**Horizon-Base Indirect Lighting**

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

February 2018 – Benoît “Patapom” Mayaux  
b.mayaux@gmail.com

In the following document, vectors will be written in boldface characters (e.g. **,** ) and scalar values will use regular characters (e.g. , ).  
The spherical coordinates used throughout the paper have an elevation angle aligned with the vertical **Z** axis, and an azimuthal angle lying in the tangent plane measured from axis **X**:

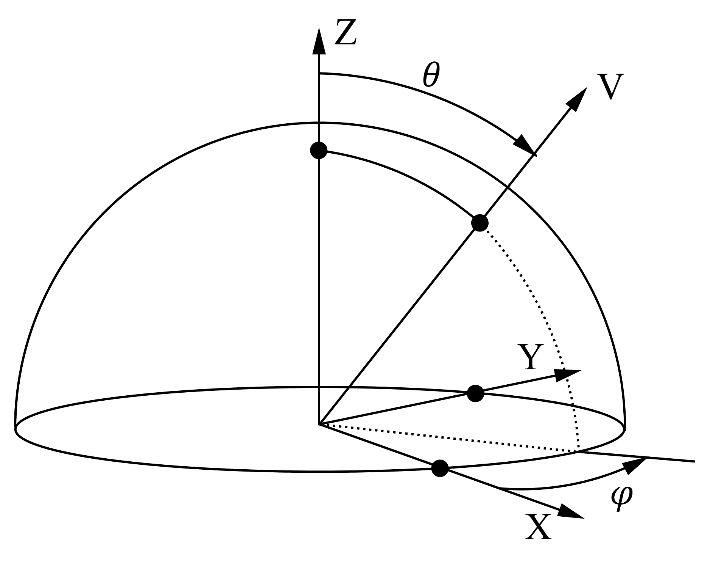


Fig. 1. The frame for spherical coordinates used in this paper.

1. Introduction

The Horizon-Based Ambient Occlusion (HBAO) technique introduced by Bavoil et al. [[1](#REF_1)] proposed to improve the computation of the Ambient Occlusion integral by skipping all the rays that we know for sure would intersect the heightfield/depth buffer and thus wouldn’t contribute to the visibility term:

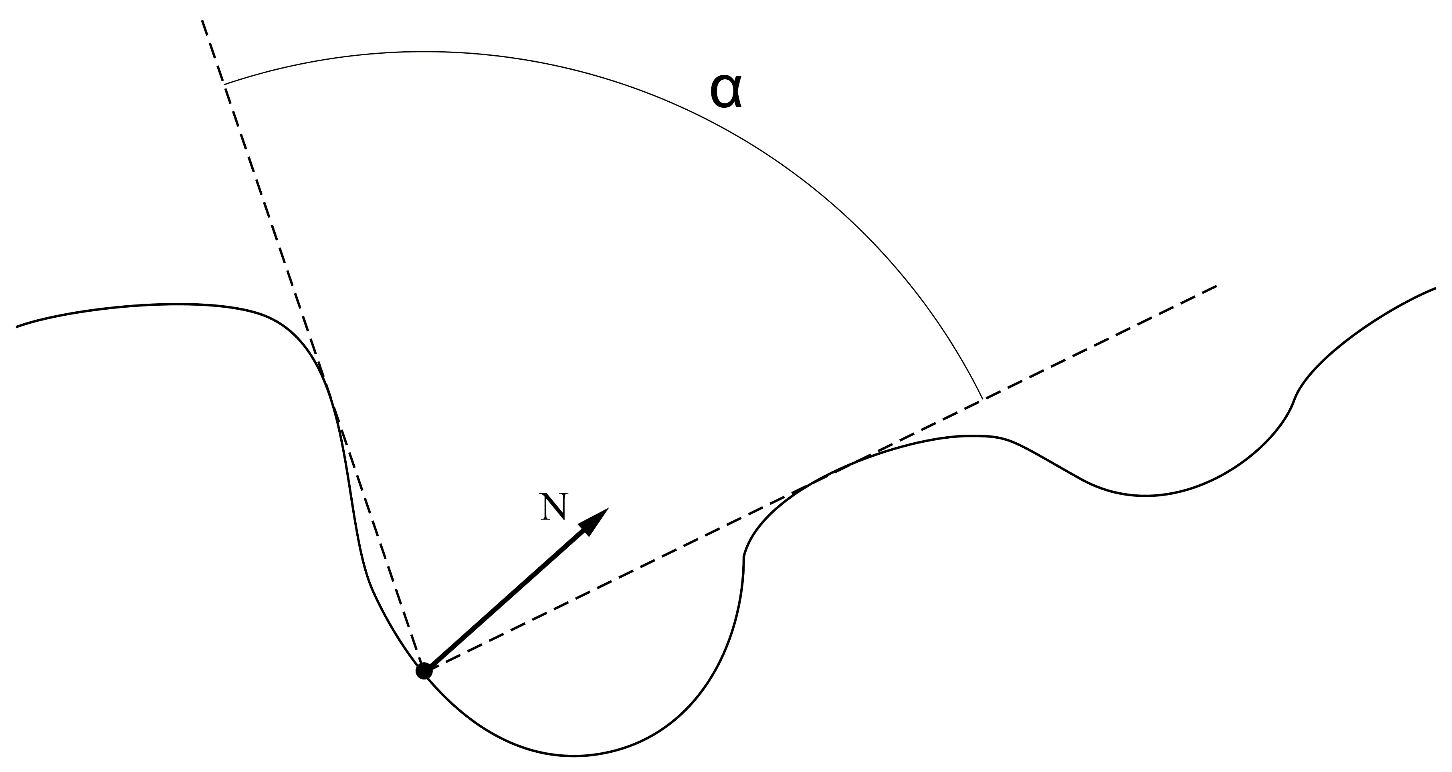


Fig. 2. The HBAO algorithm is exploiting the full visibility of the horizon cone of angle 𝛼:  
 we know all the rays inside the cone escape the surface of the heightfield and are the only ones actually contributing to the visibility term.

