Grid Creation Strategies for Efficient Ray Tracing

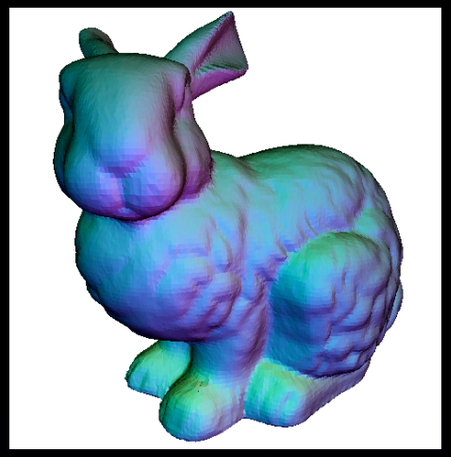
or

How to pick the best grid resolution

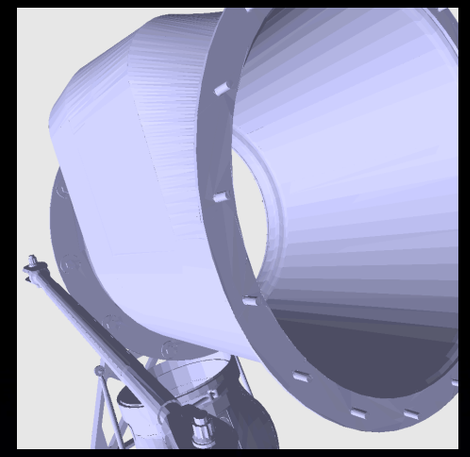
Thiago Ize
Peter Shirley
Steven G. Parker

THE UNIVERSITY OF UTAH

SCI INSTITUTE



Motivation



- Single level grids mostly solved (Cleary and Wyvill ‘89) -- O(n) cells should normally work
- Not much discussion on long skinny triangles
- Guess-and-check was the recommended method for multi-level grids
- Given the number of triangles in the grid and the current grid level, would like a formula that returns the optimal grid resolution

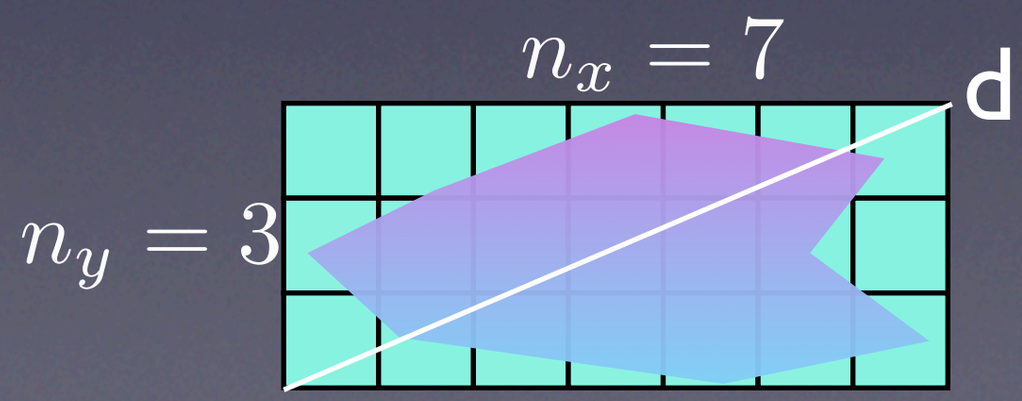
Uniform Grid Acceleration Structure

- We use the standard 3D uniform grid for ray tracing (examples are in 2D for simplicity)
- Use cubical shaped cells
- Choose number of cells in each dimension according to:
  d is the diagonal of the object (length)
  N is number of triangles (primitives)
  k is a user supplied parameter

$$n_x = d_x \sqrt[3]{\frac{kN}{d_x d_y d_z}}$$

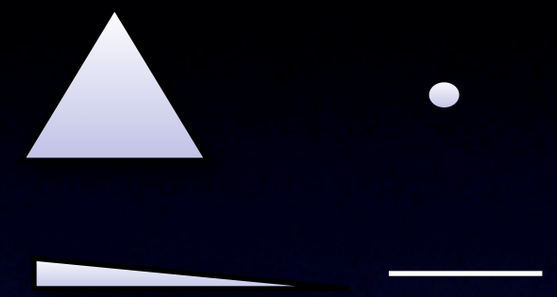
$$n_y = d_y \sqrt[3]{\frac{kN}{d_x d_y d_z}}$$

$$n_z = d_z \sqrt[3]{\frac{kN}{d_x d_y d_z}}$$

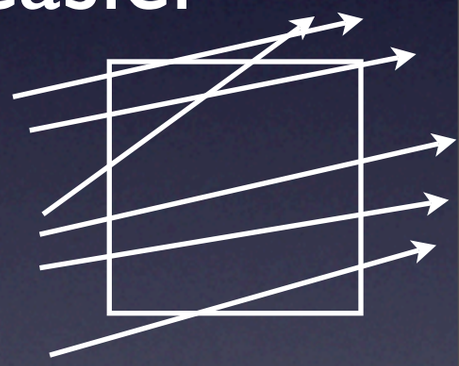


Our approach

- Assume compact triangles are points
- Assume long skinny triangles are lines
- Points and lines have zero probability of being intersected -- makes the derivations *much* easier

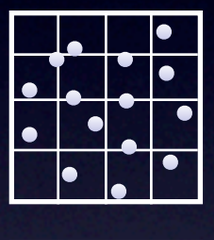


- All rays hit grid and are equally likely
- Note: we use points and lines only for our derivation, we still care only about ray tracing triangles



Scenes we solve for

We find solutions for these simple scenes and later apply it to actual triangle scenes (note these are supposed to be 3D)

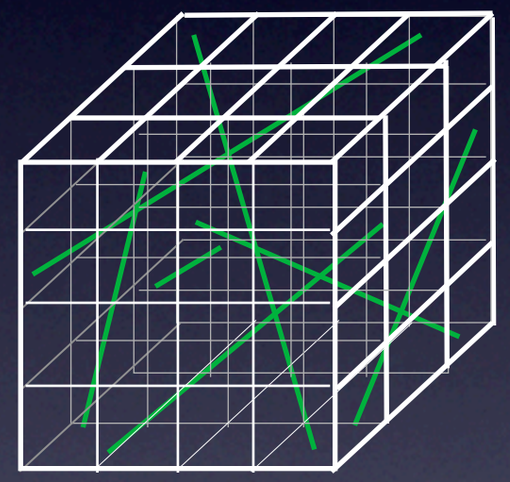


Arbitrarily scattered points

Arbitrarily scattered lines

Multi-level grids made up of points on surface

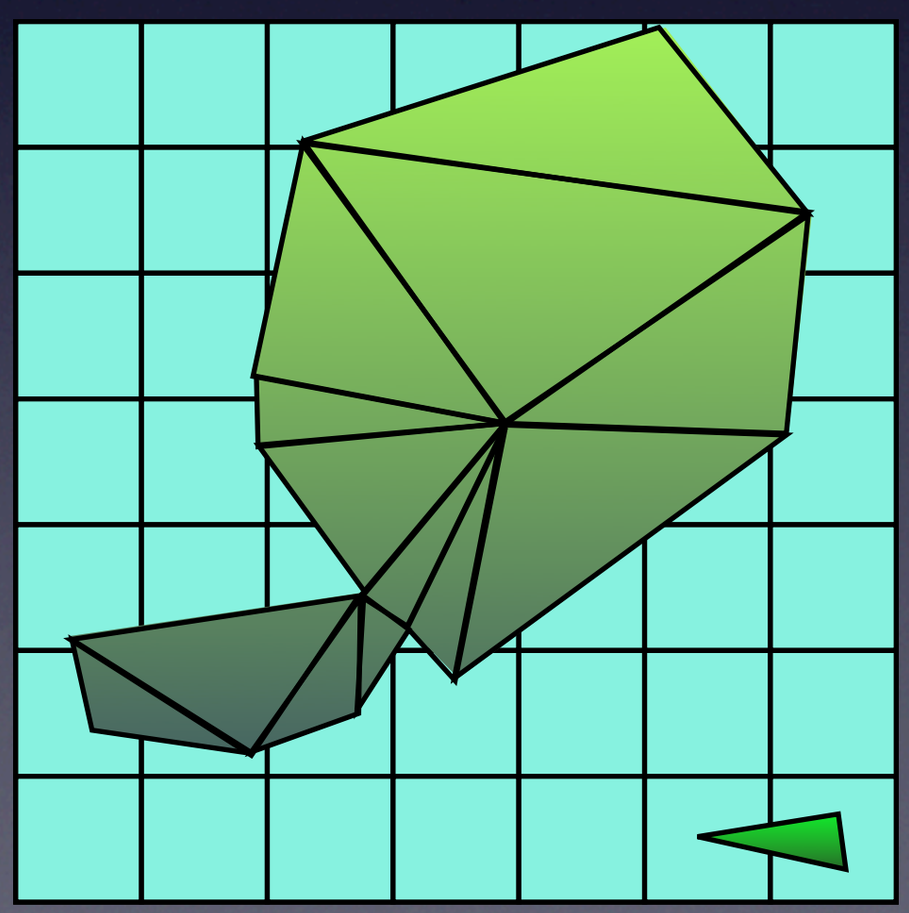
Multi-level grids made up of lines on surface



Ray/Grid cost model

- Time for a ray to enter grid, traverse, and intersect objects :

$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

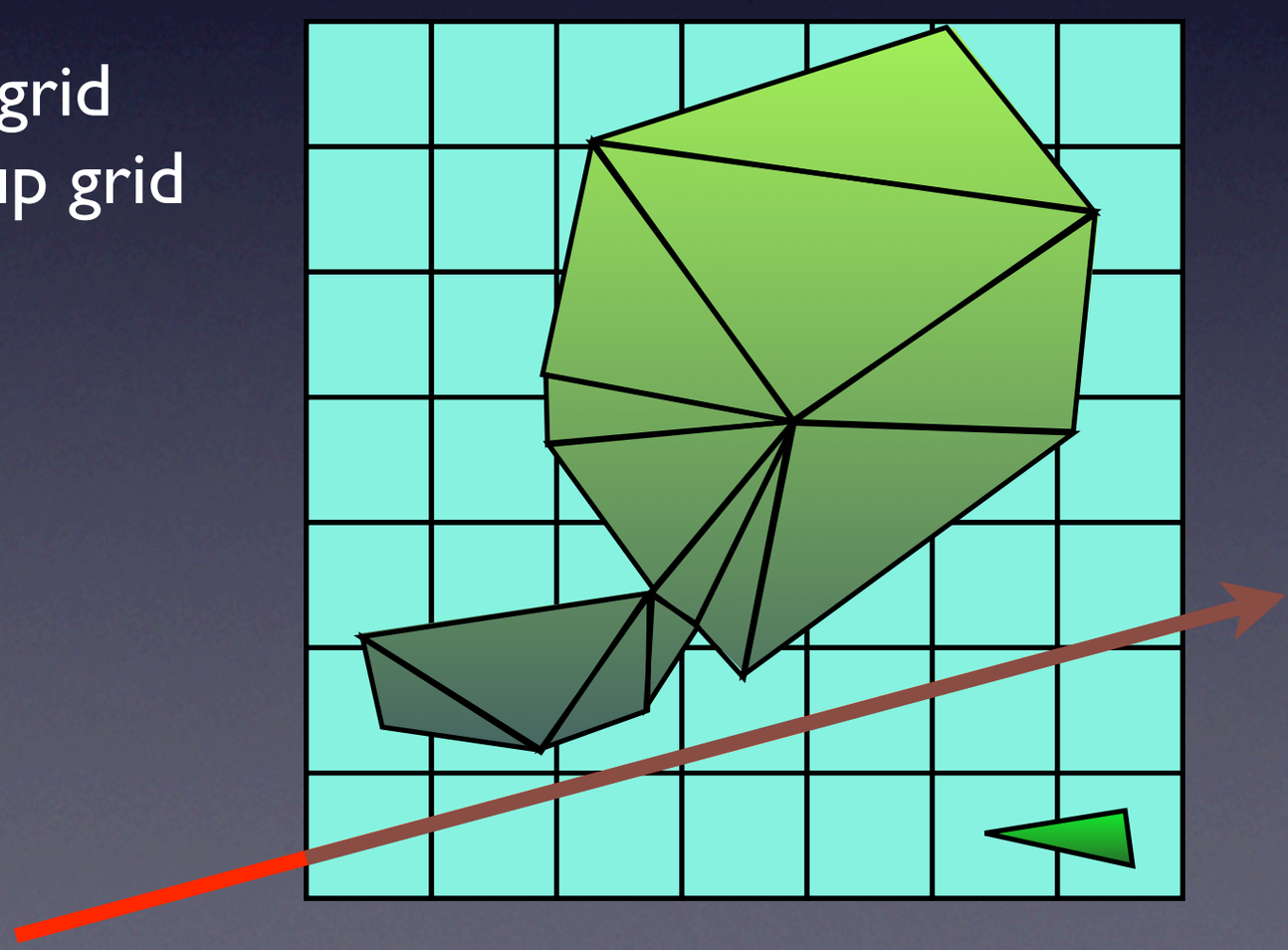


Ray/Grid cost model

- Time for a ray to enter grid, traverse, and intersect objects :

