Scalability with many lights II

(row-column sampling, visibity clustering)

Miloš Hašan

Scalability with Many VPLs

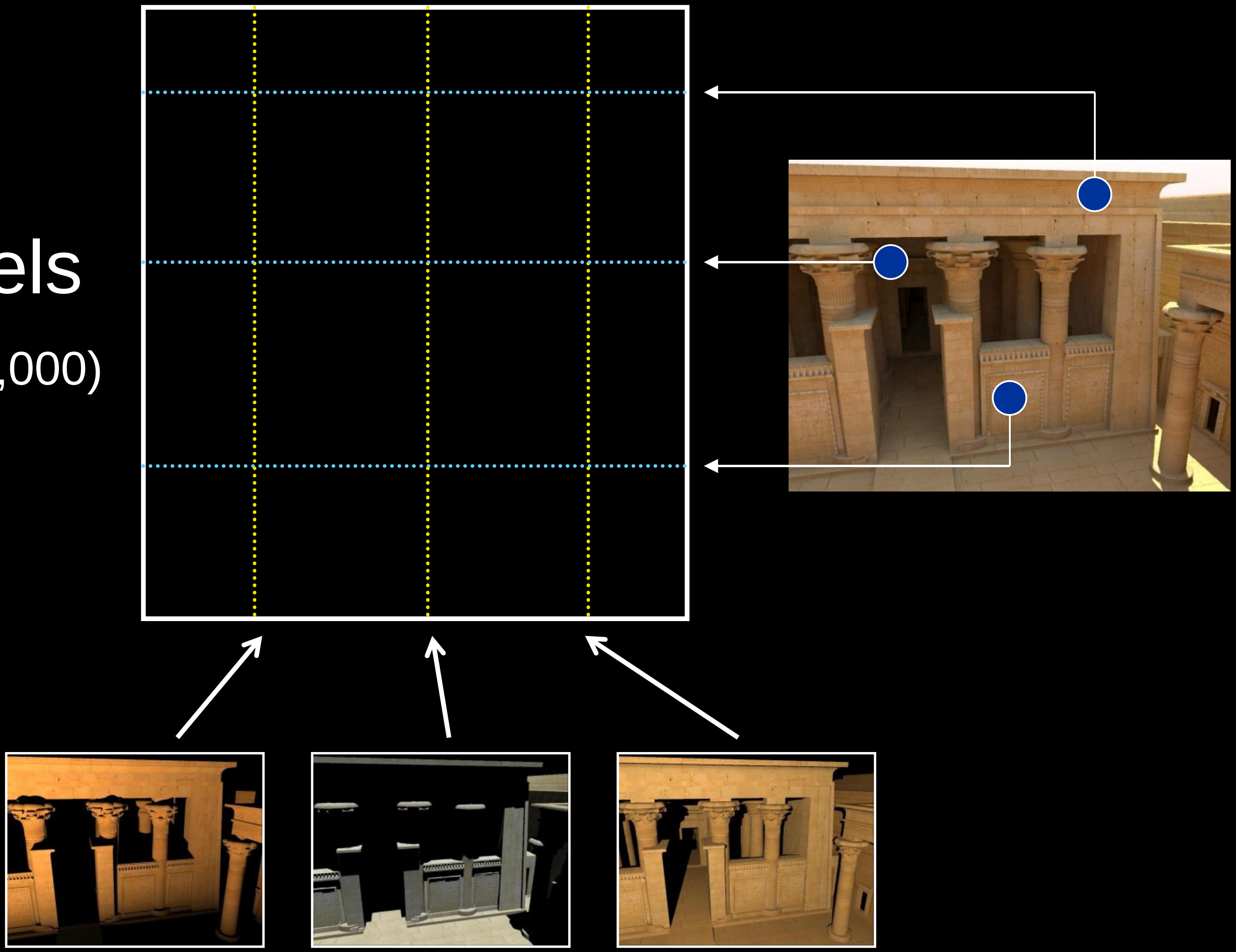
- Alternatives to lightcuts
  - Matrix row-column sampling
  - Visibility clustering
- Potential advantages
  - Shadow mapping instead of ray tracing
  - Simpler to implement
  - No bounds on BRDFs required
  - Faster in occluded environments

A Matrix Interpretation

Lights (100,000)

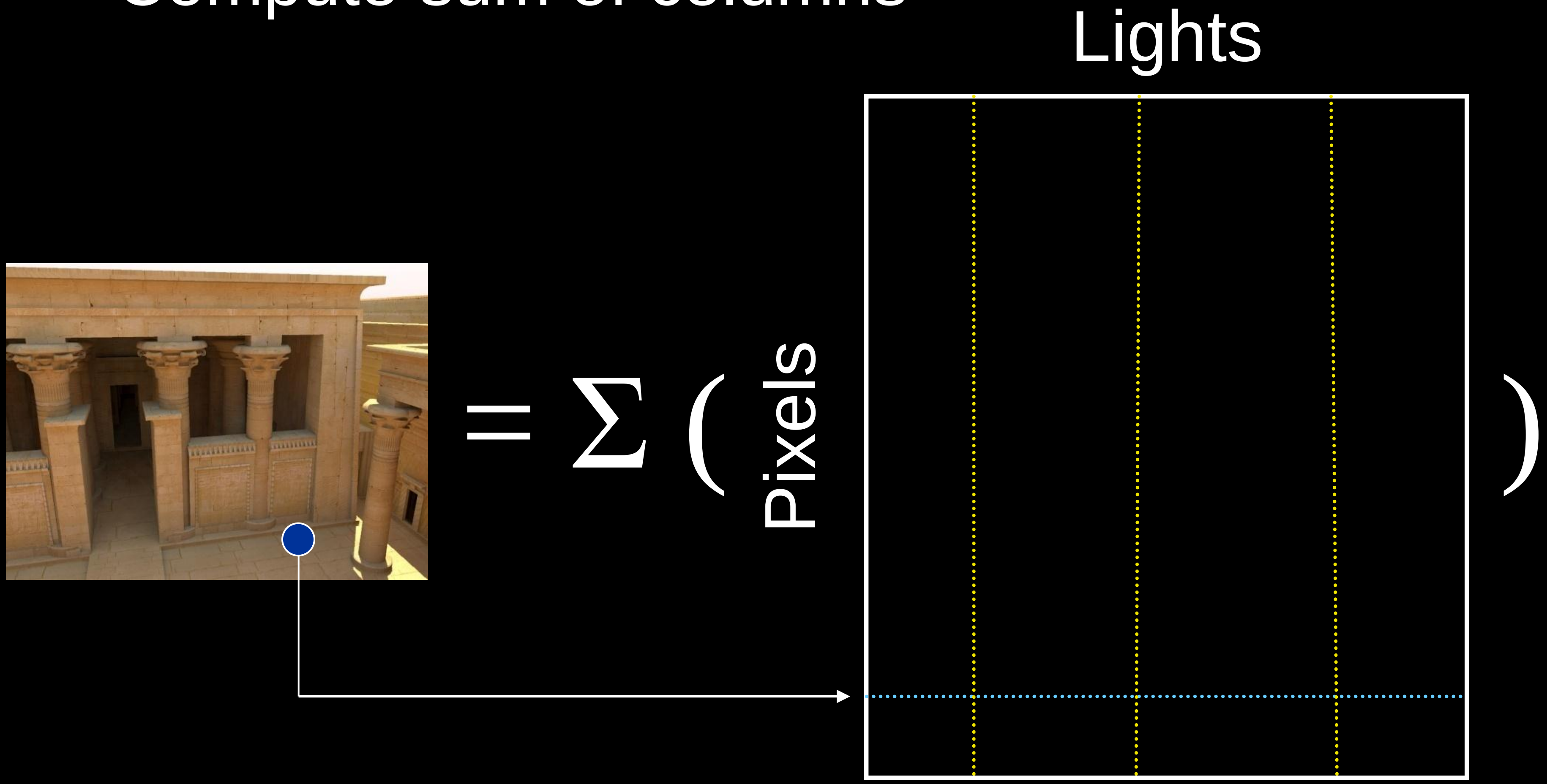
Pixels

(2,000,000)



Problem Statement

- Compute sum of columns



Lights

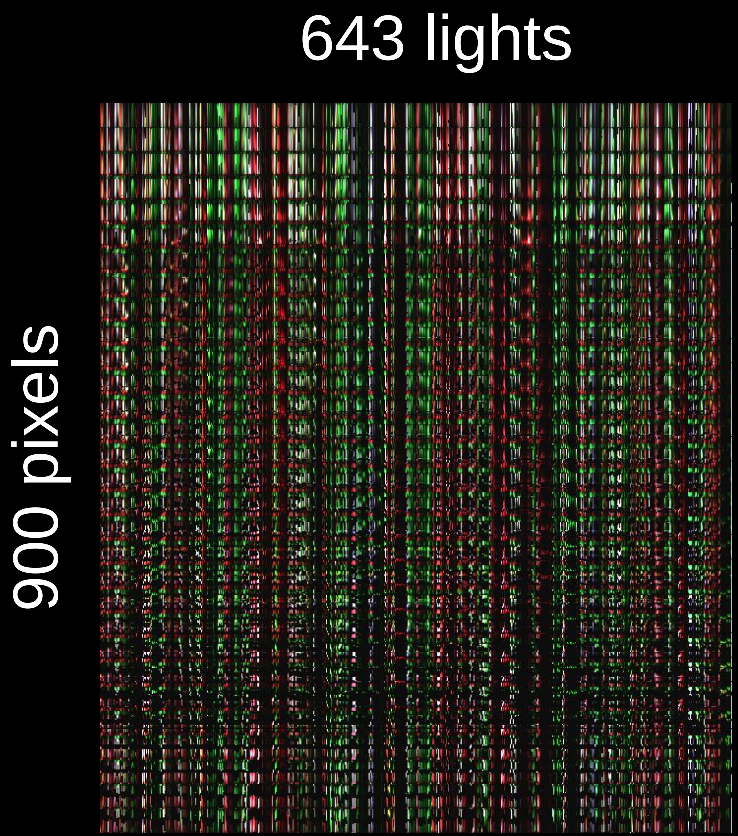
$= \sum ($ Pixels $)$

- Note: We only have oracle A(i,j)