The ambient occlusion term is then simplified into:

Where:

* is the location of the pixel for which we are calculating the AO
* is the incoming ray direction
* is the upper hemisphere of directions whose solid angle is 2π
* is the portion of solid angle covered by the vector
* is the visibility term returning 1 if the ray escapes to infinity, 0 if the ray intersects the heightfield
* is the horizon angle in azimuthal direction

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

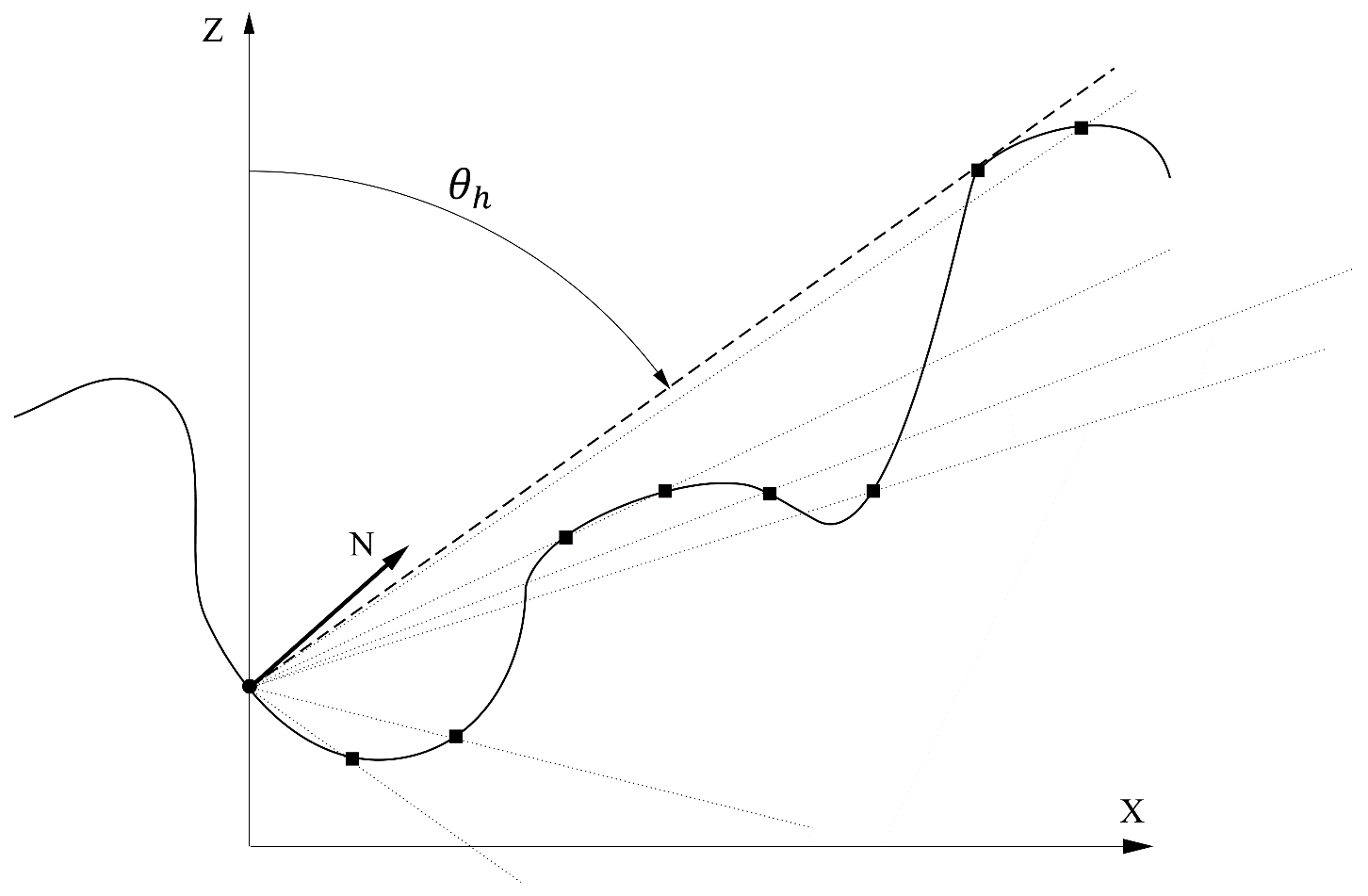


Fig. 3. The heightfield is sampled along the X direction and the horizon is updated along the way by reducing the horizon angle each time.