$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

T_{setup} :Time to intersect grid bounding box and setup grid traversal



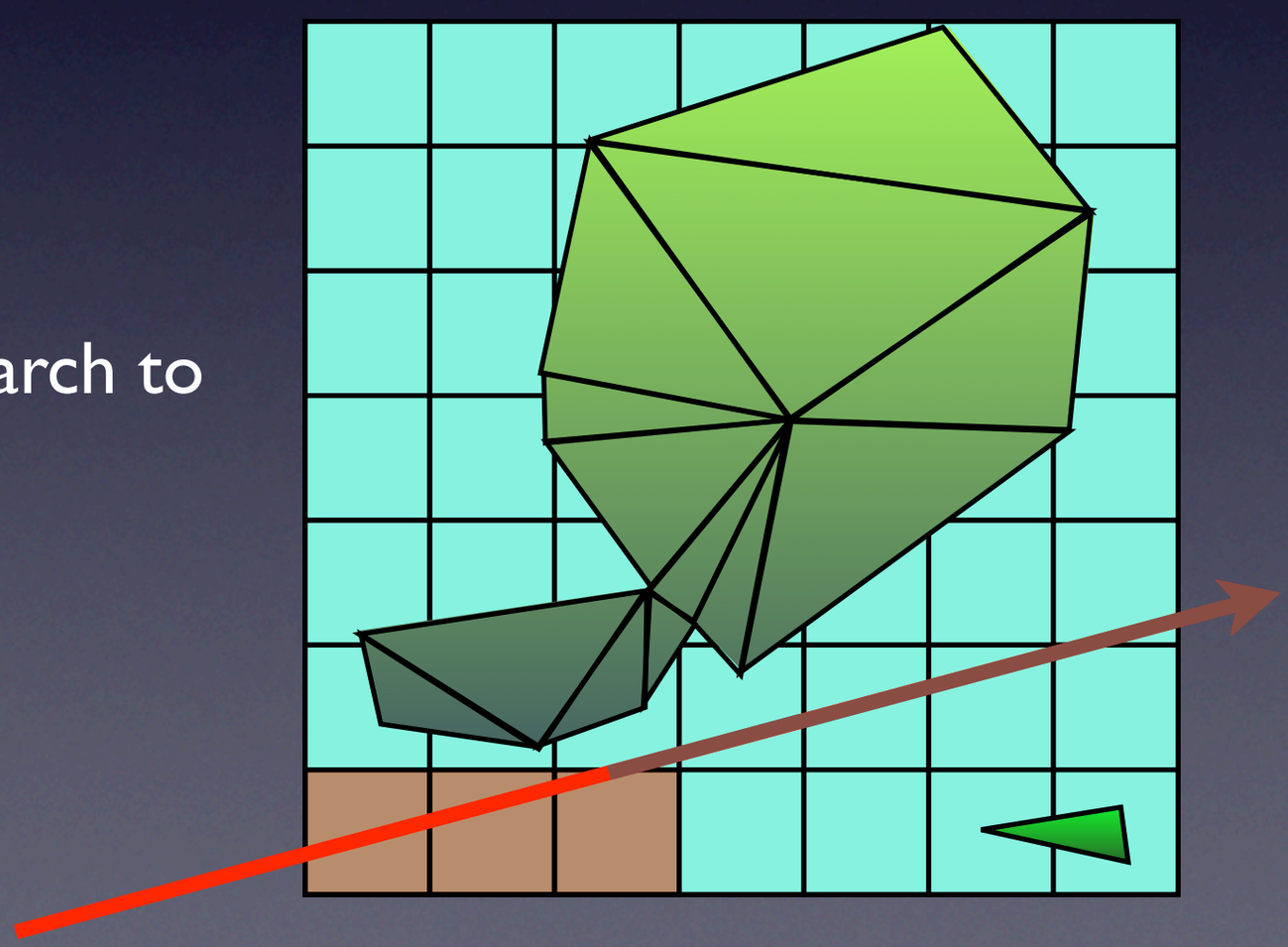
Ray/Grid cost model

- Time for a ray to enter grid, traverse, and intersect objects :

$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

3

T_{step} :Time for ray to march to next cell



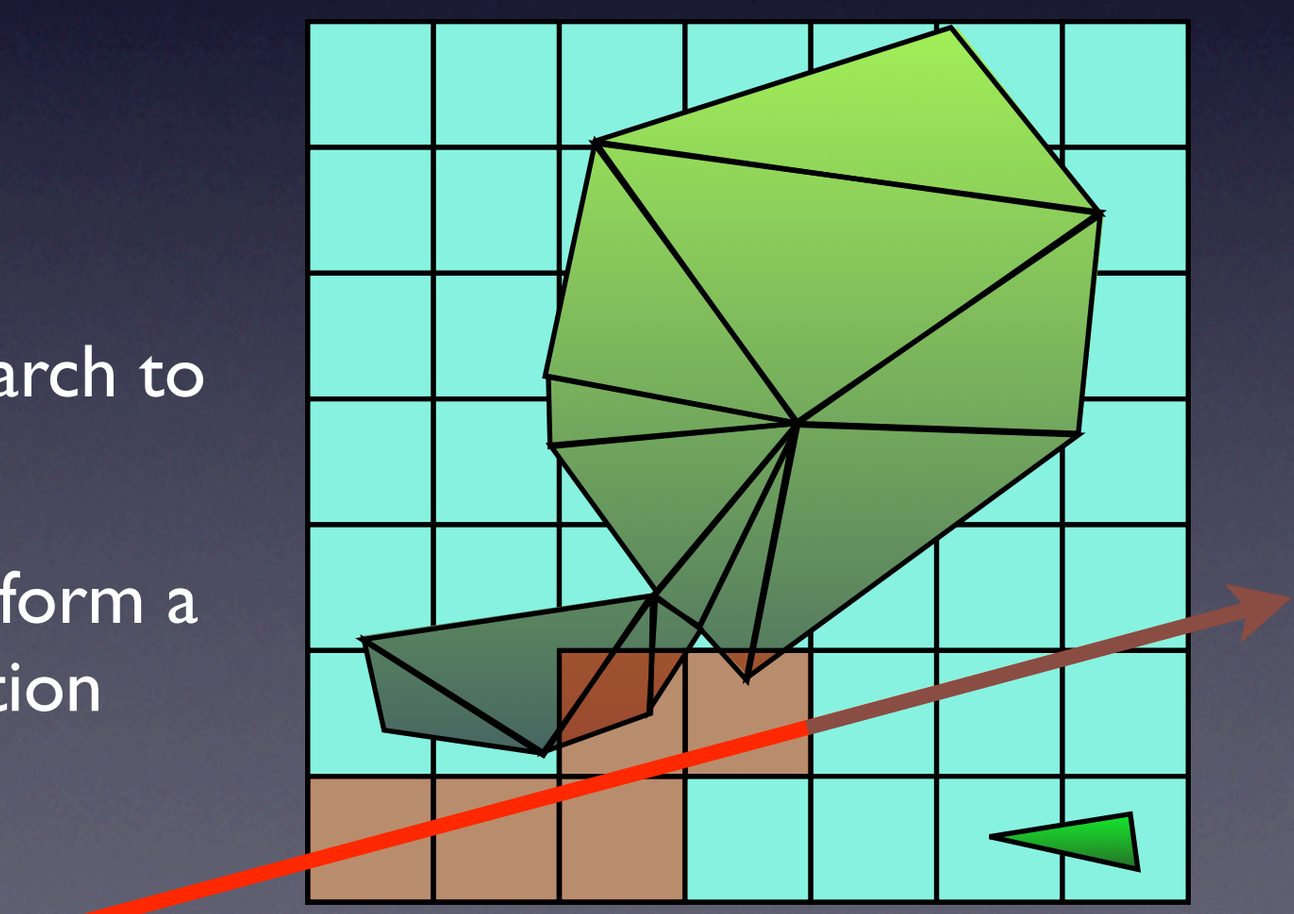
Ray/Grid cost model

- Time for a ray to enter grid, traverse, and intersect objects :

$$T = T_{\text{setup}} + \underbrace{(\#\text{ cells traversed})T_{\text{step}}}_{5} + \underbrace{(\#\text{ intersection tests})T_{\text{intersection}}}_{5}$$

T_{step} :Time for ray to march to next cell

$T_{\text{intersection}}$:Time to perform a ray/primitive intersection



Ray/Grid cost model

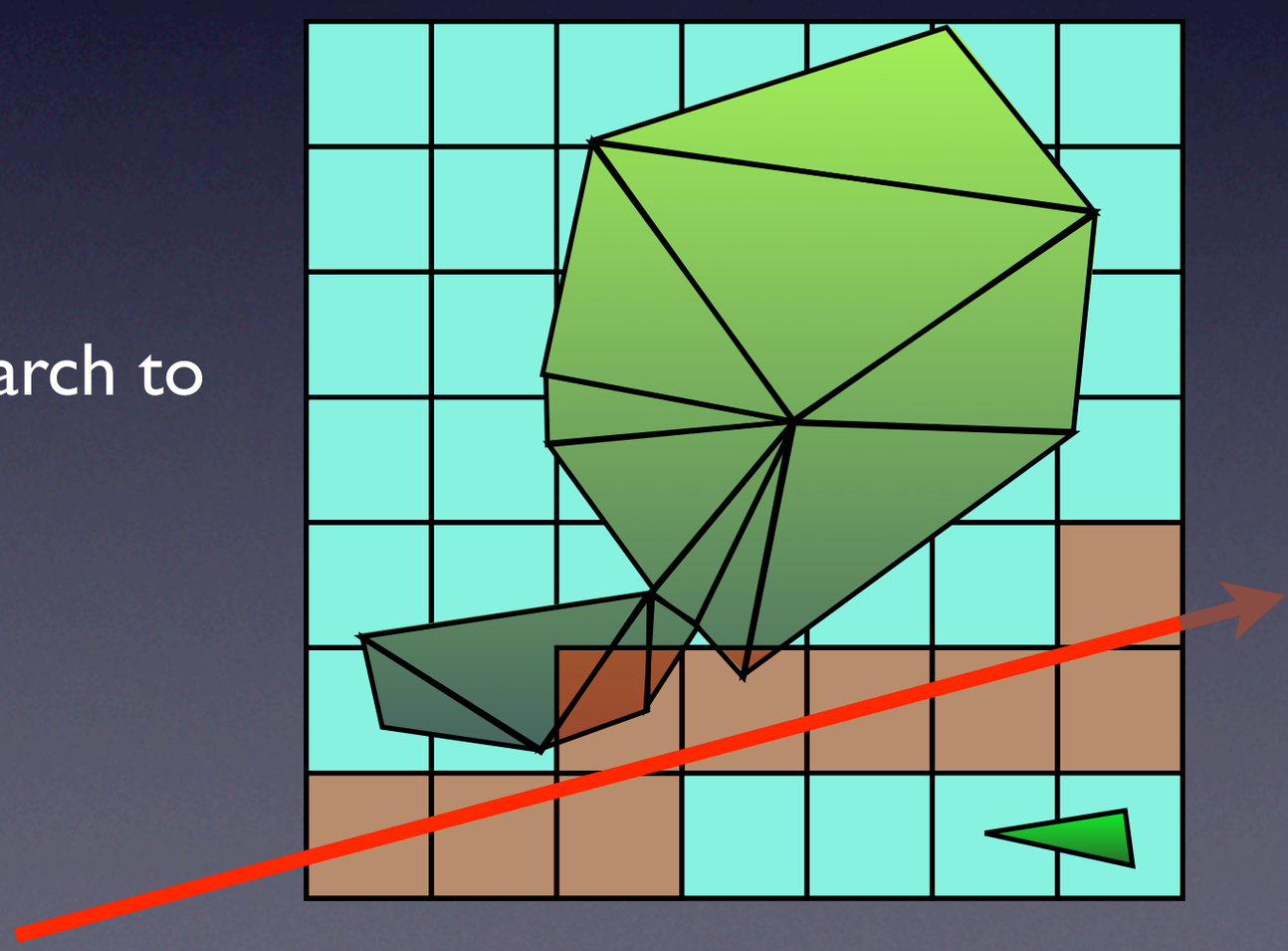
- Time for a ray to enter grid, traverse, and intersect objects :

$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

9

5

T_{step} :Time for ray to march to next cell



Ray/Grid cost model

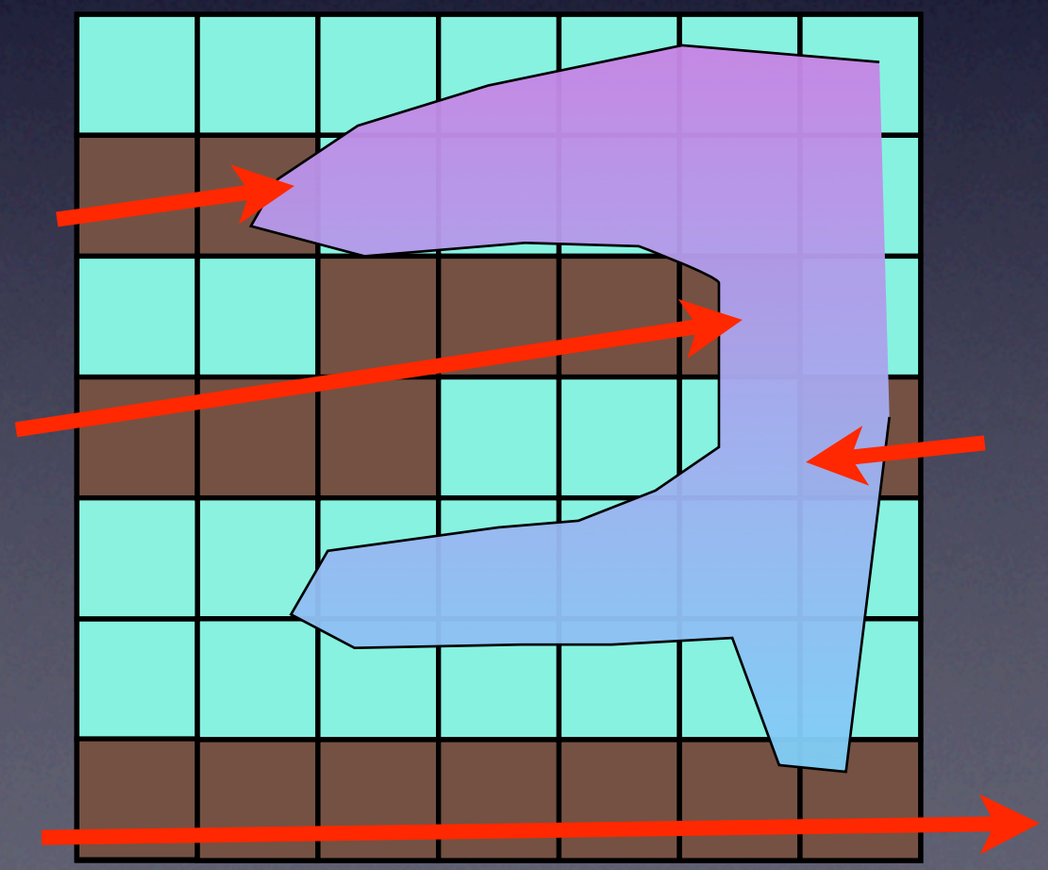
$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

