Row Tracing with Hierarchical Occlusion Maps

Ravi P. Kammaje, Benjamin Mora

August 9, 2008

Introduction
- Related Work

Row Tracing
- Concept
- Tree Traversal
- Leaf Node Processing
- HOMs

Packet Row Tracing

Results

Conclusion

Rendering methods - Rasterization

- Widely used
- Brute force approach
- High per pixel complexity
- Very fast per pixel due to fast scanline intersections
- Exceptional performance for relatively small models due to hardware acceleration
- Visibility determination
  - Z-buffer algorithms – E. E. Catmull (1974) , N. Green et al. (1993) , M. Wand et al. (2001), etc
  - Hierarchical Occlusion Maps (HOMs) – Zhang et al. (1997)

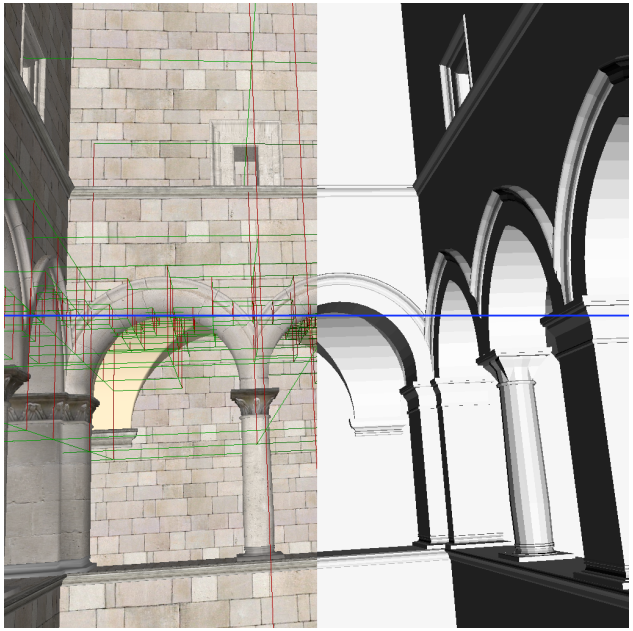
Rendering methods - Ray Tracing

- Acceleration structures
  - Spatial Subdivision Structures – kd-trees, BSP trees, octrees, grids, etc
  - Bounding Volume Hierarchies
- Scales well with increase in model size – logarithmic complexity
- Parallelization

Acceleration

- kd–trees for ray tracing – D. S. Fussell and K. R. Subramanian (1988), D. J. MacDonald and K. S. Booth (1988), V.Havran (2000), I. Wald et al. (2001)
- Packet Ray Tracing – I.Wald et al. (2001), Reshetov et al. (2005), P.S.Heckbert and P.Hanrahan (1984), R.Overbeck et al.(2007), S.Boulos et al.(2006), I.Wald et al.(2007)

Row Tracing



- Combines rasterization and ray tracing concepts
- A form of Packet ray tracing – Packets of rays spanning an entire row
- Row can be
  - A 2D plane
    - Simpler traversal
    - Easy row / triangle intersection – per-pixel cost less than ray / triangle intersections
  - A 1D line – Simplifies clipping, occlusion testing and frustum testing

Comparison with other methods

	Rasterization	Ray Tracing	Row Tracing
Logarithmic Complexity	✗	✓	✓
Cheaper per-pixel scanline algorithm	✓	✗	✓
Multi-core Parallelization	✗	✓	✓
Shadow Rays	✓ (with shadow mapping / shadow volumes)	✓	✓ (with shadow mapping / shadow volumes)

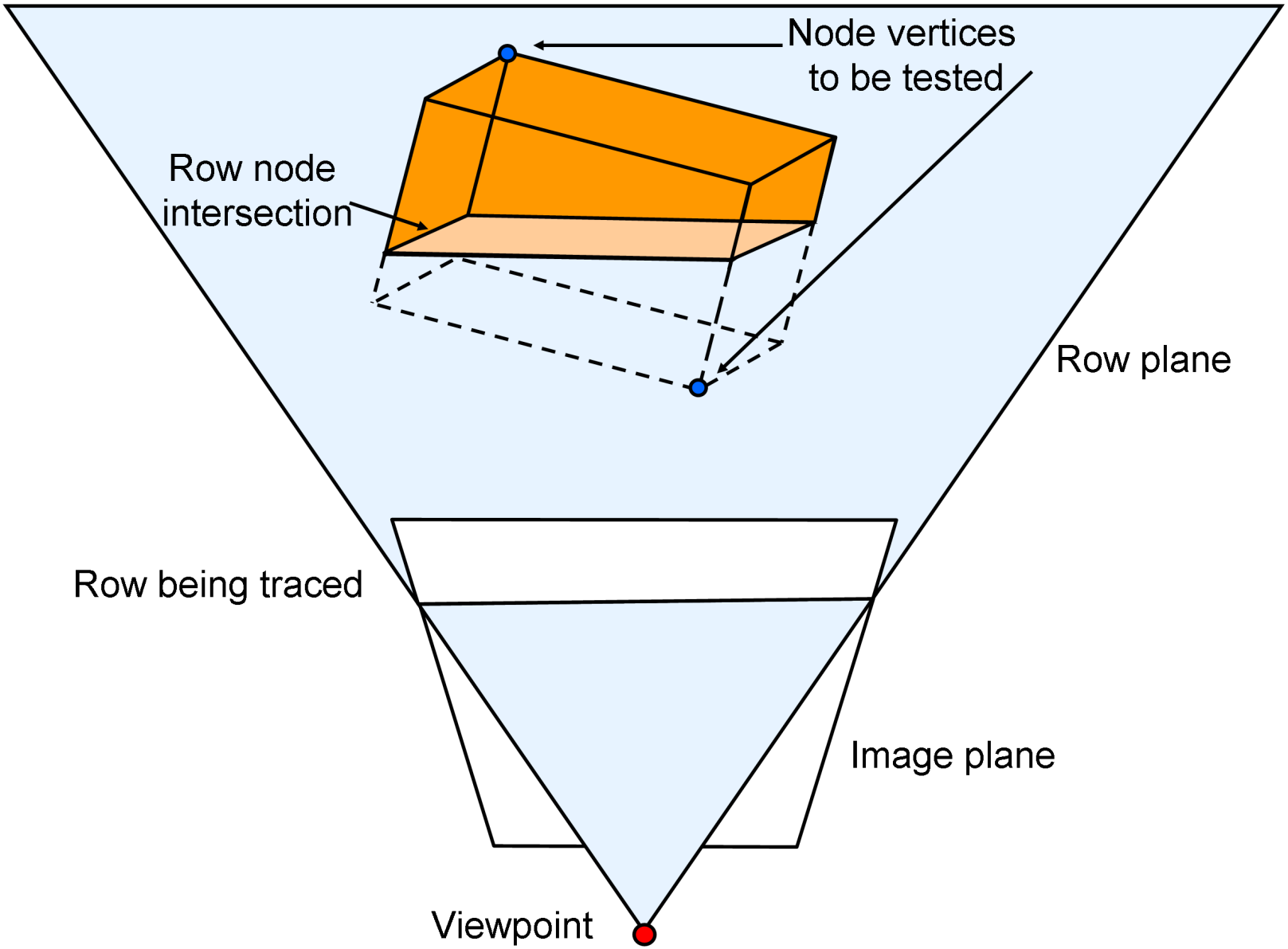
Row Tracing

- High level algorithm
  - Traverse row-plane through kd-tree or octree
  - Rasterize leaf node triangles with scanline algorithm
- Very similar to Ray tracing
- Early ray termination not possible
- Use 1D Hierarchical Occlusion Maps to achieve this

