part $\frac{\rho_d}{\pi}$ which is the only part we have considered so far.

$\boldsymbol{\omega_h} = \frac{\boldsymbol{\omega_i}+\boldsymbol{\omega_o}}{\|\boldsymbol{\omega_i}+\boldsymbol{\omega_o}\|}$ is the well-known half-direction vector between $\boldsymbol{\omega_i}$ and $\boldsymbol{\omega_o}$.

Assuming we are using the simplified Schlick's approximation to the actual Fresnel term:

$$F(\boldsymbol{\omega_i}, \boldsymbol{\omega_h}) = F_0 \ + (1 - F_0)(1 - \boldsymbol{\omega_i} \cdot \boldsymbol{\omega_h})^5$$

F_0 being the classical "reflectance at normal incidence" and is related to the medium's index of refraction by:

$$F_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$

The diffuse integral 18 then becomes:

$$\Delta E = L_d(\boldsymbol{x'}) \int_{\theta_0}^{\theta_1} \big(1 - F(\boldsymbol{\omega_i'}, \boldsymbol{\omega_h'})\big)(\boldsymbol{n'}.\boldsymbol{\omega_i'}) \sin(\theta)\, d\theta$$

Where:

$$\boldsymbol{\omega_h'} = \frac{\boldsymbol{\omega'}_{\boldsymbol{i}} + \boldsymbol{\omega'}_{\boldsymbol{o}}}{\|\boldsymbol{\omega'}_{\boldsymbol{i}} + \boldsymbol{\omega'}_{\boldsymbol{o}}\|}$$

$$\boldsymbol{\omega'}_{\boldsymbol{i}} = \begin{cases} \sin(\theta) \\ \cos(\theta) \end{cases} \qquad \boldsymbol{\omega'}_{\boldsymbol{o}} = \begin{cases} 0 \\ 1 \end{cases}$$

So ΔE becomes:

$$\begin{aligned} \Delta E &= L_d(\boldsymbol{x'}) \int_{\theta_0}^{\theta_1} (1 - F_0 - (1 - F_0)(1 - \boldsymbol{\omega_i'}.\boldsymbol{\omega_h'})^5)(\boldsymbol{n'}.\boldsymbol{\omega_i'}) \sin(\theta)\, d\theta \\ &= L_d(\boldsymbol{x'})(1 - F_0)\left[\int_{\theta_0}^{\theta_1} (\boldsymbol{n'}.\boldsymbol{\omega_i'}) \sin(\theta)\, d\theta - \int_{\theta_0}^{\theta_1} (1 - \boldsymbol{\omega_i'}.\boldsymbol{\omega_h'})^5 (\boldsymbol{n'}.\boldsymbol{\omega_i'}) \sin(\theta) d\theta\right] \end{aligned}$$

The first integral is equation 18 that we already solved, focusing on the new green expression we obtain:

$$\int_{\theta_0}^{\theta_1} (1 - \boldsymbol{\omega_i'}.\boldsymbol{\omega_h'})^5 (\boldsymbol{n'}.\boldsymbol{\omega_i'}) \sin(\theta)\, d\theta = \int_{\theta_0}^{\theta_1} \left(1 - \sqrt{\frac{1 + \cos(\theta)}{2}}\right)^5 \big(n_x \sin(\theta) + n_y \cos(\theta)\big) \sin(\theta)\, d\theta$$

Noticing that this complex integral will influence at most 0.03% of the final irradiance value, we decide the benefits aren't worth the cost of estimating such an expensive expression and we drop the term entirely.

Overall, the main takeaway here is that equation 18 really needs to be rewritten as:

$$\Delta E = L_d(\boldsymbol{x'})\ (1 - F_0) \int_{\theta_0}^{\theta_1} (\boldsymbol{n'}.\boldsymbol{\omega_i'}) \sin(\theta)\, d\theta \qquad (\mathbf{19})$$

The F_0 value is usually available from the G-Buffer as it is required for specular lighting computations. This result makes a lot of sense if you take the case of a metal for which the F_0 value is close to 1, it is known that metals have no diffuse component and, as such, weighing the final diffuse irradiance by 1-F_0 will indeed reduce the indirect diffuse lighting coming from metals to 0.