- Assume T_{setup}, T_{step}, and $T_{\text{intersection}}$ are constants
- Can empirically find average values
- Even easier, we'll find that in the end we only need to find a single value, which is a ratio of the above parameters

Ray/Grid cost model: # cells traversed

$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

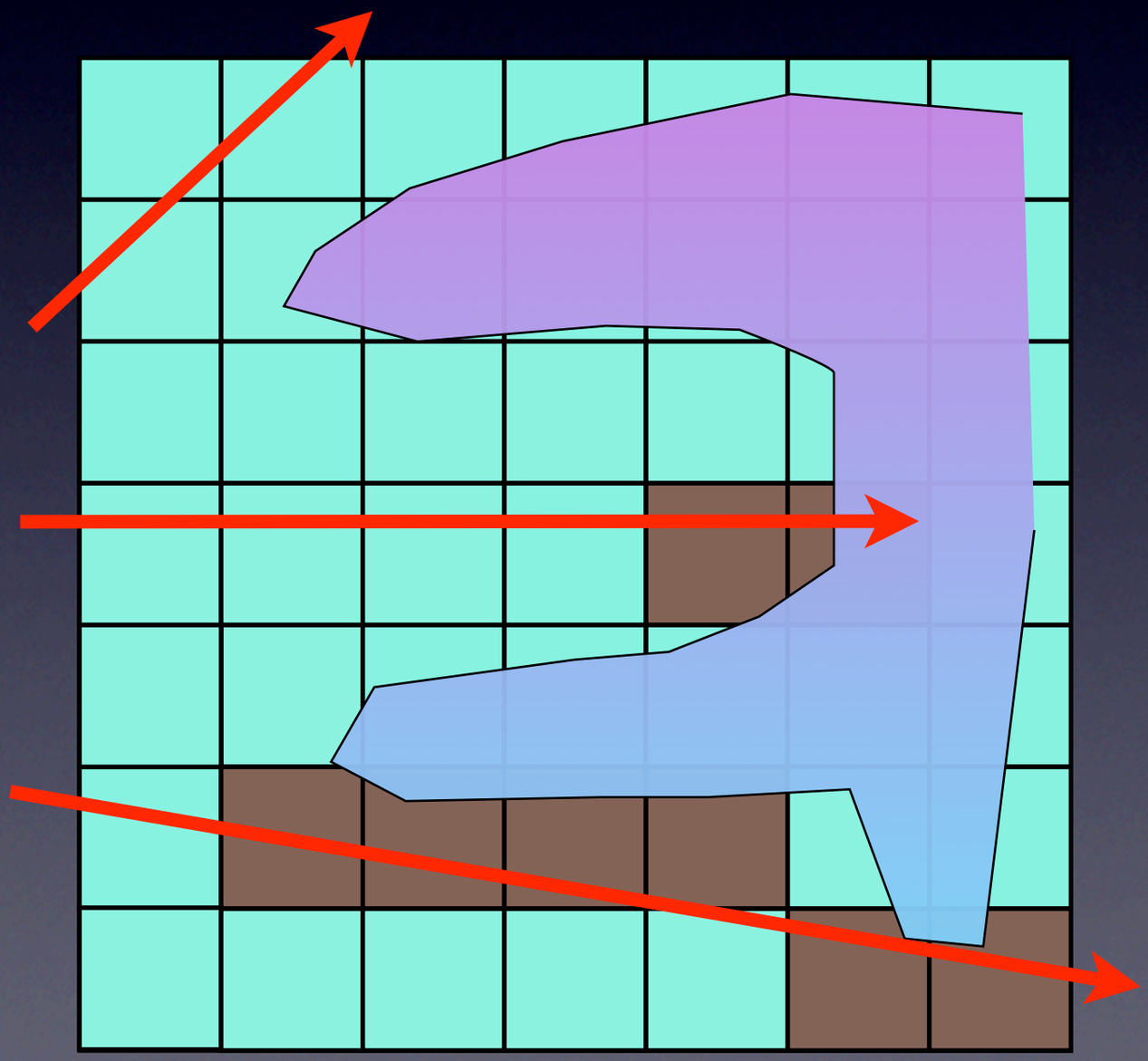
- Very complicated to find actual value
  - Cell traversal stops when primitive is hit
  - Varies with scene and camera view



Ray/Grid cost model: # intersection tests

$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

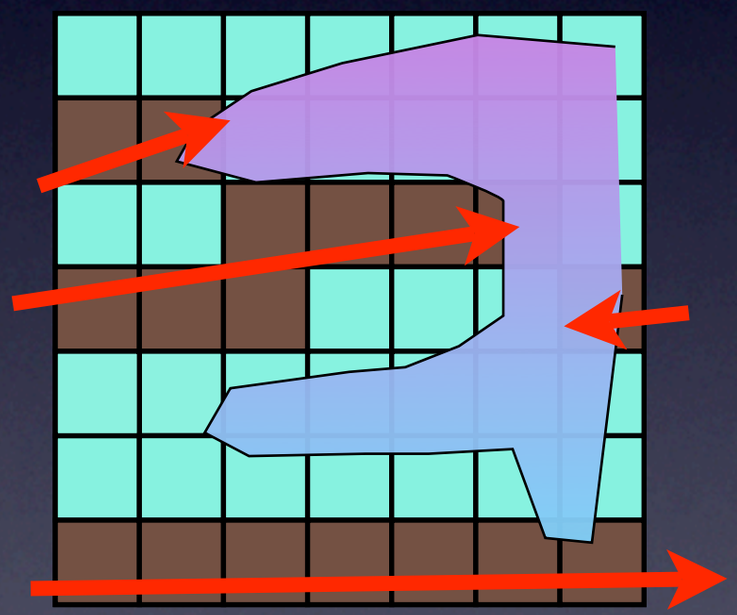
- Very complicated to find actual value



Ray/Grid cost model: # cells traversed

$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

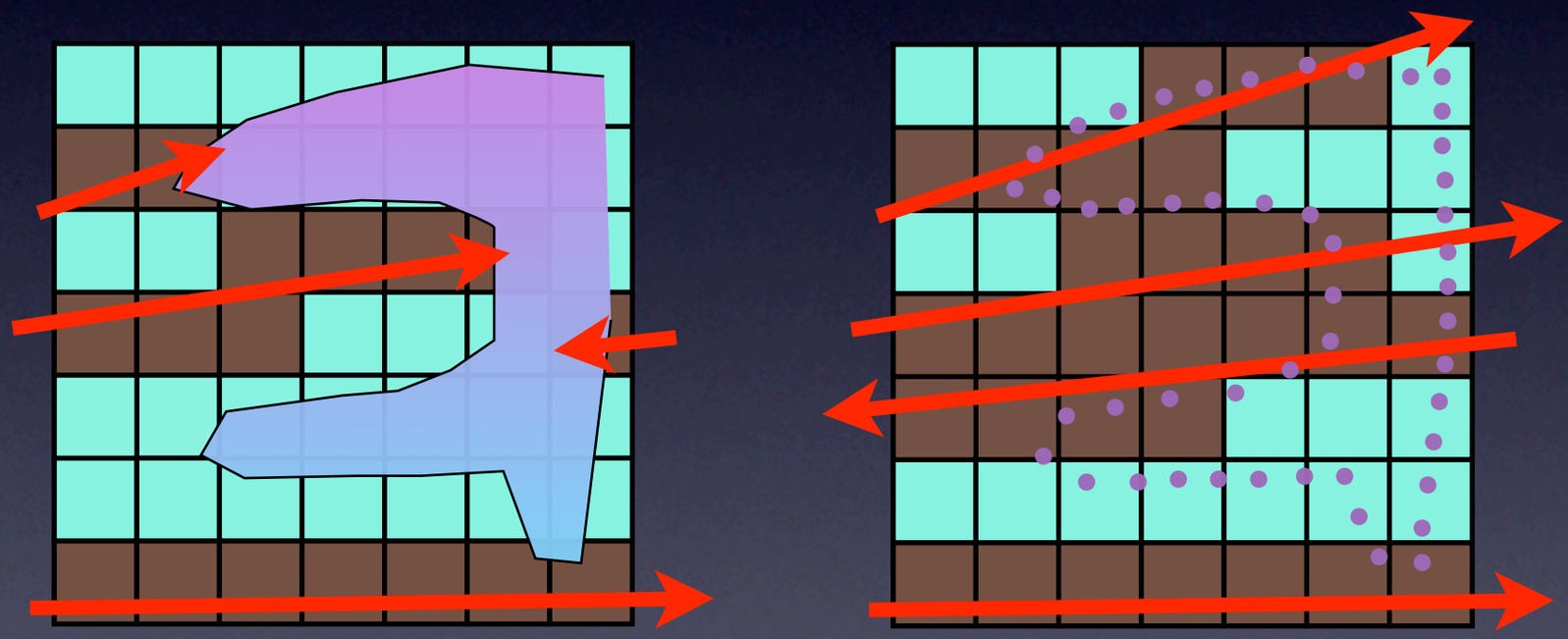
- We simplify by only looking at points and lines which probabilistically will never be hit



Ray/Grid cost model: # cells traversed

$$T = T_{\text{setup}} + (\#\text{ cells traversed})T_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

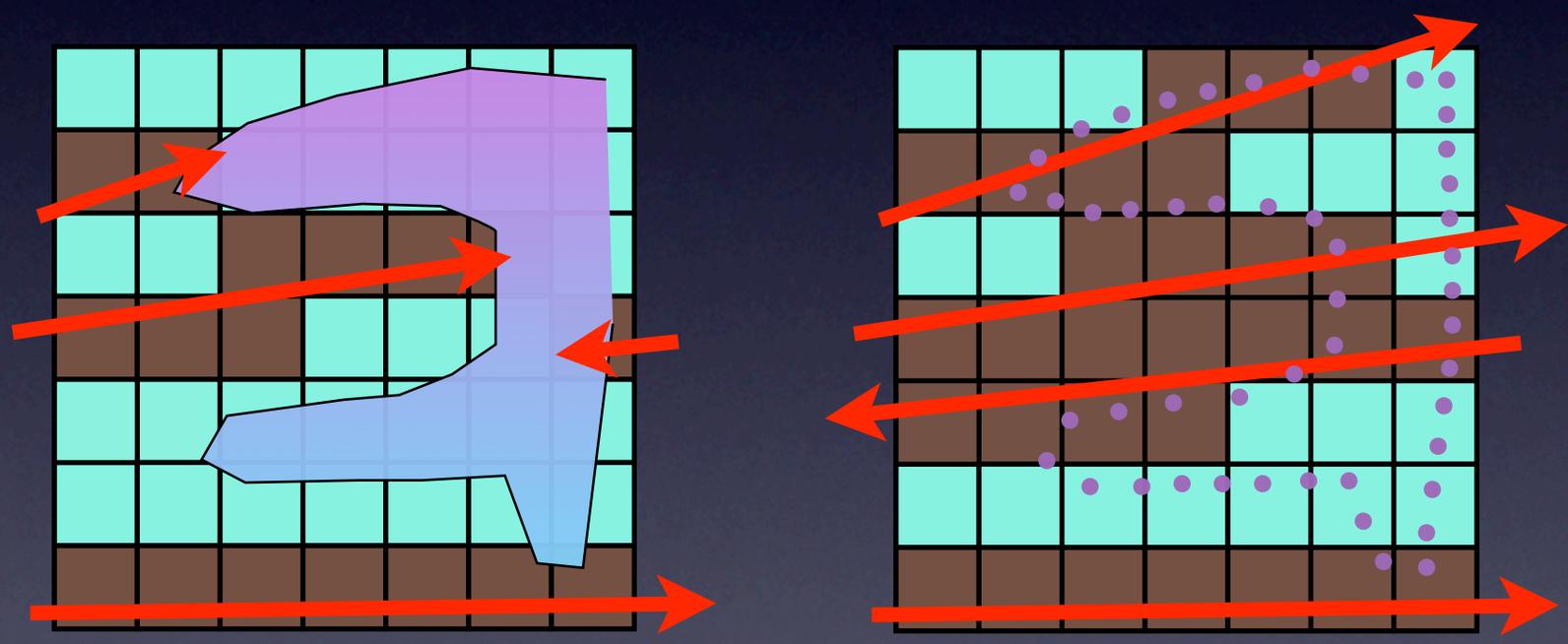
- We simplify by only looking at points and lines which probabilistically will never be hit



Ray/Grid cost model: # cells traversed

$$T = T_{\text{setup}} + mT_{\text{step}} + (\#\ \text{intersection tests})T_{\text{intersection}}$$

- We simplify by only looking at points and lines which probabilistically will never be hit

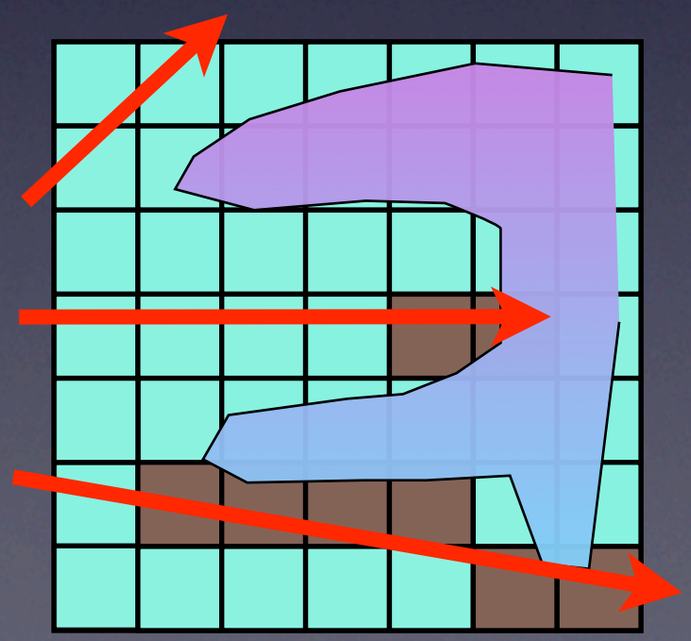


- Average # cells traversed = *m*, for an *m* x *m* x *m* grid

Ray/Grid cost model: # point intersection tests

$$T = T_{\text{setup}} + mT_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

