Deriving the analytical formula for a diffuse response from a polygonal light source

Michał Iwanicki

michal.iwanicki@activision.com

Introduction

It's been known since the 18th century that the diffuse response from a polygonal light source can be computed analytically [1]. [2] intoduced it to computer graphics and recently, with a clever trick, it has been extended to include response from different BRDFs (see [3]), but the method still relies on the same formula as the diffuse case.

Most source mention just the final form of the formula. The derivation is an interesting application of some more advance theory from calculus. Below is the step-by-step explanation where the final result comes from.

Assumptions

Let's assume that we have a flat polygonal light source emitting constant, unit, radiance in all directions. The shape of the polygon doesn't have to be convex, but its edges cannot intersect. Let's call the polygon P and its vertices $p_0, p_1, ..., p_n$. Let's assume that P is entirely in the positive Z half-space and we will be computing the diffuse response at the origin $O = (0, 0, 0)$ assuming that the normal direction points along the positive Z axis $\vec{n} = (0, 0, 1)$

What we want to compute is:

$$I = \frac{1}{\pi} \int_P cos(\vec{n}, \vec{\omega}) dA \tag{1}$$

so the integral of the cosine of the angle between the normal vector and the vector pointing to the differential area dA over the area of P. The $\frac{1}{\pi}$

factor comes from normalization - so that when integrating over the entire hemisphere we get 1 (technically this is the diffuse BDRF - energy preserving and having unit of 1/steradian).

Instead of integrating over the area of the polygon, we can also integrate over the solid angle subtended by the polygon - which in turn is the same as integrating over the area of a spherical polygon P' that's a projection of P onto a surface of the unit sphere. The vertices of P' will be marked $p'_0, p'_1, ..., p'_n$. Using spherical polygon might seem weird at first, but it simplifies a bunch of things later on (for instance the cosine of the angle spanned by the edge of P' is simply a dot product of p's and doesn't need any normalization)

Stokes' theorem

The derivation is based on Stokes' theorem:

$$\int_{\Sigma} \nabla \times F \cdot d\Sigma = \oint_{\partial\Sigma} F \cdot dr \tag{2}$$

It might look intimidating at first, but it's actually not that bad. We start with a vector field F - function defined over 3d space with values being 3d vectors (of course, as usual, there's a bunch of assumptions about this function, but let's not worry about this for now) and some surface Σ with a boundary $\partial\Sigma$.

The left-hand side is an integral of a curl of the vector field F: $\nabla \times F$ (nothing to be scared of so far - curl is just an operator that takes a vector field and turns it into another vector field - the exact formula will be shown later) dotted with a surface normal $d\Sigma$ (just a plain, old, well known unit vector perpendicular to the surface) over the surface. So we just take the surface, compute the curl of F, compute the dot product between the normal to the surface and the curl at every point on the surface and sum this up.

The right-hand side is an integral of the function F but this time along the boundary of the surface Σ - so we take the function F again, at every point on the boundary we compute the dot product of F at that point with a vector along the boundary curve (we assume that the boundary is oriented in some consistent direction - so there's always know how to move "forward" along the curve and since it's a closed curve - it's a boundary of some surface after all - we will eventually get to the point where we started) and sum that up over the whole boundary.

Stokes' theorem states that those two integrals are equal. Instead of calculating integral of some function over a surface we can compute the integral

of slightly different function over the boundary of that surface and vice versa.

This is the exact trick that we'll use for the diffuse response integral - we will replace the integration over the surface of the spherical polygon P' with the integration over its boundary.

The plan is as follows:

- figure out the vector field when put into the left hand side of Stokes' theorem (2) will give us the integral from (1)
- figure out how to compute the integral of that vector field along the boundary of the light source polygon

Vector field and the surface integral

Easy things first: we want the integral on the left-hand side of (2) to look like our diffuse response integral. They are both surface integrals - like mentioned before the integral in (1) can be thought of as integral over the surface of the polygon projected onto a unit sphere. That projected spherical polygon P' will be our Σ surface in Stokes' theorem.