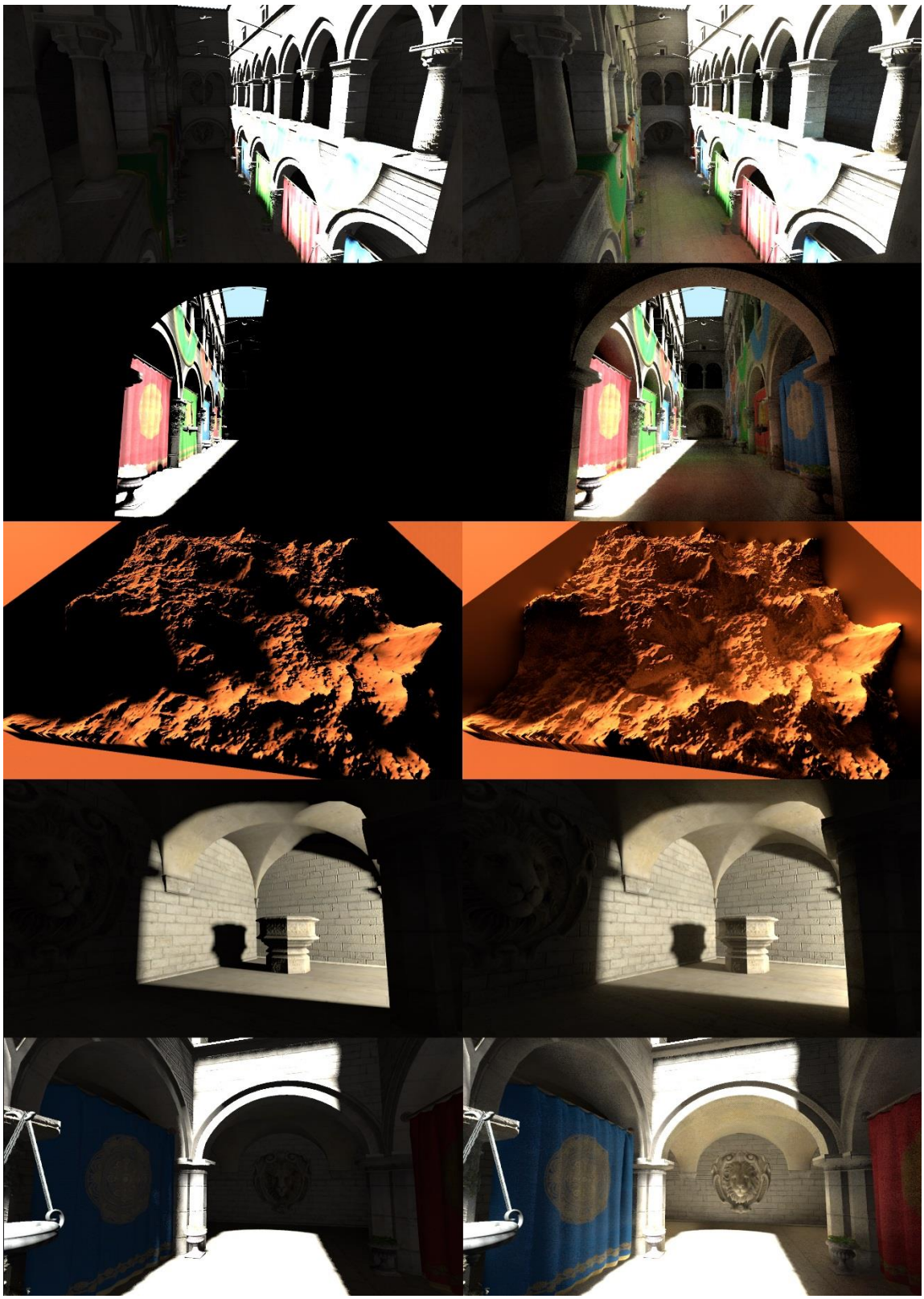

Figure 19. *The effect of the HBIL technique. Left: Direct only. Right: Direct + HBIL.*