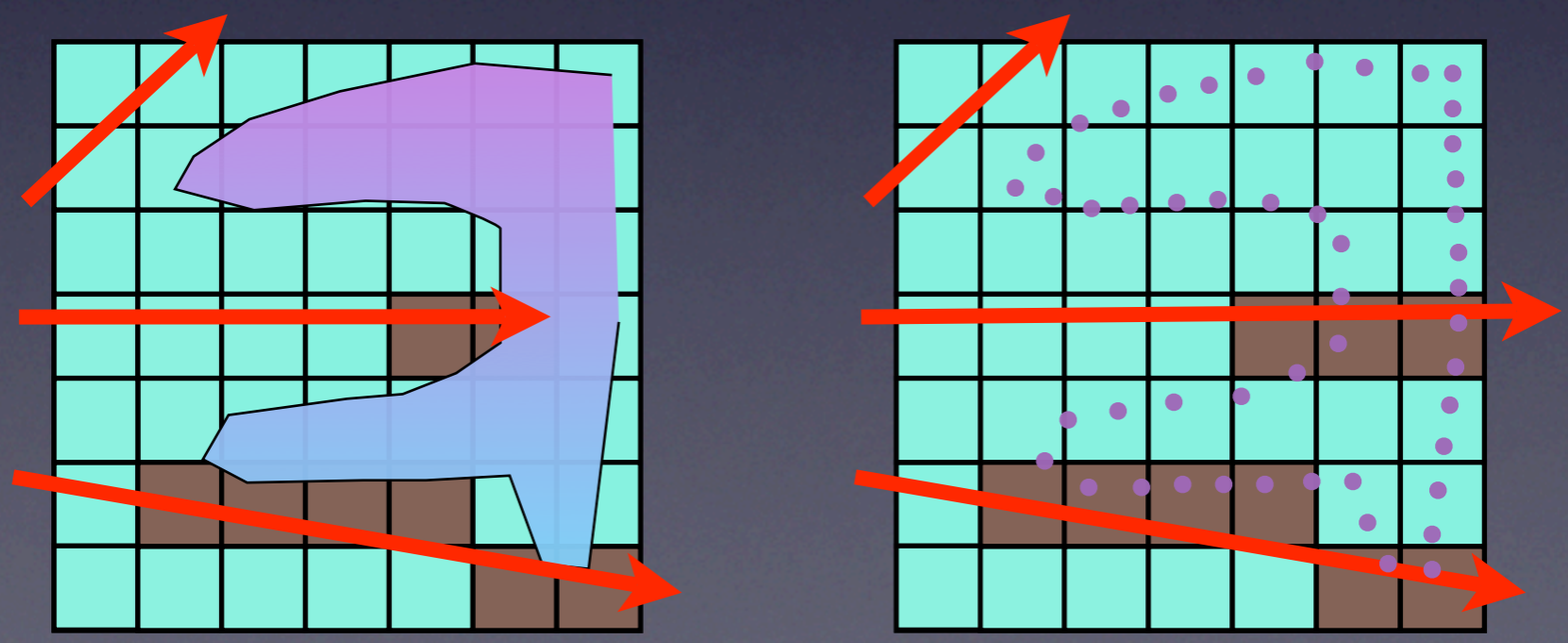
- Given N points and m^3 cells, there are an average of $\frac{N}{m^3}$ points in a cell
- A ray traverses m cells on average
- $\frac{N}{m^3}m = \frac{N}{m^2}$ intersection tests per ray



Ray/Grid cost model: # point intersection tests

$$T = T_{\text{setup}} + mT_{\text{step}} + (\#\text{ intersection tests})T_{\text{intersection}}$$

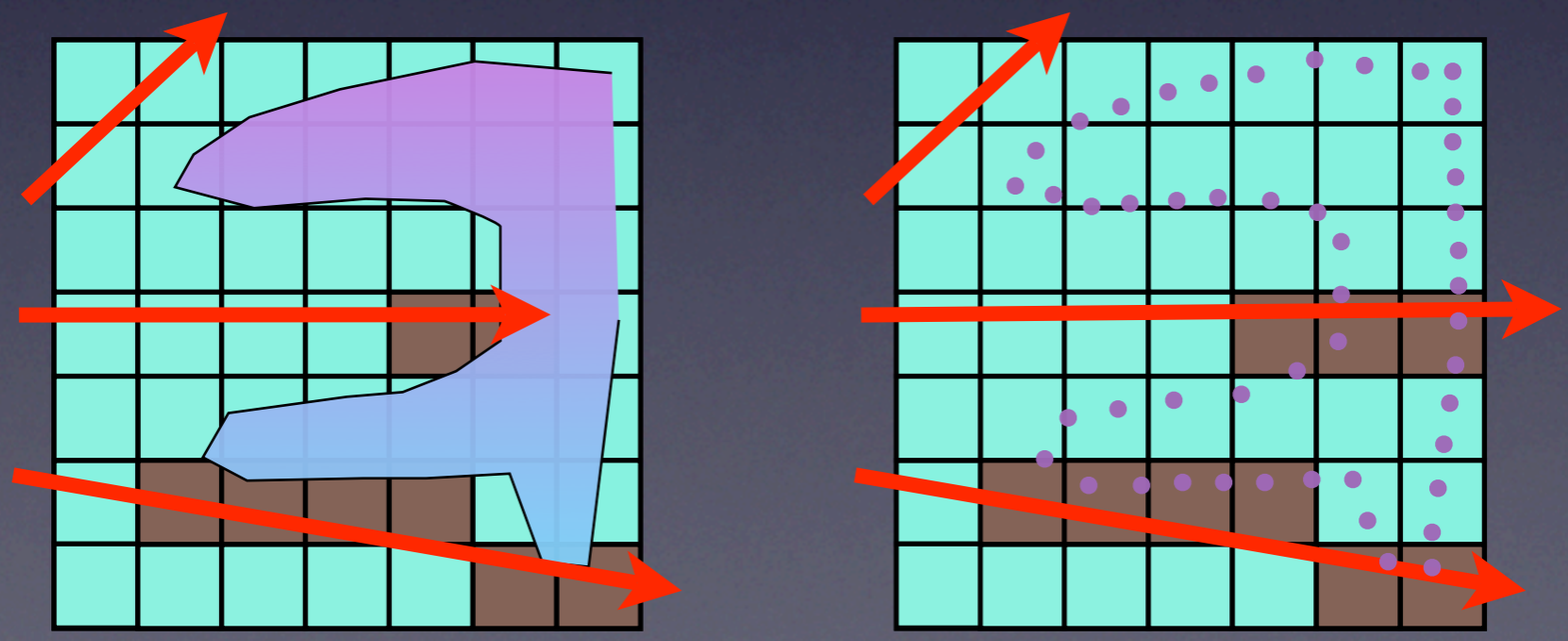
- Given N points and m^3 cells, there are an average of $\frac{N}{m^3}$ points in a cell
- A ray traverses m cells on average
- $\frac{N}{m^3}m = \frac{N}{m^2}$ intersection tests per ray



Ray/Grid cost model: # point intersection tests

$$T = T_{\text{setup}} + mT_{\text{step}} + \frac{N}{m^2} T_{\text{intersection}}$$

- Given N points and m^3 cells, there are an average of $\frac{N}{m^3}$ points in a cell
- A ray traverses m cells on average
- $\frac{N}{m^3} m = \frac{N}{m^2}$ intersection tests per ray

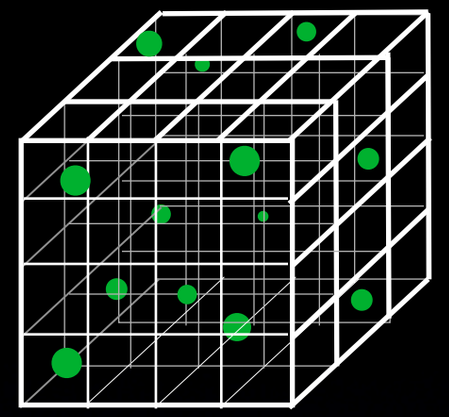


Ray/Grid cost model: # line intersection tests

$$T = T_{\text{setup}} + mT_{\text{step}} + \frac{N}{m}T_{\text{intersection}}$$

- A line covers m cells on average
- Given N lines and m^3 cells, there are an average of $\frac{N}{m^3}m = \frac{N}{m^2}$ lines in a cell
- A ray traverses m cells on average
- $\frac{N}{m^2}m = \frac{N}{m}$ line intersection tests per ray

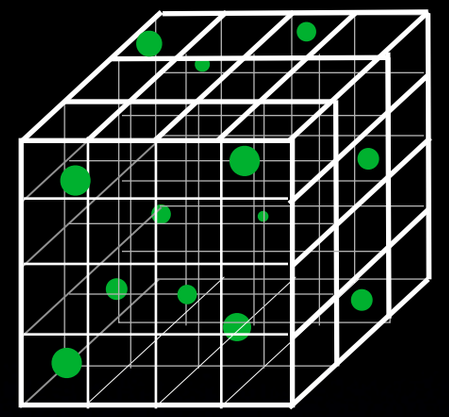
Optimal Grid for Points



- Minimize ray/grid cost to find optimal *m*

$$T = T_{\text{setup}} + mT_{\text{step}} + \frac{N}{m^2}T_{\text{intersection}}$$

Optimal Grid for Points

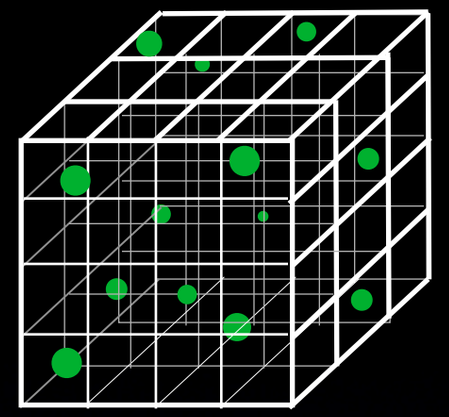


- Minimize ray/grid cost to find optimal *m*

$$T = T_{\text{setup}} + mT_{\text{step}} + \frac{N}{m^2}T_{\text{intersection}}$$

$$\frac{dT}{dm} = T_{\text{step}} - \frac{2N}{m^3}T_{\text{intersection}}$$

Optimal Grid for Points

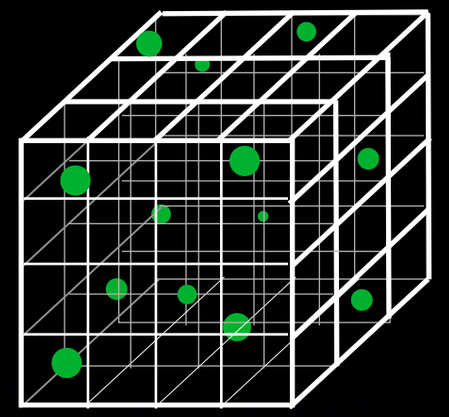


- Minimize ray/grid cost to find optimal *m*

$$T = T_{\text{setup}} + mT_{\text{step}} + \frac{N}{m^2}T_{\text{intersection}}$$

$$\frac{dT}{dm} = T_{\text{step}} - \frac{2N}{m^3}T_{\text{intersection}} = 0$$

Optimal Grid for Points



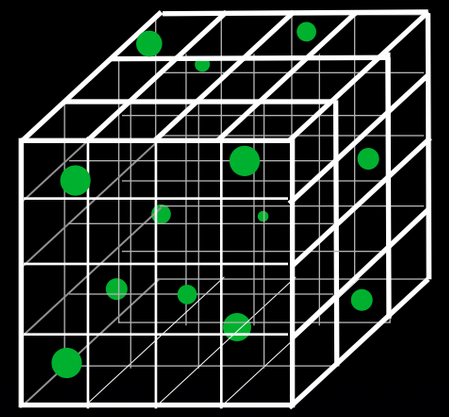
- Minimize ray/grid cost to find optimal *m*

$$T = T_{\text{setup}} + mT_{\text{step}} + \frac{N}{m^2}T_{\text{intersection}}$$

$$\frac{dT}{dm} = T_{\text{step}} - \frac{2N}{m^3}T_{\text{intersection}} = 0$$

$$m = \sqrt[3]{N\frac{2T_{\text{intersection}}}{T_{\text{step}}}}$$

Optimal Grid for Points



- Minimize ray/grid cost to find optimal *m*

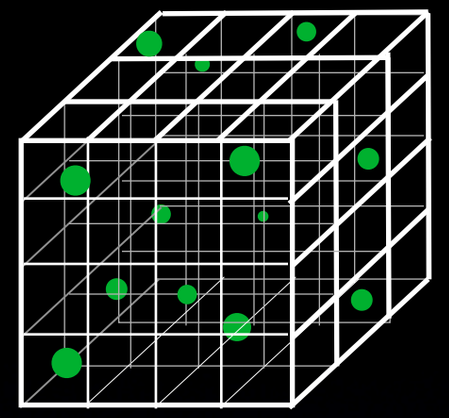
$$T = T_{\text{setup}} + mT_{\text{step}} + \frac{N}{m^2}T_{\text{intersection}}$$

$$\frac{dT}{dm} = T_{\text{step}} - \frac{2N}{m^3}T_{\text{intersection}} = 0$$

$$m = \sqrt[3]{N\frac{2T_{\text{intersection}}}{T_{\text{step}}}}$$

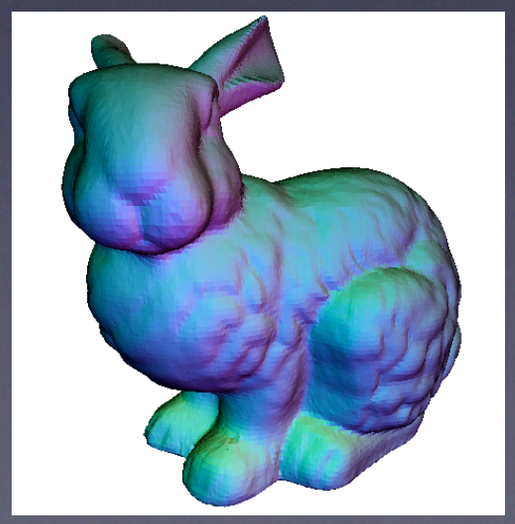
- Grid should thus have $m^3 = N\frac{2T_{\text{intersection}}}{T_{\text{step}}}$ cells
- This holds for any single level grid of points