Tree Traversal

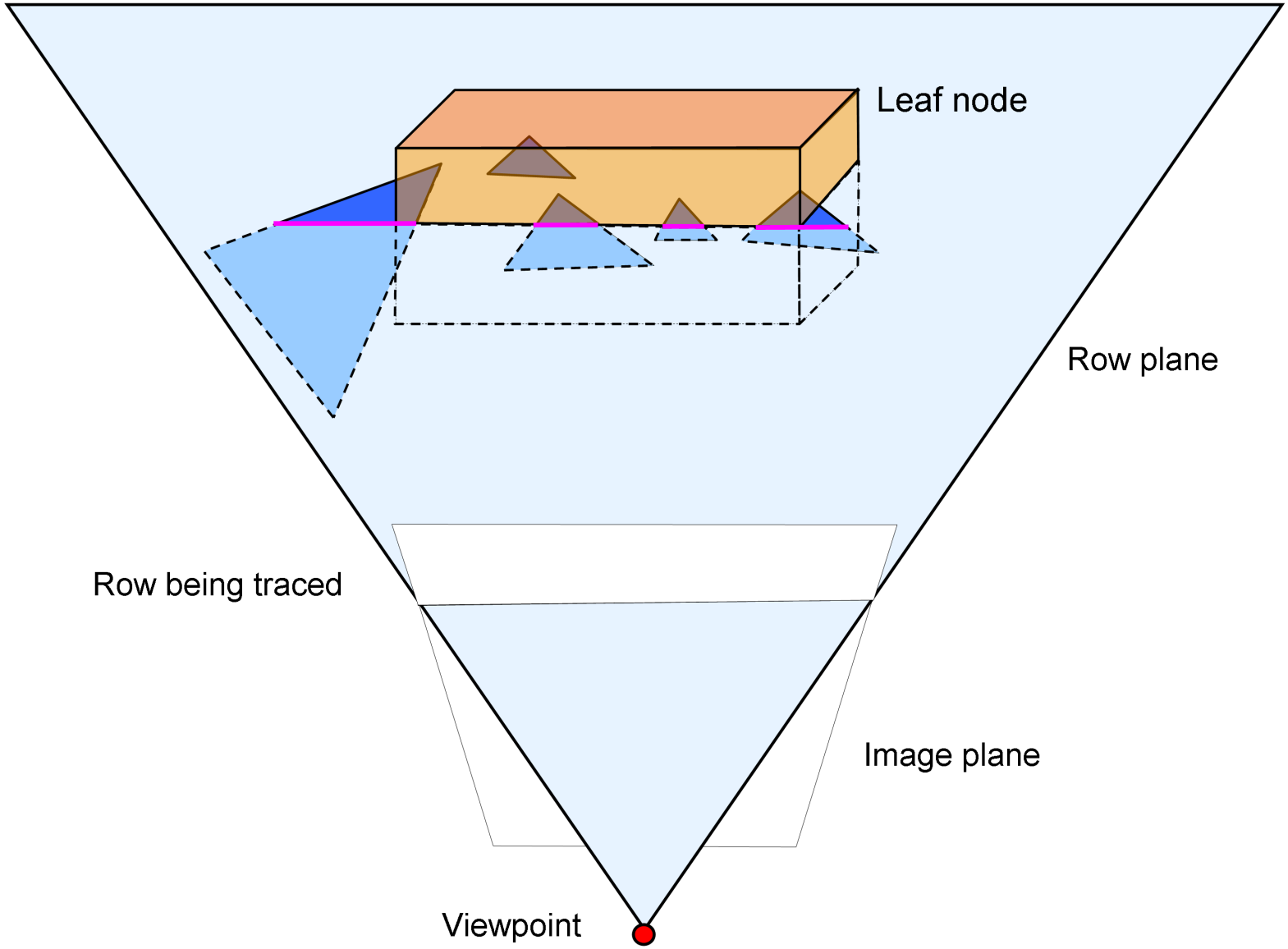
- Similar to ray tree traversal
- Start traversal from root node
- Traverse nodes in front-to-back order
- Traversal stops if
  - Row-plane does not intersect node
  - Node's projection is occluded
  - Node's projection is outside frustum (1D frustum)
- At leaf node – rasterize leaf node triangles

