3D Complexity Class Notes

Brandon Whitley

April 20th, 2006

Given an area light source and a potential occluder, we can visualize how to compute the occluder's umbra volume (Figure 1). The umbra is the set of all points that receive no light at all (or are completely occluded).

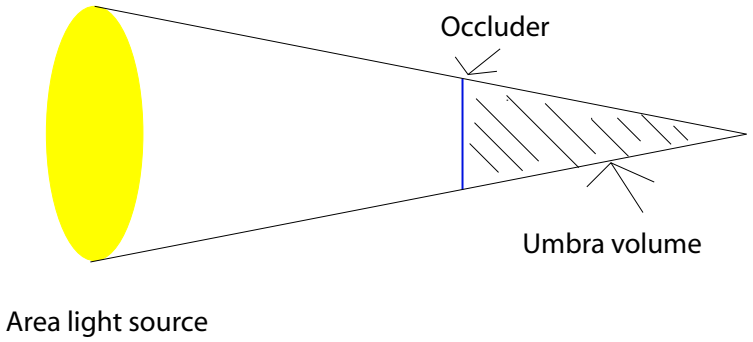

Figure 1: Computing umbra volume.

But what if there are many potential occluders (Figure 2)?

How do we compute their combined umbra? A nice easy solution would be to just take the union of their individual umbras. But we can show that this is not sufficient (Figure 3).

So this is a more complicated problem. We do know, however, that the union of the individual umbras is at least a subset of the actual combined umbra.

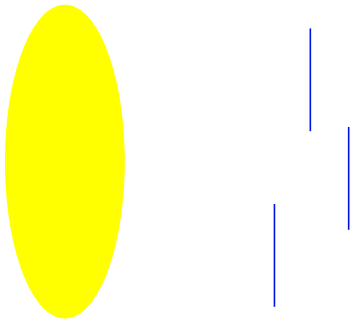

Figure 2: Multiple occluders.

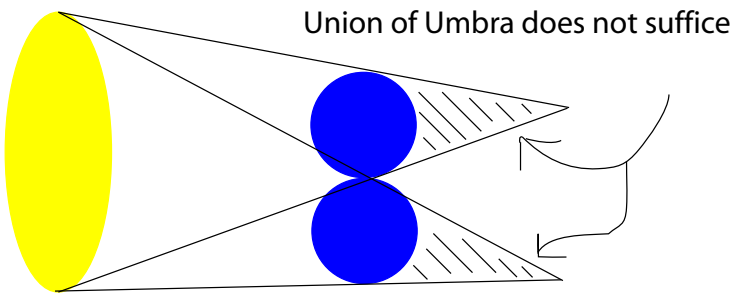

Figure 3: Nipun's counter-example.

The umbra volume can be thought of as the intersection of 3 half-spaces. Imagine the area light source (or window) is composed of a set of vertices V_i. Now, imagine finding the umbra of a point light source, where each vertex is a point light source. The umbra of the area light source will then be the intersection of all the individual umbra (Figure 4). This is true because of continuity and complexity (convex objects).

Moving on to 3D, how to compute the umbra of a triangle. Check out figure 5.

Perhaps we could try the intersection of 4 cones that start at the 4 points of our window, but let's look for an easier way.

We don't want to just test every point by shooting rays or something similar. We want some explicit model of the umbra volume. But how to generate the

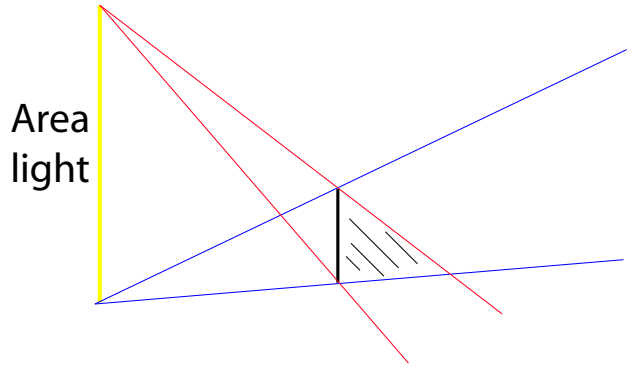

Figure 4: Intersecting Umbra from point lights to create area light umbra.

planes (half-spaces)?