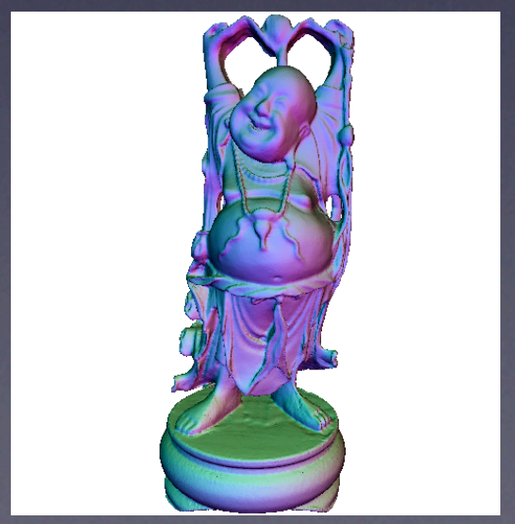
Optimal Grid for Points Applied to Triangle Based Scenes



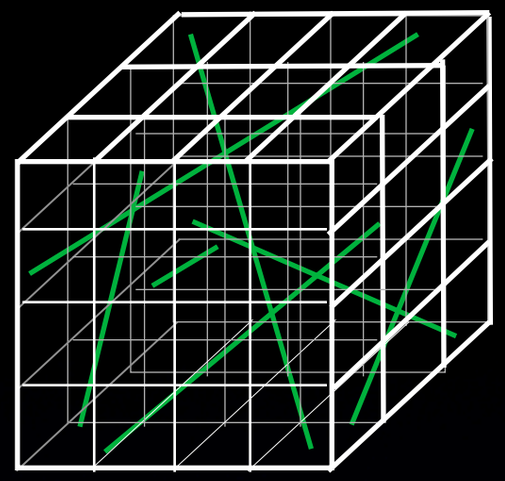
- From $m^3 = N\frac{2T_{\text{intersection}}}{T_{\text{step}}}$, we empirically find $\frac{T_{\text{intersection}}}{T_{\text{step}}}$
- Get within 5% of optimal for tested scenes
  - Varied triangle count for laser scanned models
  - Use same ratio of $\frac{T_{\text{intersection}}}{T_{\text{step}}}$ for all scenes

- Laser scanned models with small compact triangles are handled very well, but even conference room does well

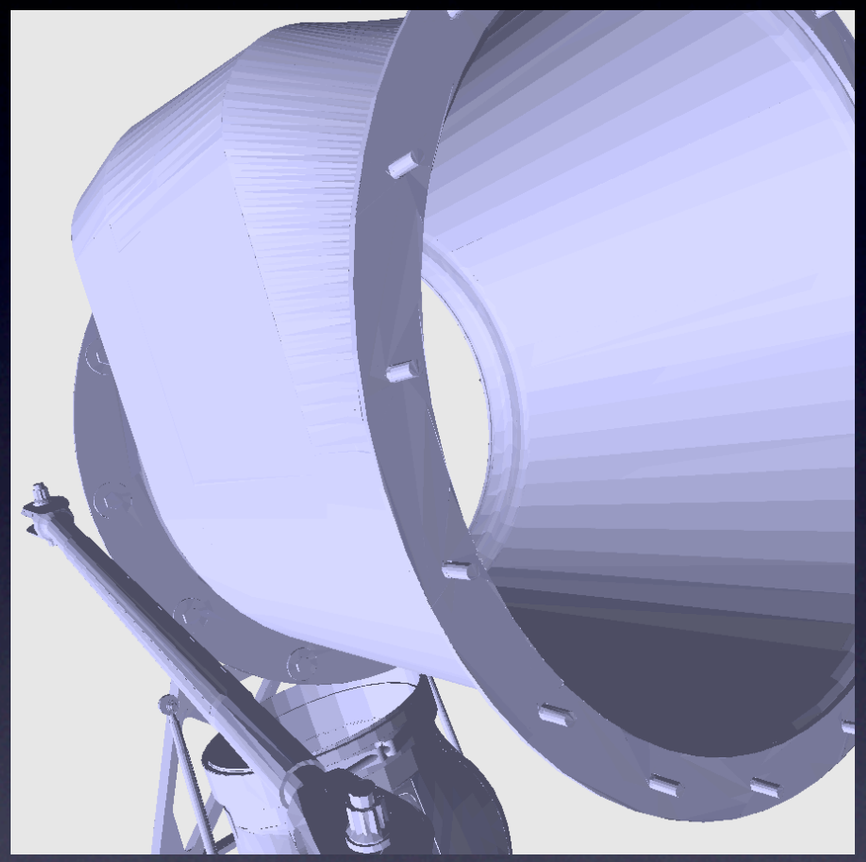


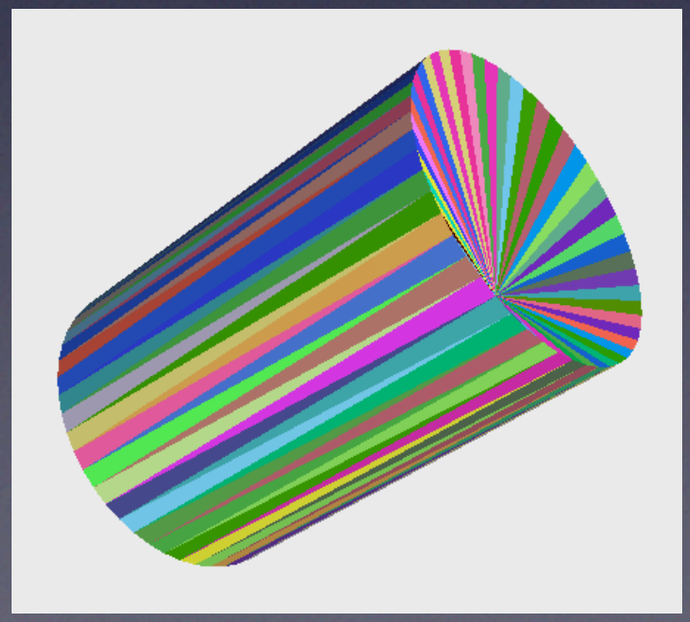


Optimal Grid for Lines Applied to Long Skinny Triangle Based Scenes

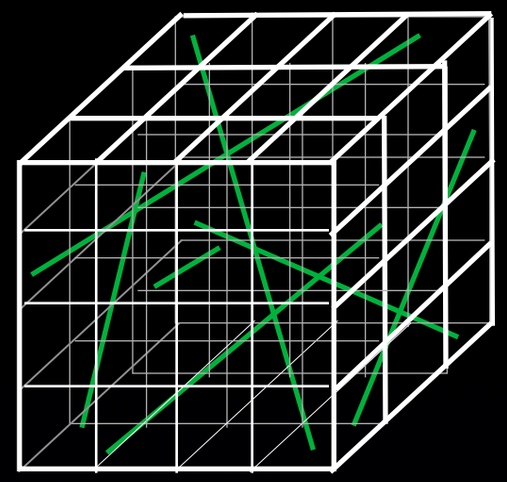


- Same derivation as for points gives $m^3 = (N\frac{T_{\text{intersection}}}{T_{\text{step}}})^{1.5}$ cells
- $O(N^{1.5})$ space-complexity
- Long skinny triangles common in CAD models
- We use a cylinder to simulate this

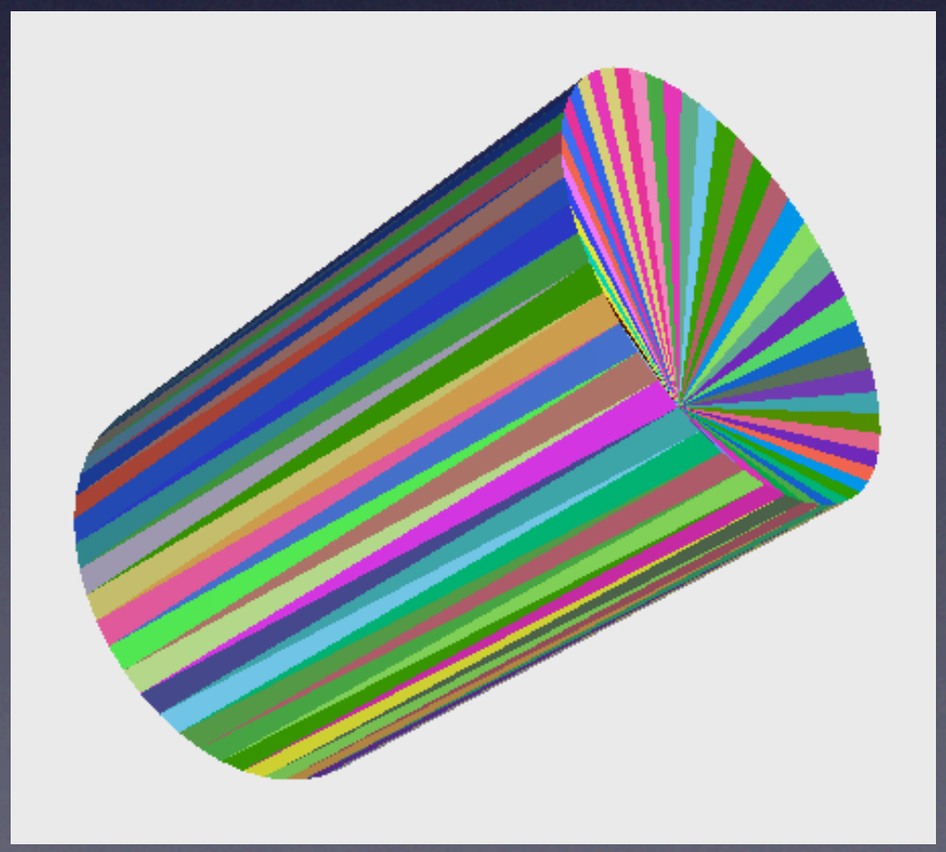




Optimal Grid for Lines Applied to Long Skinny Triangle Based Scenes

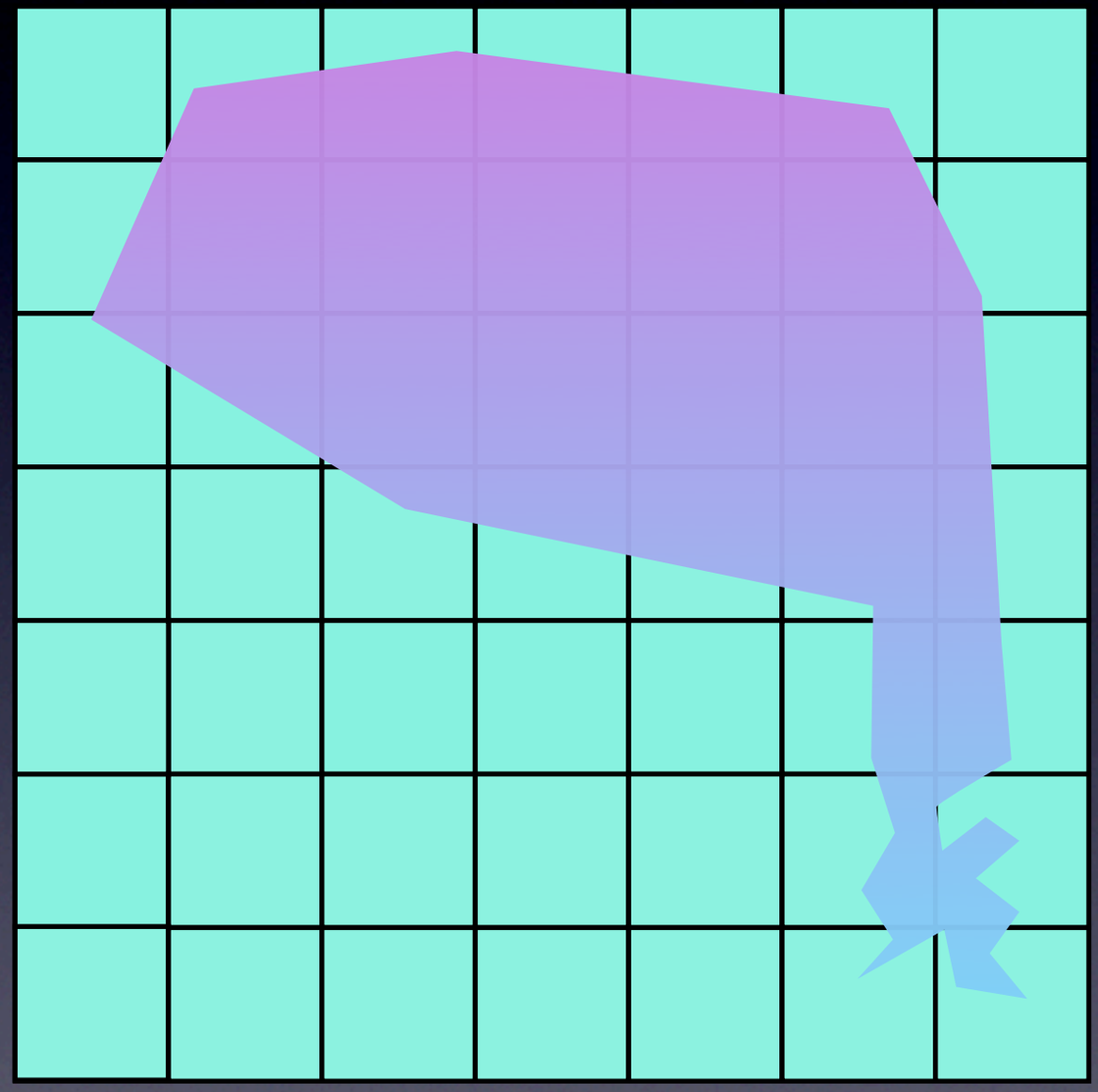


- Only practical for small number of triangles
- $O(N^{1.5})$ cells produces grids within 3% of optimal performance
- $O(N)$ cells results in a $1.25\times - 2\times$ slowdown
- Extremely large 19,996 tri cylinder within 10% of optimal