Row-plane / Node intersection



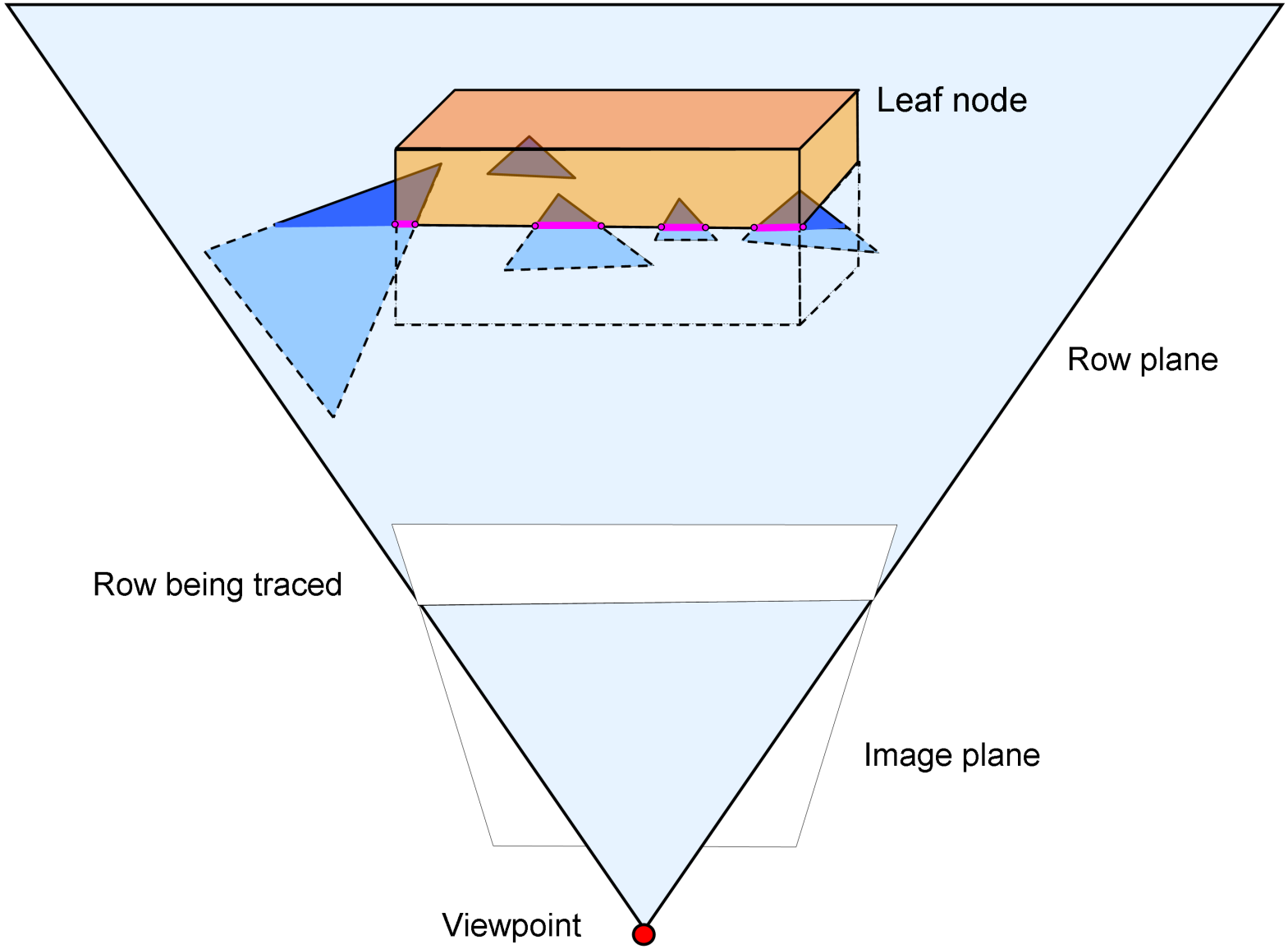
Leaf node processing

Intersect Row-plane with leaf node triangles



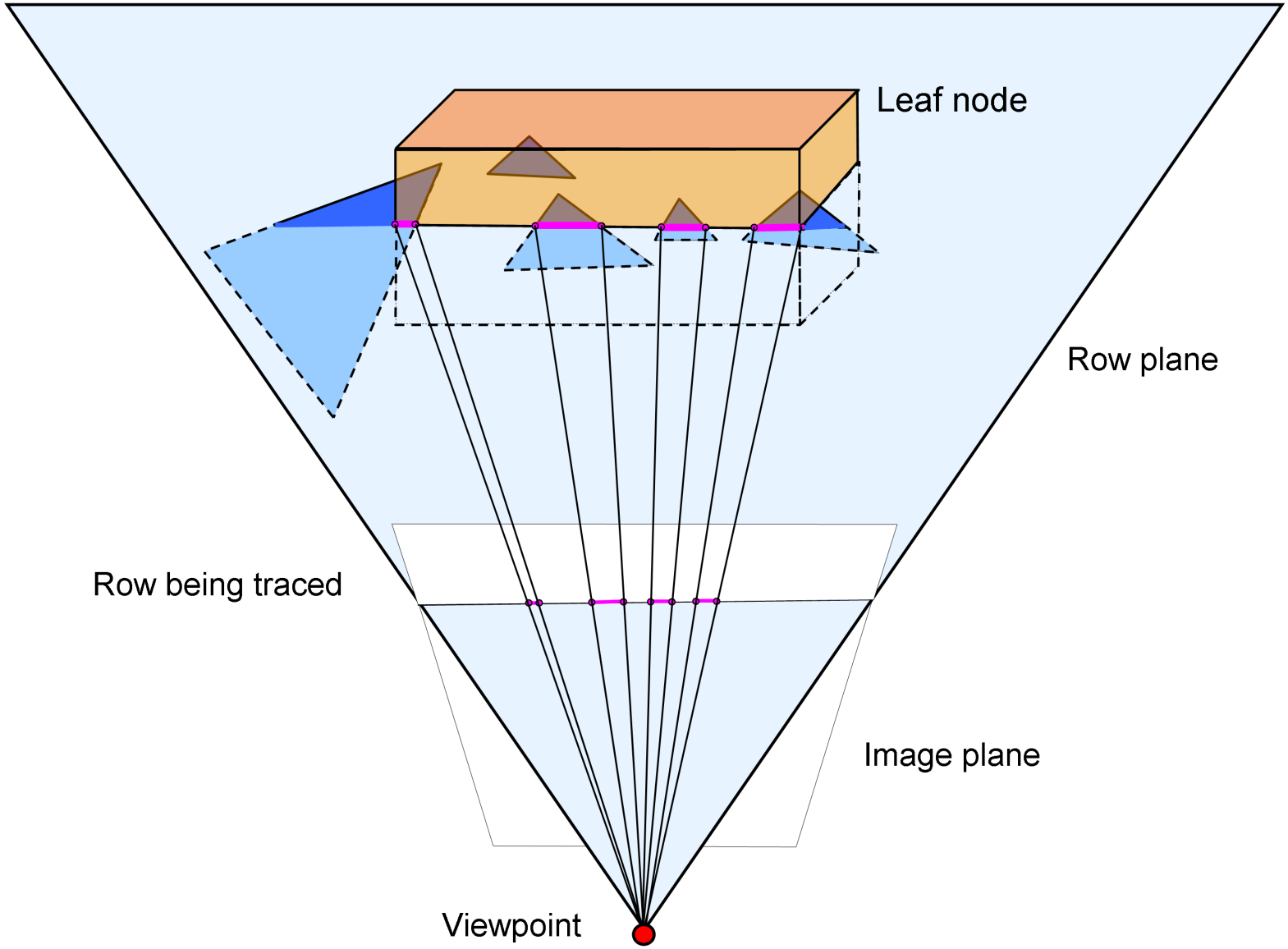
Leaf node processing

Clip intersection segments



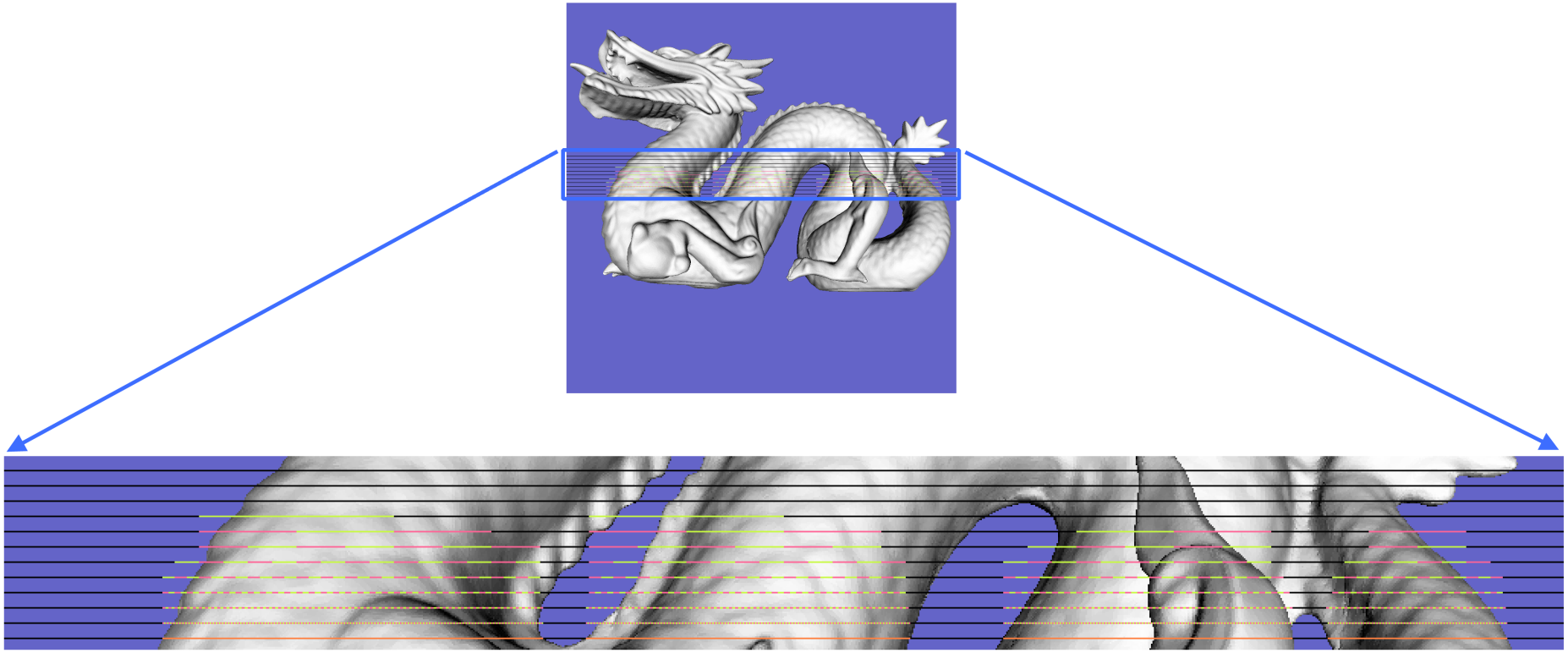
Leaf node processing

Project clipped segments



Hierarchical Occlusion Maps (HOMs)

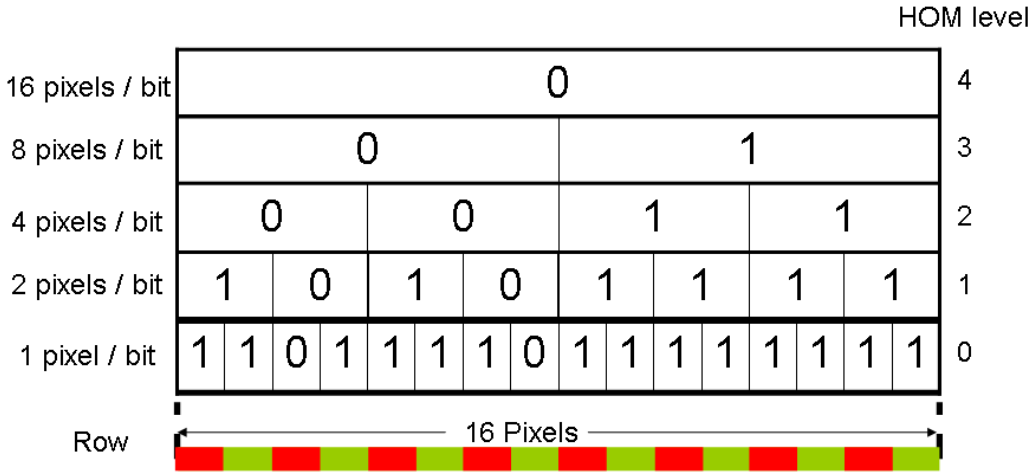
- Important optimization
  - Indicates already occluded parts of a Row
- 1D version of HOM by Zhang, et al. (1997)



HOM for Dragon model

HOMs

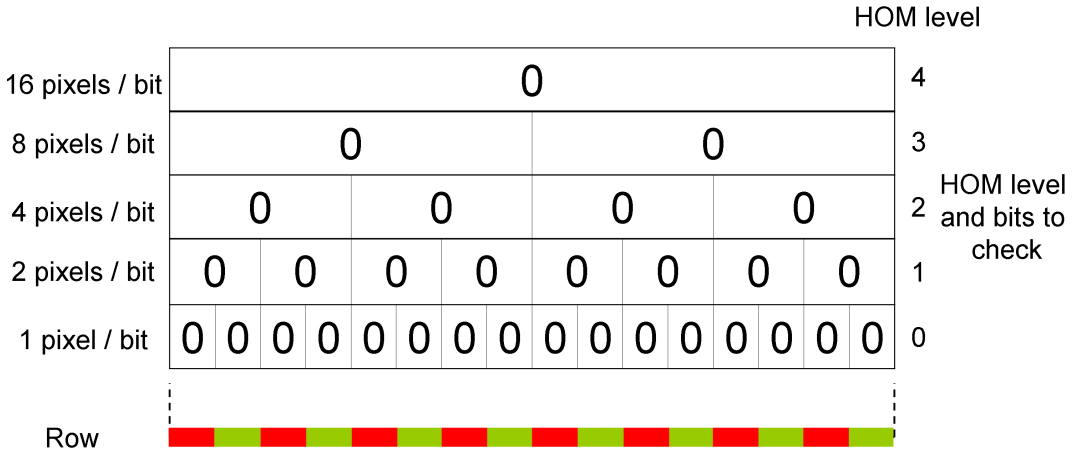
- Lowest level – 1 pixel
- Each upper level – 2 bits of lower level
- For a row with 1024 pixels, lowest level – 128 chars
- Entire HOM – 256 chars



Example HOM for a 16 pixel row

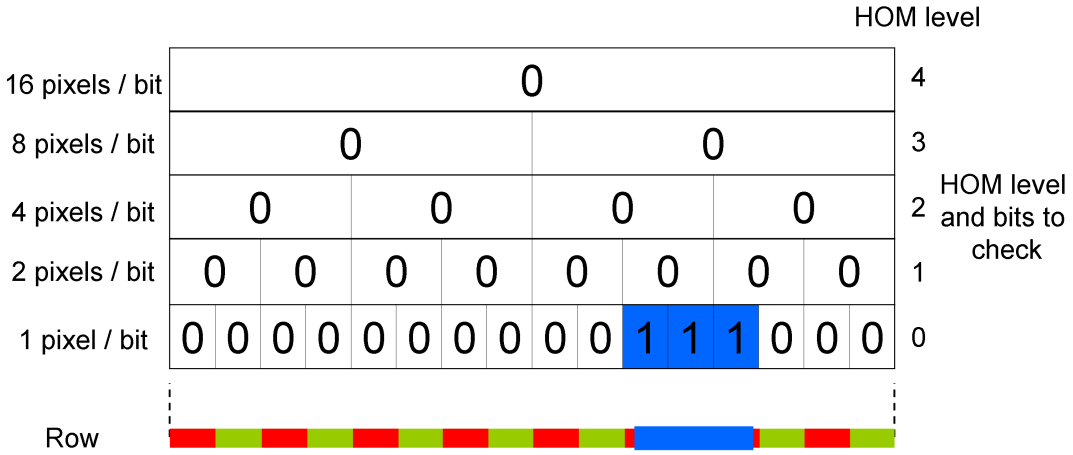
HOMs – Usage

- Initialize prior to traversal
  - Set all bits to zero
  - The entire row is unoccluded