The integral in the Stokes' theorem uses the dot product between the surface normal and the curl of the vector field. Our surface is on a unit sphere, the normal at any point is just a vector from the origin to that point - which happens to be the same as $\vec{\omega}$ in (1). Since the dot product of two unit vectors is a cosine of the angle between them, if we find a vector field F that has a curl equal to $(0, 0, 1)$ at all point at the surface of the unit hemisphere, the left-hand side of the Stokes' theorem will look exactly the same as the integral in (1).

Given vector field $F = (F_x, F_y, F_z)$ (where F_x, F_y, F_z are scalar valued functions $R^3 \to R$, representing the three components of the vector field) the curl is define as follows:

$$
\begin{aligned}
(\nabla \times F)_x &= \frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z} \\
(\nabla \times F)_y &= \frac{\partial F_z}{\partial x} - \frac{\partial F_x}{\partial z} \\
(\nabla \times F)_z &= \frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}
\end{aligned}
$$

Where $\frac{\partial F_i}{\partial j}$ is the (partial) derivative of the i-th component of the vector field with respect to variable j.

We want our function to have the curl equal to $(0, 0, 1)$ over the surface of the unit sphere. We can create multiple different functions with such properties, but to make it simpler, we'll create one that has the curl equal to $(0, 0, 1)$ everywhere in the domain.

We're looking for three functions of (x, y, z) such that:

$$\begin{aligned}\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z} &= 0 \\ \frac{\partial F_z}{\partial x} - \frac{\partial F_x}{\partial z} &= 0 \\ \frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y} &= 1\end{aligned}$$

Simplest solutions are always good, and the simplest functions I could think of here are:

$$\begin{aligned}F_x(x, y, z) &= 0 \\ F_y(x, y, z) &= x \\ F_z(x, y, z) &= 0\end{aligned} \tag{3}$$

with the partial derivatives being:

$$\begin{aligned}\frac{\partial F_x}{\partial x} &= 0 \quad \frac{\partial F_x}{\partial y} = 0 \quad \frac{\partial F_x}{\partial z} = 0 \\ \frac{\partial F_y}{\partial x} &= 1 \quad \frac{\partial F_y}{\partial y} = 0 \quad \frac{\partial F_y}{\partial z} = 0 \\ \frac{\partial F_z}{\partial x} &= 0 \quad \frac{\partial F_z}{\partial y} = 0 \quad \frac{\partial F_z}{\partial z} = 0\end{aligned}$$

and the curl equal to:

$$\begin{aligned}(\nabla \times F)_x &= 0 \\ (\nabla \times F)_y &= 0 \\ (\nabla \times F)_z &= 1\end{aligned}$$

Now the left hand side of (2), with $\vec{n} = (0, 0, 1)$ and Σ (the surface) being our projected spherical polygon P', becomes:

$$\int_\Sigma \nabla \times F \cdot d\Sigma = \int_\Sigma \vec{n} \cdot d\Sigma = \int_{P'} cos(\vec{n}, \vec{\omega}) dA \tag{4}$$

which is exactly what we need for (1)

The boundary integral

We now need to figure out what to do with the right hand side of (2) - the integral of the vector field along the boundary of our polygon. First the obvious - our surface is a polygon and integral is linear - so the integral along the whole boundary will be equal to the sum of integrals along individual edges of the polygon:

$$\oint_{\partial\Sigma} F \cdot dr = \sum_i \int_{e'_i} F \cdot dr \tag{5}$$

where e'_i is the edge between points p'_i and p'_{i+1}.

The integral of a vector function along a curve, like the ones above, is actually an integral of a dot product of a vector tangent to the curve and the value of the function. If the curve is parameterized in some way - so we have a function $g : R \to R^3$ which takes some scalar argument and gives us a point on the curve for argument in range (a, b), such integral can be written as:

$$\int_{e_i} F \cdot dr = \int_a^b F(g(t)) \cdot \frac{dg}{dt}(t)dt \tag{6}$$

This way we replace the integration along the curve with integration over a scalar argument ($g(t)$ gives us point on the curve at t and $\frac{dg}{dt}(t)$ is the derivative of the parametrization - so it gives us vector tangent to the curve at t)

The edges of P' are arcs of great circles on the unit sphere (projection of a vertex of the original polygon P onto the surface of the unit sphere is finding the intersection of a ray from the origin to the given point with a surface of the sphere - so the plane spanned by two such rays passes through the origin too - and cuts the unit sphere into two equal hemispheres, forming a great circle). We can parametrize such curves with SLERP, which given the two endpoints of an edge will form a shortest path between them on the surface of the sphere as we change the interpolation control variable from 0 to 1.