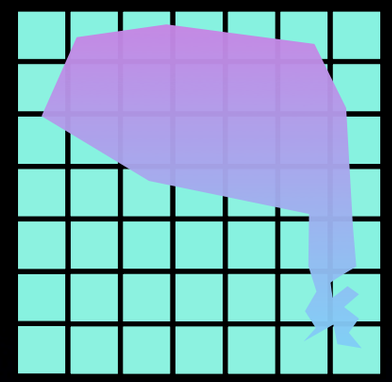


2-Level Grid

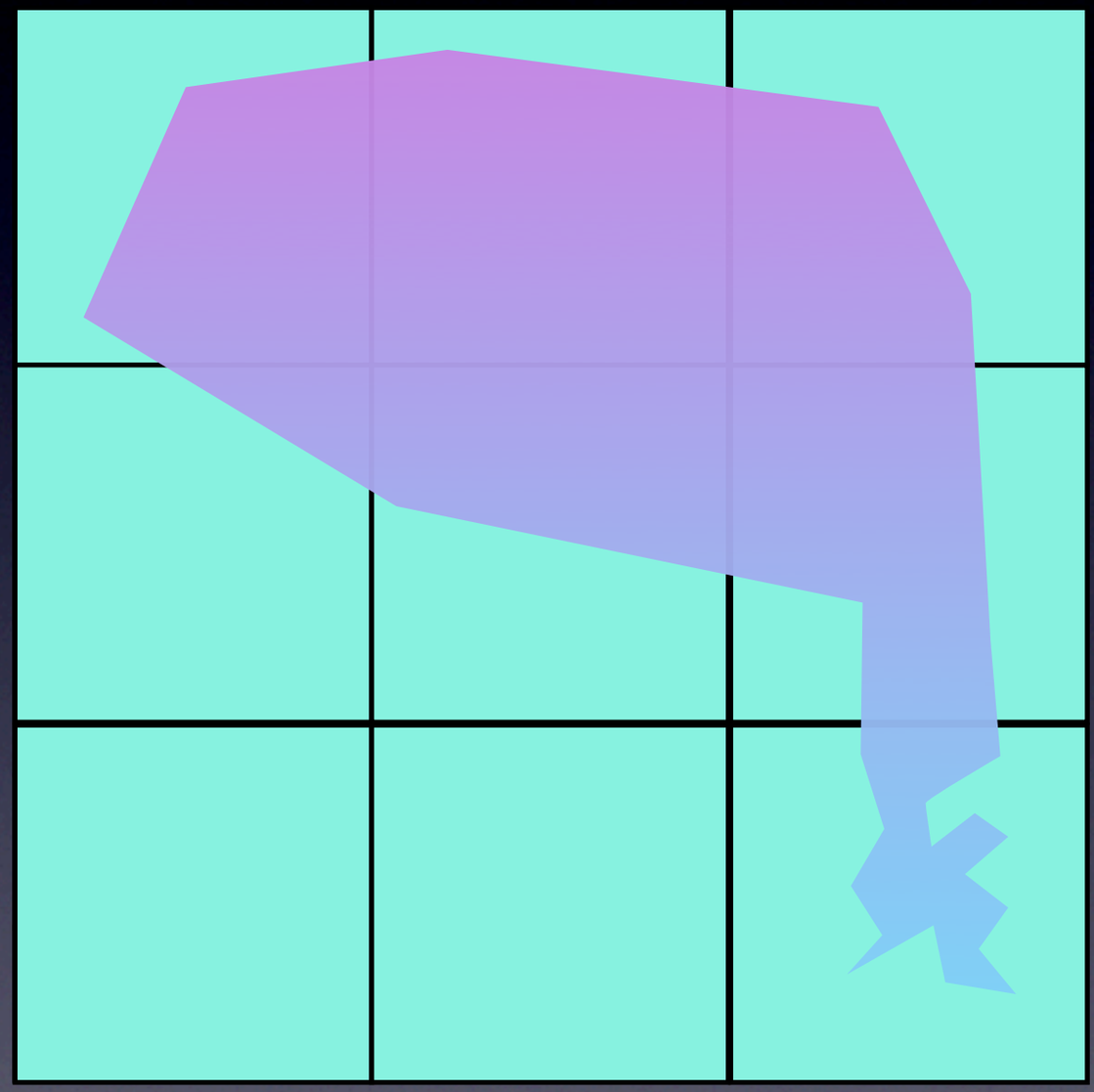
- Recursive (nested) Grid
- Cube shaped cells
- Cell size can vary between grids
- Subgrid resides only in parent cell



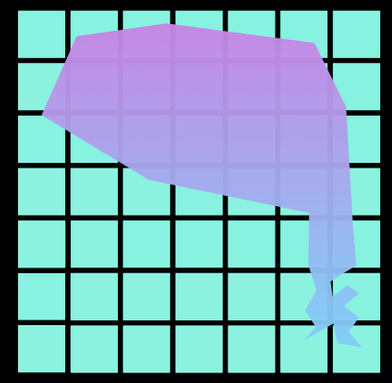
2-Level Grid



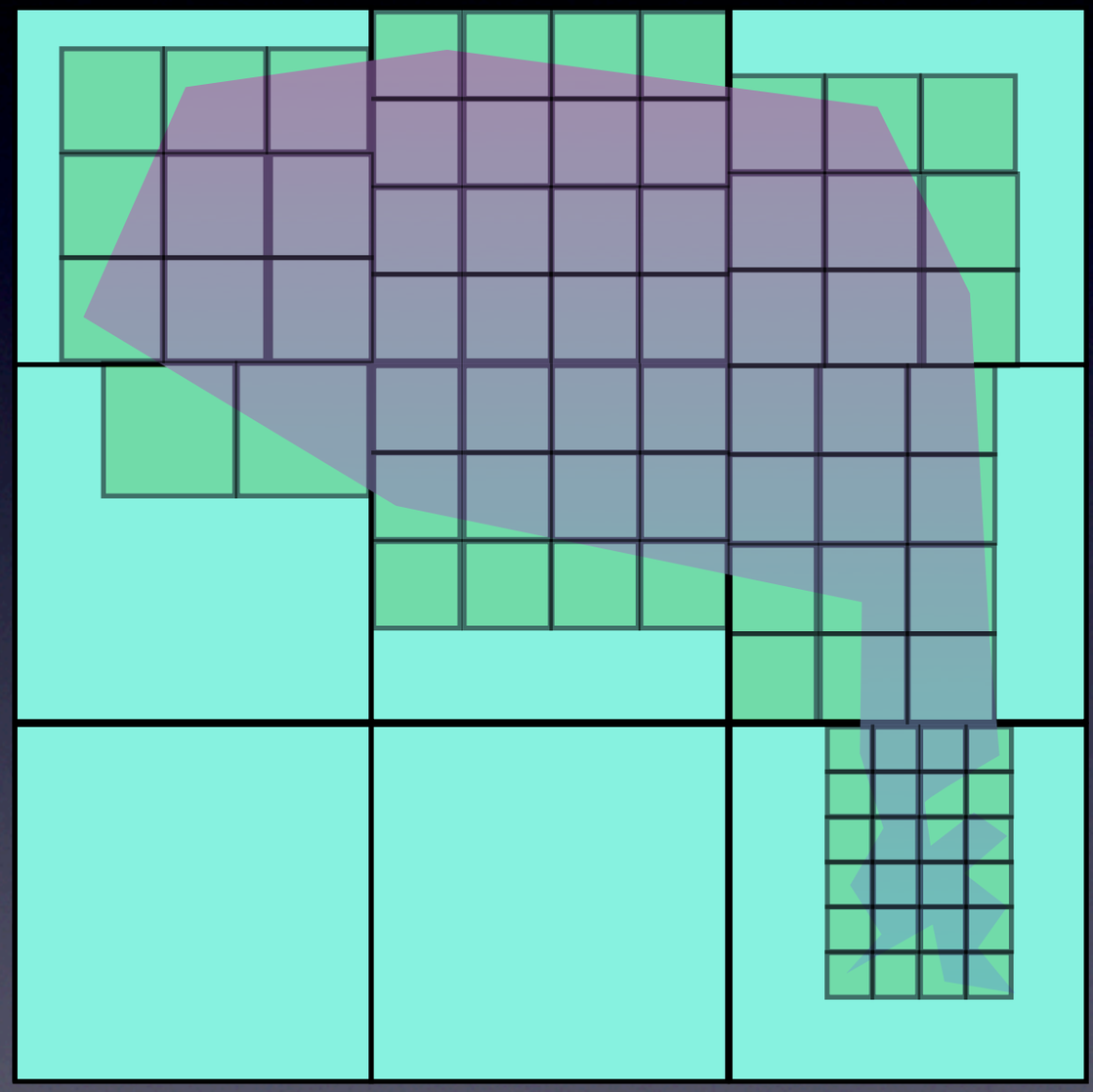
- Recursive (nested) Grid
- Cube shaped cells
- Cell size can vary between grids
- Subgrid resides only in parent cell



2-Level Grid

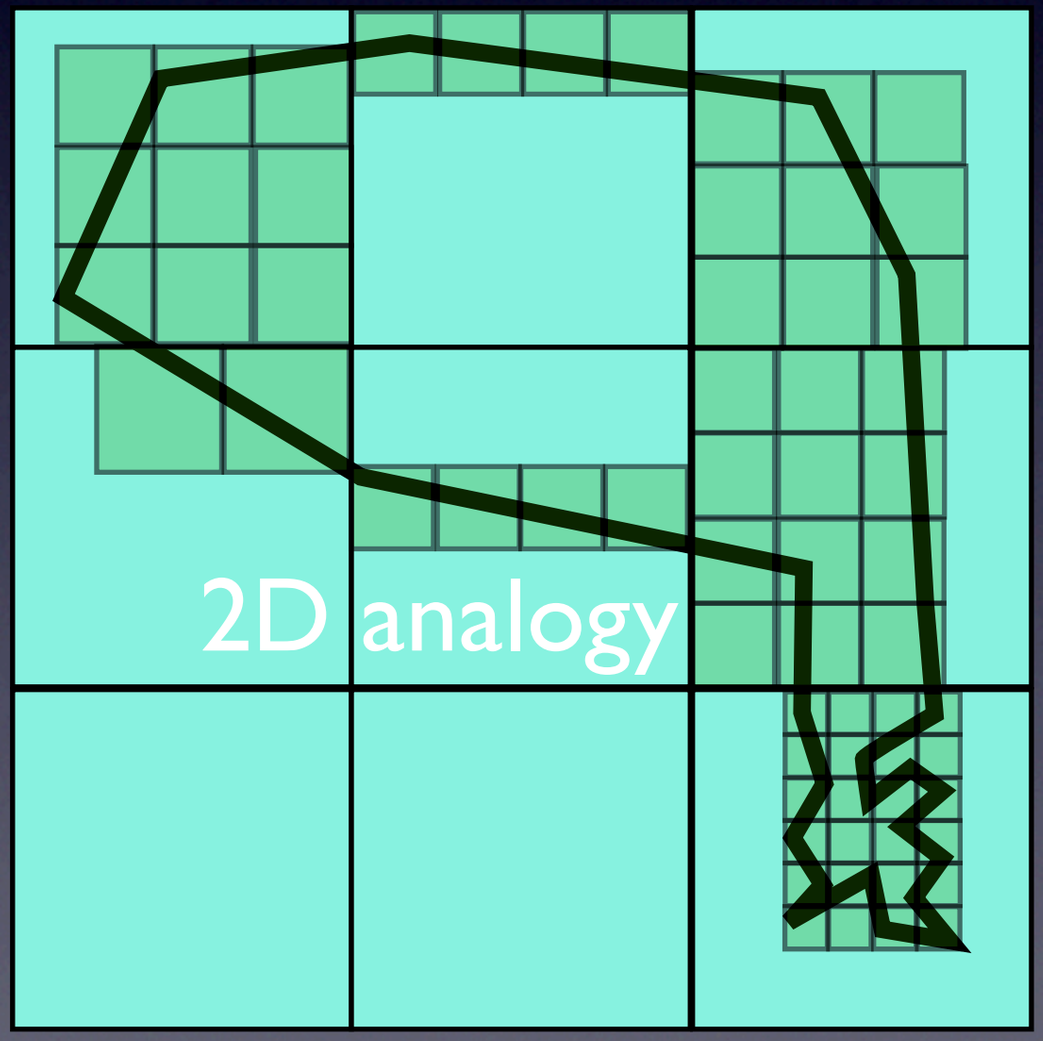


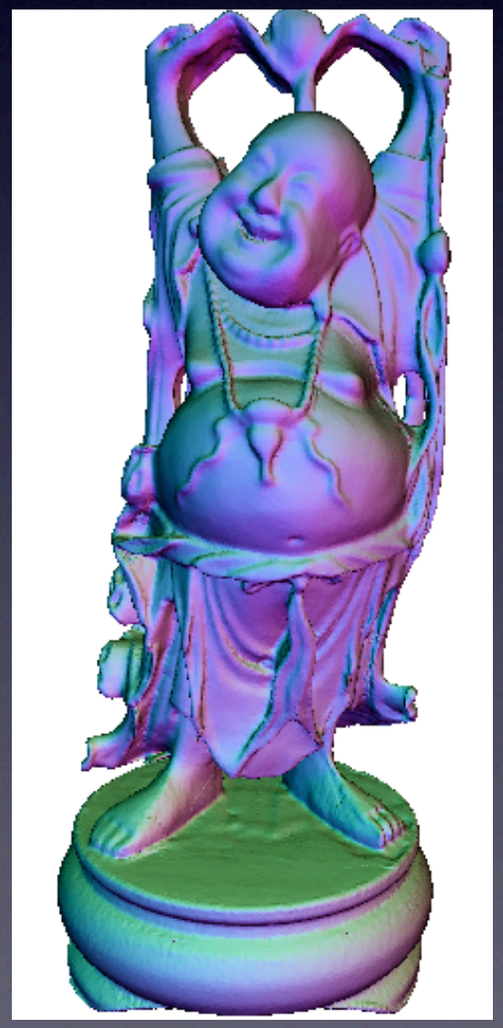
- Recursive (nested) Grid
- Cube shaped cells
- Cell size can vary between grids
- Subgrid resides only in parent cell