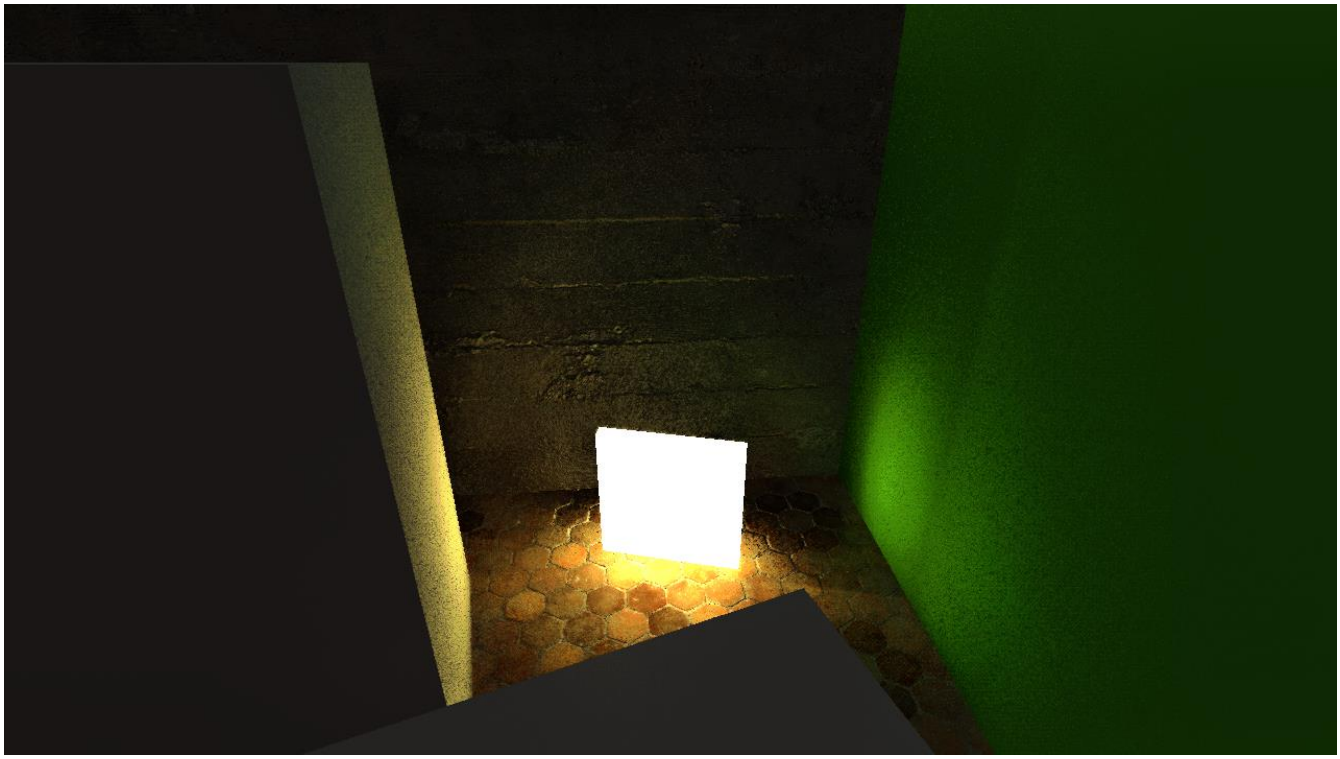

FIGURE 20. *AN ADDITIONAL BENEFIT OF THE TECHNIQUE: IT IS POSSIBLE TO INJECT EMISSIVE SURFACES INTO THE SOURCE RADIANCE BUFFER TO OBTAIN SIMPLE AREA LIGHTS.*

3. Integration with the lighting pipeline

Integrating HBIL into a deferred pipeline is not a lot of work but strongly relies on having an efficient Temporal Anti-Aliasing (TAA) technique to smooth out the noise introduced by the fact that only 2 screen-space directions are traced per-pixel, as we will see in section 3.3 about interleaved rendering.

The rendering pipeline for HBIL is shown below:

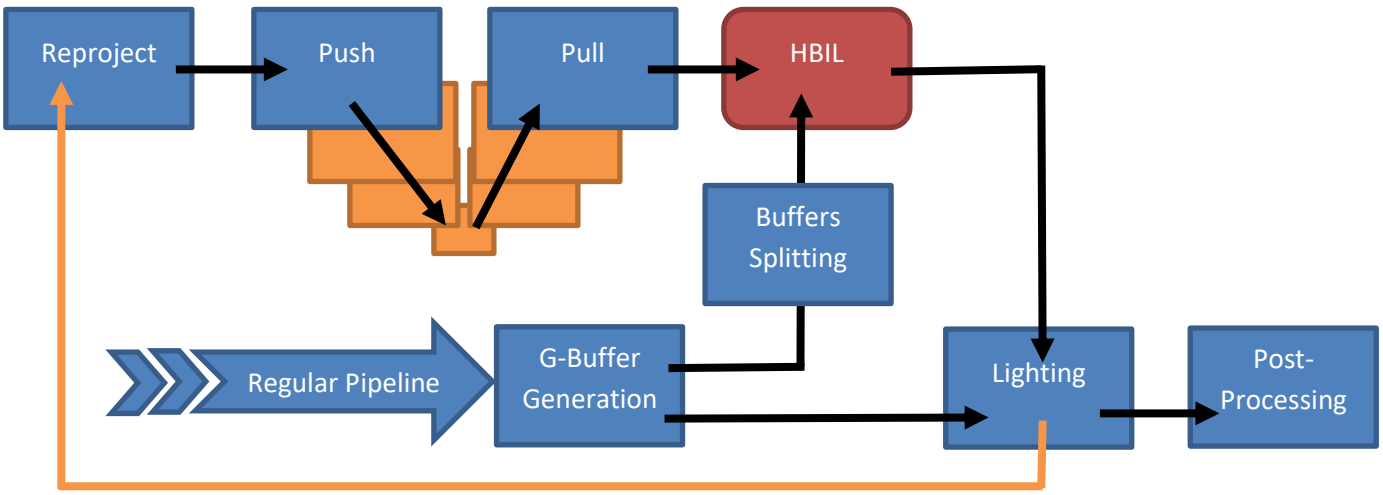

*FIGURE 21. INTEGRATION OF **HBIL** INTO A REGULAR DEFERRED-LIGHTING PIPELINE.*

3.1. Memory Requirements

Here is the list of G-Buffer fields required by the technique:

- Albedo **(RGB)**
- Normal **(RG)**, usually stored in the camera-space described in section 1.1.
  The 3rd component of the normal is then reconstructed via $z = \sqrt{1 - x^2 - y^2}$
- F0, or the equivalent "Metal" parameter (either **luminance** only or full **RGB**).

Here is the list of additional buffers required by the technique:

- Diffuse lighting **(Fullscreen, RGB, HDR, All Mips)**, used to store the diffuse part of the lighting that is re-projected and re-injected next frame (orange arrow in figure 21). Successfully tested with either RGB10A2 and R11G11B10 high-dynamic range formats.
- Push/Pull-Buffer **(Fullscreen, RGB, HDR, All Mips)**, serves as the 2nd ping-pong buffer (diffuse lighting buffer being the first one) to compute the push/pull chain after reprojection (cf. Section 3.2).
- 4x4 Split Radiance (**1/4 Resolution, RGB, HDR, No Mip)**, temporary quarter-res buffers where the reprojected/reconstructed radiance from last frame is stored (allocated as Texture2DArray)
- 4x4 Split Normal (**1/4 Resolution, R8G8_SNORM, No Mip)**, temporary quarter-res buffer where the G-Buffer normal is stored as HBIL input (allocated as Texture2DArray)
- 4x4 Split Depth (**1/4 Resolution, R16F, No Mip)**, temporary quarter-res buffer where the G-Buffer depth is stored as HBIL input (allocated as Texture2DArray)

- 4x4 Split Irradiance (**1/4 Resolution, RGB, HDR, No Mip)**, temporary quarter-res buffer where we store the irradiance output by the HBIL (allocated as Texture2DArray)
- 4x4 Split Bent Cone (**1/4 Resolution, R8G8B8A8_SNORM, No Mip)**, temporary quarter-res buffer where we store the bent-cone output by the HBIL (allocated as Texture2DArray)

3.2. Reconstructing Last Frame Radiance

I'm using a compute shader to scatter last frame's radiance into current frame's point of view. Due to the nature of the scatter process, the resulting image has holes (disocclusion events) and pixels that fight to scatter to the same position (occlusion events), as seen in figure 22.

The occlusion events are taken care of by simply using a temporary Z-buffer to prioritize last frame's reprojected Z values and accept only the reprojected radiance closest to this frame's camera.

The disocclusion events, on the other hand, appear as holes that need to be filled. We use the push/pull algorithm described in [8] to compute mips from valid radiance values, then pulling these newly computed smooth values to the higher mips again and using them whenever a radiance value is unavailable.