The portion of AO computed by the inner integral is:

And so, the final expression for the AO becomes:

# Back & Front Sampling

In this paper, to avoid doing the same work twice, we will assume that we always trace the front and back parts of the slices of the tangent-space disk at the same time:

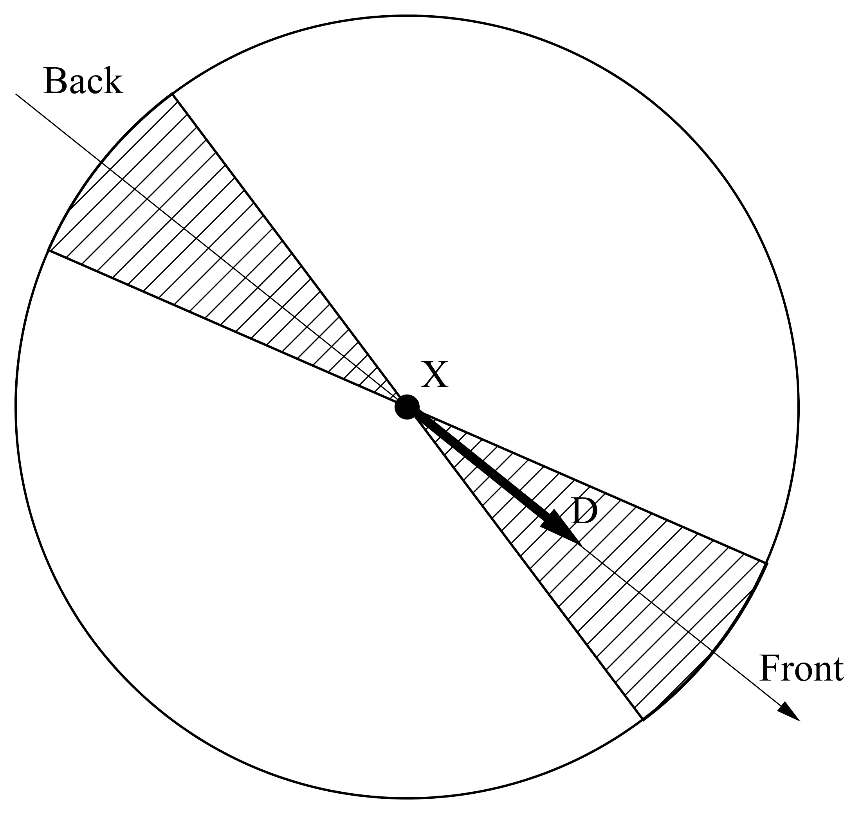


Fig. 4. The tangent-Space disk is sampled both in the forward and backward direction

Estimating the AO in such a simultaneous back & front manner simply becomes:

# Local Camera Space

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 :

Where:

* is the normalized view vector pointing from our world-space location toward the camera
* is the world-space camera position
* is the world-space camera up vector

From which we obtain:

* the new local camera-space “right” vector
* the new local camera-space “up” vector

