Efficient Ray Traced Soft Shadows using Multi-Frusta Tracing

Carsten Benthin, Ingo Wald
Intel Labs

Goals

- Ray-traced soft shadows
- Game-like scenarios
- Want real-time performance
- Target architecture: Larrabee
  - Many-core
  - n-wide SIMD (n = 16)

Previous Work

- Special approaches for fast ray traced soft shadows
  - [Laine ‘05, Lehtinen ‘06, Overbeck ‘07,…]
- Complex global data structures
- Complex state handling
- Unclear how to efficiently map to many-core + wide SIMD

Monte Carlo (MC) - Sampling

- Robust
- Scales in #cores
- Need high #samples for sufficient quality
- Traditional ray tracing performance not high enough for real-time

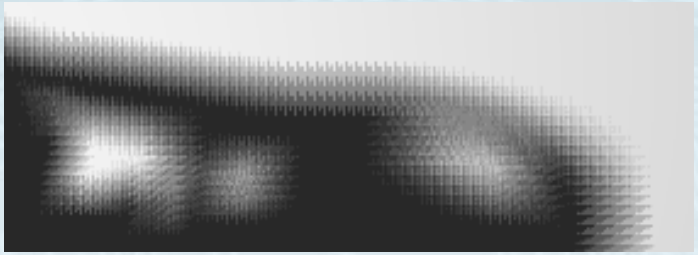
Monte Carlo (MC) - Sampling

- Robust
- Scales in #cores
- Need high #samples for sufficient quality
- Traditional ray tracing performance not high enough for real-time

- Reduce #samples without large quality impact
- Speedup ray tracing performance

Sampling

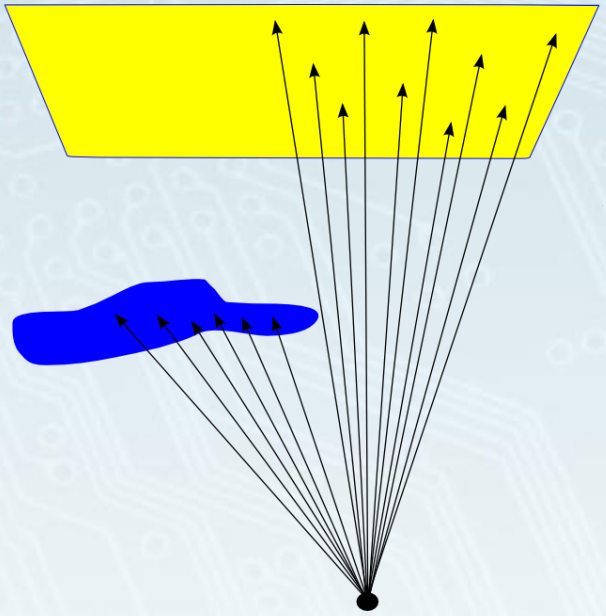
- **Interleaved Sampling** [Keller ‘01]
  - 4 x 4 pixel pattern
  - 16 different sampling sets, 16 samples each



- **Discontinuity Buffer** [Kollig ‘02]
  - Combines sampling sets
  - Filters neighboring irradiance in 1-2 pixel radius
  - Filtering is post-process, done after rendering