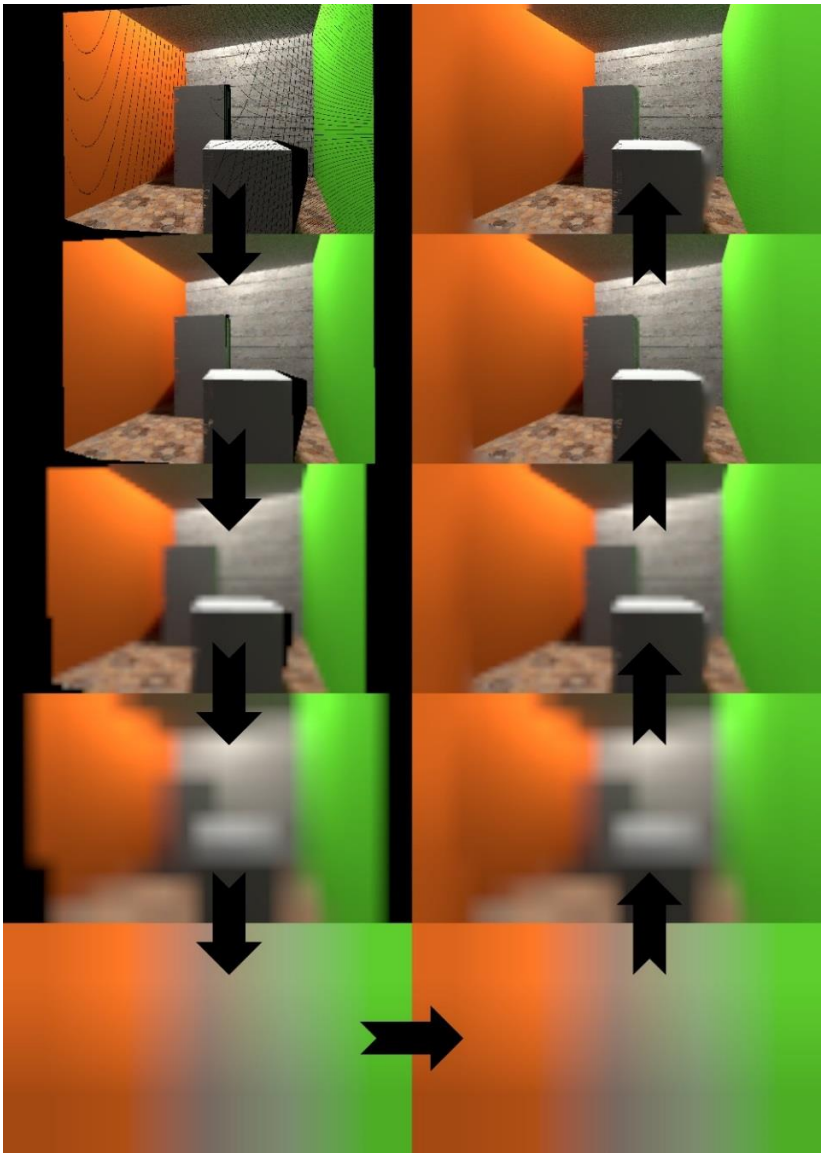

Figure 22. *Top-left: raw reprojected radiance with holes. top-right: Final result after push/pull reconstruction. The raw reprojection goes through the entire process of pushing valid values down to the mips, then pulling the newly computed values back again up to the final result where holes are filled.*

Of course, it's also entirely possible to entirely bypass the forward (i.e. scattered) reprojection + push/pull reconstruction and simply use a gathering technique with screen-space motion vectors that are often readily available with Temporal Anti-Aliasing algorithms. The source radiance from last frame would then be "gathered" instead of "scattered".

3.3. Interleaved Rendering

Ideally, we would like to split the integral over the hemisphere from equation 1 into as many slices as possible for each pixel, and use as many samples as possible. In reality, taking many texture taps from fullscreen render targets is largely too expensive and quickly ruins cache coherence, making overall performance very poor.

The core optimization for the HBIL algorithm to be tractable is of course to use interleaved rendering as described by Bavoil et al. in [2] and [6].

The idea is to separate the many slices used for integration into only a few slices per pixel that are slightly offset in rotation from one pixel to the next, all the while computing the samples into smaller render targets, thus drastically improving cache coherence.
In my current implementation of HBIL, I split the buffers into quarter-resolution buffers and interleaved rendering is computed in small 4x4 squares so, even though only 2 hemisphere slices 90° from each other are used per pixel, when merged together, the Monte-Carlo integration is actually using 4x4x2 = 32 slices.

