Ray Tracing Depth Maps Using Precomputed Edge Tables

Kevin Egan Ivan Neulander
Rhythm and Hues Studios

Introduction

We present a new data structure and algorithm for parallelizing the ray tracing of depth maps to accurately render soft shadows. Instead of caching shadow rays or optimizing intersection computations for a single ray [Agrawala et al. 2000], we instead trace many shadow rays to the light in parallel by using a lookup table at each filter pixel edge.

Implementation

We introduce the *edge table mask* to accelerate shadow ray intersections within a perspective depth map. The novel property of an edge table mask is that, given a planar light source perpendicular to the depth map's axis of projection, the z-distances stored in the edge table mask can be used for any pixel in the depth map (due to the z-distances being unaffected by shearing in the *xy* plane of the depth map). To create a new mask for a shading position, we intersect each shadow ray with the neighboring pixel frusta, marching along an epipolar ray through the depth map. At each intersection we record the z-distance between the intersection point and the shading position. We call this z-distance ζ as shown in Figure 1. If the intersection point is closer to the light than any occluder in the depth map, we record the intersection and stop ray marching.

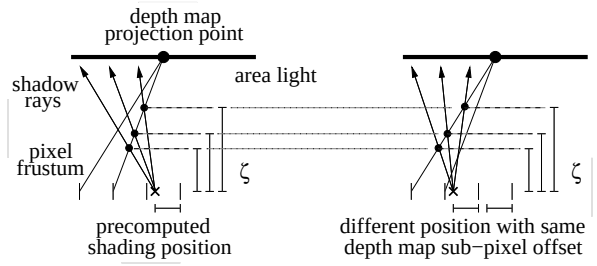

Figure 1: Left: For each depth map pixel frustum we store the z-distance for all ray intersections. Right: A new shading sample with the same depth map sub-pixel offset will have the same ζ values for neighboring pixel frusta.

For each pixel frustum side that has an intersection we create an *edge table* to efficiently query intersection information. The edge table stores a list of intersections sorted in increasing ζ order (non-intersecting rays have $\zeta = \infty$). For each intersection depth d, the edge table also stores a bitmask where all rays with $\zeta \leq d$ have a corresponding 0 bit and all others have 1 bits. Given a query distance q, a binary search is all that is necessary to efficiently return the appropriate bitmask that stores all rays with $\zeta \leq q$. To query for rays with $\zeta > q$ we simply invert the bitmask.

Each edge table mask is specific to the set of sample positions on the area light surface, the shading sample's z-depth, and the shading sample's sub-pixel offset within the depth map. We uniformly sample in the three dimensions of *xy* sub-pixel offset and zDepth, and compute a new mask for each sample. We stochastically resample the light surface for each mask.

At render time we match a new shading sample with the closest edge table mask based on depth map sub-pixel offset and zDepth, (multiple masks can be used for trilinear interpolation). Using the simplified *wall* and *floor* representation of depth map geometry [Agrawala et al. 2000], the edge table mask data structure can efficiently process all depth map pixels and pixel edges in the filter region using the `IntersectFloor()` and `IntersectWall()` methods shown below (0 bits represent occluded rays). Each filter pixel edge is classified as being an "in" or "out" edge for an adjacent pixel based on whether the epipolar shadow rays enter or exit through the edge. We bitwise AND all intermediate results from walls and floors to get the final set of occluded rays.

```
INTERSECTFLOOR(zeta, inEdges, outEdges)
1  inMask ← (anyInEdges) ? FULL_MASK : EMPTY_MASK
2  outMask ← FULL_MASK
3  foreach e ∈ inEdges
4     inMask ← inMask AND e.Below(zeta)
5  foreach e ∈ outEdges
6     outMask ← outMask AND e.Below(zeta)
7  return (inMask OR (NOT outMask))

INTERSECTWALL(zetaMin, zetaMax, edge)
1  return ((NOT edge.Below(zetaMin)) OR edge.Below(zetaMax))
```

We are currently investigating a variety of methods to improve efficiency, including grouping nearby rays so that edge tables can store partial bitmasks, storing fewer masks for lights with 8-way symmetry, and using a hierarchical depth map.

Results

The image in Figure 2A took 76 seconds to render on an Athlon 2133MHz, using a 256x256 depth map with 16 edge table masks (sampled 4x4x1 in depth map sub-pixel space). Each edge table mask stored 1024 rays and used an average filter region of 394 depth map lookups. The reference ray traced image took 1211 seconds to render using 1024 rays per pixel. Most of the error in Figure 2A comes from the discretization of silhouettes in the depth map and the relatively small number of edge table masks.

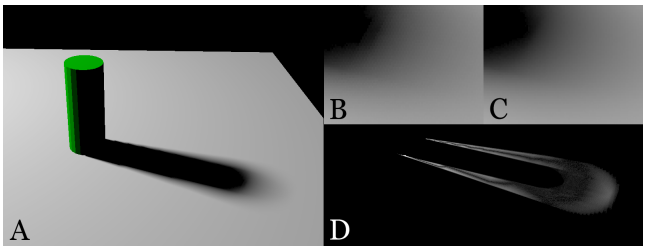

Figure 2: A: Soft shadow rendered using edge table masks. B: Closeup from left image. C: Closeup from ray traced image. D: Error in left image (brightness multiplied by 4).

References

AGRAWALA, M., RAMAMOORTHI, R., HEIRICH, A., AND MOLL, L. 2000. Efficient image-based methods for rendering soft shadows. In *Proceedings of ACM SIGGRAPH 2000*, 375–384.