A Divide-and-Conquer Algorithm for Simultaneous Photon Map Queries

Alexander Keller* NVIDIA
Marc Droske† NVIDIA
Leonhard Grünschloß‡ NVIDIA / Weta Digital
Daniel Seibert§ NVIDIA

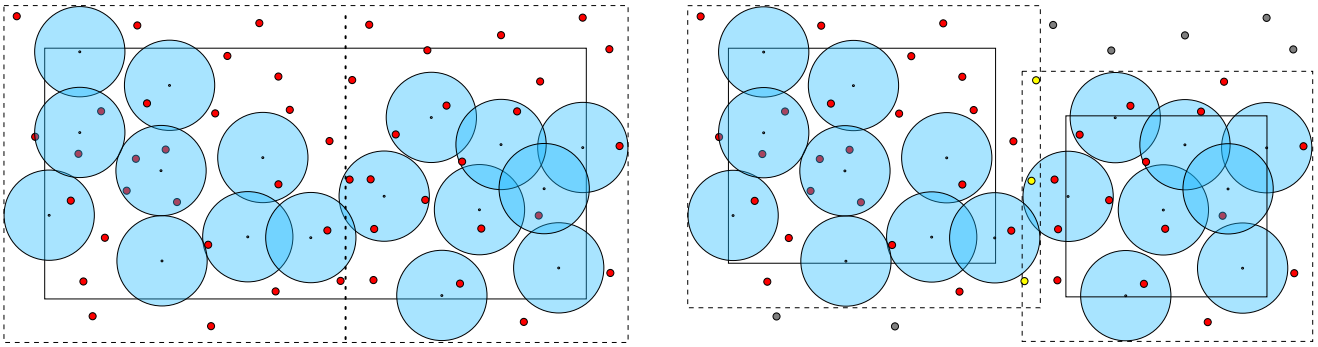

Figure 1: *Instead of single photon map queries, a batch of range queries (blue disks centered at black dots) for photons (red dots) is executed simultaneously, which increases efficiency, because similar queries can share search results. In addition, partitioning the set of query locations on the left by a split plane allows for early discarding of photons (gray dots) outside the resulting search ranges (dashed boxes on the right). The yellow dots (on the right) are photons that need to be considered in both search ranges.*

1 Progressive Photon Mapping

Following the paths of photons emitted from lights is the most natural way to simulate light transport. Storing the flux ϕ_p with the direction ω_p and location of incidence whenever a photon hits a surface allows one to determine the radiance

$$L(x,\omega) = \lim_{n\to\infty} \frac{1}{n} \frac{\sum_{p\in\mathcal{B}(x,r(n))} f_s(\omega,x,\omega_p)\phi_p}{\pi\cdot r^2(n)}$$

in a point x seen from direction ω: The flux of all photons p in a ball $\mathcal{B}$ of radius $r(n) := r_0\sqrt{n^{-\frac{1}{2}}}$ is colored by the physical transport properties f_s of the surface and averaged across a disk of the same radius, where n is the total number of photon paths.

As defined above, the squared radius $r^2(n)$ decreases slower than the increase in the number of paths, which yields consistent algorithms and even allows for deterministic progressive photon mapping [Keller et al. 2010, and references therein].

Simulating the process of taking a photograph consists of computing the color of a pixel in an image by averaging samples of the radiance L in query locations seen through that pixel. In order to stay within a finite amount of memory, the computation proceeds in passes. For each pass a set of photons is generated in order to compute the radiance in a set of query locations, which is accumulated in the pixels.

2 Implicit Hierarchy for Range Queries

Simultaneously considering all range queries of a pass results in a recursive divide-and-conquer Algorithm 1. Recursion is terminated whenever the number of remaining query locations falls below a selected threshold or the set cannot be partitioned further. Upon termination, the contribution of each remaining photon to each remaining query location is computed.

*e-mail: keller.alexander@gmail.com
†e-mail: marc@mental.com
‡e-mail: leonhard.gruenschloss@gmail.com
§e-mail: daniel@mental.com

Algorithm 1: Simultaneous hierarchical range search.

```
Integrate(QueryLocations, Photons)
if QueryLocations ≠ ∅ and Photons ≠ ∅ then
    PhotonsInBBox ← Photons ∩ BBox(QueryLocations);
    if PhotonsInBBox then
        if Terminate(QueryLocations, PhotonsInBBox) then
            AddRadiance(QueryLocations, PhotonsInBBox);
        else
            (QLoc1, QLoc2) ← Partition(QueryLocations);
            Integrate(QLoc1, PhotonsInBBox);
            Integrate(QLoc2, PhotonsInBBox);
        end
    end
end
```

In the recursion step overlapping range queries share results and computations are restricted to regions where both query locations and photons reside, as illustrated in Figure 1. The recursive algorithm in fact traverses a bounding volume hierarchy, which resembles collision detection algorithms, however, this hierarchy is never explicitly stored. As the working set becomes more localized with recursion depth, performance benefits from processor caches.

A parallel version has been implemented in CUDA for NVIDIA GPUs. In a breadth first approach, each step of the recursive algorithm is parallelized, while large nodes are partitioned into smaller chunks in order to increase occupancy and such efficiency. Partitioning the query locations uses a segmented scan, while ordering the photons accordingly must use two segmented scans as replication may occur. Finally, active nodes are compacted.

In summary, the new algorithm significantly improves the performance of progressive photon mapping.

References

KELLER, A., GRÜNSCHLOSS, L., AND DROSKE, M. 2010. Quasi-Monte Carlo progressive photon mapping. under revision.