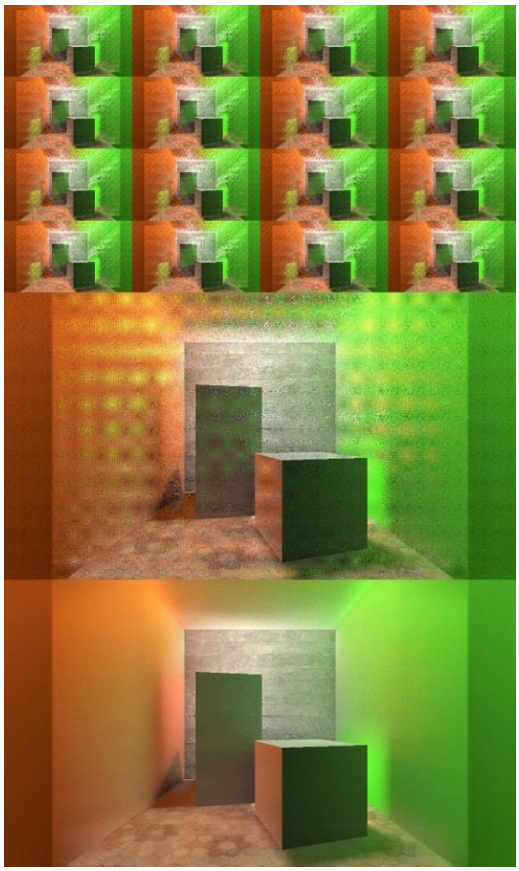

FIGURE 23. *TOP: MOSAIC OF 4X4 INTERLEAVED RENDERINGS OF THE IRRADIANCE BUFFER. MIDDLE: RAW RECOMPOSED RESULT. BOTTOM: RECOMPOSED RESULT AFTER TEMPORAL FILTERING.*

It is certainly possible to obtain a smoother result by recomposing the final buffer with a bilateral spatial filter that uses neighbor samples as well, at the cost of using more bandwidth and reducing cache efficiency once again. This is left as future research.

3.4. Depth Filtering

Care must be taken when sampling the radiance buffer in order to reject samples that clearly don't contribute to the lighting. The only true valid criterion I could find is somewhat expensive as it requires sampling the normal buffer on top of existing depth and radiance buffers, but it has strong geometric grounds that make it robust as it doesn't involve too many "free parameters" (rather a once for all setup that "looks good™" for all scenes).

You can see in the figure below the need for such a filter:

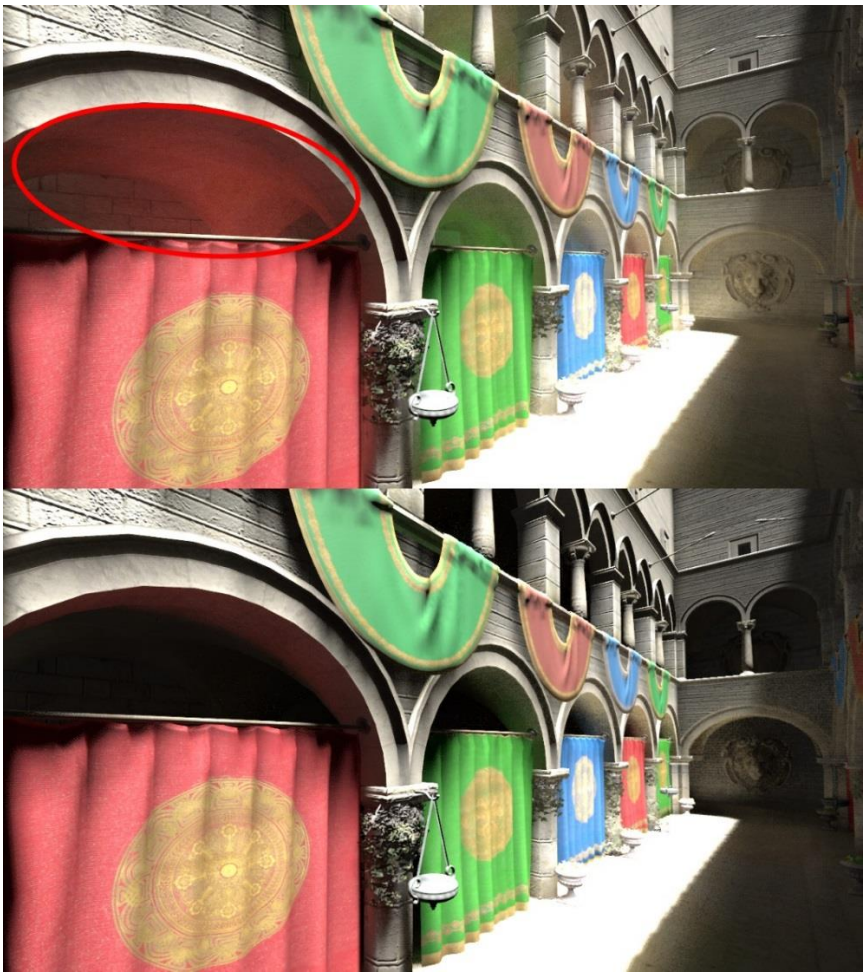

FIGURE 24. TOP: UNFILTERED RESULT WHERE INCORRECT SAMPLES ARE USED, LEADING TO A PLEASING ALTHOUGH WRONG LIGHTING UNDER THE ARCHES. BOTTOM: CORRECTLY FILTERED OUT RESULT.

