A Vectorized Traversal Algorithm for Ray-Tracing

José María Noguera Rozúa
Carlos Ureña Almagro
Rubén J. García Hernández

University of Jaén
University of Granada
Contents

- Introduction
- Previous Work
- Our Proposal
- Results
- Conclusions & Future work
Ray-Tracing:
- Very Realistic image synthesis.
- Complex models supported.
- Basis of many algorithms.
- Traditionally slow.
Parallel Raytracers
  - Breadth-first traversal.
  - SIMD aware.
  - Some approaches use GPU.
Previous work

Classic algorithm

- Traversal algorithm: Once per ray
Previous Work
Wald's algorithm I

- Coherent rays:
  - Rays with the same origin and very similar direction will very probably hit the same objects.

- The algorithm uses a kd-tree and traverses it with 4-ray packets.

- Parallelism using Intel SIMD: SSE.
Four ray traversal.
Wald's algorithm limitations:

- The gain in efficiency increases with ray coherency.
- It's easy to select coherent primary rays in Ray-Tracing or Path-Tracing.
- For secondary rays in Ray-Tracing, or other applications, rays are not coherent.
  - Gain is therefore smaller.
With non-coherent rays, packets tend to break, decreasing parallelism.
Our Proposal

Description

- We propose traversing the kd-tree only once but with all rays at the same time.

- Rays are classified during the traversal process:
  - No additional effort needed.
  - Coherence is increased in terminal nodes.

- We aim for maximum parallelization.
Our Proposal

Traversals Algorithm 1

- Traversal for $n$ rays.
Our Proposal

Traversal Algorithm II

- **Problem:**
  - Appears when looking for the first intersection (typical in raytracing and other applications).
  - We want the ability to stop the traversal of a ray when finding its first intersection.
  - Different rays may mean different traversal order of the tree.
Solution:

- Classify the rays in eight groups according to their director vectors.
- Run the algorithm for each group.
Advantages:

- If 2 or more rays cross the same node, they will do it together.
- Increases the possibilities of having rays available to process in parallel.
- Non-coherent rays will be classified when going down the tree.
- The number of traversals of the tree, and the number of triangle accesses is reduced.
Our Proposal
Memory layout of rays I

- A global data structure for the rays:
  - Stores all rays and their attributes (director vector, origin, intersected triangle id...)

- A ray stack:
  - Stores only the values for the ray traversal depending on the current node.
  - Pointer to the rest of the information in the structure above.
  - SSE optimized "Structure of Vectors".
Our Proposal
Memory layout of rays II

State of the Ray Stack in node C:
Performance I

- Tests with coherent rays (Ray-Tracing's primary rays – 1 ray per pixel)
  - Classic algorithm, 1 traversal per ray.
  - Wald's algorithm, 1 traversal per 4-ray packet.
  - Wald's algorithm, 1 traversal per 64-ray packet.
  - Proposed algorithm, 1 traversal for each group of rays with direction vector in the same octant.

- Test with non coherent rays
  - Random rays with uniform distribution.
Performance II

Dragon
0,8 M tri.

Stanford Bunny
0,069 M tri.

Happy Buddha
1,1 M tri.
Performance III

Buddha, 1.1MTris, 1280x1024
Conclusions & Future Work

Our technique allows:

- More parallelization to be extracted even for non-coherent rays.
- Visiting each node at most eight times. Less access to nodes and their triangles.

Future work: larger SIMD width → GPU