Ray Maps for Global Illumination

Vlastimil Havran* Jiří Bittner† Hans-Peter Seidel*
*MPI Informatik, Saarbrücken, Germany †VUT, Vienna, Austria

Fast rendering of high quality images with global illumination is still a challenging task. A popular technique addressing this topic is photon mapping [Jensen 2001]. Photon mapping uses a global photon map for indirect illumination and a caustics map for caustics. In order to achieve visually pleasing results a costly **final gathering** step is commonly used with the global photon map since a simpler direct visualization would produce significant **boundary artifacts**. The artifacts are caused by using only photon hits on object surfaces: photons passing nearby an object are not accounted for which underestimates indirect illumination on the object boundaries (see Figure 2, right top).

We have developed an efficient technique that eliminates the boundary artifacts and thus it helps to avoid the final gathering. We have extended the concept of **photon maps** to **ray maps**. Instead of storing only the photon hits the ray map stores also the photon paths represented as sequences of rays. The ray map allows to efficiently answer various **ray proximity queries** that cannot be answered using the photon map. For example we can locate also photons passing in vicinity of an object and use them for density estimation.

The idea of using photon paths instead of photon hits was presented recently by Lastra et al. [Lastra et al. 2002]. They use a tangent plane at the point of illumination and search for rays intersecting a disc using a *ray cache* based on dynamic list of spheres. However, the ray cache results in a performance decrease up to two orders of magnitude compared to the direct visualization of photon maps. Additionally, the ray cache performance highly depends on the coherence of subsequent intersection queries.

We propose to implement ray maps using **lazily built kd-trees**. The interior nodes of the tree are associated with axis aligned splitting planes, the leaves store references to rays. The kd-tree is constructed lazily in order to adapt to the actual queries on the fly. We use a caching strategy that allows the user to specify a maximum memory usage for tree; we keep only the most recently accessed tree paths and collapse the unused parts. The query performance is improved using several optimization techniques. For example we use directional splitting nodes that separate non-feasible rays by exploiting directional coherence of the queries. For the nearest neighbor queries we use a priority queue and an efficient N-median search. The priority queue is reused by the subsequent coherent queries in order to eliminate repeated tree traversals.

Our implementation of ray maps allows to efficiently compute ray proximity queries that use different intersection domains and nearest neighbor distance metrics:

I. **Intersection domains**
   - disc,
   - hemisphere,
   - axis aligned bounding box,
   - sphere.

II. **Nearest neighbor distance metrics**
   - distance to the point of intersection of the ray with a tangent plane,
   - distance to the ray segment,
   - distance to the supporting line of the ray.

For example using the distance of point to ray segment results in a hemispherical neighborhood where the nearest rays are found (see Figure 1). Our measurements show that the radius of a hemisphere is approximately 30% smaller than the radius of a disc for the same number of nearest rays, which allows to reduce the bias of density estimation. Additionally, the ray map queries can be issued in an arbitrary point in space: for example we can use the spherical ray intersection domain for volumetric illumination effects.

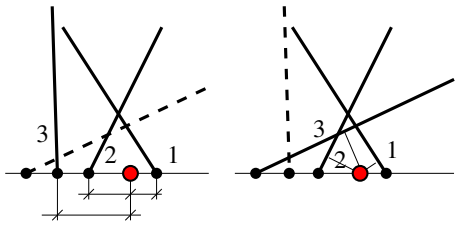

Figure 1: Three nearest neighbors according to different ray distance metrics. (left) Euclidean distance between the center of the query and the intersection of the ray with a tangent plane. (right) Euclidean distance between the query center and the ray itself.

Another application of ray maps is **light path invalidation** for scenes with moving objects. We determine all rays intersecting the moving objects and reshoot only the affected light paths.

Performance: Our ray map implementation is 3.9 to 5.2 times slower than the photon map implementation described in [Jensen 2001]. However, the ray map fully eliminates the disturbing boundary artifacts. Additionally, the ray map is from 10 to 51 times faster than the ray cache described in [Lastra et al. 2002].

Conclusions: We have developed a concept of ray maps allowing efficient computation of ray proximity queries. The ray maps allow to eliminate the boundary artifacts of the classical photon mapping at a moderate increase of the computational cost. Being able to efficiently answer various ray proximity queries the ray map opens new application directions such as using other metrics for density estimation or new lighting update strategies for scenes with moving objects.

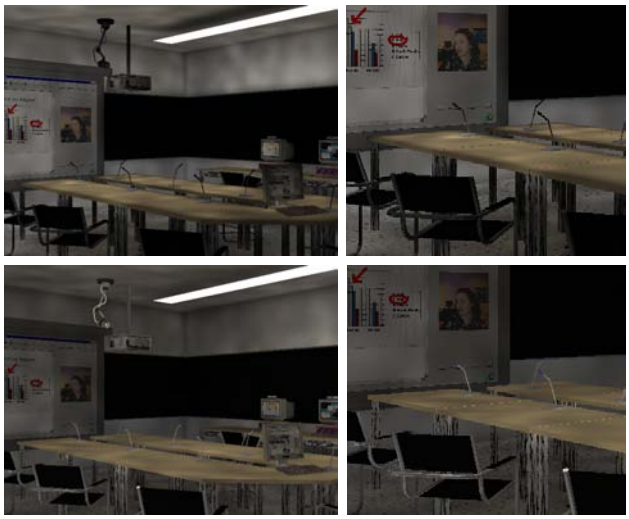

Figure 2: Rendering without final gathering: (top row) indirect illumination computation by direct visualization of photon maps. (bottom row) indirect illumination computed using ray maps.

References

JENSEN, H. W. 2001. *Realistic Image Synthesis using Photon Mapping*. A. K. Peters, Ltd.

LASTRA, M., URENA, C., REVELLES, J., AND MONTES, R. 2002. A Particle-Path Based Method for Monte Carlo Density Estimation. In *Rendering Techniques 2002 (Poster Papers Proceedings).*