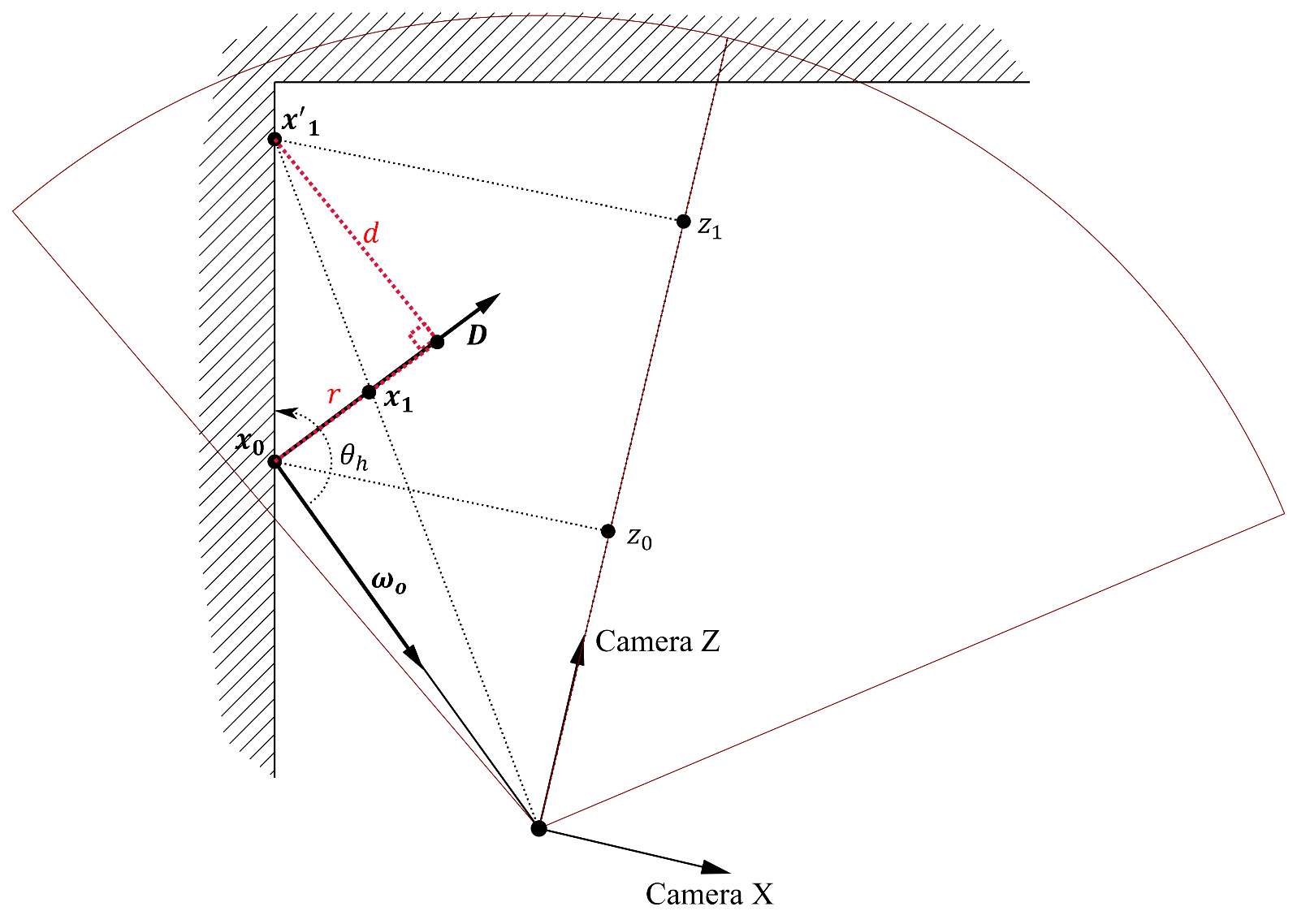


Fig. 5. The important difference between the “global” camera space and the “Local” camera space reconstructed from point and camera view vector .  
Moving to a neighbor location implies some reprojection steps to obtain , and used to compute the new horizon angle . These steps are described in section 2.1.2.

This new local camera space will be used to express the normal, the horizon angles and later, the bent cones.

1. Improvements over HBAO

In this paper, we will propose several techniques to make the most out of the information gathered while performing the HBAO algorithm:

* The first section will discuss how to use the camera-space normal, if it’s available, to make the result more robust.
* The second section will discuss how to obtain the bent-cone from our samples  
  The generated bent-cone buffer will then replace the normal buffer and become a very efficient tool to compute a better scene lighting.
* Finally, the third section 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 to improve the scene lighting.2.

# 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 camera-space normal vector is often available from the G-Buffer produced by most deferred renderers.

The gray areas of the tangent-space disk from figure 4 are approximated by a “slice” rotating about the axis of the local camera-space described earlier:

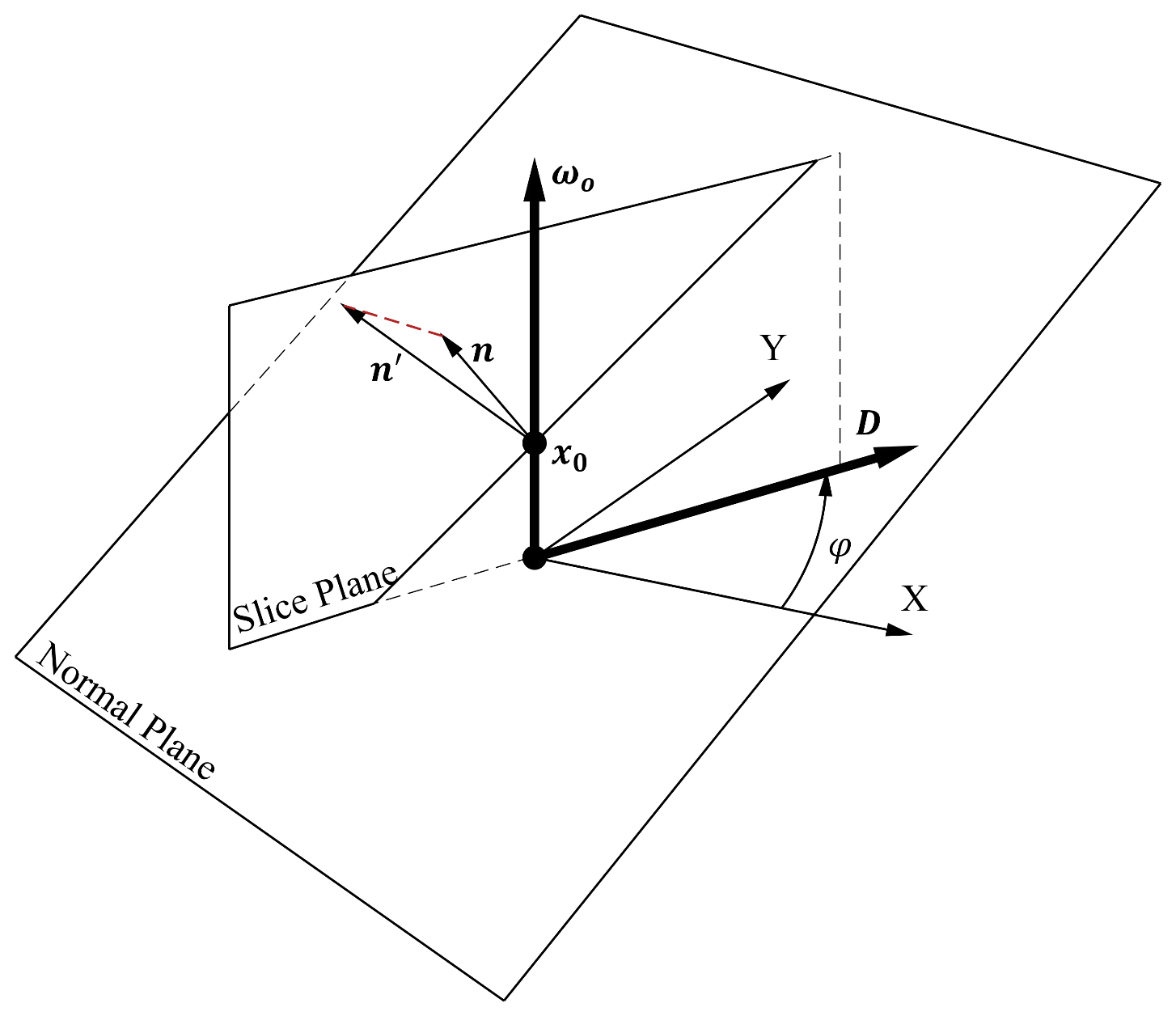


Fig. 5. a camera-space slice rotating by an angle about the axis pointing toward the camera**.**  
We point out the important fact that the normal is not necessarily lying in the plane of the slice and will need to be projected onto the slice, yielding the  vector.