The samples we want to reject are the ones facing away from our central location, as seen in the figure below:

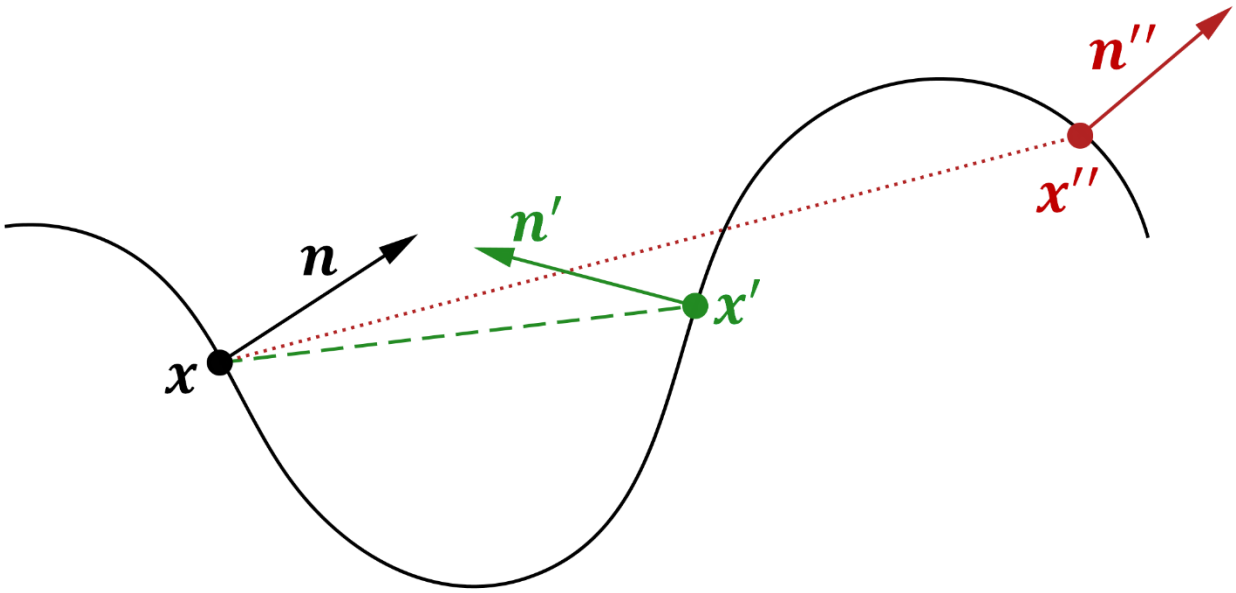

FIGURE 25. REJECTION FILTER IS BASED ON WHETHER A POINT IS FACING THE CENTRAL LOCATION $\boldsymbol{x}$ WHERE THE INTEGRAL IS TAKING PLACE. POINT $\boldsymbol{x'}$ WITH NORMAL $\boldsymbol{n'}$ IS A VALID CANDIDATE WHILE POINT $\boldsymbol{x''}$ WITH NORMAL $\boldsymbol{n''}$ IS CLEARLY NOT CONTRIBUTING THE RADIANCE PERCEIVED BY $\boldsymbol{x}$.

In figure 24 (top) we notice the pixels in the area circled in red will attempt, during screen-space sampling, to use the colors from the bright red drape below whereas the drape is clearly facing away from these pixels and doesn't contribute at all to their lighting.

So, although the depth samples from the drape should participate to the increase in ambient occlusion, the radiance samples, on the other hand, are filtered with the criterion:

$$t = smoothstep(\, a, b, -\boldsymbol{n}' \cdot \boldsymbol{\omega_i})$$

With $\boldsymbol{n}'$ the normal at the neighbor's location and $\boldsymbol{\omega_i} = \frac{\boldsymbol{x}' - \boldsymbol{x}}{\|\boldsymbol{x}' - \boldsymbol{x}\|}$ the vector pointing from the central location $\boldsymbol{x}$ to the neighbor's location $\boldsymbol{x}'$. a and b are tolerance values set so that the fade value t goes to 0 when the dot product is too negative.