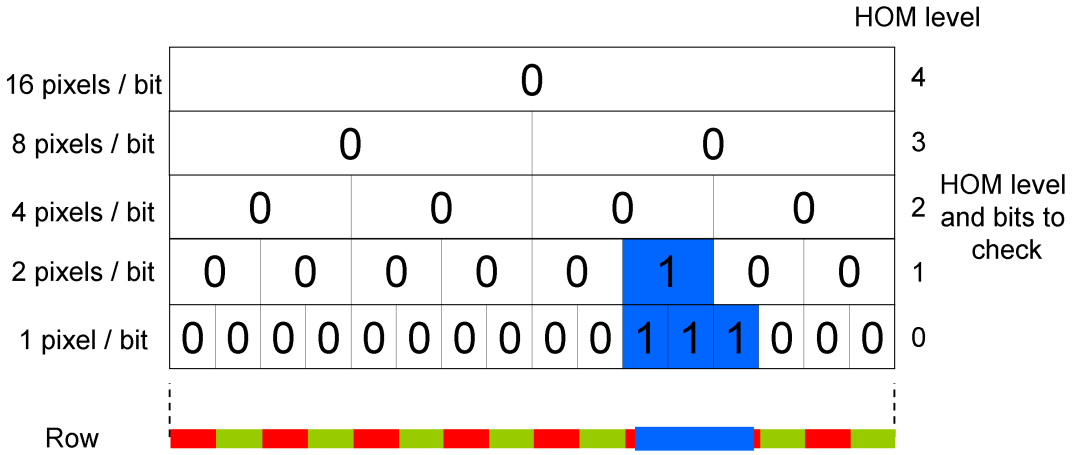
HOMs – Usage

- Initialize prior to traversal
  - Set all bits to zero
  - The entire row is unoccluded
- Updating the HOM
  - Triangles rasterization
  - Corresponding lowest level bits are set to 1



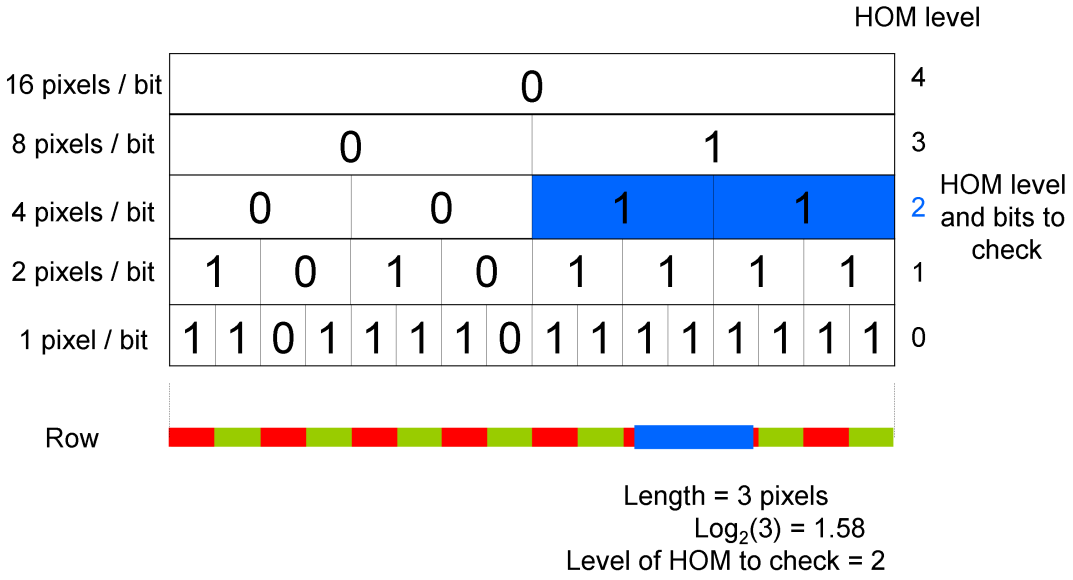
HOMs – Usage

- Initialize prior to traversal
  - Set all bits to zero
  - The entire row is unoccluded
- Updating the HOM
  - Triangles rasterization
  - Corresponding lowest level bits are set to 1
  - Upper levels updated if necessary



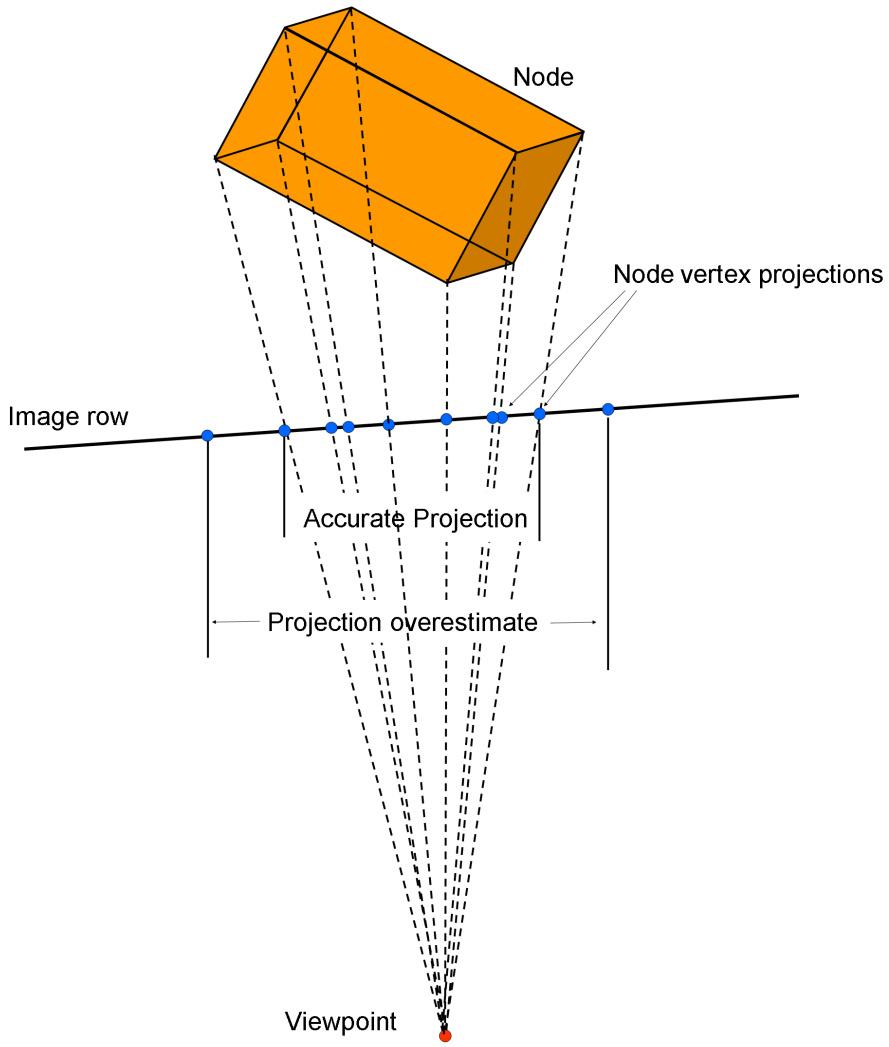
HOMs – Usage

- Initialize prior to traversal
  - Set all bits to zero
  - The entire row is unoccluded
- Updating the HOM
  - Triangles rasterization
  - Corresponding lowest level bits are set to 1
  - Upper levels updated if necessary
- Testing for Occlusion
  - Skip occluded nodes
  - Optimize rasterization