## Initial Horizon Angles

We initialize the horizon angles for the slice by computing the projection of the vector onto the normal plane by following the axis:

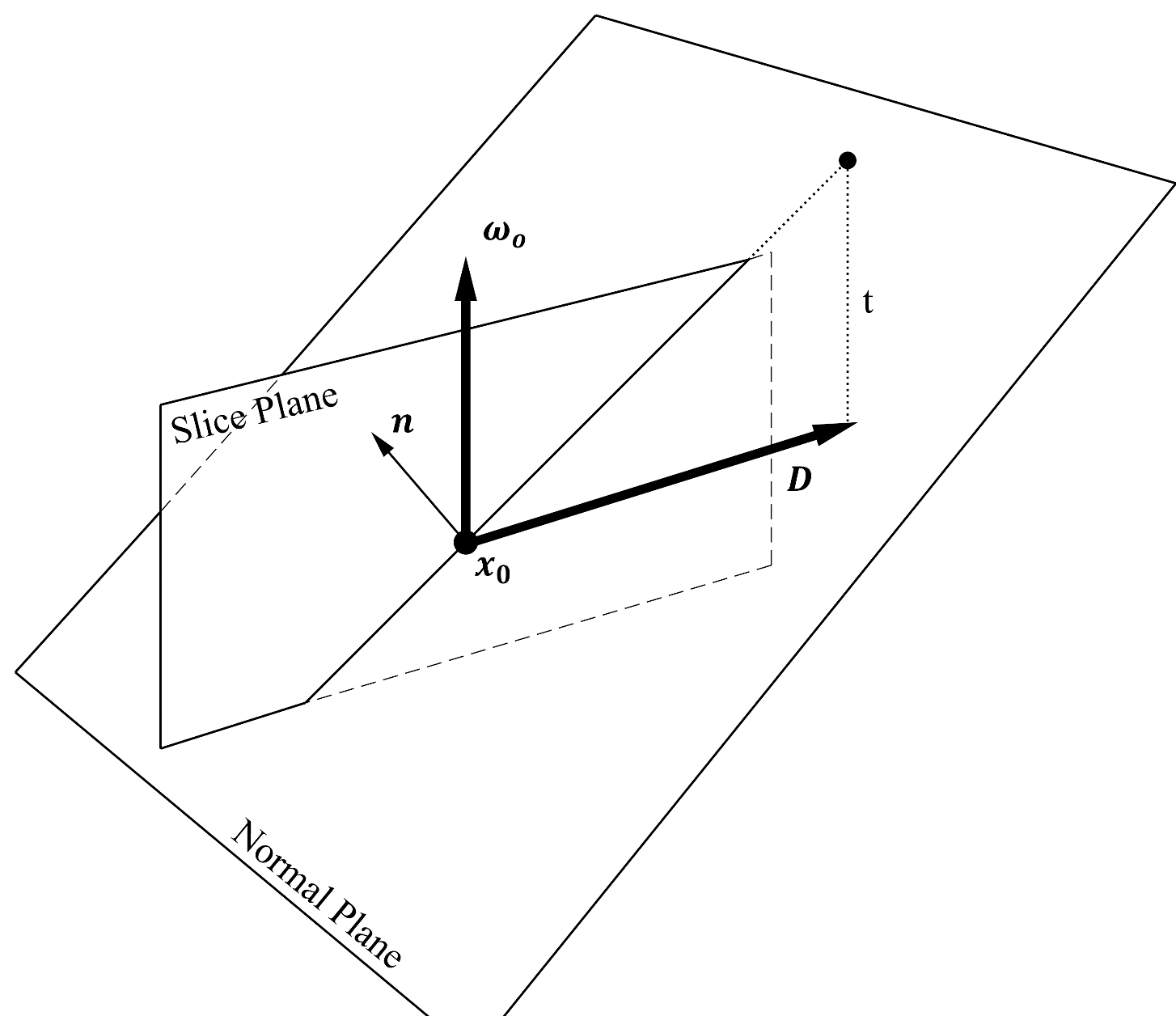


Fig. 6. Projection of the unit vector onto the normal plane following the axis to find the initial “minimal” horizon angles for the slice.