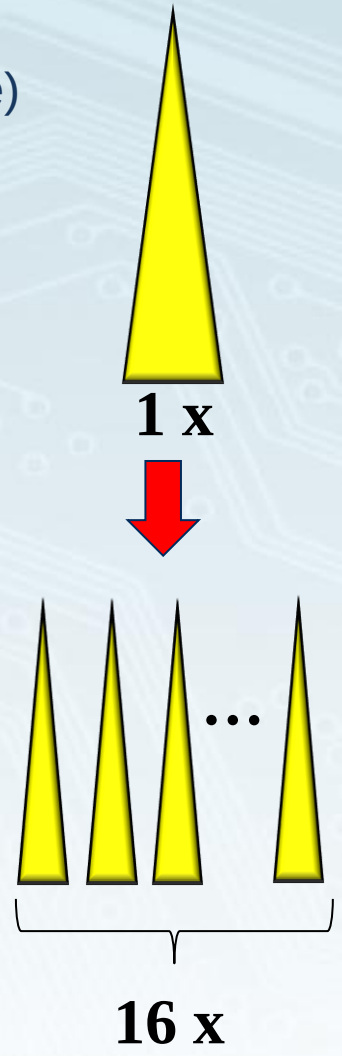
Soft Shadow Ray Distributions

- Distributions exhibit coherence
- Packet tracing
  - Good SIMD utilization
  - Speedup is bound by SIMD width
- Beam/frustum techniques
  - More rays (than SIMD width) at once
- New (old) problems
  - Low utilization of wide SIMD
  - Many rays ➔ large state



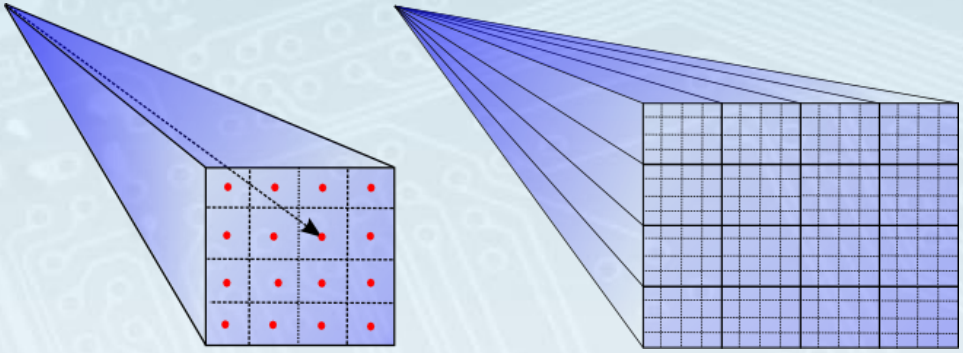
Multi-Frusta Tracing (MFT)

- For n-wide SIMD trace ‘n’ frusta in parallel (n = 16 for Larrabee)
  - Generalization of SIMD packet tracing to SIMD frusta tracing
  - High SIMD utilization
  - Optimizations due to common origin per frustum
- Pure frusta-based tracing
  - Traversal is independent of individual rays
- For intersection tests generate rays quickly on demand
  - Minimal memory consumption
- MFT for **soft shadows** and **primary visibility**

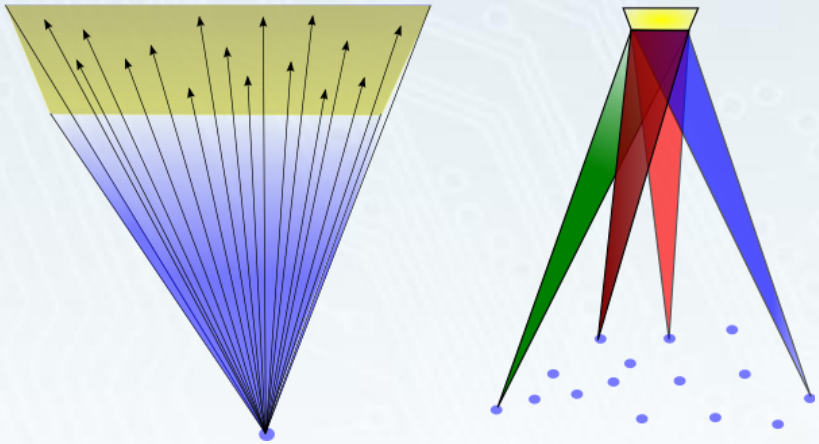


Multi-Frusta Generation

- Primary Visibility
  - Single common origin
  - Single frustum = 4x4 pixel grid
  - 16 frusta = 16x16 pixel grid



- Soft shadows
  - Rectangular light source
  - Traces 16 frusta **from** 16 **different** intersection points **to** the light source
  - Only track occlusion state