Matrix has structure

A simple scene

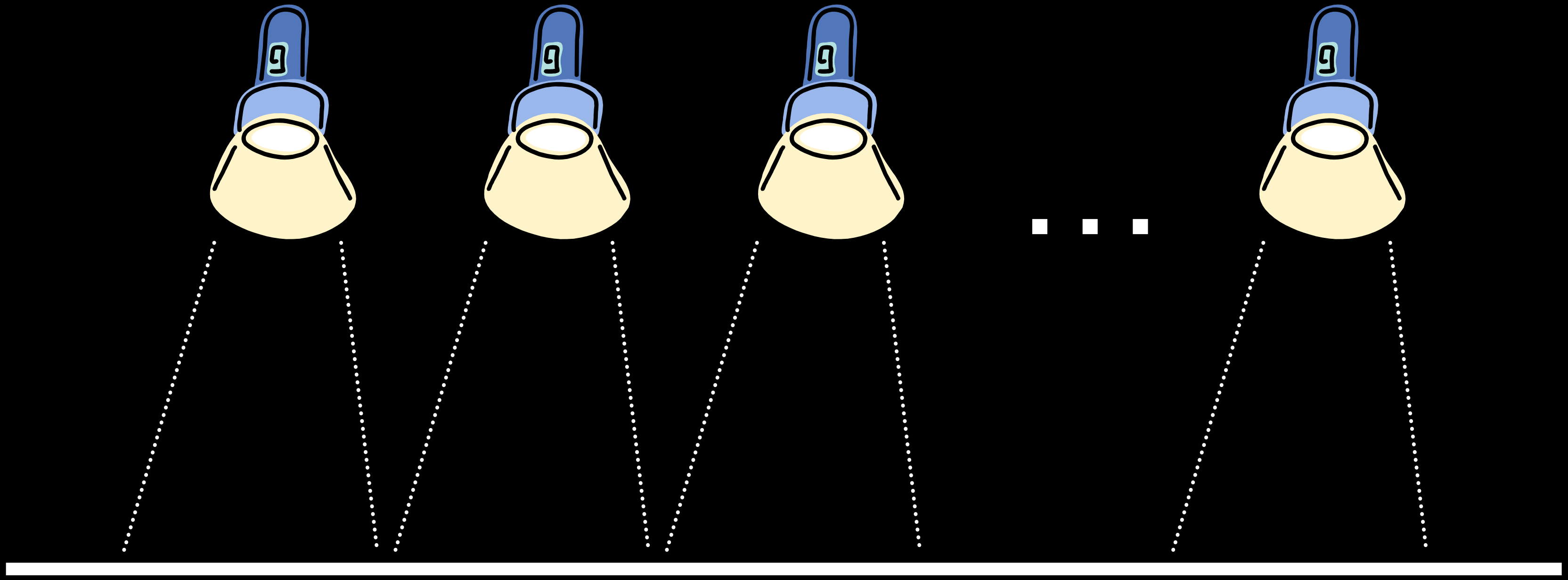
30 x 30 image



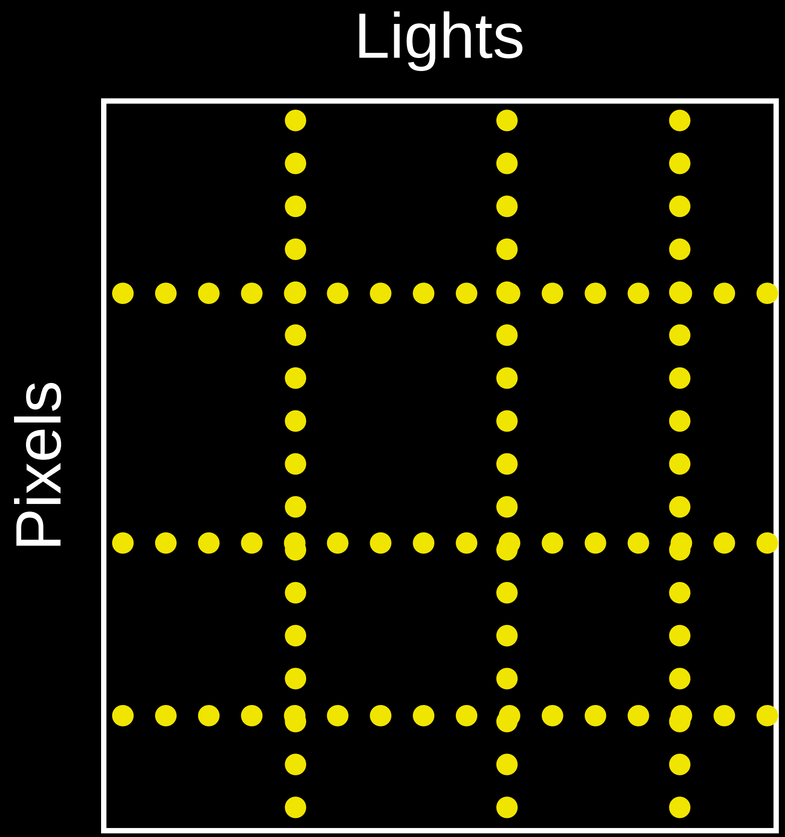
The matrix

Low Rank Assumption Violation

- Bad case: lights with very local contribution



Sampling Pattern Matters

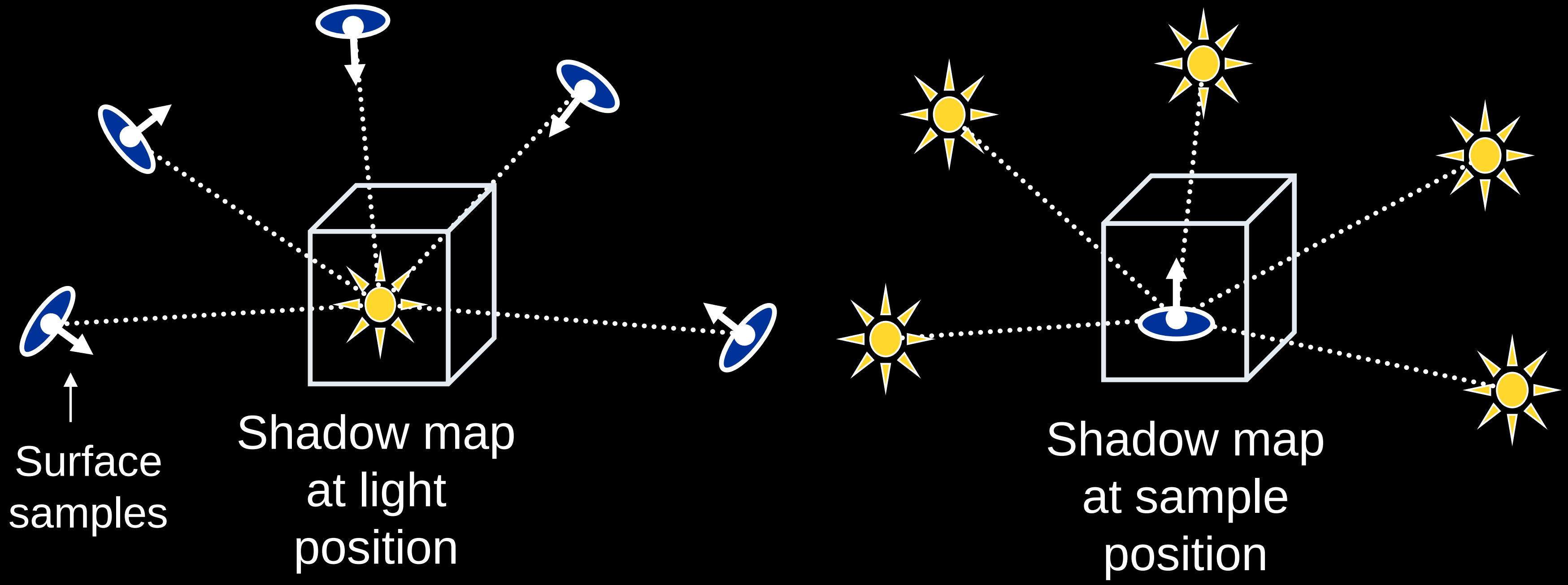


Point-to-point visibility: Ray-tracing

Point-to-many-points visibility: Shadow-mapping

Row-Column Shadow Duality

- Columns: Regular Shadow Mapping
- Rows: Also Shadow Mapping!

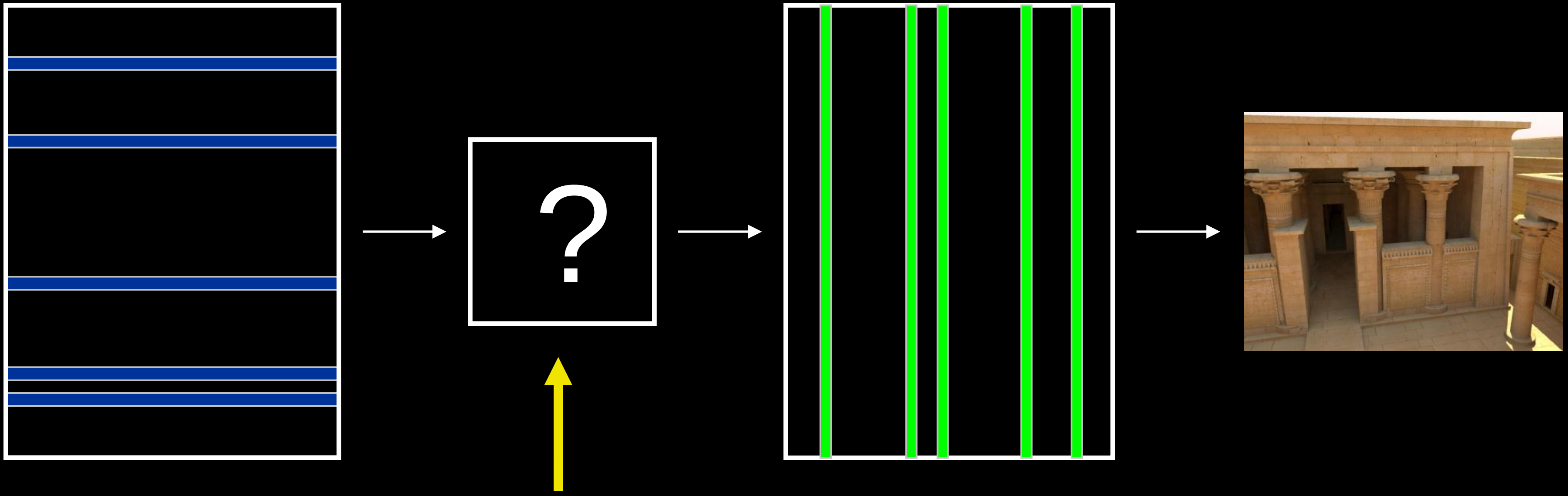


Surface samples

Shadow map at light position

Shadow map at sample position

Exploration and Exploitation



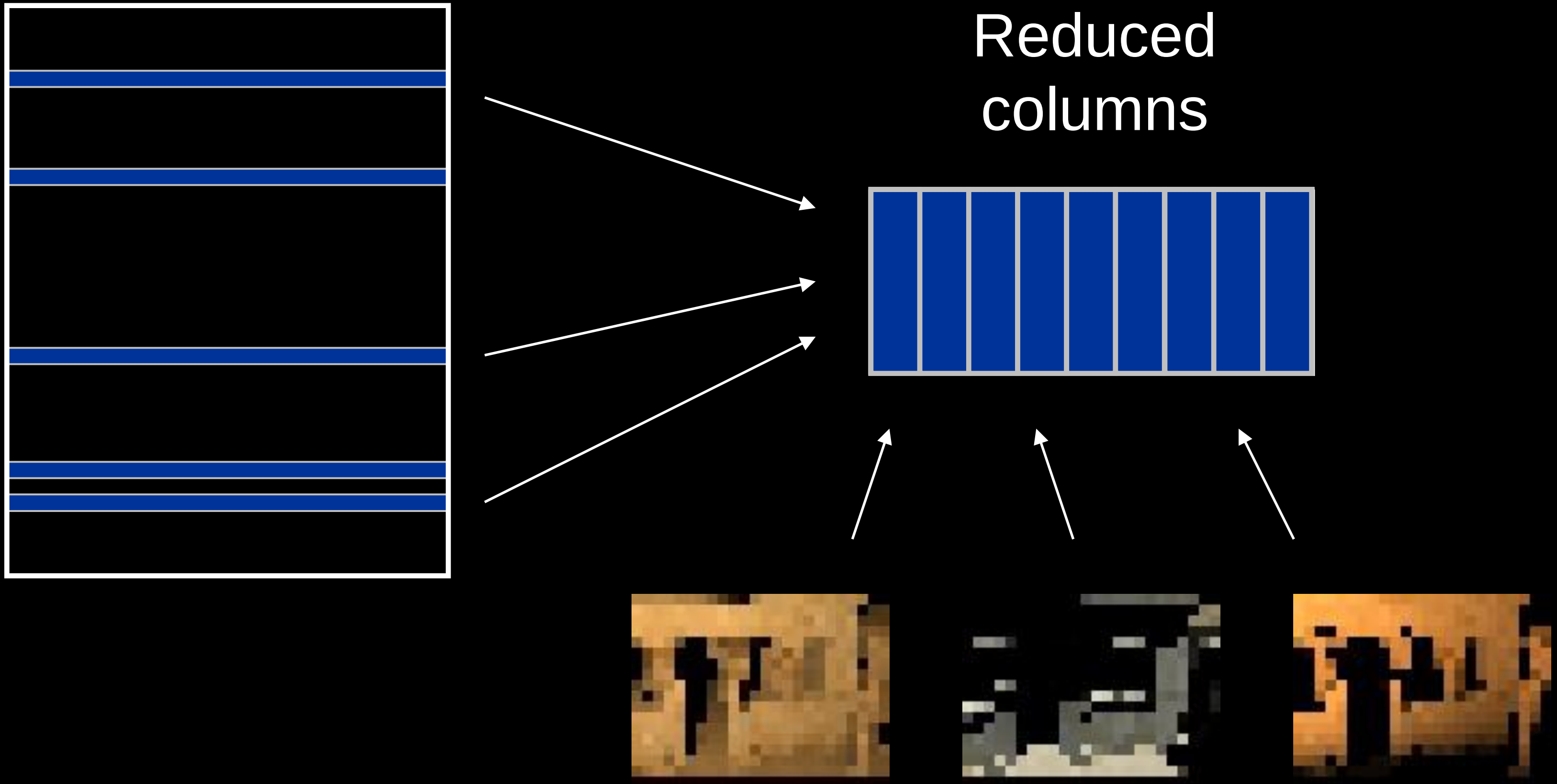
compute rows
(explore)

how to choose
columns and
weights?