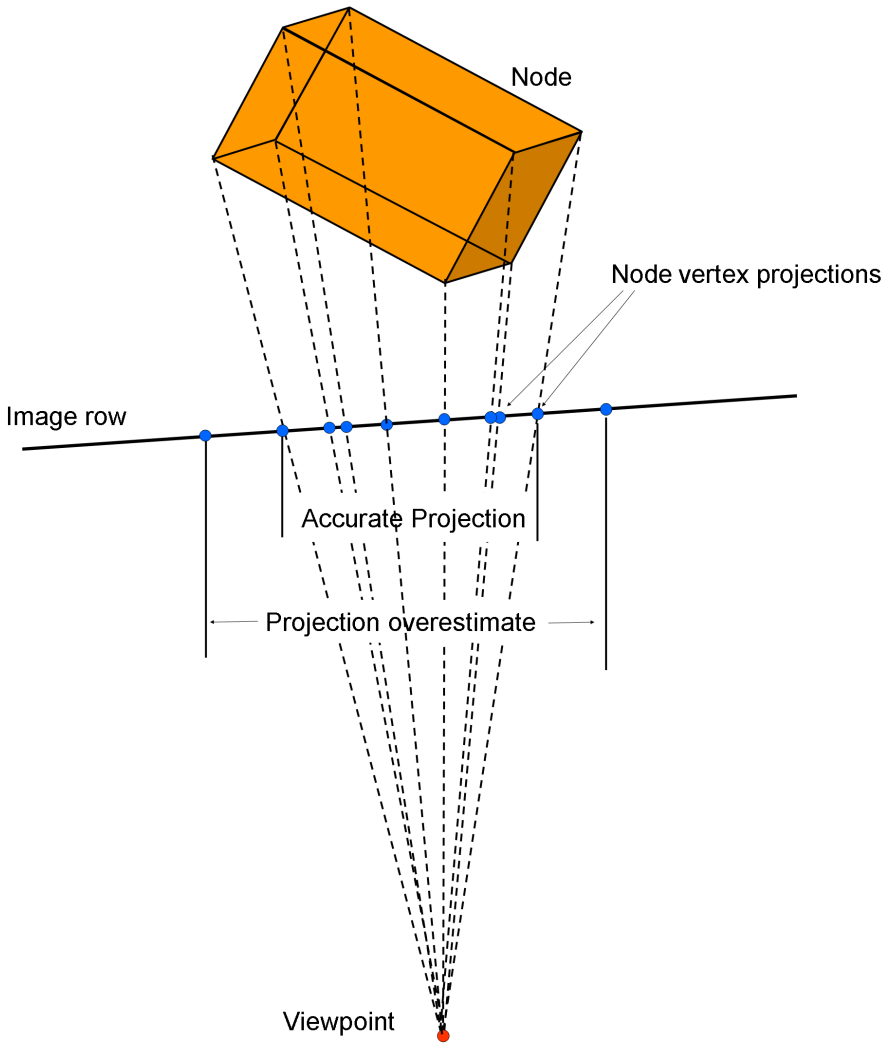
Node occlusion test

- Use HOM to find if node is occluded

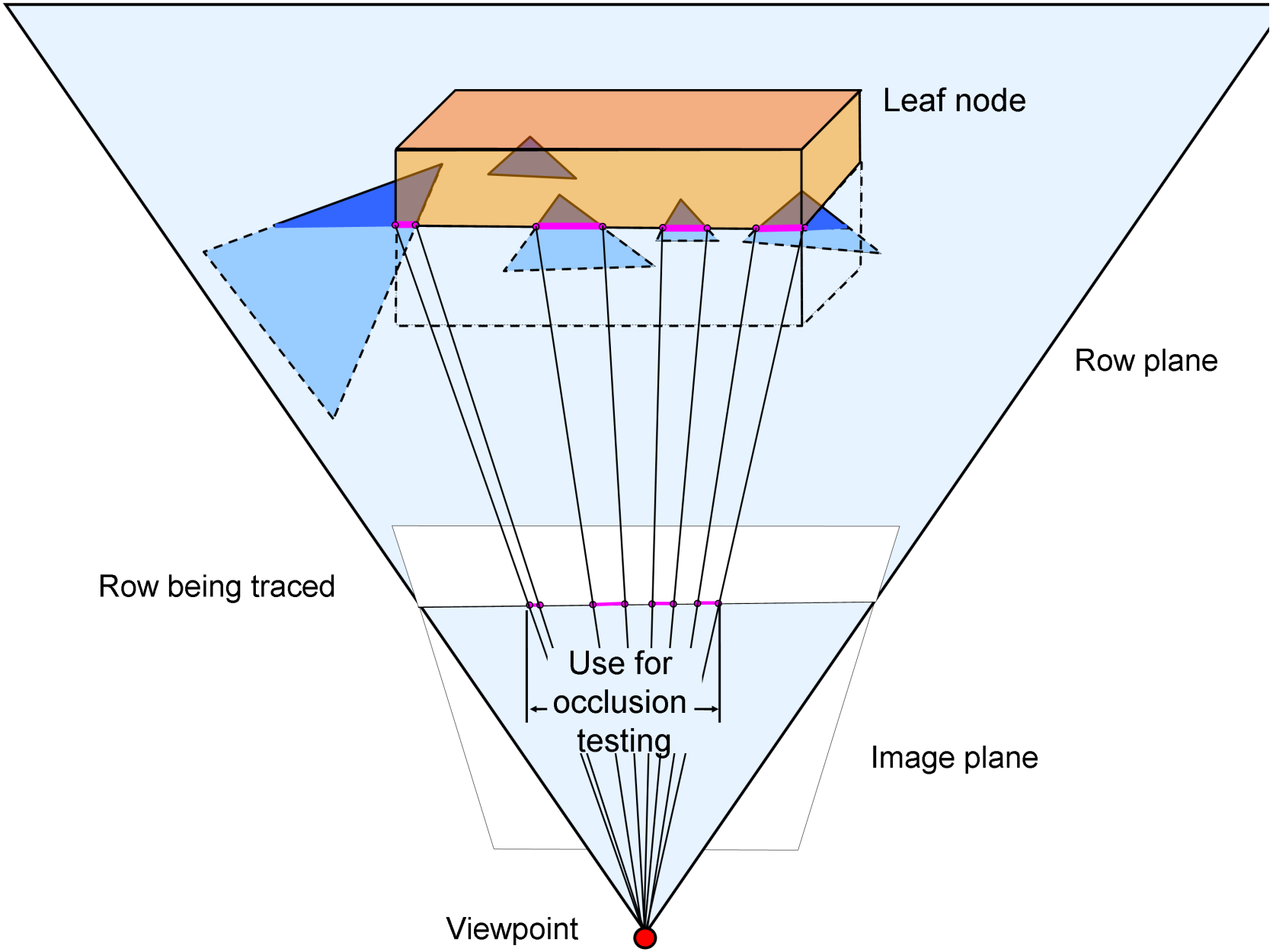


Node occlusion test

- Use HOM to find if node is occluded
- Main challenge – to find the pixels of the node projection
  - Accurate projection
    - Needs 8 vertices' projection
    - Too expensive – 16 dot products and 8 divides
  - Calculate an overestimate
  - Much cheaper – 4 dot products and 2 divides

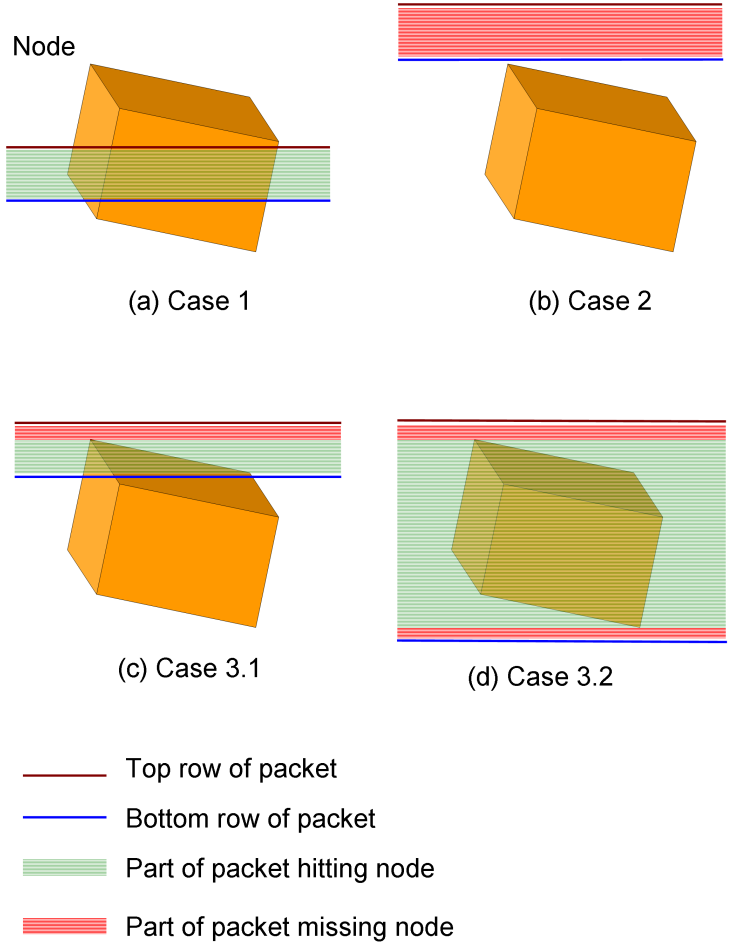


Occlusion testing leaf node triangle projections



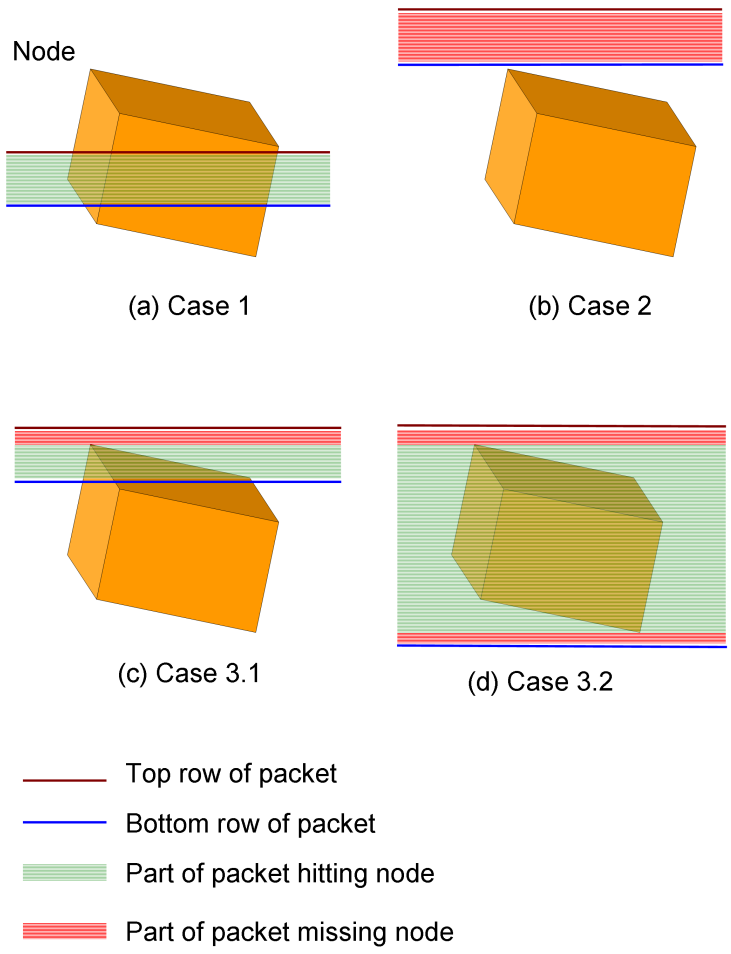
Packet Row Tracing

- Row-Packet / Node intersection
  - Case 1 – All rows in packet hit the node
  - Case 2 – Row packet misses node
  - Case 3 – Divergence nodes - Trace individual rows from these nodes