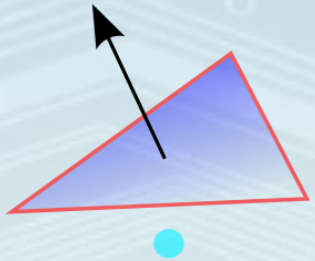


MFT Traversal

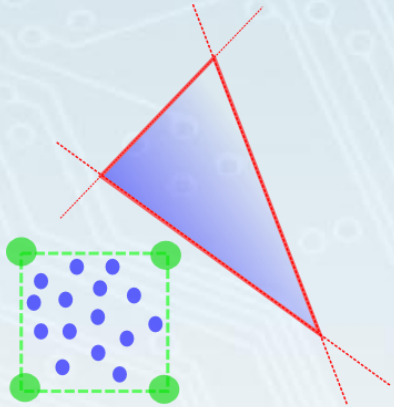
- Shallow BVH
- Pure interval arithmetic-based culling of AABBs
- Complexity: 16 frusta vs. AABB ~ 16 rays vs. AABB
- Tracking of active/inactive frusta
- Early frustum termination

At The Leaf

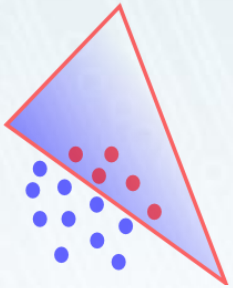
- Iterate over triangles
- Two culling tests to reduce #active frusta per triangle
  - Back-face and corner ray culling
  - For all 16 frusta done in parallel (SIMD)
- Iterate over all active frusta
  - Generate rays quickly on demand
  - 16-wide ray triangle intersection test (SIMD)