compute columns
(exploit)

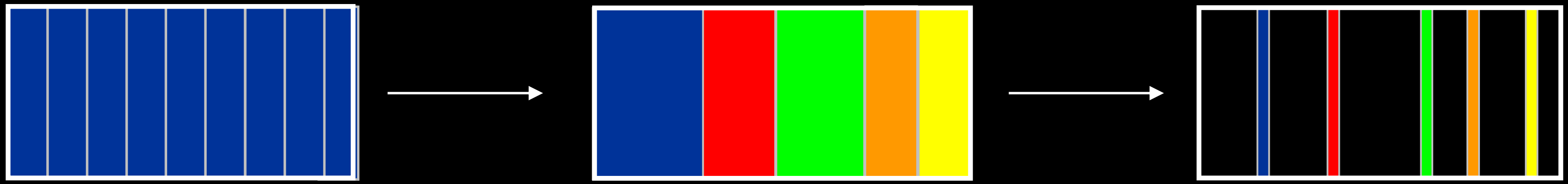
weighted
sum

Reduced Matrix



Reduced columns

Clustering Approach



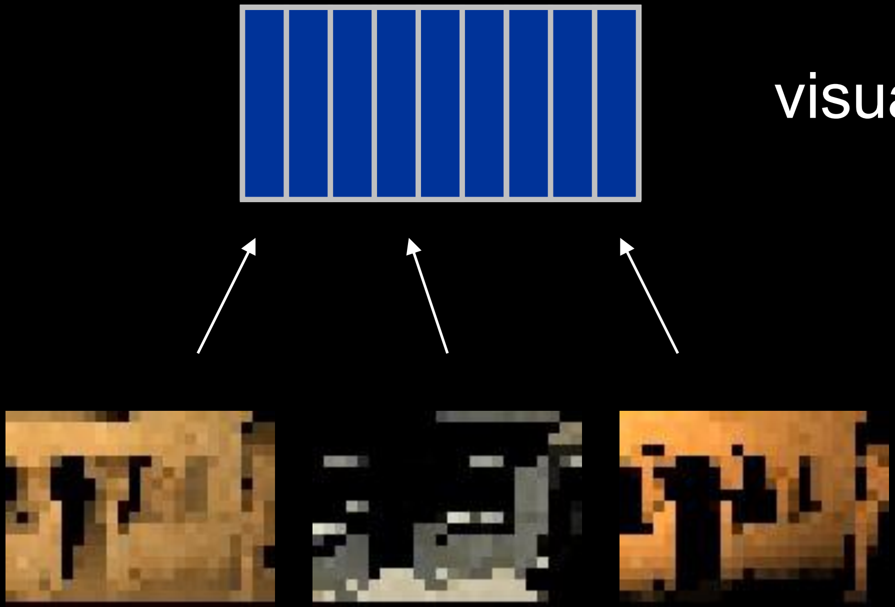
Reduced columns

Choose k clusters

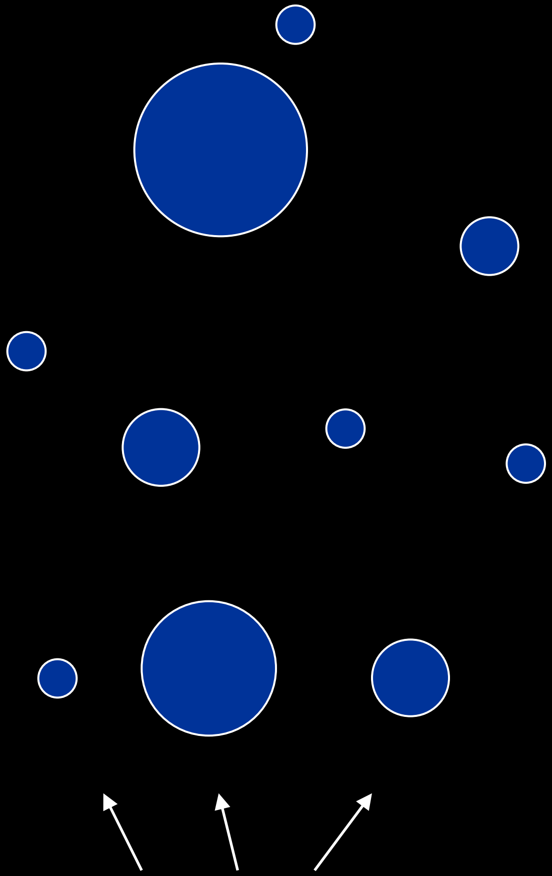
Choose representative columns

Visualizing the Reduced Columns

Reduced columns: vectors in high-dimensional space



visualize as …



radius = norm

The Clustering Metric

- Minimize: $$\underbrace{\sum_{p=1,\ldots,k} cost(C_p)}_{\text{total cost of all clusters}}$$

- where: $$cost(C) = \sum_{i,j \in C} w_i \; w_j \; \|\mathbf{x}_i - \mathbf{x}_j\|^2$$

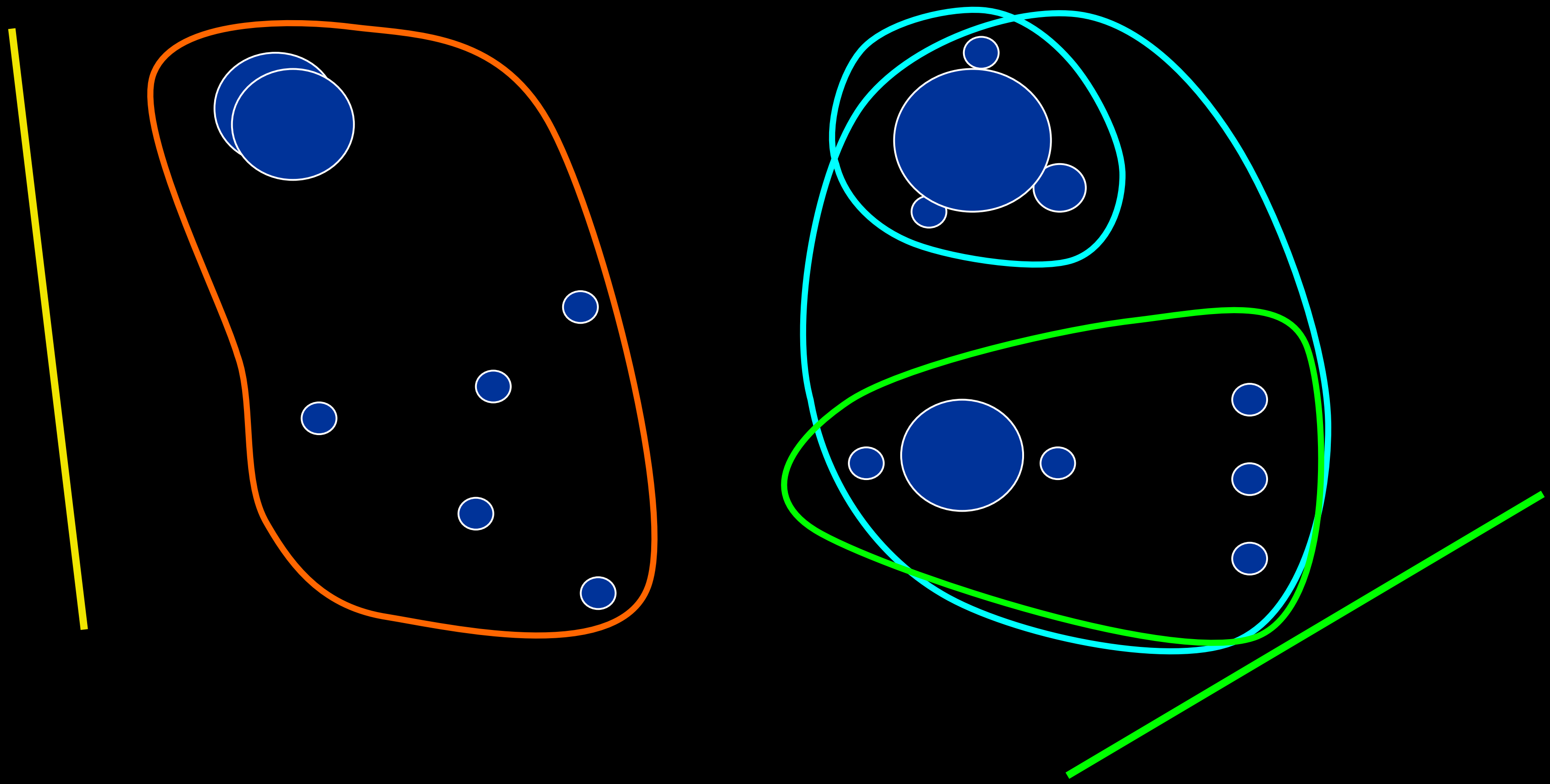
cost of a cluster

sum over all pairs in it

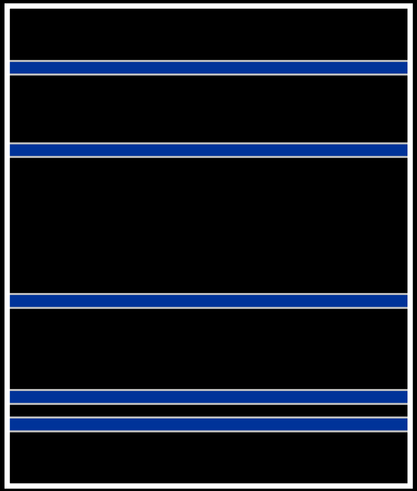
norms of the reduced columns

squared distance between normalized reduced columns

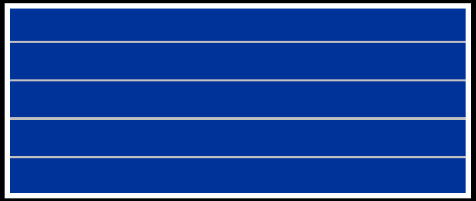
Clustering by Divide & Conquer



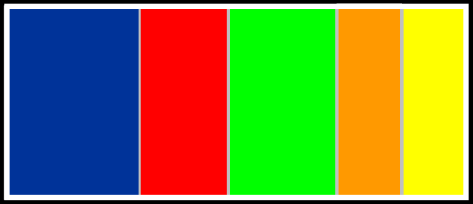
Full Algorithm



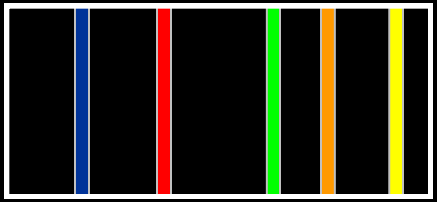
Compute rows (GPU)