Where:

* is the slice’s azimuthal direction expressed in local camera-space
* is some azimuthal angle in the tangent plane of the new local camera-space
* is the view direction expressed in local camera-space
* is the normal expressed in camera-space
* is the intersection distance with the normal plane
* and are respectively the initial front and back horizon angles

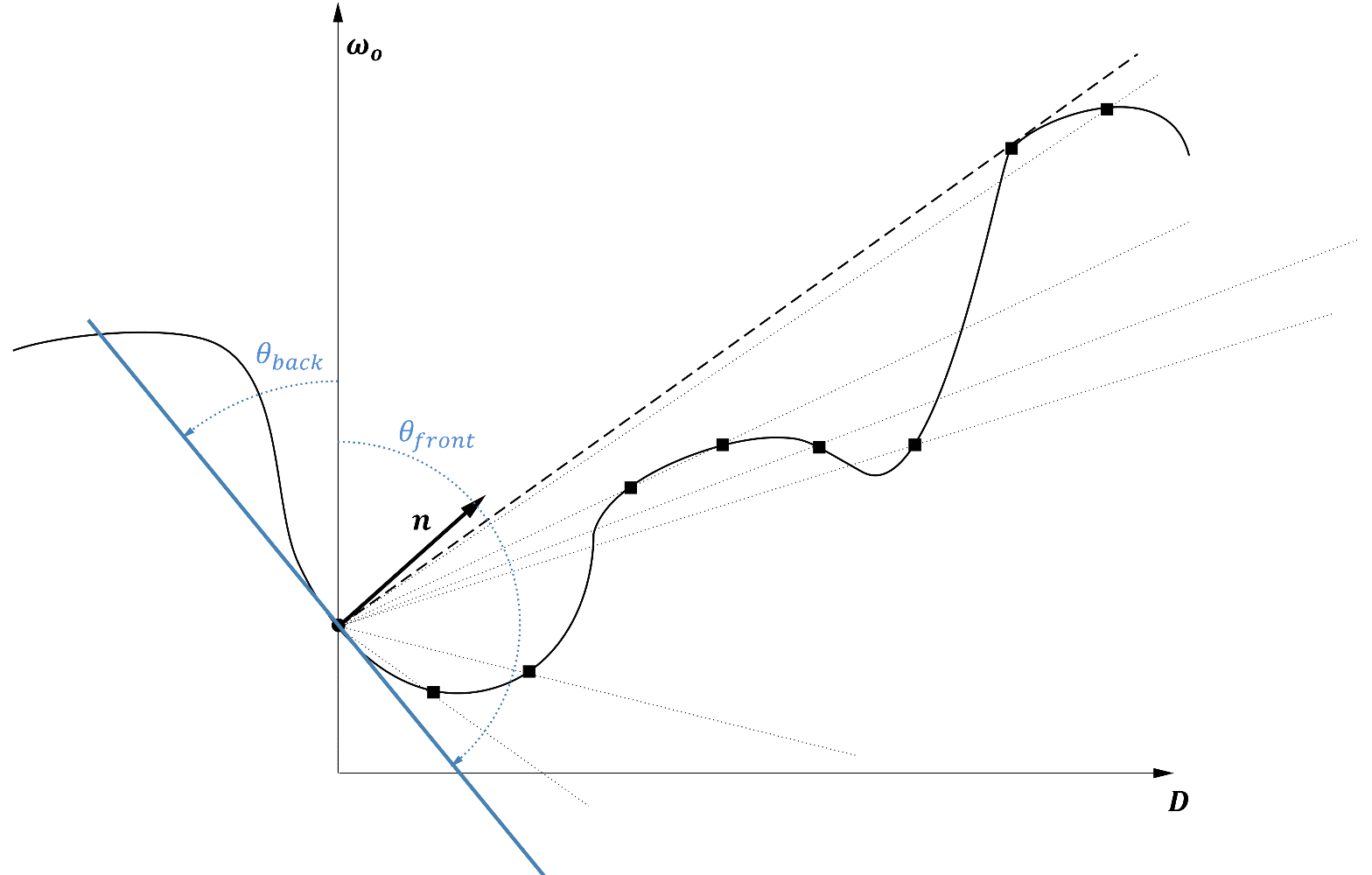


Fig. 7. Using the normal to determine and as our initial horizon values for each slice of the camera-space sampling disk.

**TODO: Show ON/OFF image**

## Updating the Horizon Angles

In figure 5 we saw that when we move from our central location to the neighbor location by following the world-space vector for a small distance we get**:**

If we sample the depth-buffer at the screen-space location corresponding to this new location , it will give us depth from which we obtain the new world-space location that we can finally express back into the local camera space:

From which we can finally obtain the cosine of the horizon angles that we will use all along this paper:

Where:

* is the measured elevation angle for the neighbor sample at
* is the horizon angle we keep updating as we move along

Of course, the horizon angles are now expressed off from the local camera axis and for each slice of the camera-space sampling disk, we are given the two horizon angles in the form of their cosine and with and :

