Out-of-Core Photon Mapping: When More isn't Enough

CC Envelopes and Solar Energy
Lucerne University of Applied Sciences and Arts

Roland Schregle (until May 2016) - presented by C. Bauer -
Present: roland.schregle@gmail.com

15th Radiance Workshop

Motivation

Applications demand high photon counts

- Photon count is the key parameter for accurate simulations
- Contribution Photon Maps need higher counts than normal ones (demand approx. scales with number of sources/source patches)
- Applications like EvalDRC with additional sun coefficients further increase the demand

Limitation

Photon counts of several 100 Million up to a few Billion use up too much memory on today's commodity hardware (8 GB RAM)

Strategy

Provide efficient mechanism for Out-of-Core storing and on demand loading of photon data
Overview

Out-of-Core Photon Map Generation

Photon Map generation (mkpmap)

Scene

Forward raytracer

Unsorted photon heap (out-of-core)

Morton code generator

Out-of-core sort

Morton codes

Photons in Morton order

Octree build

Photons Map

Photon primaries

Leaf file

Octree nodes
Overview

Out-of-Core Photon Map Use

Photon Map
- Photon primaries
- Leaf file
- Octree nodes

- In-core load
- Photons loaded on demand
- In-core load

- Photon cache
- Cached photons
- NN search
- Search buffer

- Photon density estimate
- Irradiance

Photon Map Lookups (rtrace/rvu/rpict/rcontrib)
Building Blocks of the Algorithm

Photon Map Generation

- Photon indexing with Space Filling Curves / Morton Coding
- Photon sorting (external sorting algorithms for large datasets are a well studied area in computer science)
- Octree construction (hierarchical volume structure for nearest neighbour search)

Photon Map Use

- Paging mechanism for loading and unloading photon data from the disk (analog to the OS handling of main memory pages)
- Intelligent caching strategy to reduce load/unload operations
- Decisions on types of data which are kept in core (iC) or out-of-core (ooC)
- Efficient nearest neighbour searches
Morton Code Example

$f_{M,1}$

$f_{M,2}$
Octree Generation

Ordered photons / Morton indices facilitates 1-pass octree generation
Nearest Neighbour Search

Logical Layout
Paging Mechanism

→ cf. Techn. Report
In and Out

Permanently in Memory

- Octree Structure (Nodes)
- Photon Primaries

Loading on demand (out-of core)

Gross photon data

Critical aspects

- Page and Cache sizes (amount of photons to load and buffer)
- I/O bottleneck / contention
Do's and Don'ts with ooC

In principle, the ooC scheme needs coherent data (i.e. photon) accesses for an efficient operation. This is generally the case for direct photon density evaluation, which can be enabled with the RADIANCE parameter -ab -1.

The evaluation of photon density via 1 ambient bounce (as recommended in the original Photon Map version) will lead to excessive loading/unloading of photon data in the ooC variant.
ooC vs. iC - Benchmarks

![Graphs showing benchmark results for ooC and iC]
ooC Parallel Processing Benchmarks

ooC mkmap Scalability, 8Gb RAM

ooC rtrace Scalability, 8Gb RAM
Radiance with ooC Photon Map

- Default Version: in Core Map
- ooC Version needs Rebuild with -DPMAP_OOC
- ooC Photon Map produces 2 Files: xyz.pmap and xyz.pmap.leaf
- iC and ooC Photon Maps are not compatible
- additional ooC Photon Map parameters: -ac/-aC for cache and page size adjustment (cf. Tech.Report)
Thank you for your attention!

Our website: http://www.hslu.ch/ccease

References / Suggested further reading:


This research was supported by the