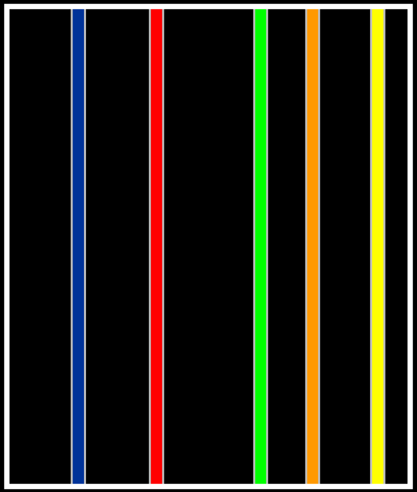
Assemble rows into reduced matrix



Cluster reduced columns



Choose representatives

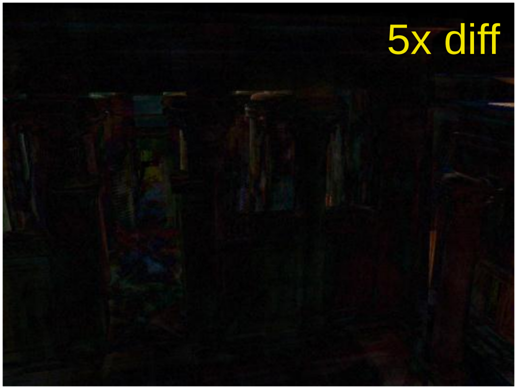


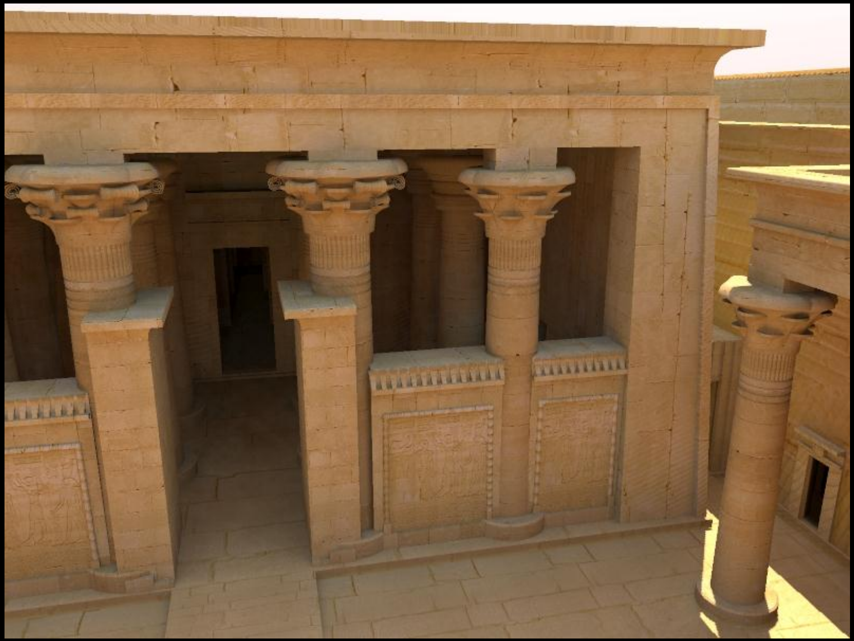
Compute columns (GPU)

Weighted sum

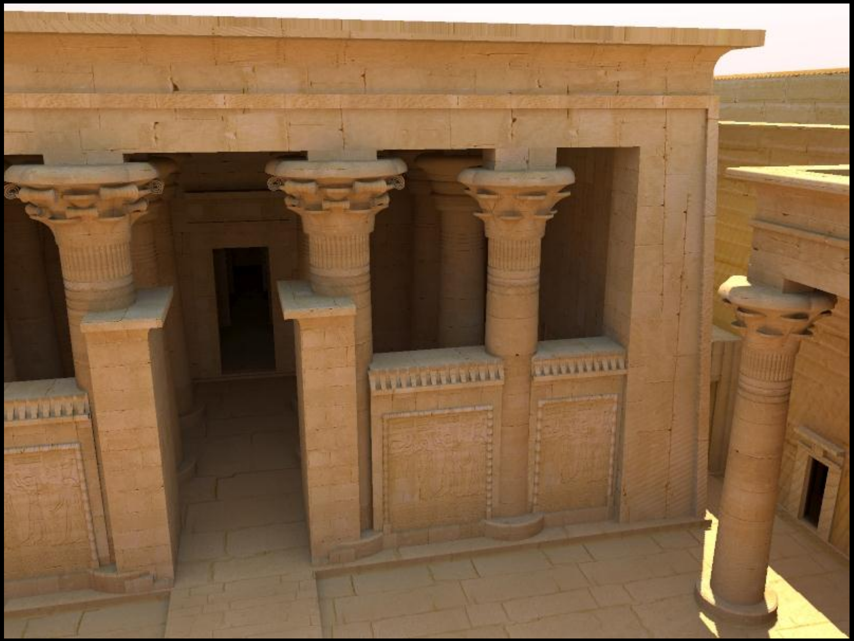
Results: Temple

- 2.1m polygons
- Mostly indirect & sky illumination
- Indirect shadows





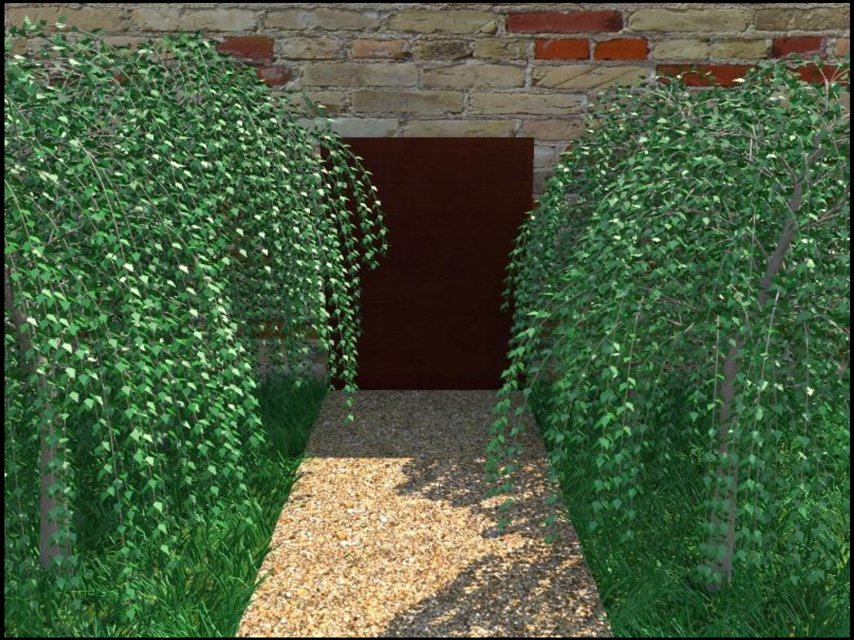
Our result: 16.9 sec
(300 rows + 900 columns)



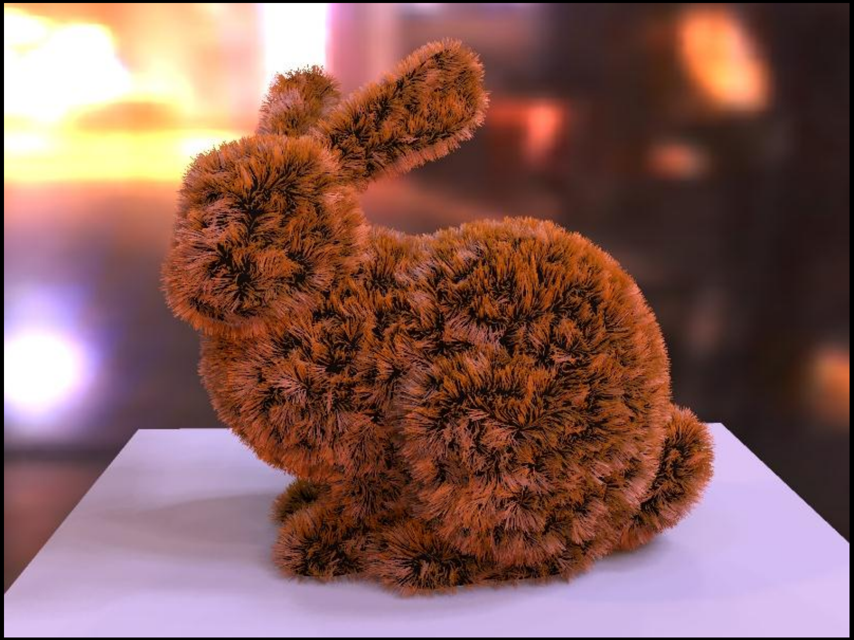
Reference: 20 min
(using all 100k lights)

Results: Trees and Bunny

- Complex incoherent geometry
- Low rank, not low frequency



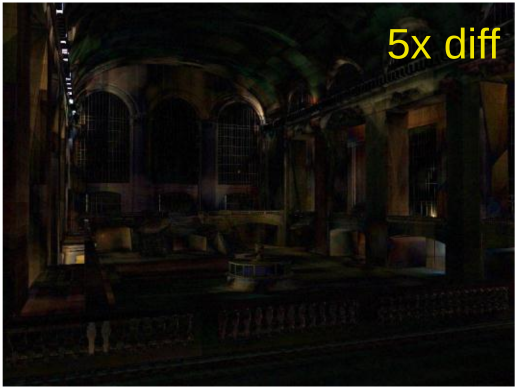
Our result: 2.9 sec
(100 rows + 200 columns)

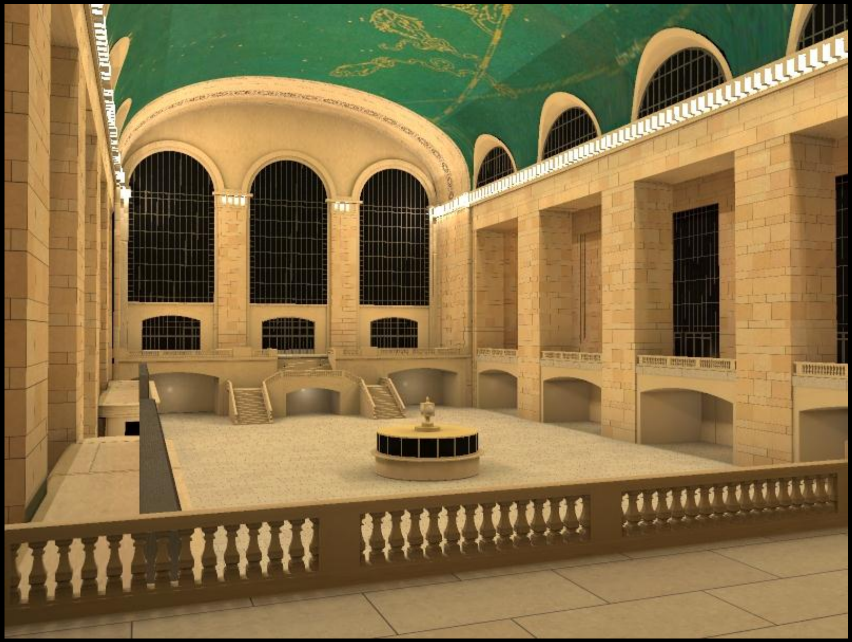


Our result: 3.8 sec
(100 rows + 200 columns)