Packet Row Tracing

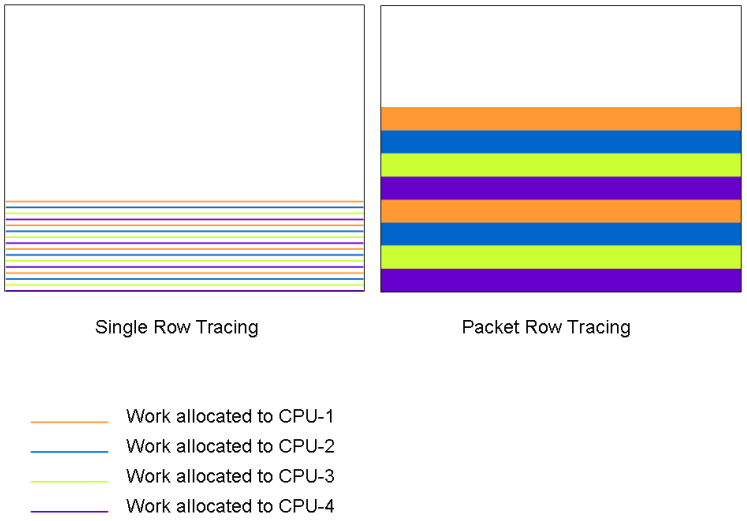
- Row-Packet / Node intersection
  - Case 1 – All rows in packet hit the node
  - Case 2 – Row packet misses node
  - Case 3 – Divergence nodes - Trace individual rows from these nodes
- Occlusion testing – Test each row individually
- Leaf node – All rows are rasterized with leaf node's triangles



SSE and Muti-threading

- SSE used to optimize key parts of algorithm
  - Triangle intersections
  - Projection calculation

- Multi-threading
  - Good load balancing – essential to obtain best speed-ups
    - Round robin allocation of work



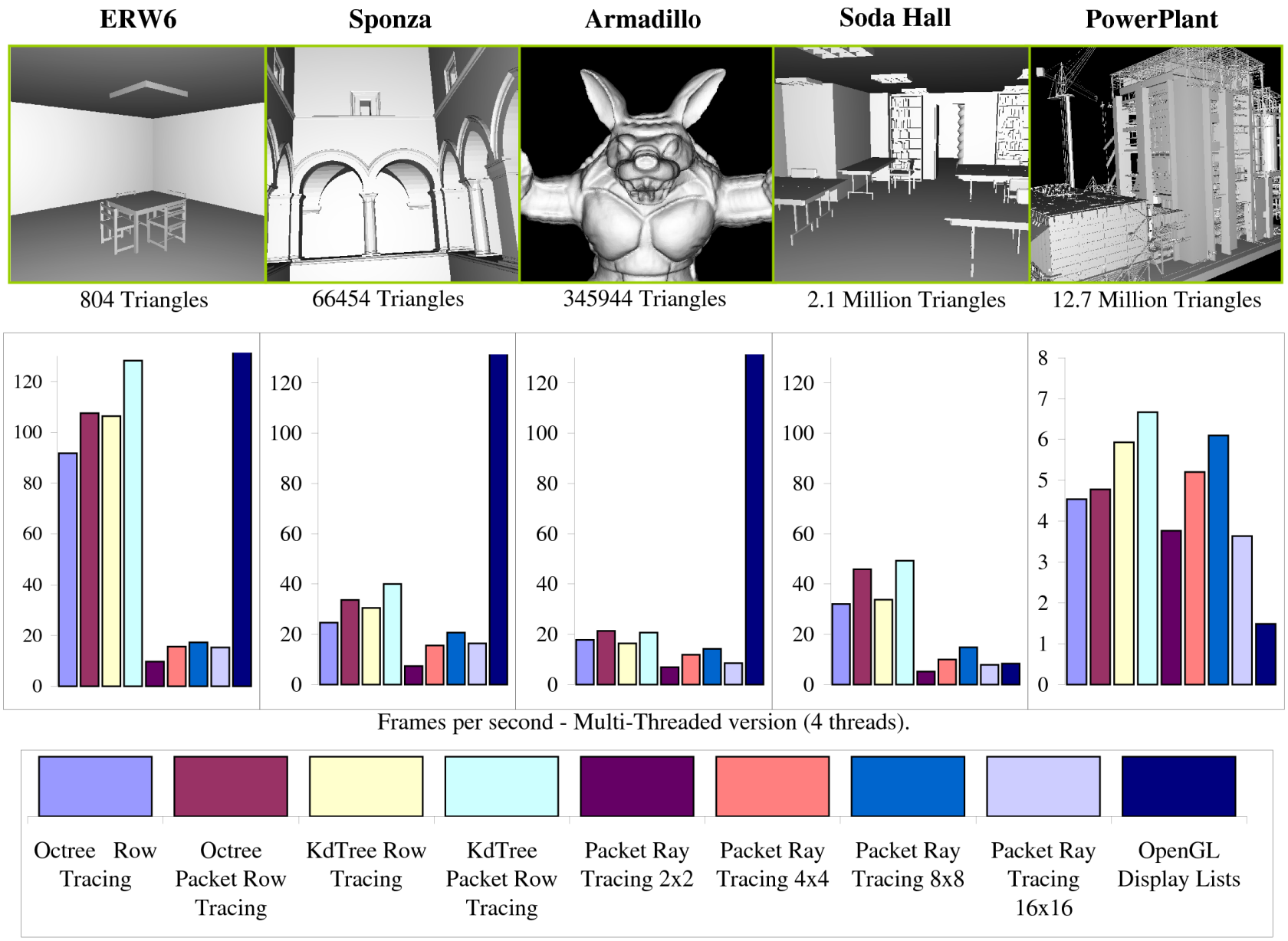
Load balancing for a Quad core processor.

Results

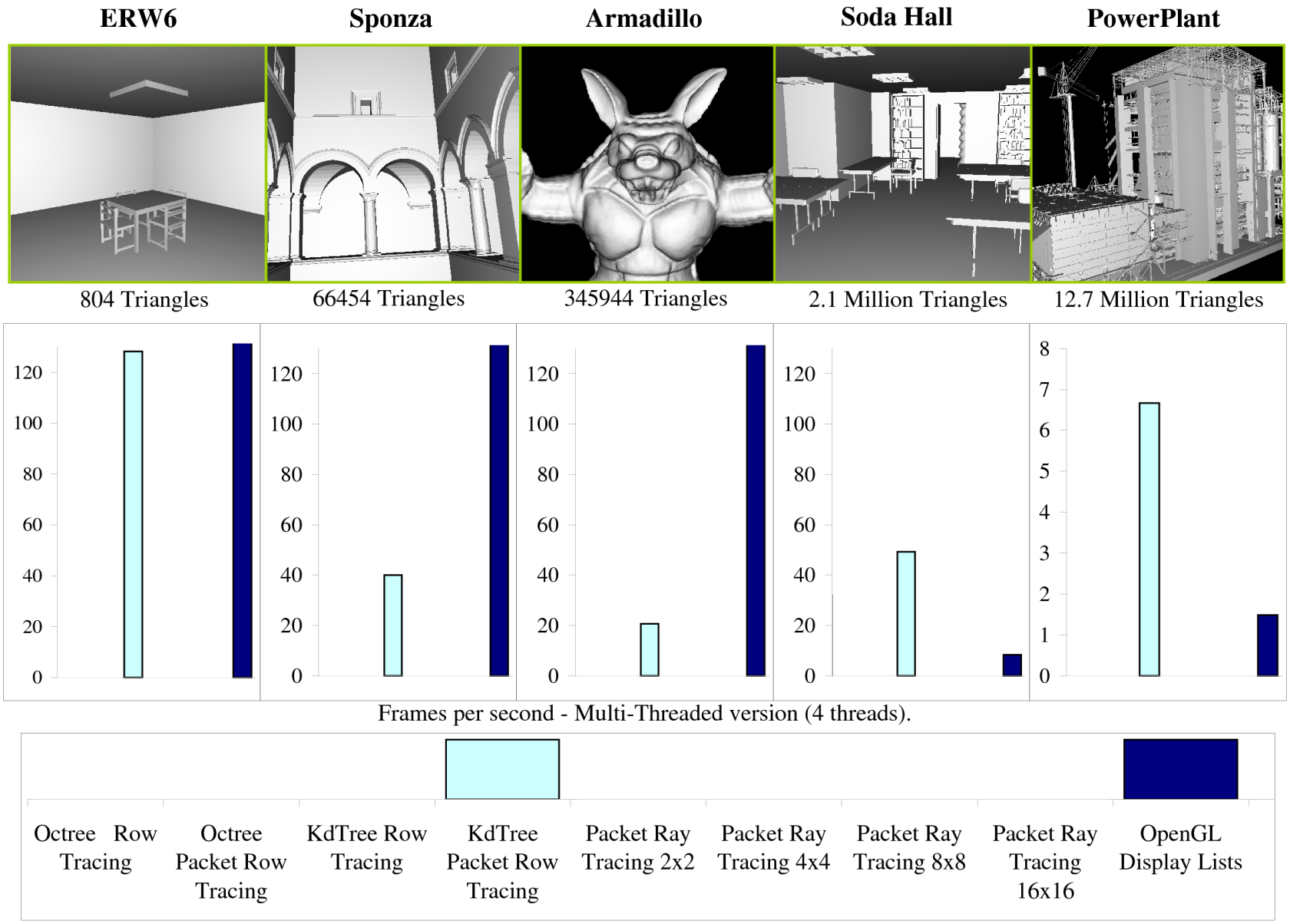
ERW6	Sponza	Armadillo	Soda Hall	PowerPlant
804 Triangles	66454 Triangles	345944 Triangles	2.1 Million Triangles	12.7 Million Triangles