2-Level Grid

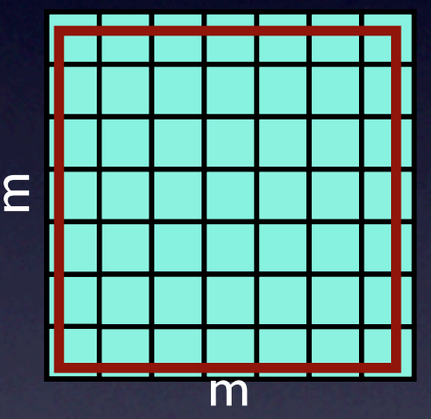
- Multi-level grids do not help for scattered data
- Focus on manifold-like models, such as laser-scanned models





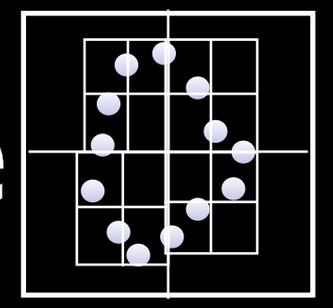
2-Level Grid for Manifolds

- For an $m \times m \times m$ cell grid, a 2D manifold passes through $O(m^2)$ cells
  - A 2D manifold goes through km^2 cells, where k depends on the specific manifold
    - Cube goes through $6m^2$ cells ($k = 6$)
    - In 2D, square goes through 4m cells



- Only need to refine the km^2 cells for bottom level grid
- Fraction of cells with subgrids: $\frac{km^2}{m^3} = \frac{k}{m}$

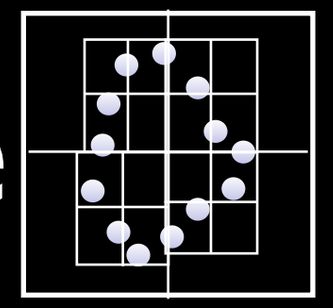
2-Level Grid: Points on Surface



- As before, we simplify the problem by looking at points -- this time points are on a surface
- Assume the km^2 cells each contain an equal number of points (each subgrid has same time complexity)

- $T = T_{\text{setup}} + mT_{\text{step}} + \frac{k}{m} mT_{\text{subgrid}}$

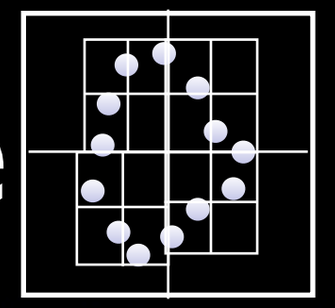
2-Level Grid: Points on Surface



- As before, we simplify the problem by looking at points -- this time points are on a surface
- Assume the km^2 cells each contain an equal number of points (each subgrid has same time complexity)

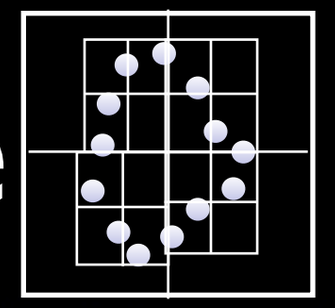
- $T = T_{\text{setup}} + mT_{\text{step}} + kT_{\text{subgrid}}$

2-Level Grid: Points on Surface



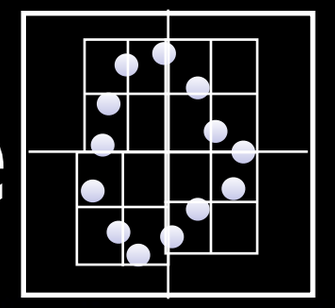
- $T = T_{\text{setup}} + mT_{\text{step}} + kT_{\text{subgrid}}$
- N points and km_1^2 subgrids give us $\frac{N}{km_1^2}$ points per subgrid
- Each subgrid has $\frac{N}{km_1^2}\frac{1}{m_2^3}$ points per subgrid cell

2-Level Grid: Points on Surface



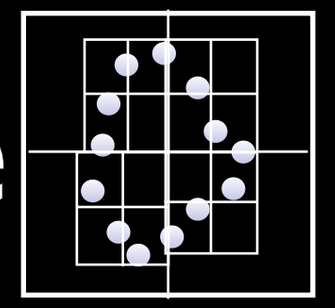
- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k T_{\text{subgrid}}$
- N points and km_1^2 subgrids give us $\frac{N}{km_1^2}$ points per subgrid
- Each subgrid has $\frac{N}{km_1^2}\frac{1}{m_2^3}$ points per subgrid cell

2-Level Grid: Points on Surface



- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k T_{\text{subgrid}}$
- N points and km_1^2 subgrids give us $\frac{N}{km_1^2}$ points per subgrid
- Each subgrid has $\frac{N}{km_1^2}\frac{1}{m_2^3}$ points per subgrid cell
- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k\left(T_{\text{setup}} + m_2 T_{\text{step}} + \frac{N}{km_1^2 m_2^2} T_{\text{intersection}}\right)$

2-Level Grid: Points on Surface

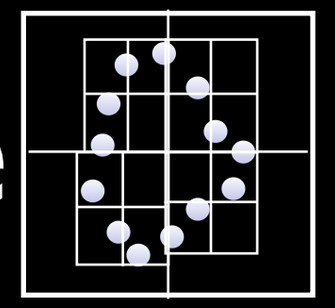


- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k T_{\text{subgrid}}$
- N points and km_1^2 subgrids give us $\frac{N}{km_1^2}$ points per subgrid
- Each subgrid has $\frac{N}{km_1^2}\frac{1}{m_2^3}$ points per subgrid cell
- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k\left(T_{\text{setup}} + m_2 T_{\text{step}} + \frac{N}{km_1^2 m_2^2} T_{\text{intersection}}\right)$

$$m_1^3 = \left(N \frac{2k^2 T_{\text{intersection}}}{T_{\text{step}}}\right)^{0.6}$$

$$m_2^3 = \left(N \frac{2T_{\text{intersection}}}{k^3 T_{\text{step}}}\right)^{0.6}$$

2-Level Grid: Points on Surface

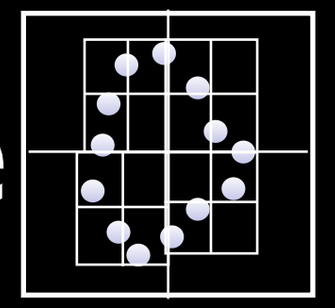


- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k T_{\text{subgrid}}$
- N points and km_1^2 subgrids give us $\frac{N}{km_1^2}$ points per subgrid
- Each subgrid has $\frac{N}{km_1^2}\frac{1}{m_2^3}$ points per subgrid cell
- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k\left(T_{\text{setup}} + m_2 T_{\text{step}} + \frac{N}{km_1^2 m_2^2} T_{\text{intersection}}\right)$

$$m_1^3 = \left(N\frac{2k^2 T_{\text{intersection}}}{T_{\text{step}}}\right)^{0.6}$$

$$m_2^3 = \left(N\frac{2T_{\text{intersection}}}{k^3 T_{\text{step}}}\right)^{0.6} \qquad m_2^3 = \frac{N}{km_1^2}\frac{2T_{\text{intersection}}}{T_{\text{step}}}$$

2-Level Grid: Points on Surface



- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k T_{\text{subgrid}}$
- N points and km_1^2 subgrids give us $\frac{N}{km_1^2}$ points per subgrid
- Each subgrid has $\frac{N}{km_1^2}\frac{1}{m_2^3}$ points per subgrid cell
- $T = T_{\text{setup}} + m_1 T_{\text{step}} + k\left(T_{\text{setup}} + m_2 T_{\text{step}} + \frac{N}{km_1^2 m_2^2} T_{\text{intersection}}\right)$

$$m_1^3 = \left(N \frac{2k^2 T_{\text{intersection}}}{T_{\text{step}}}\right)^{0.6}$$

$$m_2^3 = \left(N \frac{2T_{\text{intersection}}}{k^3 T_{\text{step}}}\right)^{0.6} \qquad m_2^3 = N_2 \frac{2T_{\text{intersection}}}{T_{\text{step}}}$$

buddha - 15536 tri

Optimal

Bottom Level Grid

$m_2/(N_2^{1/3})$

4
3.5
3
2.5
2
1.5
1
0.5

0.5 1 1.5 2 2.5 3 3.5 4

$m_1/(N_1^{1/5})$

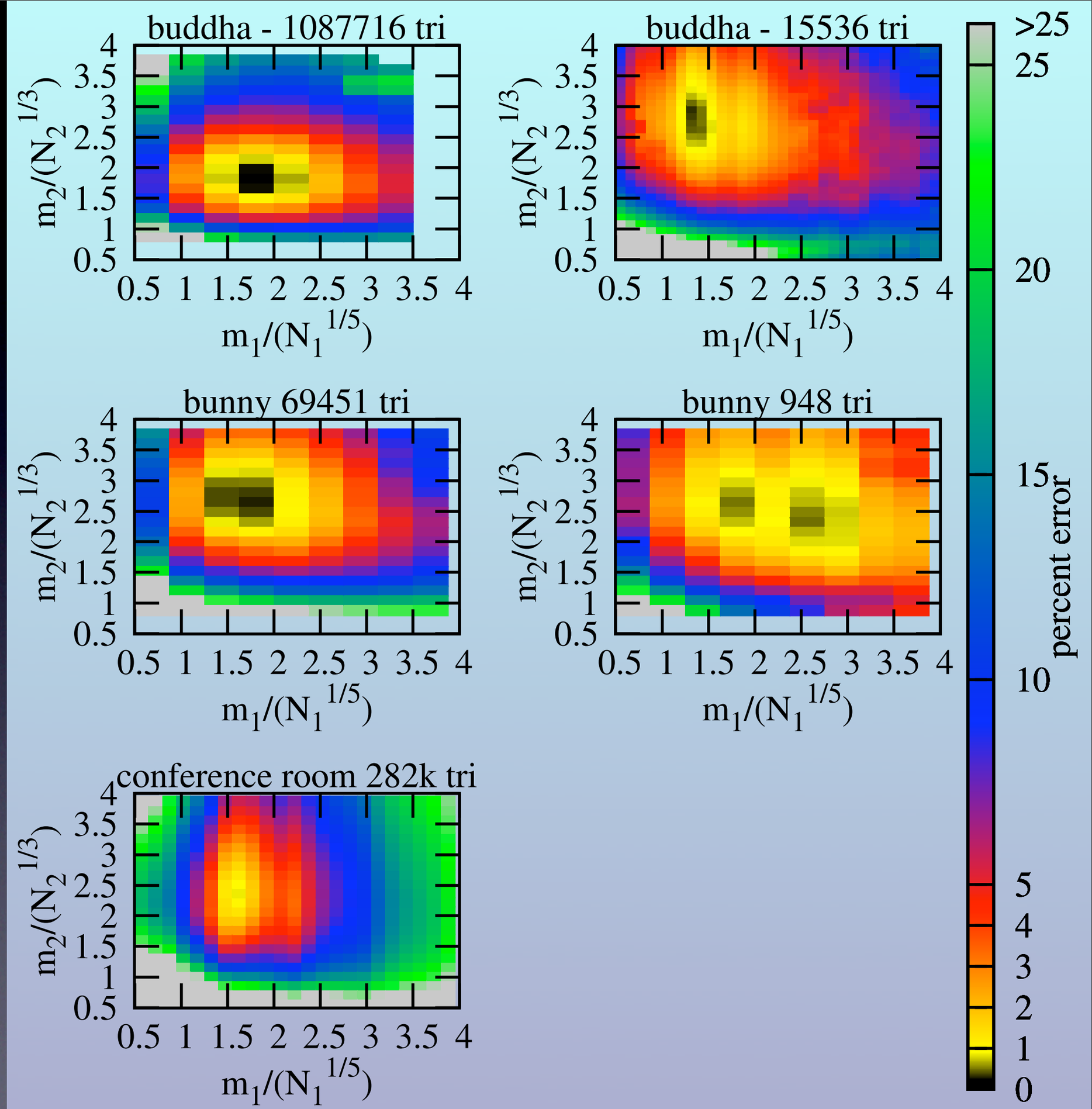
Top Level Grid

41 cells	327 cells	1104 cells	2618 cells	5112 cells	8834 cells	14028 cells	20940 cells