Results: Grand Central

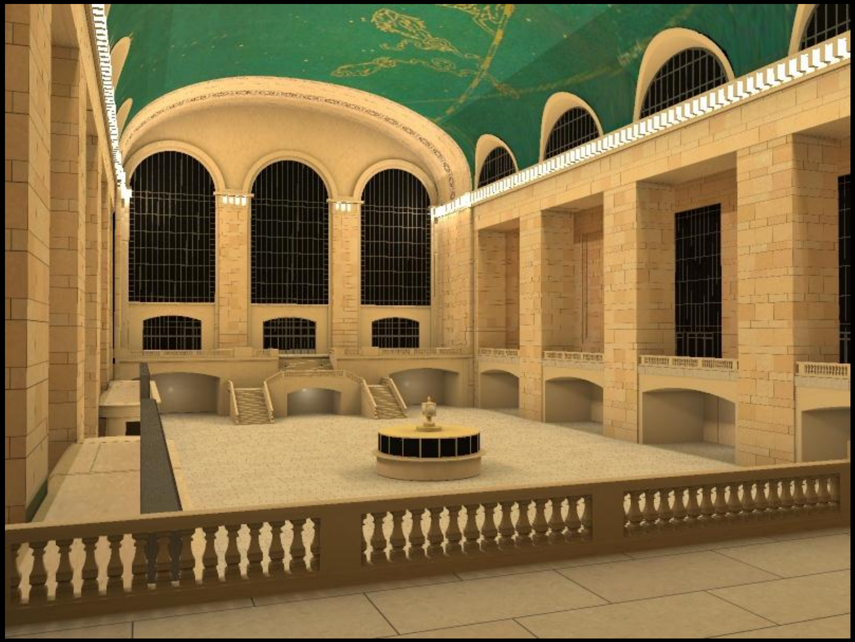
- 1.5m polygons
- Point lights between stone blocks





Our result: 24.2 sec

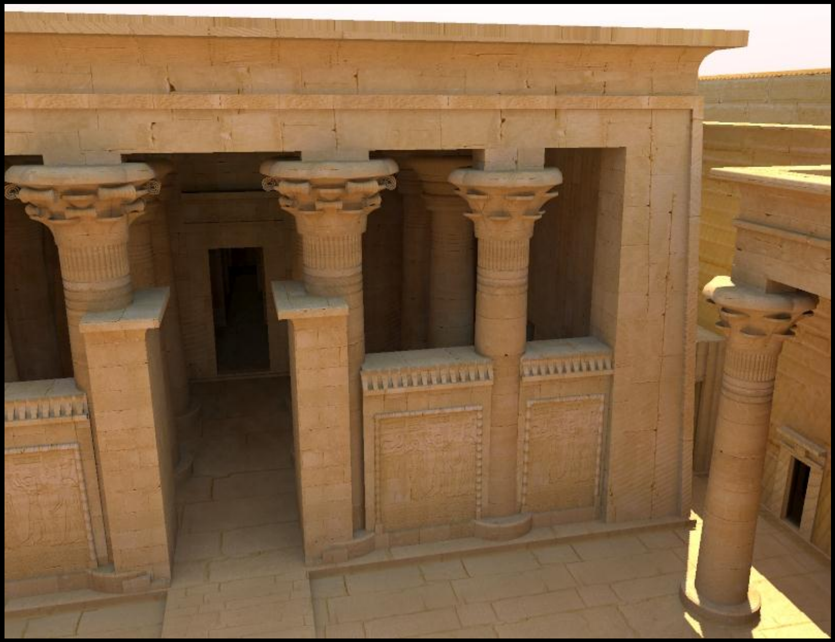
(588 rows + 1176 columns)



Reference: 44 min

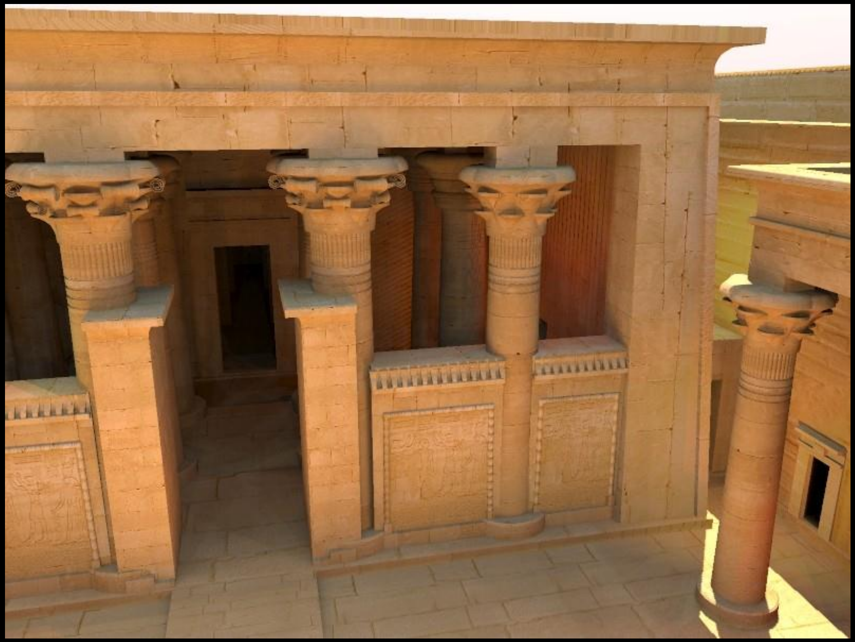
(using all 100k lights)

Advantage: Adaptive Stratification



Our result

(432 rows + 864 columns)

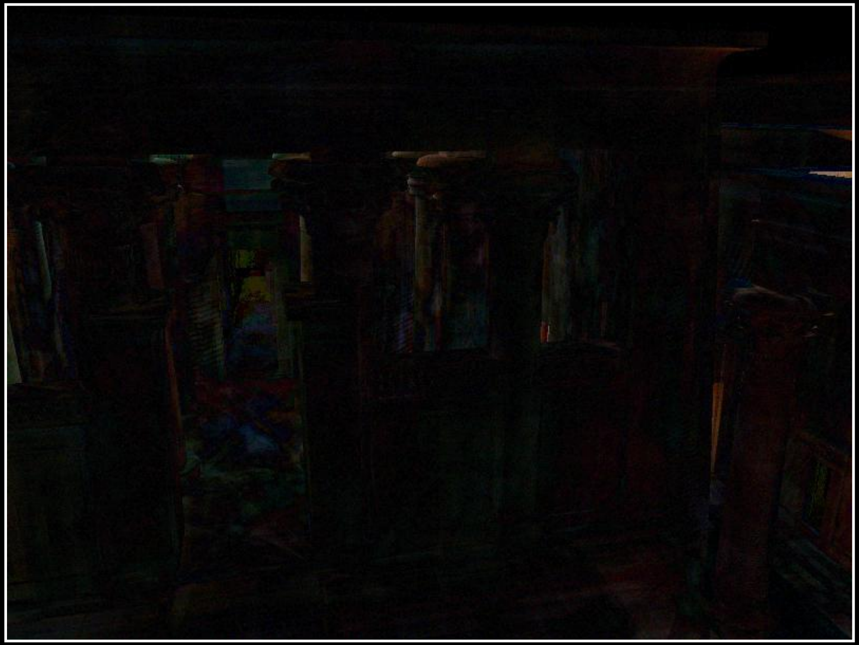


Importance sampling

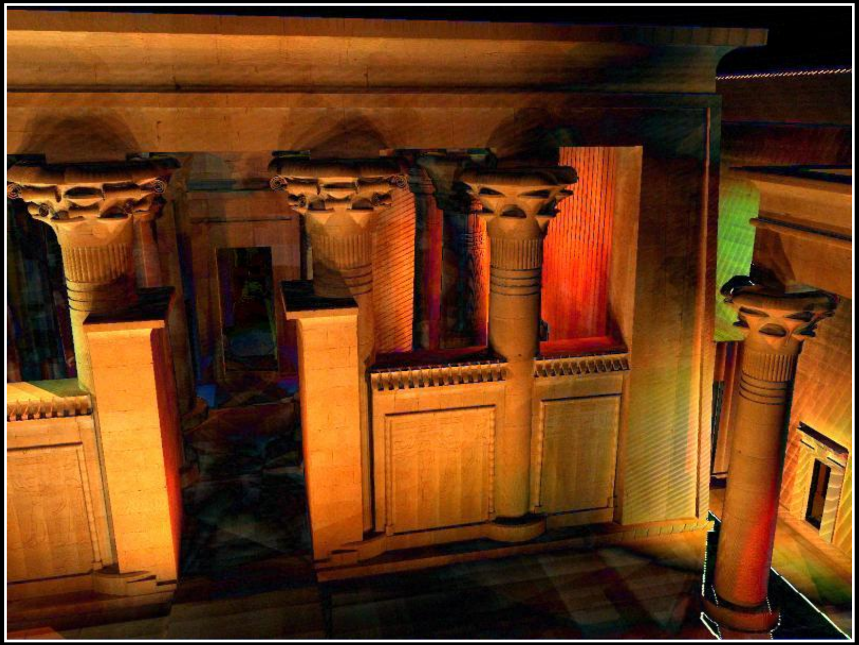
(Using 1455 lights)

Equal time comparison

Advantage: Adaptive Stratification



Our result

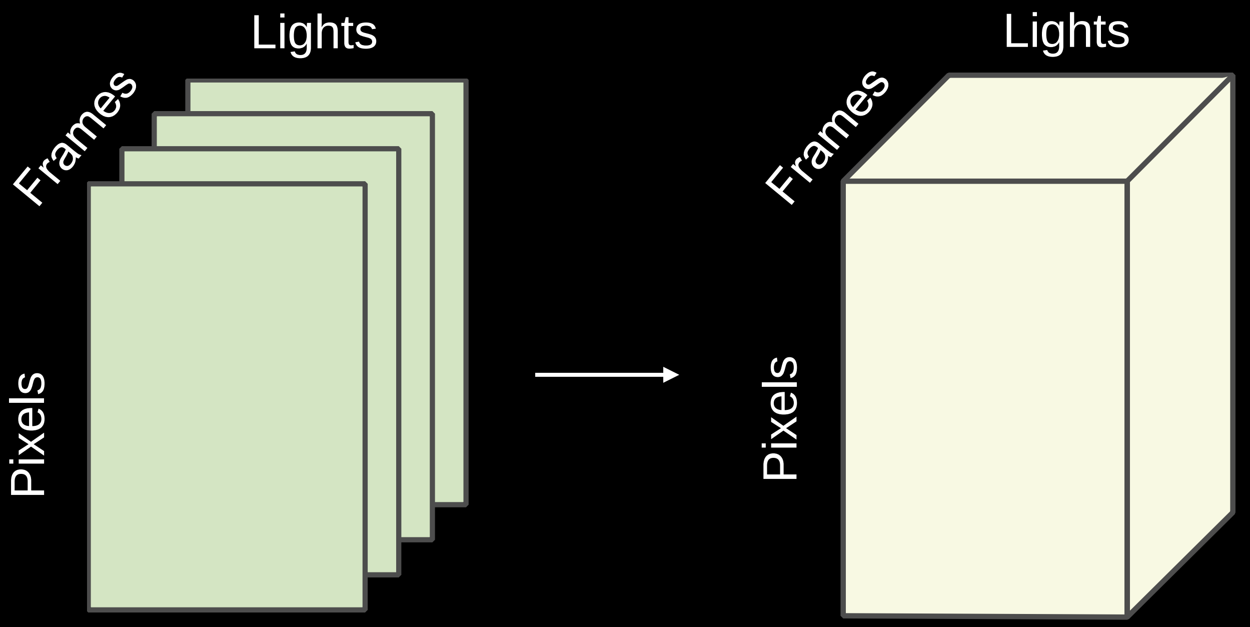


Importance sampling

Equal time comparison:

5x difference from reference

Animations: Tensor Extension

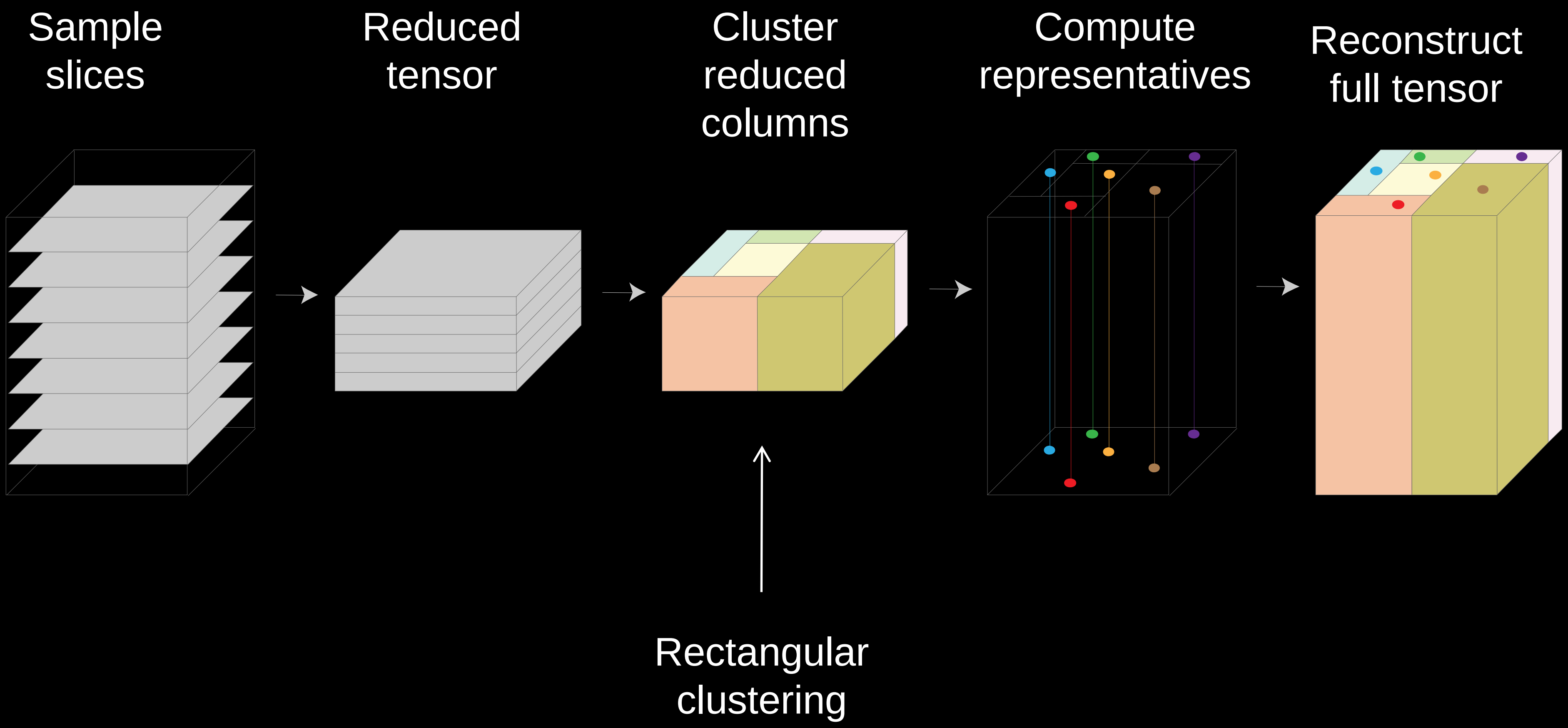


Sequence of matrices

Tensor view

- Size of tensor in our results: 307,200 x 65,536 x 40

Tensor Extension - Overview



Sample slices

Reduced tensor

Cluster reduced columns

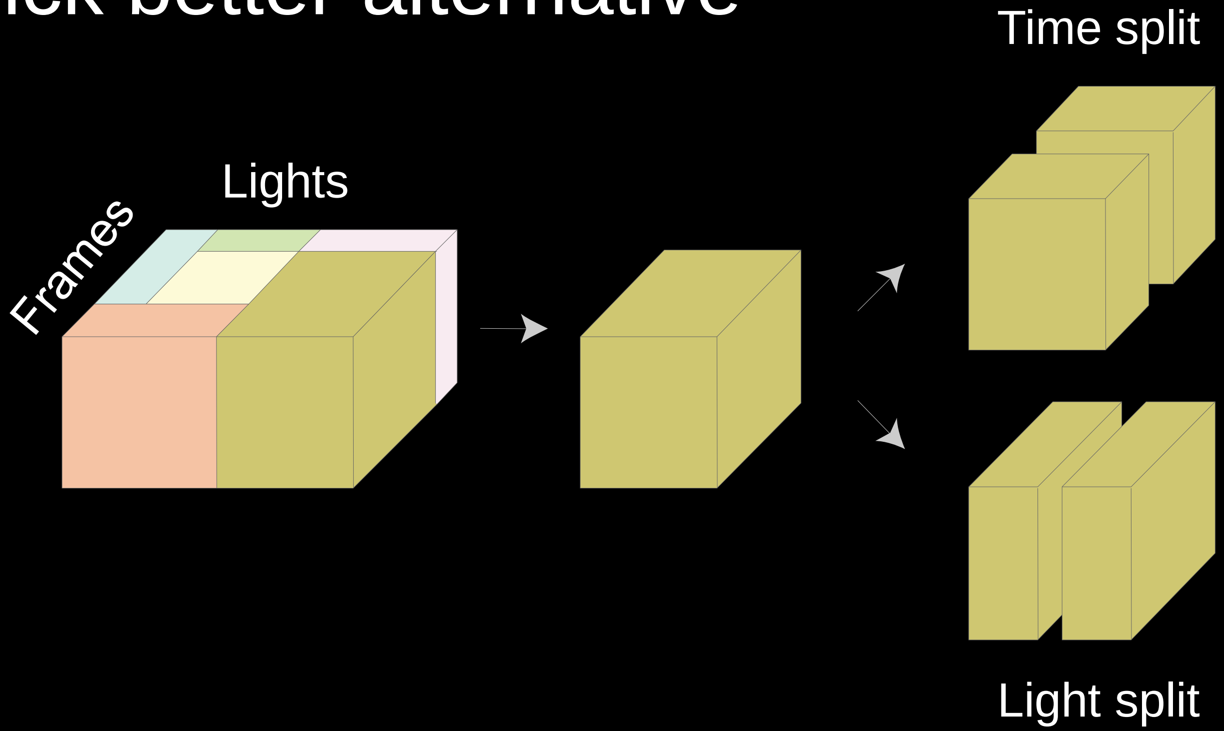
Compute representatives

Reconstruct full tensor

Rectangular clustering

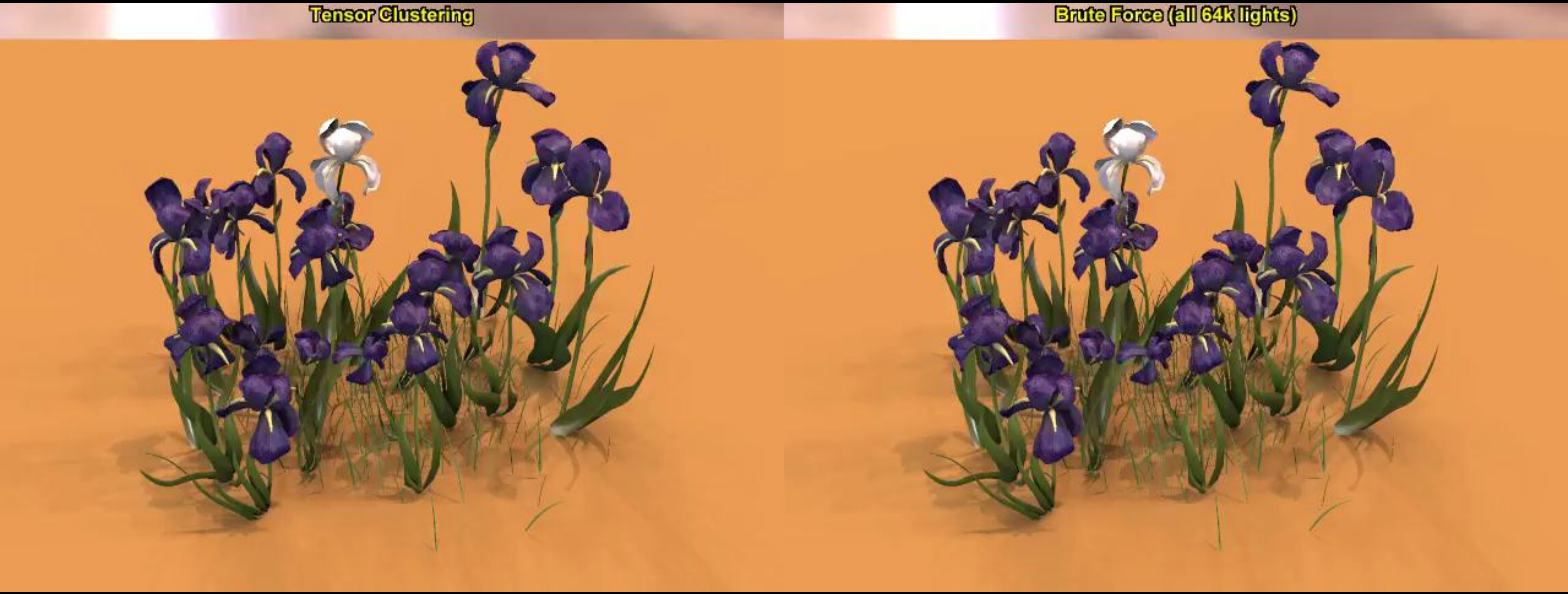
Splitting a cluster

- Pick cluster with highest cost
- Try splitting in time
- Try splitting in lights
- Pick better alternative



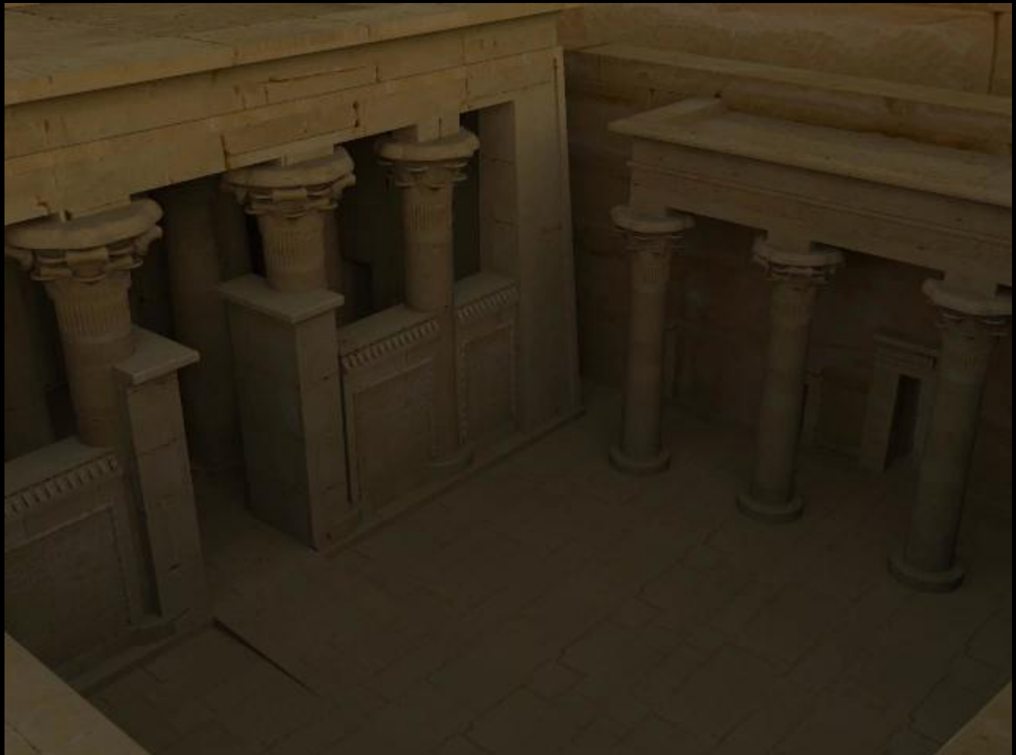
Results - Iris

- 51k triangles, 65,536 lights
- Deforming objects, high-frequency shadows
- 6.9 sec / frame (brute-force: 2 min / frame)