We would like to connect the top-most edge of our triangle with the top-most vertex of our window. Said another way, we could start with the plane of our triangle, and rotate the plane, using the edge as pivot, until the plane touches a vertex on the window. Okay, so from here, can we somehow wrap around, creating all the necessary planes?

Let's look at two convex blobs and the 3D convex hull that surrounds them (Figure 6).

We can write the tube as: $Tube = \delta H(A\ U\ B) - (\delta A\ U\ \delta B)$ Where δ means the boundary.

Essentially, we are creating a tube that connects the silhouettes of the two blobs.

Now consider polyhedra instead of convex blobs. How can we use this tube to help us understand visibility? (Figure 7, figure 8)

We will perform a conservative test. Our goal is to be able to find situations where we can say for sure that A can not see any part of B because of the occluder S. Our test is conservative, so we might say, "we don't know", for some situations.

This is the high-level algorithm to guarantee occlusion:

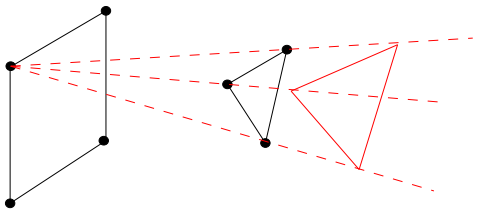

Figure 5: Repeat this process from all vertices of the window. Intersection of all volumes gives us the umbra.

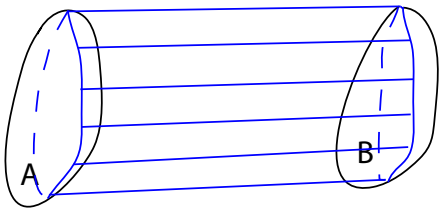

Figure 6: Blue tube connecting two convex blobs.

- Shoot ray A to B. We must intersect disk odd number of times.
- Check if curve C goes around the tube. (use main axis).
- Check that there are no holes, which implies no borders inside the tube. We will be conservative and check for no border in the disk bounded by curve in S going "inwards". Inward we will say is Counter Clockwise, facing us, turn to the left.

Can we avoid computing the curve (Figure 9)?

For more information, check out the paper *ShieldTester*.

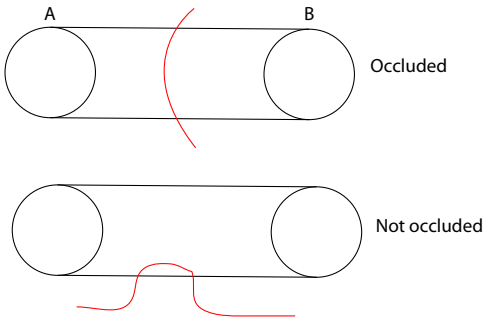

Figure 7: Red curve is the potential occluding surface.

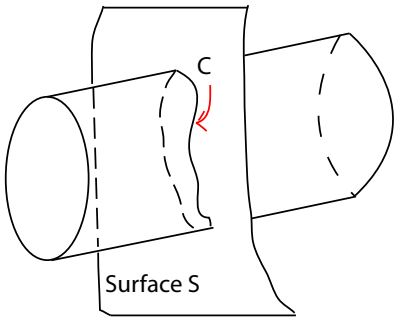

Figure 8: C is the curve created by the intersection of surface S with the tube.

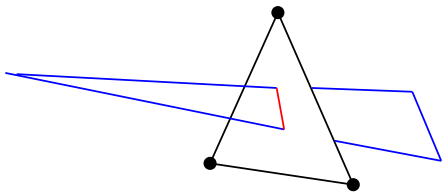

Figure 9: We just need to know if we want to leave the white triangle (surface S) or blue triangle (tube) first.