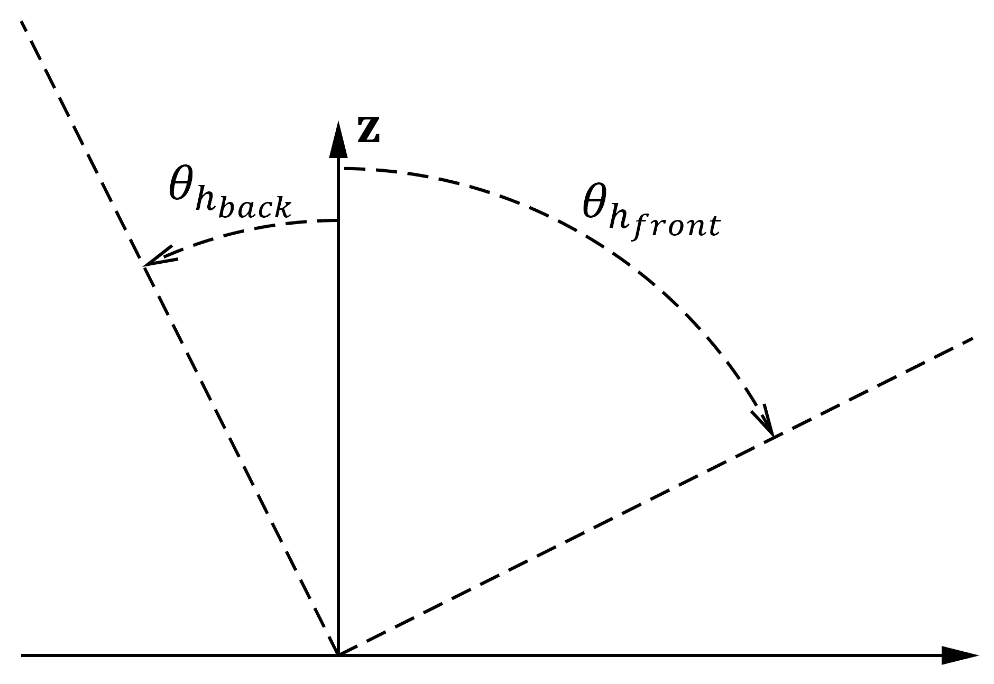


Fig. 8. Slice-Space horizon angles

# Bent Cones

We follow the methodology of the GTAO computation as described by Jimenez et al. [[3](#REF_3)] but we will not compute the ambient occlusion, rather we are interested in 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:

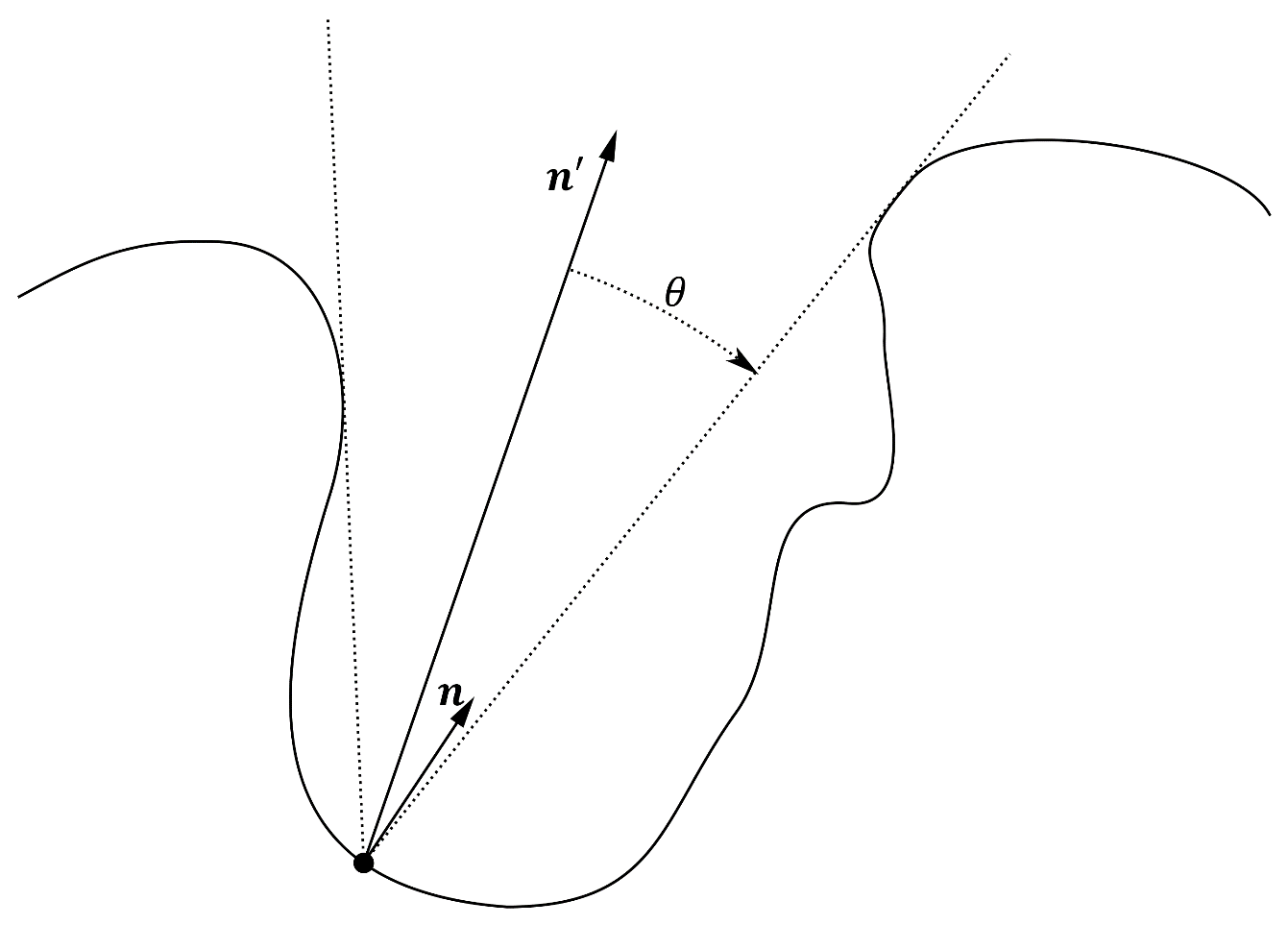


Fig. 9. The “Bent-Cone”, a bent normal with an angle.