back-face culling



corner ray culling

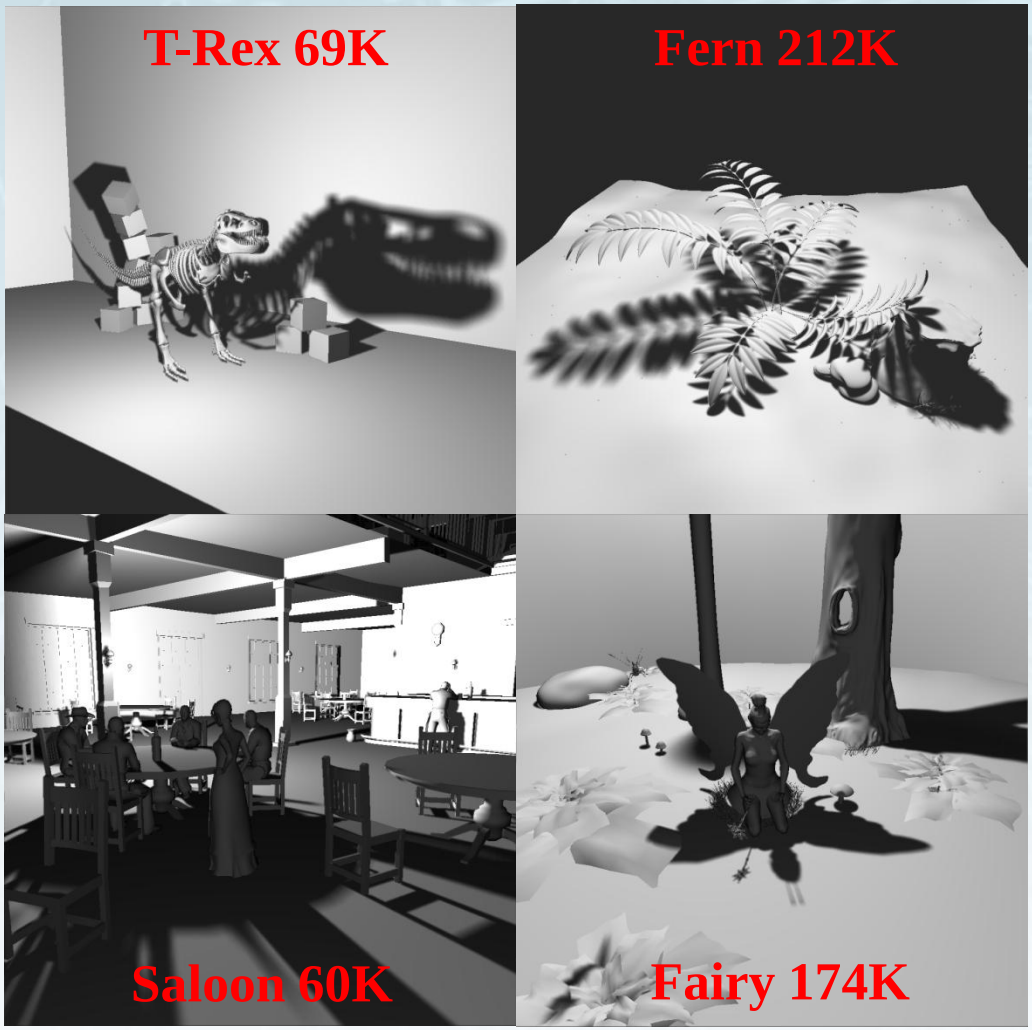


16-wide ray triangle intersection

On Demand Ray Generation

- Create only ray directions (common origin per frustum)
- Decompose ray direction into **corner ray** + **delta** vector
- **delta** = 3D offset vector on image/light source plane
- 16 x **delta** for shadow rays, 1 x **delta** for primary rays
  - Pre-computed per frame
- 3 x LOAD and 3 x ADD to create ray direction

Test Scenes



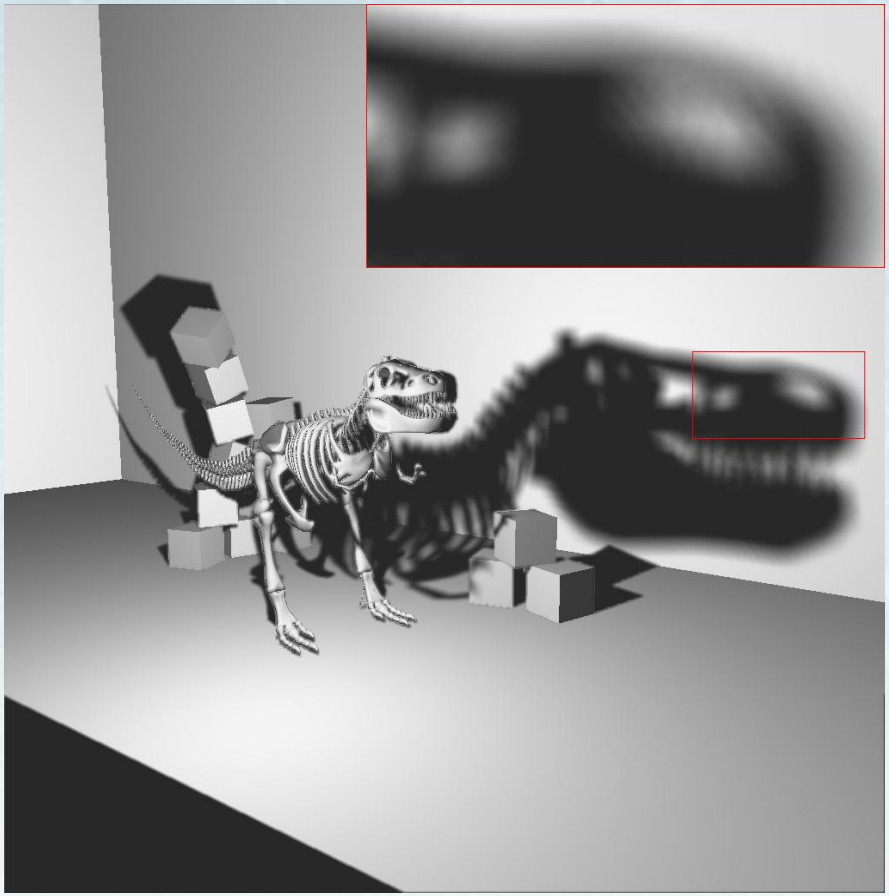
Setup

- 1024x1024 resolution, 4x4 interleaved sampling + filtering
- Single hypothetical Larrabee core @ 1 GHz (cycle accurate simulation)
- Comparison baseline: 16-wide packet tracer