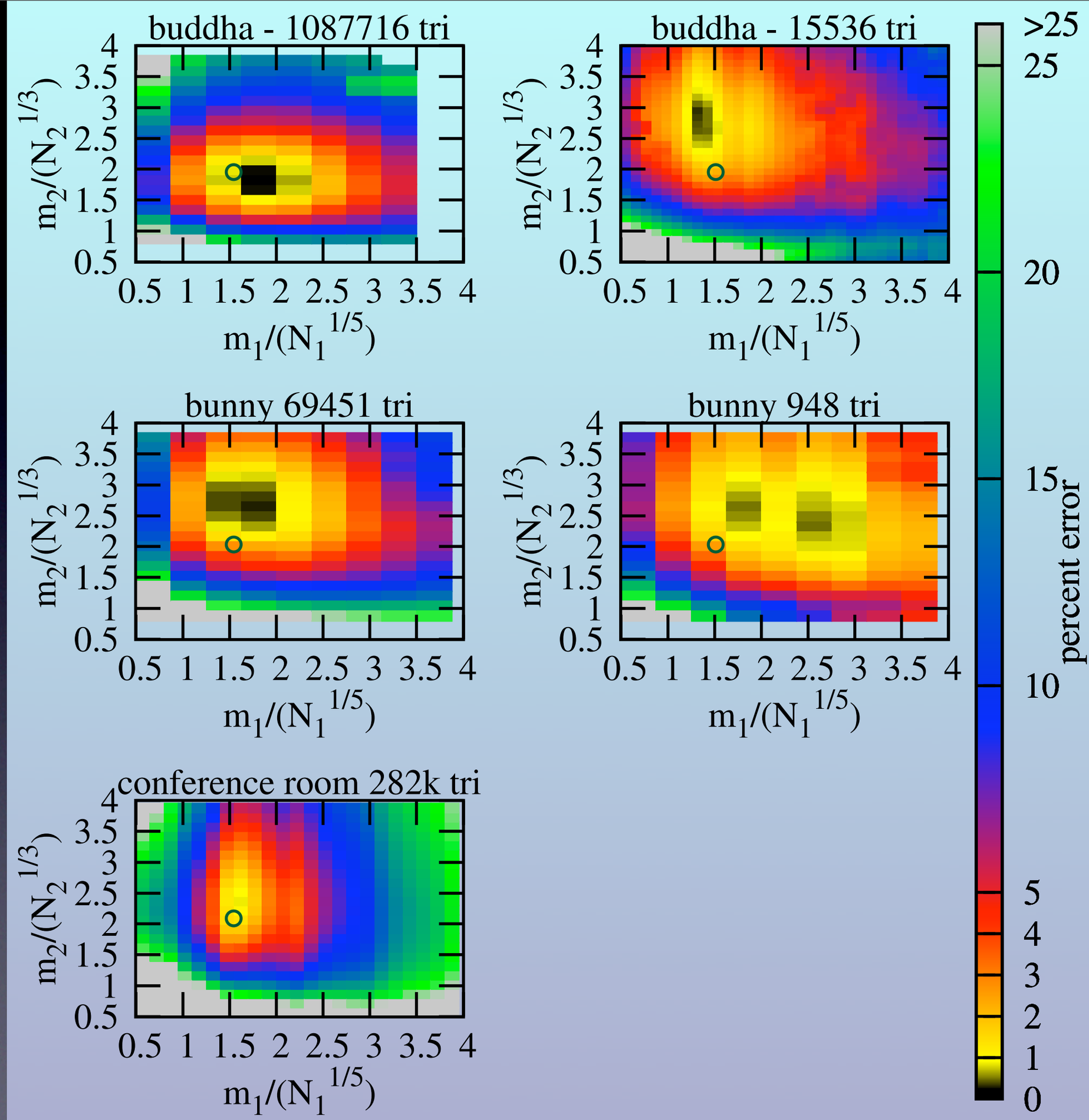
>25
25
20
15
10
5
4
3
2
1
0

percent error

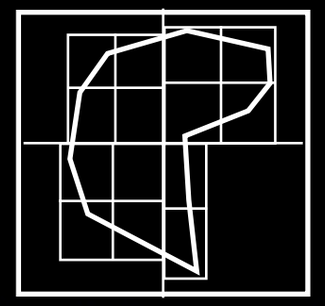
- Get within 5% of optimal for tested scenes
- About a 2x speedup over single level grid



- Get within 5% of optimal for tested scenes
- About a 2x speedup over single level grid

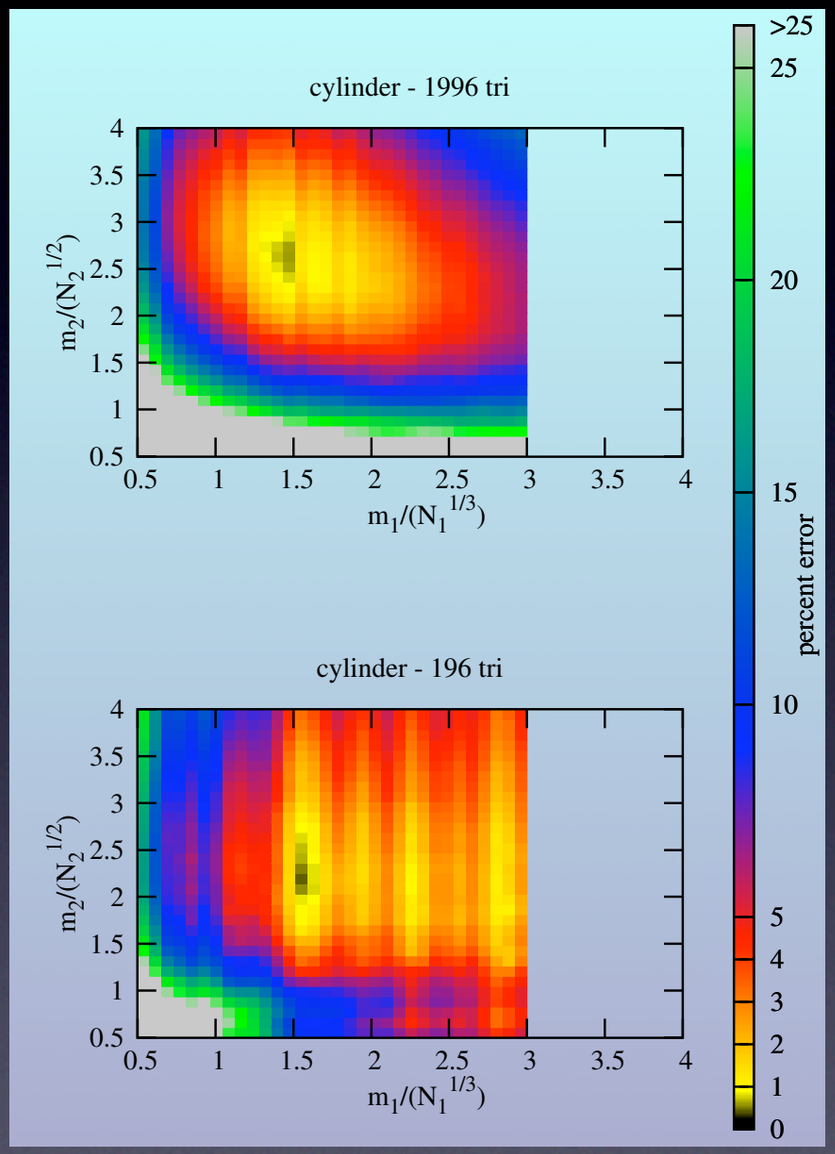


2-level grid: lines on surface Applied to Long Skinny Triangle Based Scenes

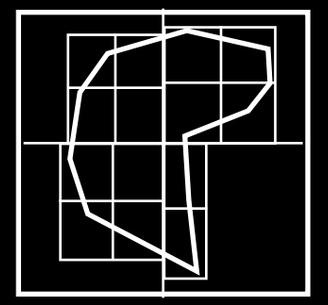


2-level grid is 1.4x faster than 1-level for 1996 tri cylinder

2-level grid is just 1.04x faster than 1-level for 196 tri cylinder



2-level grid: lines on surface Applied to Long Skinny Triangle Based Scenes



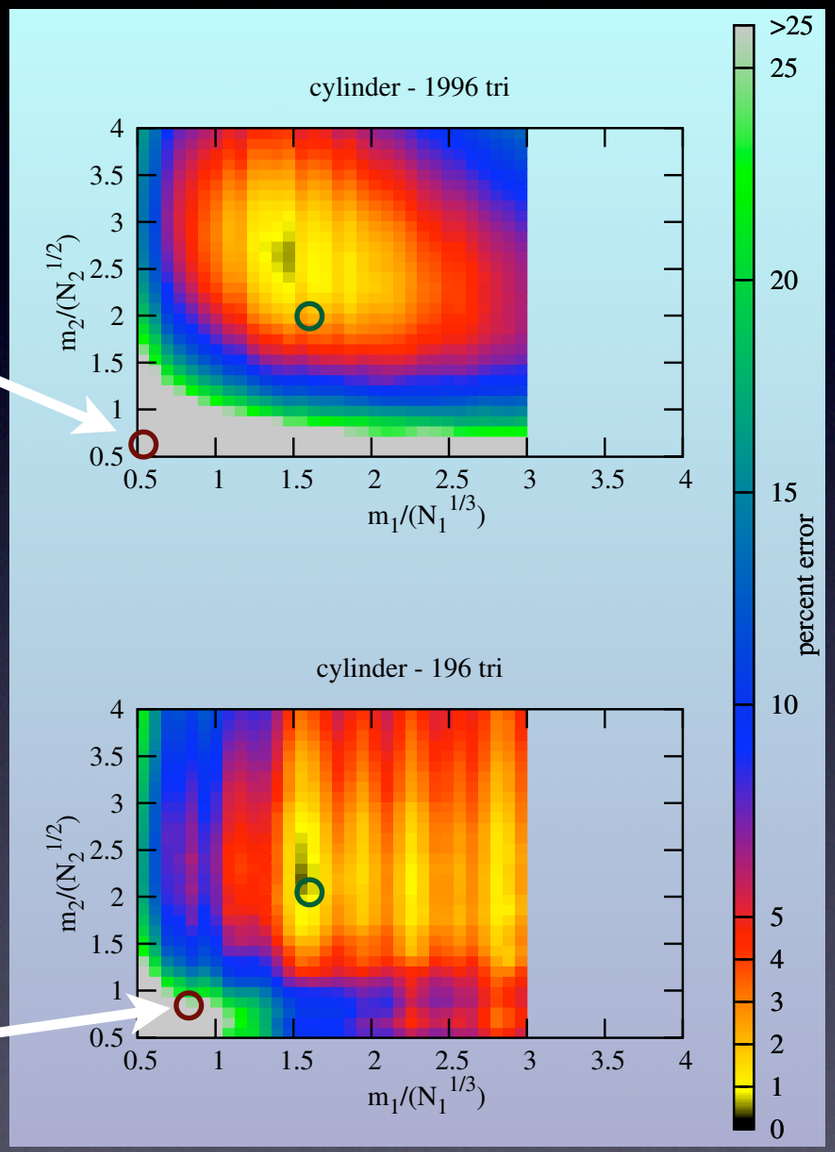
2-level grid is 1.4x faster than 1-level for 1996 tri cylinder

60% slower

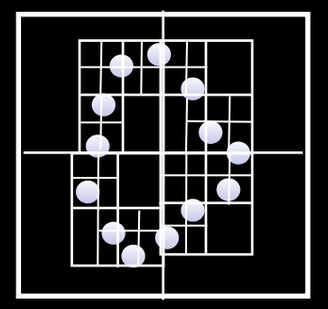
2-level compact triangle grid resolution

2-level grid is just 1.04x faster than 1-level for 196 tri cylinder

20% slower



L-level grid



- Same process as for 2-level grids
- Compact triangle top level grid has $O\left(N^{\frac{3}{2L+1}}\right)$ cells
- The next level would recursively behave as an L-1 level grid

- Can do same analysis for long skinny triangles

Better Single Level Grid

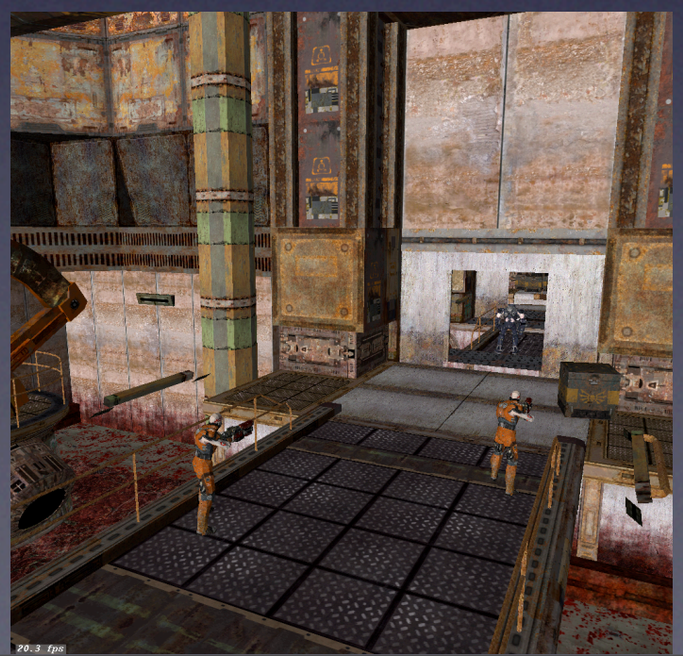
- For manifold like models empirically found
  - $O(N^{7/9})$ cells (sublinear grid storage!) for compact triangles
  - $O(N^{4/3})$ cells for long skinny triangles

  Results in even better performance

- Results in almost perfect grids for all our tests:
  - With $O(N^{3/2})$ cells, 19,996 tri cylinder has 10% penalty
  - With $O(N^{4/3})$ cells, 19,996 tri cylinder has 0% penalty
- Use this if you don't mind the lack of theory
- Theory is future work

Limitations

- Good for laser scanned models and simple scenes
- Not as good as empirically found formula
- Assumptions might not hold for some scenes
  - Assumed we never hit the primitive -- clearly false
  - Triangle distributions not evenly distributed about simple surface
- Might not work at all for more complex scenes



Conclusion

- Nested grids can lower time complexity below $O(\sqrt[3]{N})$
- Number of grid levels depends on model and setup cost of entering grid
- Long skinny triangle scenes can only achieve sub-linear time with super-linear memory use
- For scenes that do not deviate too much from our assumptions, finding a close-to-optimal multi-level grid resolution is very easy