- Intel Core 2 Quad 2.4 GHz – 4 threads
- Geforce 8800 GTX with 768 MB Video RAM
- Performance comparison
  - Packet ray tracing (2×2, 4×4, 8×8, 16×16 rays)
  - OpenGL with display lists

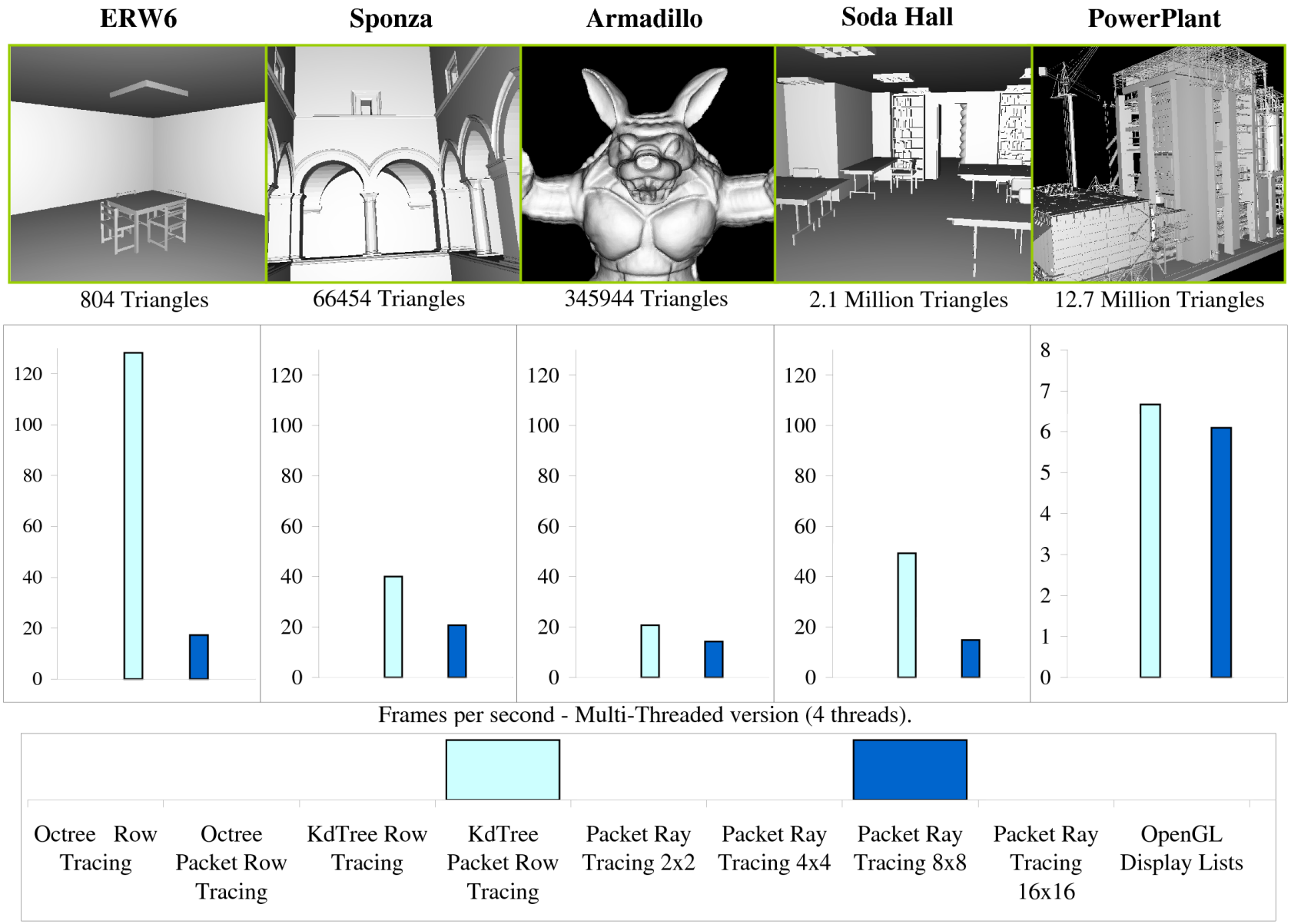
Results


ERW6
Sponza
Armadillo
Soda Hall
PowerPlant
804 Triangles
66454 Triangles
345944 Triangles
2.1 Million Triangles
12.7 Million Triangles
0 20 40 60 80 100 120
0 1 2 3 4 5 6 7 8
Frames per second - Multi-Threaded version (4 threads).
Octree Row Tracing
Octree Packet Row Tracing
KdTree Row Tracing
KdTree Packet Row Tracing
Packet Ray Tracing 2x2
Packet Ray Tracing 4x4
Packet Ray Tracing 8x8
Packet Ray Tracing 16x16
OpenGL Display Lists

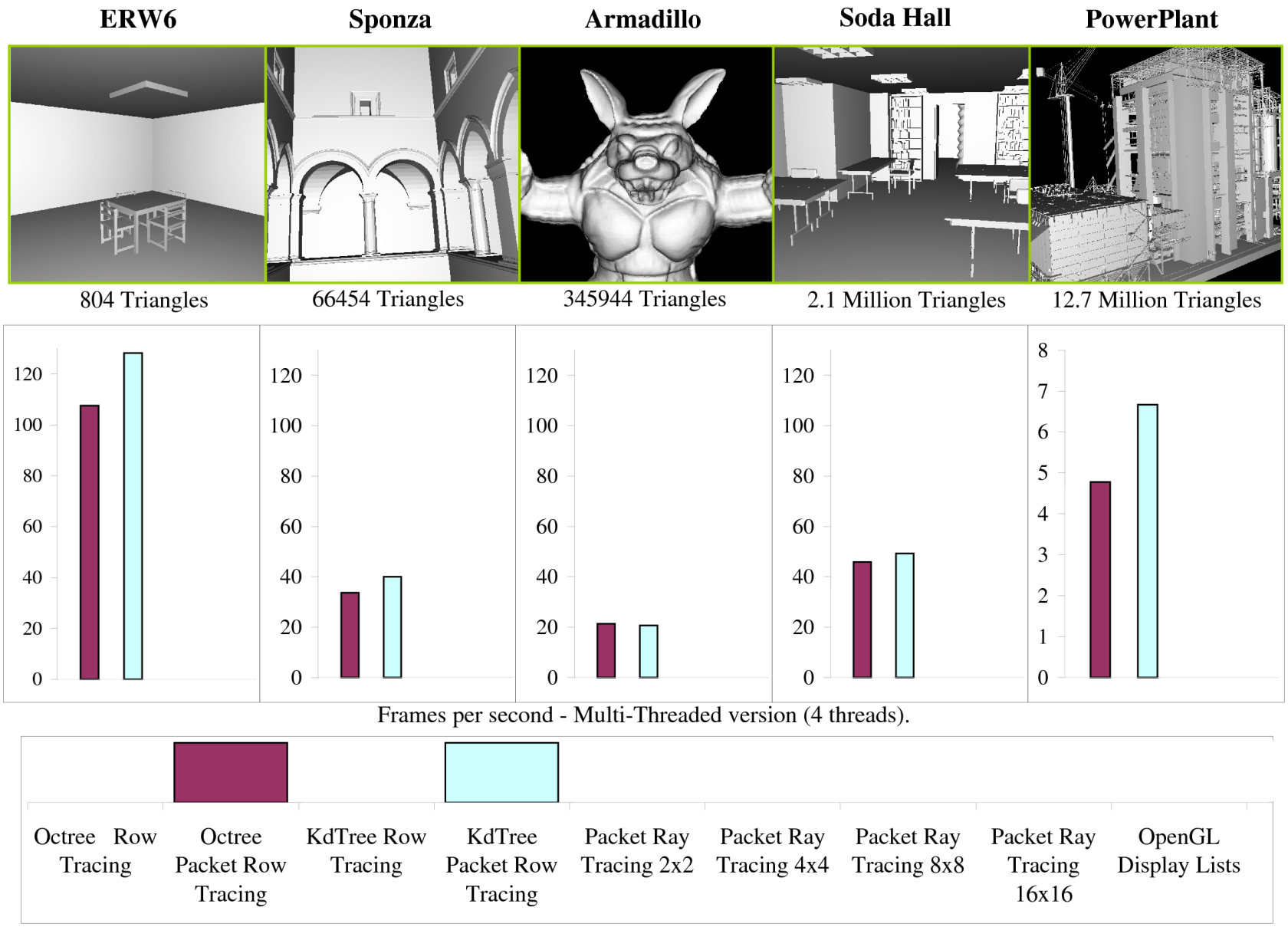
Results – Fastest version of Row tracing vs OpenGL