Then the radiance value can be updated as:

$$L_{new} = lerp(\, L_{old}, L(\boldsymbol{x}'), t\,)$$

Where L_{old} is the radiance value from the previous sample that we keep in an "*inout*" register that keeps getting updated, $L(\boldsymbol{x}')$ is the radiance that we sampled from the source radiance buffer. We can see the parameter t serves as a fallback to a "safe" radiance value we used for an earlier sample.

3.5. Performance

Performance measurements were done on a NVIDIA 680 GTX, 2GB of DDR5 Video Memory, Driver version is 388.13.
Current version of the algorithm uses **2 sampling slices per pixel** with **16 samples** per direction at **1280x720**.
Time measurements are averaged over many scenes.

Pass Name	Time (ms)
Reprojection (opt.)	0.16
Push/Pull (opt.)	0.32
Buffers Split	0.78
HBIL	1.97
TOTAL (with optional push/pull phase)	3.23
TOTAL (core HBIL only)	**2.75**

4. Conclusion & Future Work

This paper presented new improvements over the existing horizon-based algorithms and showed how to make the most out of the information gathered during the horizon sampling phase. This additional information allows us to produce the very important bent-normal direction as well as gathering the indirect radiance from our samples, which drastically enhances the overall indirect lighting of the scene.

There are of course limitations due to the screen-space nature of the method, like large sources of lighting disappearing when off-screen, but I found out the fading effect due to the progressive reduction in surface due to camera movement usually masks this effect well. In a future work, it would be interesting to keep these off-screen values in some sort of special "off-screen buffer", during the re-projection phase for example. This off-screen buffer could possibly be a paraboloid-projection of the half-space behind the camera, that would accumulate values as the camera stands still and rotates, but would fade away as soon as it starts a translation movement (invalidating the cached off-screen environment).

Due to the small amount of sampling directions per pixel, additional work is needed to further reduce the noise that is produced.

An interesting and very promising line for future research would be to further exploit the dependence on the F_0 Fresnel term to "focus" the sampling directions into the "rough" specular direction, the idea would be to merge the HBIL and specular screen-space reflection into a single algorithm, thus avoiding the cost of tracing the screen twice, once for the diffuse reflection, another time for the specular reflection.

Overall, although it is not the definitive answer to the difficult problem of global illumination, HBIL provides a significant enhancement to the indirect lighting part of an image and constitutes a nice extension to far-field indirect lighting techniques like the wildly-used probe-based solutions.

5. Acknowledgments

Special thanks to Eko Software for financing this research. HBIL will appear in the next Eko Software ARPG game based on Warhammer Fantasy Battles IP.

My thanks go to Stéphane Le Boeuf for proof reading this paper, Benjamin Lalisse for his clever remarks and general support, Martin Gérard for his precious help with my math, Léon Denise for his beautiful 3D scenes, Geoffrey Rosin for his amazing textures, David Toyou, Laurent Tartinville for his help, and Sandra for moral support 😊.

6. References & Bibliography

[1] Bavoil, L. and Sainz M. 2008. "Image-Space Horizon-Based Ambient Occlusion"

[2] Bavoil, L. and Jansen, J. 2013. "Particle Shadows & Cache-Efficient Post-Processing"

[3] Jimenez, J. Wu, X-C. Pesce, A. and Jarabo, A. 2016. "Practical Realtime Strategies for Accurate Indirect Occlusion"

[4] Mayaux, B. 2018, "Improved Ambient Occlusion"

[5] Mayaux, B. 2016, "Spherical Harmonics Irradiance Estimate for a Cone" (personal blog)

[6] Bavoil, L. 2014, "Deinterleaved Texturing for Cache-Efficient Interleaved Sampling"

[7] Ramamoorthi, R. 2001, "On the Relationship between Radiance and Irradiance: Determining the illumination from images of a convex Lambertian object"

[8] Kraus, M. 2009, "The Pull-Push Algorithm Revisited"

[9] Vidiger, D. 2017 "SSRTGI: Toughest Challenge in Real-Time 3D"

McGuire, M. 2011, "The Alchemy Screen-Space Ambient Obscurance Algorithm"

McGuire, M. 2012, "Scalable Ambient Obscurance"