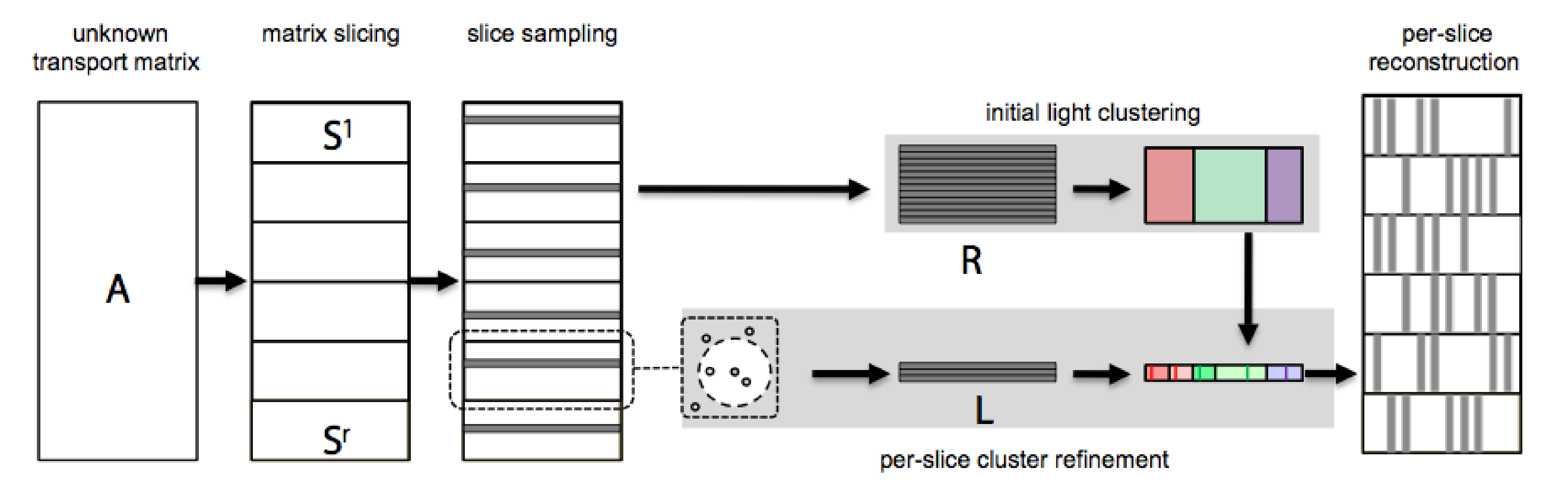
Results - Temple

- 2.1m triangles, 65,536 lights
- Sun & sky lighting, moving sun
- Multiple indirect light bounces
- 26 sec / frame (brute-force: 33.5 min / frame)

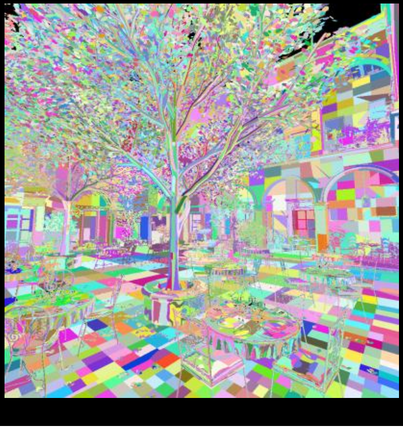


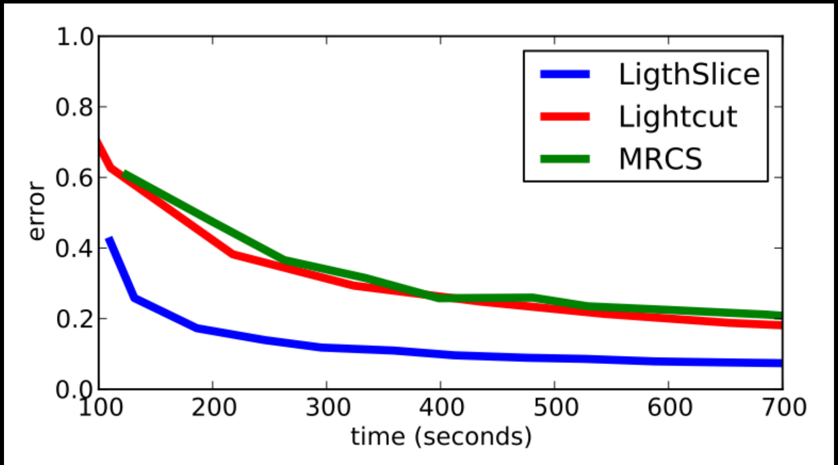
LightSlice [Ou and Pellacini 2011]

- Compute initial clustering
- Refine it differently in different “slices”
- Use neighboring slices to get more rows



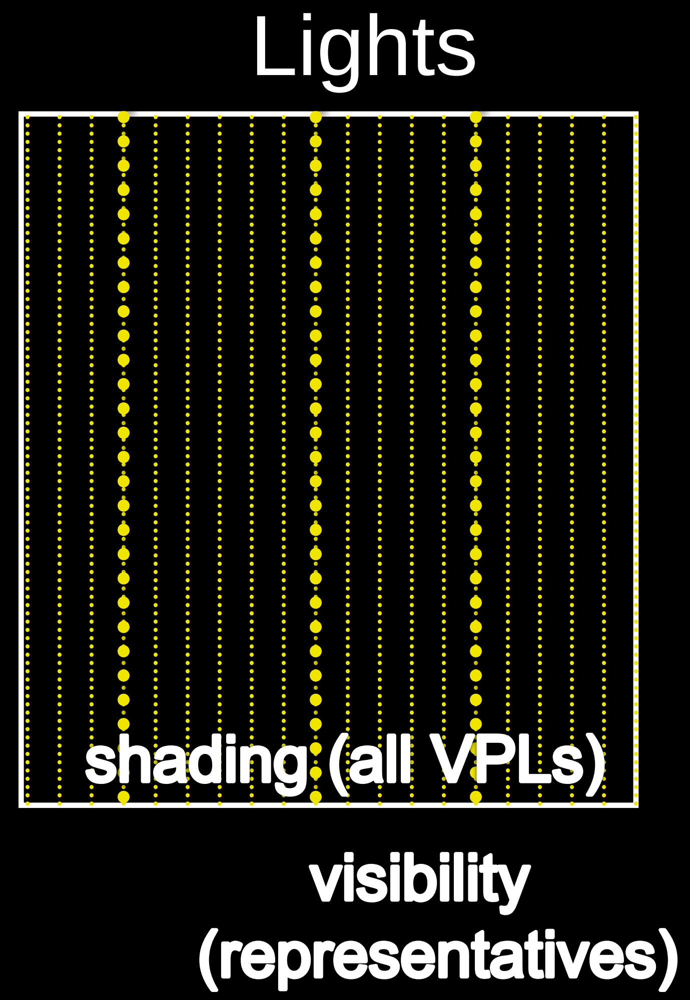
LightSlice: Results





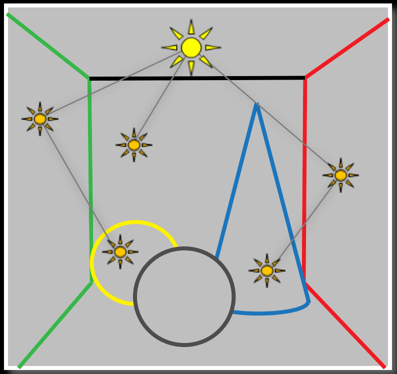
Visibility Clustering [Davidovič et al 2010]

- Separate shading from visibility
- Cluster only visibility
- Shade from all VPLs