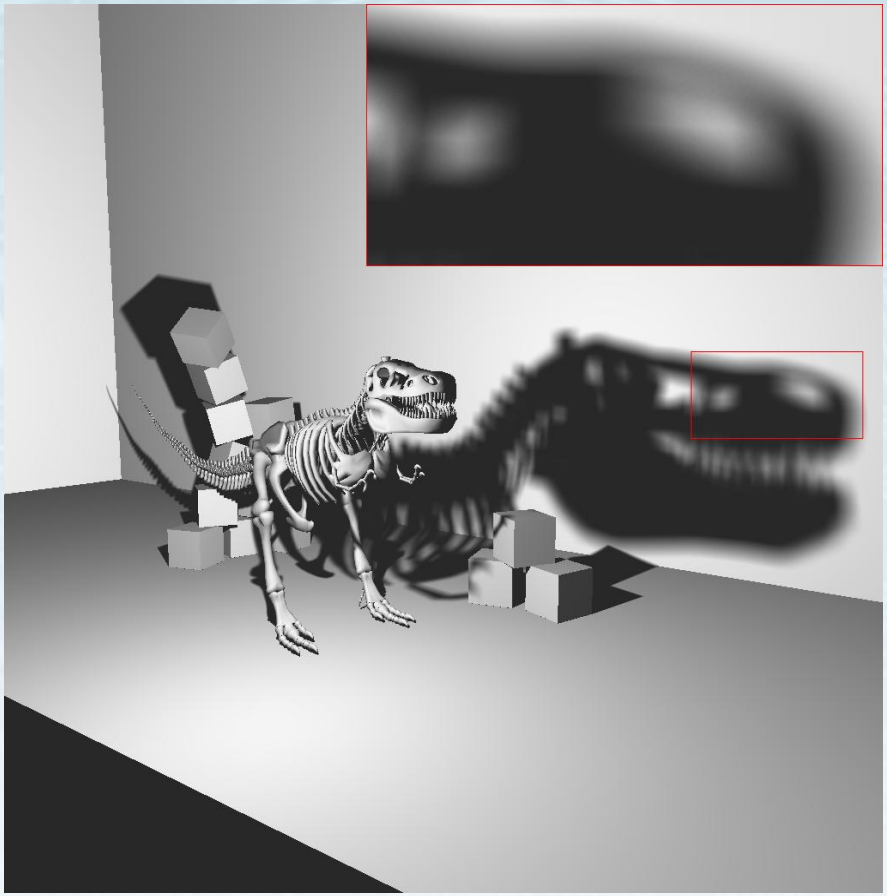
Results

- MFT reduces #AABB test by **11-23x**
- Primary visibility: **20-51 fps (3.3-7.8x)**
  - Excluding shading
- Soft shadows (16 samples per pixel): **1-2.2 fps (3.3-6.1x)**
  - Including primary visibility, shading, and filtering
- For 16 cores: **16-32 fps** (more than **half a billion rays/s**)

Soft Shadow Quality



16 samples, 4x4 interleaved sampling, discontinuity buffer



256 samples, reference image

Other Applications

- Hard shadows (256 intersection points vs. single point light)
- Instant Global Illumination (16 intersection points vs. 16 VPLs)
- Super-sampling for primary visibility, e.g. MSAA
- Combination with other algorithms, e.g. collect silhouette edges in frusta

Where MFT Breaks Down…

- Too much geometry in frusta, e.g. for very large light sources
  - Insufficient culling
  - Quick frusta (corner ray) shrinking helps a bit
  - Split one big frustum into many smaller
- Cannot handle incoherent ray distributions
  - *"Faster Incoherent Rays: Multi-BVH Ray Stream Tracing"* by John Tsakok

Conclusion

- MFT is a general framework
- Efficiently maps to wide SIMD
- Significant reduction in traversal steps (AABB tests)
- Very fast for coherent ray distributions
- Very memory efficient with on demand ray generation

Remarks & Future Work

- Paper implementation targets integration not performance
  - Example: current code for primary visibility is ~30% faster
  - Still a lot of room for improvement ☺
- Future work
  - Speedup current bottleneck: culling & intersection tests
  - Support transparent shadows
  - Adaptive or more approximate techniques
  - Non-common origin frusta

Thank you !