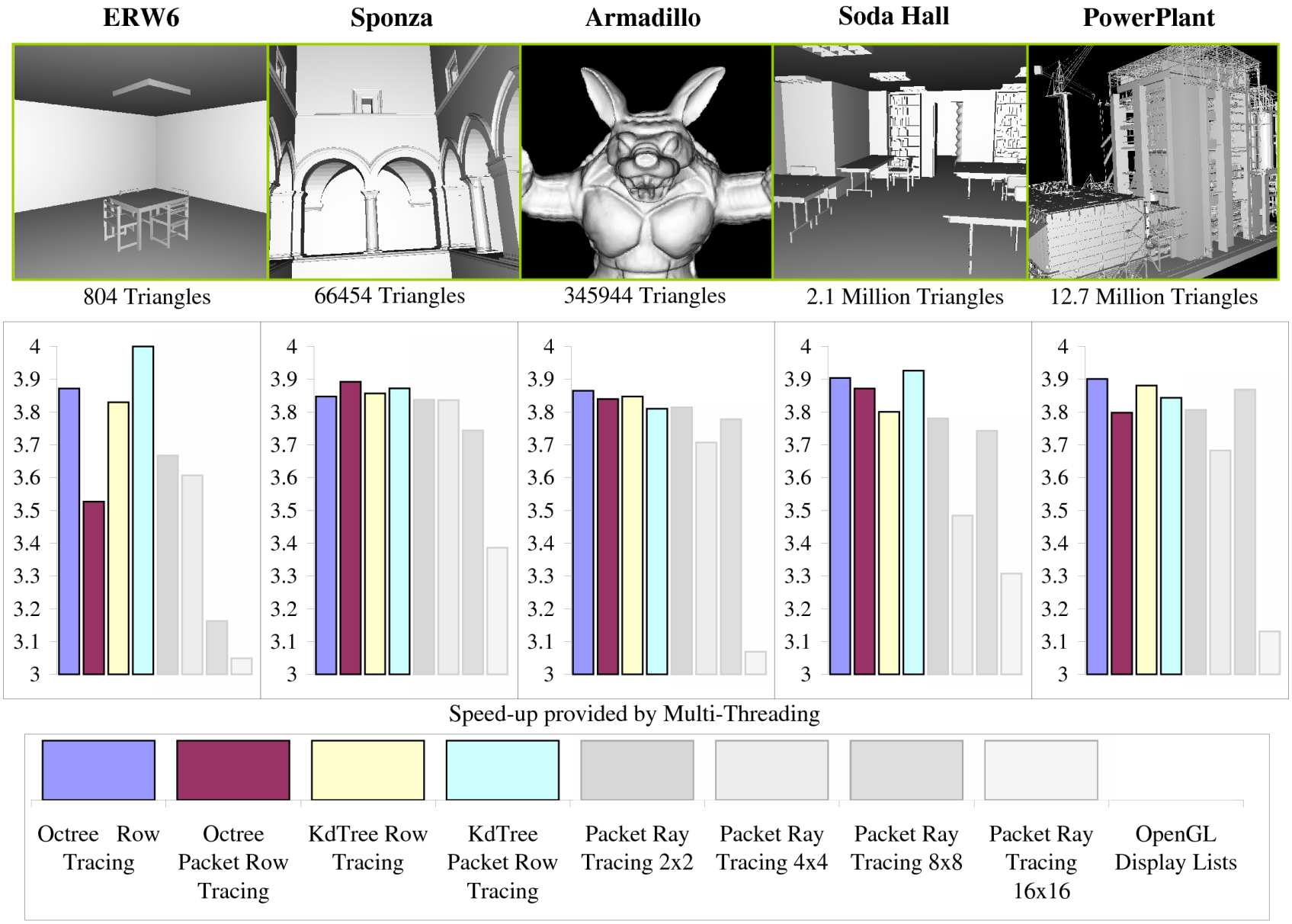
Results – Fastest versions of Row tracing vs Ray tracing



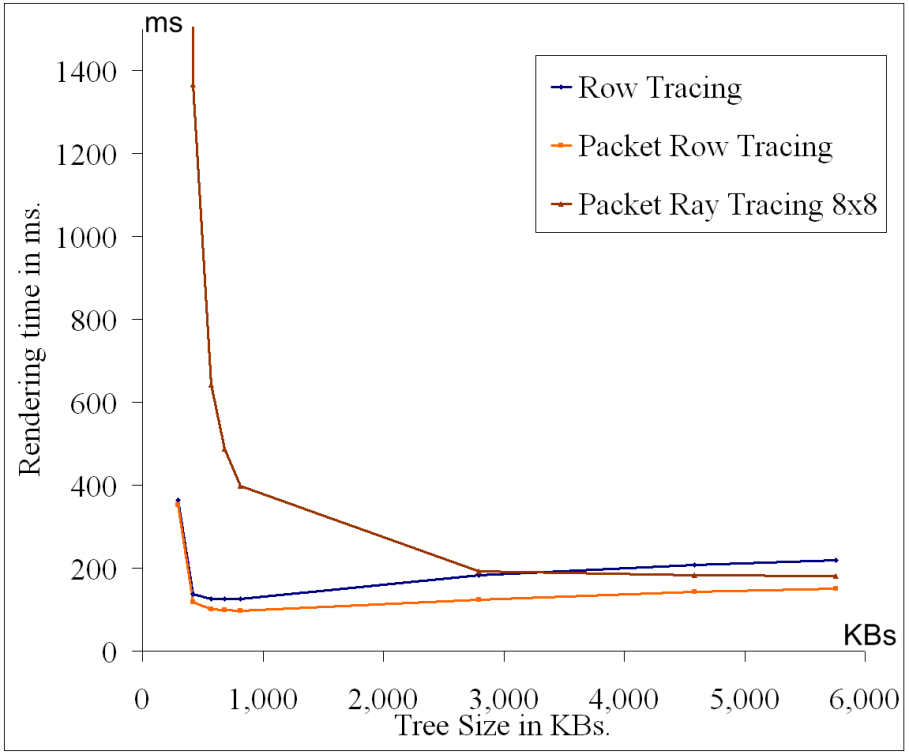
Results - Performance on Octrees

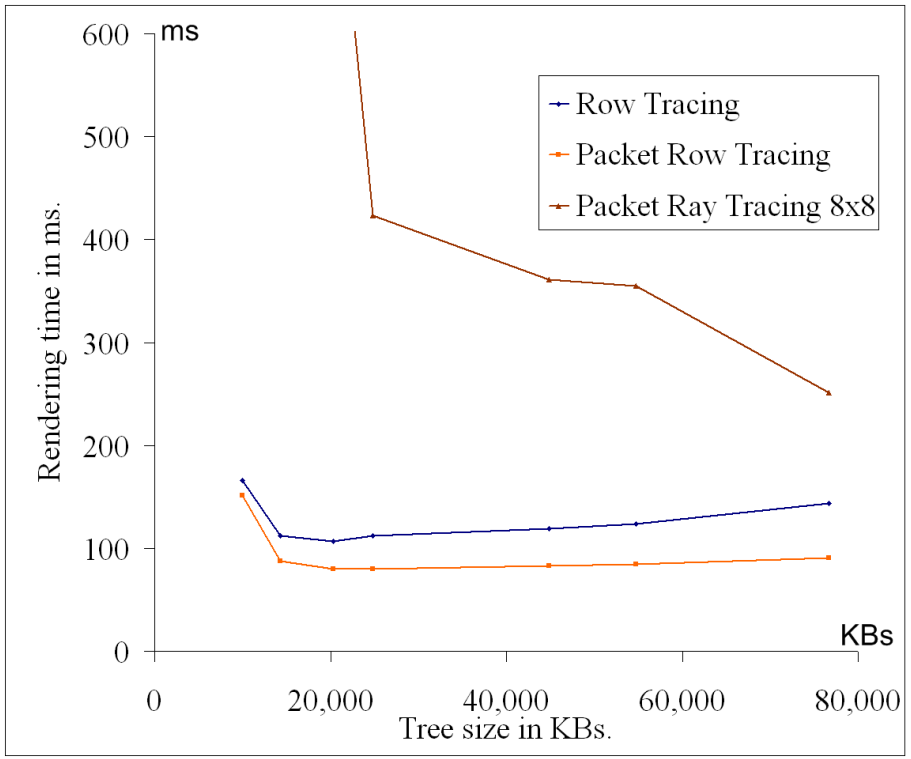


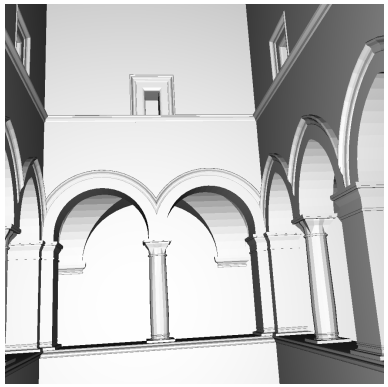
Results - Scalability over number of Cores



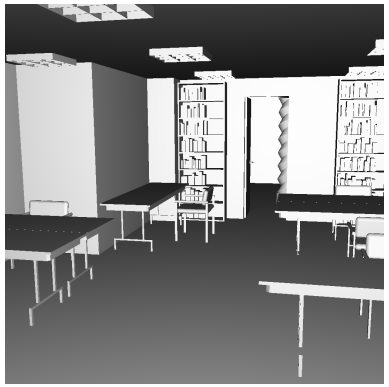
Results - Performance vs Tree Size







a) Sponza



b) Sodahall

Conclusion and Future work

Conclusion

- Row tracing combines concepts from ray tracing and rasterization
- Low per pixel cost
- Excellent performance
- Works well on smaller trees
- Easy and effective parallelization – near perfect speed-up achieved

Future work

- Further optimization – faster row packet occlusion testing
- Dynamic scenes – combine row tracing with fast construction methods
- Shadow rays – shadow mapping technique

Acknowledgements

- I.Wald
  - Models and information about importing models.
- Stanford Computer Graphics Laboratory
  - Models.
  - PLY import libraries.
- Eric Haines et al.
  - Standard Procedural Databases set of models.

Questions?