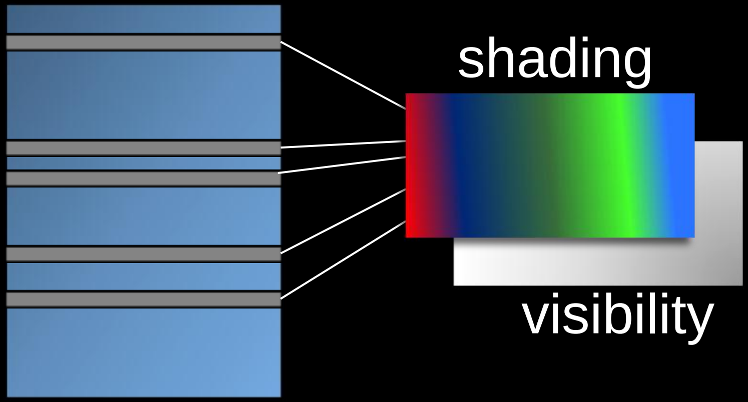
Visibility Clustering Overview

VPL tracing



Row sampling

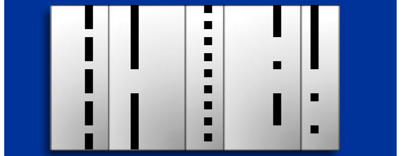
Reduced matrix



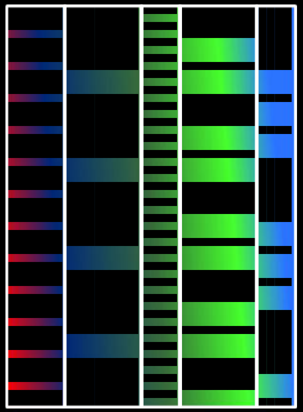
shading

visibility

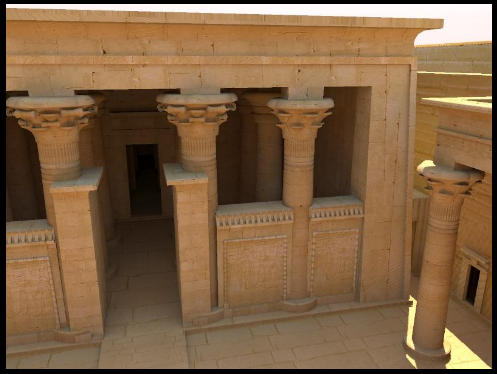
Visibility clustering



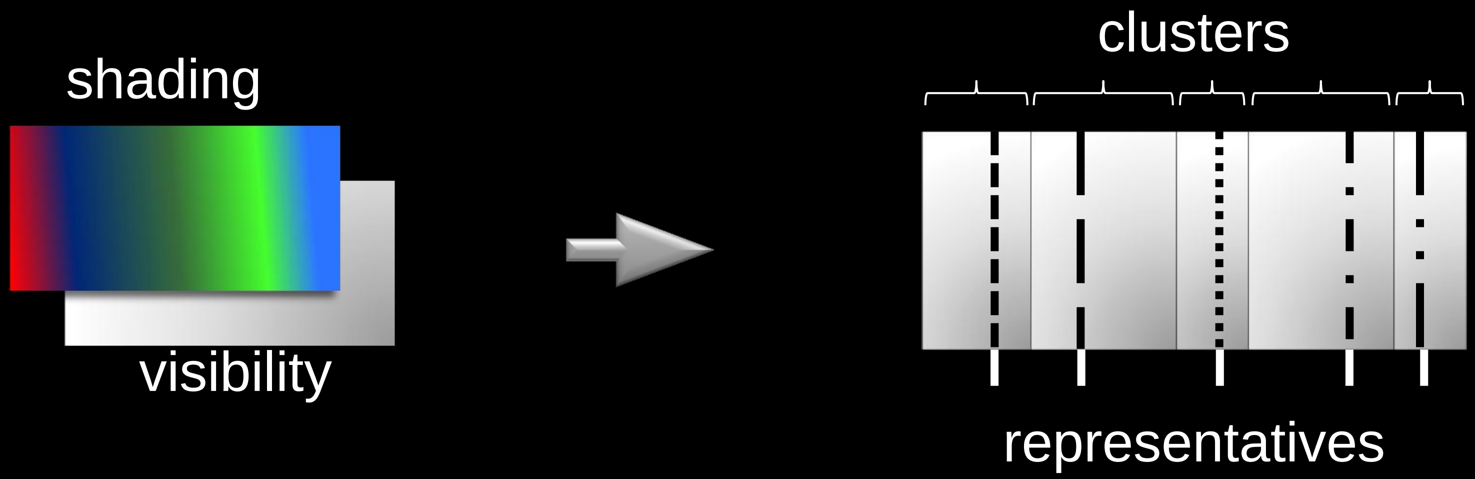
Render lights with reps’ visibility



Result



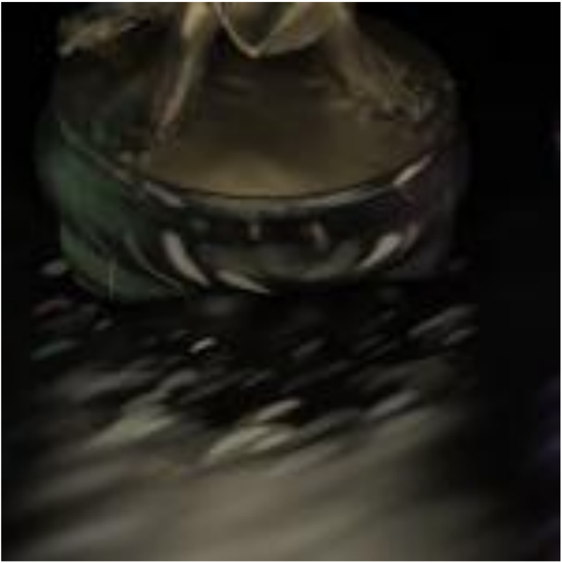
Visibility clustering

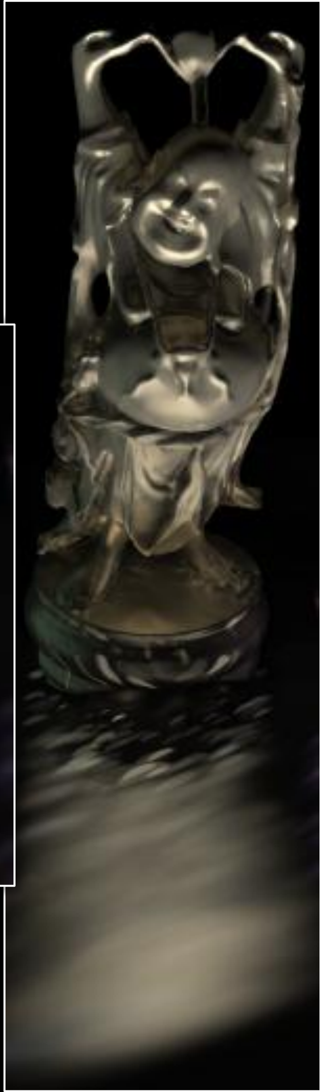


- Clustering algorithm
  - Divide & conquer (top-down splitting)
  - Modified clustering cost
    - L2 error of reduced matrix due to visibility approximation

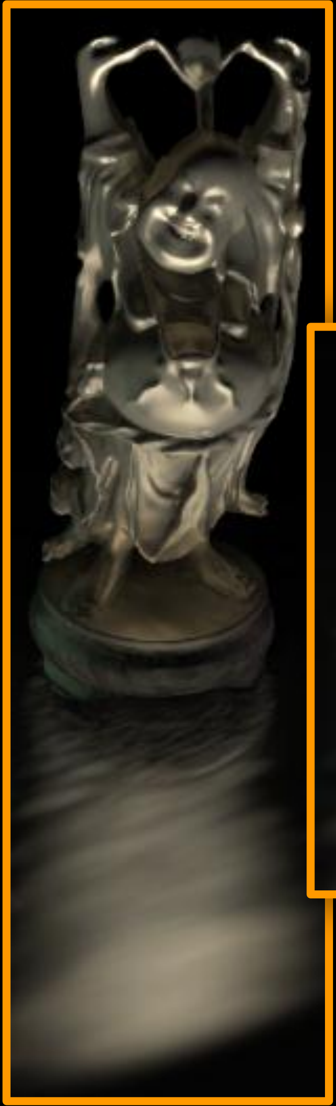
Visibility clustering result

Matrix row-column sampling

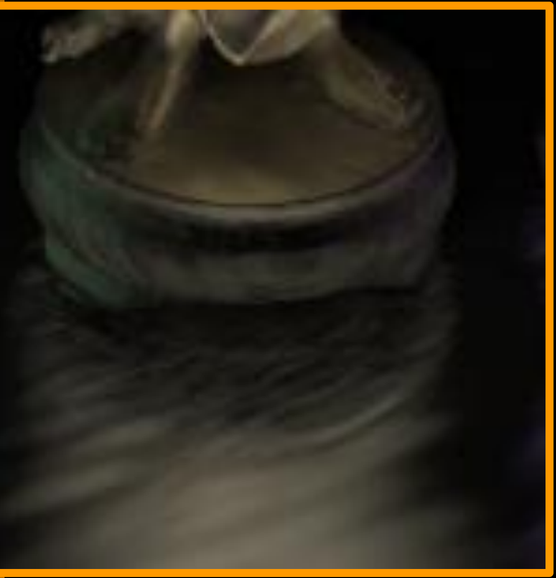




10k shadow maps
10k shading lights

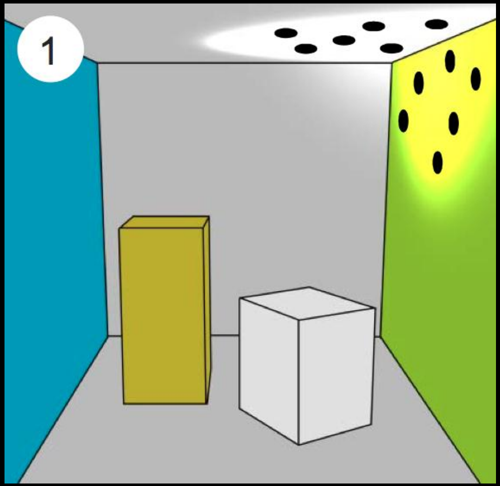


Our visibility clustering

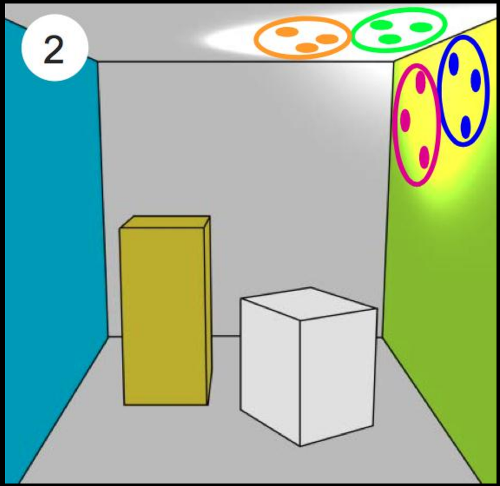


5k shadow maps
200k shading lights

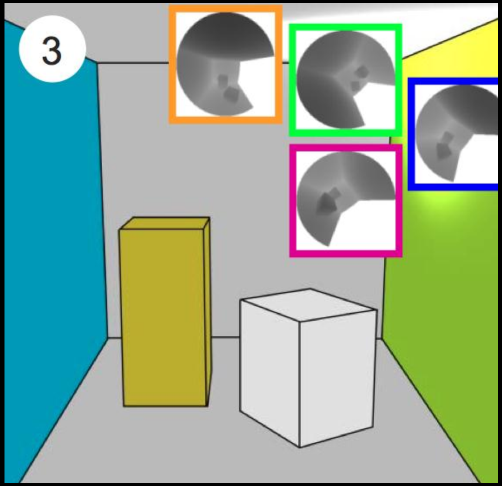
Clustered Visibility [Dong et al 2009]



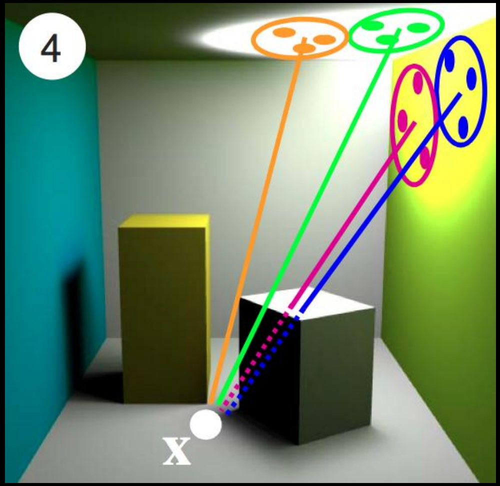
Trace VPLs



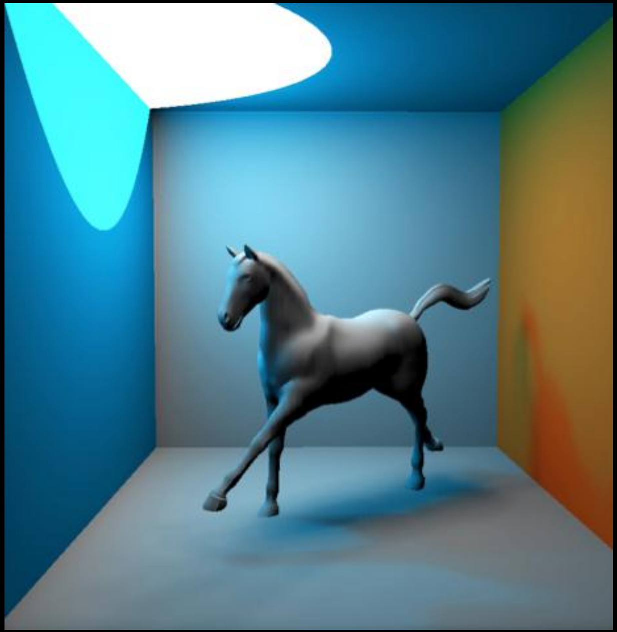
K-means clustering



Soft shadow maps



Compute full shading



Real-time diffuse indirect illumination

Conclusion

- Row-column sampling algorithms
  - Handle large numbers of VPLs
  - Alternatives to lightcuts
- Open Problems
  - How many rows + columns?
    - Pick automatically
  - Row / column alternation
  - Progressive algorithm:
    - stop when user likes the image