In order to get the bent normal, we need to compute the average direction of a vector weighted by the cosine of the angle with the normal (since directions at grazing angles don’t contribute much to the bending) and unobscured by the heightfield:

Where:

* is the resulting average “bent normal”
* N is the amount of angular samples
* is the direction of the ith sample
* is the direction of the normal to the surface

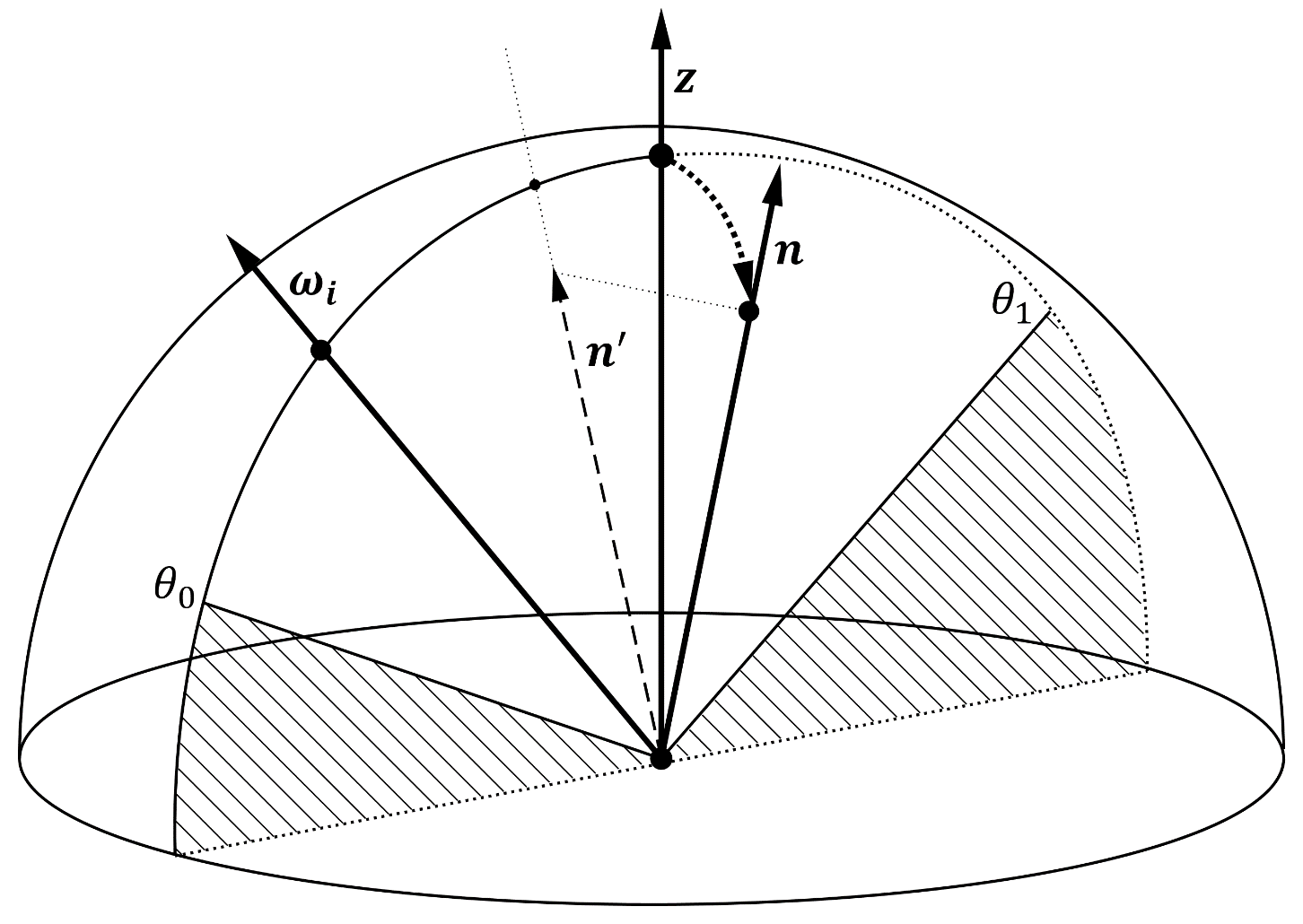


Fig. 10. Integration of the vector in the slice between angles and weighted by the dot product with normal vector . Notice that the normal vector is not necessarily lying in the slice’s plane.

We can compute the exact integral for equation (2) in our 2D “slice space” by writing:

Equation 3 can be expanded into:

Identically, equation 4 gives:

We can then rebuild a camera-space normal:

# Indirect Lighting

The classical lighting equation to compute the outgoing radiance from a pixel at in viewing direction is essentially given by:

Where:

* is the incoming radiance at from direction
* is the surface’s BRDF
* is the surface normal
* is the set of all directions covering the upper hemisphere
* is the solid angle covered by the surface perceived along direction

Focusing only on diffuse Lambertian reflection, we can rewrite eq. (1) as:

Where:

* is the *irradiance* arriving at surface location and normal **.**
* represents the diffuse BRDF for a surface with albedo . The division by is here to guarantee energy conservation since .  
  NOTE: although a RGB quantity will be noted simply in the rest of the document

# Acknowledgments

Special thanks to Eric Arnebäck for proof reading this paper, Benjamin Lalisse for his clever remarks, Martin Gérard for his precious help with my math, [Geoffrey Rosin](https://www.artstation.com/kikette) for his amazing concrete texture, and Sandra for moral support 😊.

# References

*[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] Cook, J. D. “*[*Accurately computing running variance*](https://www.johndcook.com/blog/standard_deviation/)*”*