SLERP has the form:

$$SLERP(p_i, p_j, t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)} p_i + \frac{\sin(t\Omega)}{\sin(\Omega)} p_j \tag{7}$$

where Ω is the angle between the vectors p_i and p_j.

To use it for calculating integrals along the edges we also need its derivative with respect to t:

$$\frac{dSLERP}{dt}(p_i, p_j, t) = -\Omega\frac{\cos((1-t)\Omega)}{\sin(\Omega)}p_i + \Omega\frac{\cos(t\Omega)}{\sin(\Omega)}p_j \tag{8}$$

(both SLERP and its derivative are of course $R \to R^3$ as p_i and p_j are 3d vectors)

We can now take (7) and (8) and put them in (6):

$$\int_a^b F(g(t))\cdot\frac{dg}{dt}(t)dt = \int_0^1 F(\frac{\sin((1-t)\Omega)}{\sin(\Omega)}p'_i + \frac{\sin(t\Omega)}{\sin(\Omega)}p'_j)\cdot(-\Omega\frac{\cos((1-t)\Omega)}{\sin(\Omega)}p'_i + \Omega\frac{\cos(t\Omega)}{\sin(\Omega)}p'_j)dt$$

Now we expand the dot product, using the definition of the vector field - because the vector field has a very simple structure, most of the components of the dot product disappear (the only non zero component of F is $F_y = x$ component - it gets multiplied by the y component of the derivative of the SLERP):

$$\int_{e'_i} F\cdot dr = \int_0^1 (\frac{\sin((1-t)\Omega)}{\sin(\Omega)}p'_i.x + \frac{\sin(t\Omega)}{\sin(\Omega)}p'_j.x)(-\Omega\frac{\cos((1-t)\Omega)}{\sin(\Omega)}p'_i.y + \Omega\frac{\cos(t\Omega)}{\sin(\Omega)}p'_j.y)dt$$

If you calculate the above integral (or make Mathematica do it for you...) you get:

$$\int_{e'_i} F \cdot dr = \frac{1}{2}(p'_j.x\, p'_j.y - p'_i.x\, p'_i.y + \frac{\Omega}{\sin(\Omega)}(p'_i.x\, p'_j.y - p'_i.y\, p'_j.x))$$

You might notice that when we put the above into the sum in (5) the $p'_j.x\, p'_j.y - p'_i.x\, p'_i.y$ terms are canceled between neighboring edges (the component with the negative sign becomes the one with positive sign for the previous edge). In the end we get:

$$\oint_{\partial\Sigma} F \cdot dr = \frac{1}{2}\sum_i \frac{\Omega_i}{\sin(\Omega_i)}(p'_i.x\, p'_j.y - p'_i.y\, p'_j.x) \tag{9}$$

where p'_j is $p'_{(i+1)\%n}$ and Ω_i is the angle between p'_i and p'_j vectors.

Putting it all together

Plugging (4) and (9) into (2) and then into (1) we get our final result:

$$I = \frac{1}{\pi}\int_{P'} cos(\vec{n}, \vec{\omega})dA = \frac{1}{2\pi}\sum_i \frac{\Omega_i}{\sin(\Omega_i)}(p'_i.x\, p'_j.y - p'_i.y\, p'_j.x)$$

References

[1] Johann Heinrich Lambert. *Photometria, sive de mensura et gradibus luminus, colorum et umbrae*. 1760.

[2] D. R. Baum, H. E. Rushmeier, and J. M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. *SIGGRAPH Comput. Graph.*, 23(3):325–334, July 1989.

[3] Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. Real-time polygonal-light shading with linearly transformed cosines. *ACM Trans. Graph.*, 35(4):41:1–41:8, July 2016.