Scalable many-light methods

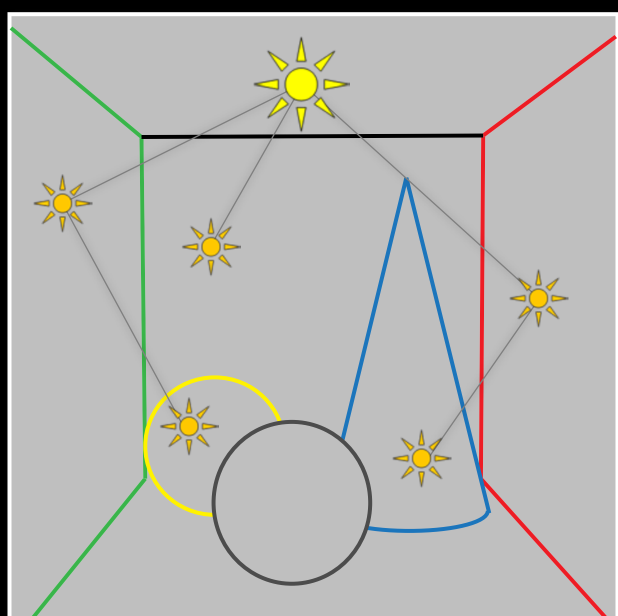
Jaroslav Křivánek

Charles University in Prague

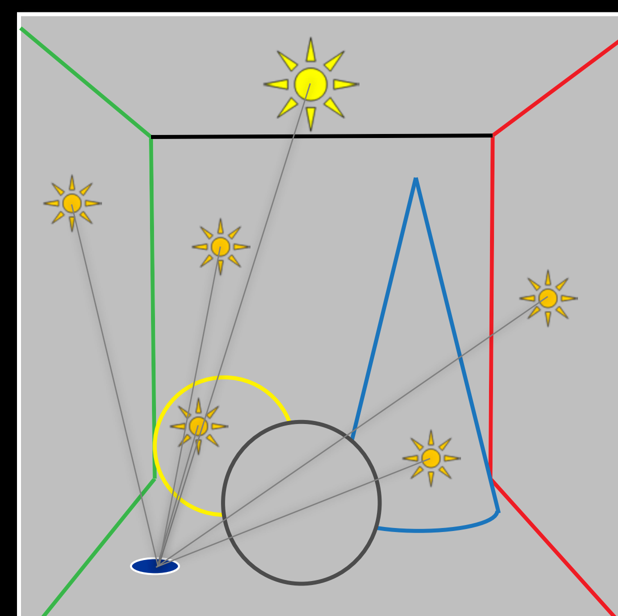
Instant radiosity

- Approximate indirect illumination by
  Virtual Point Lights (VPLs)

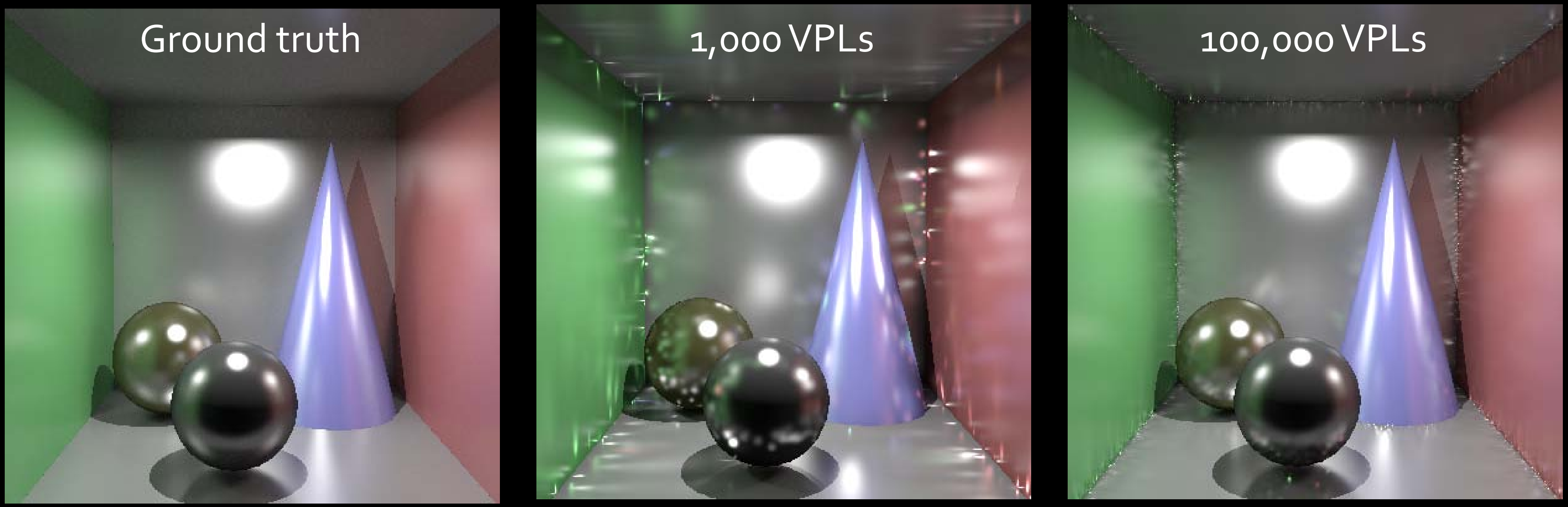
1. Generate VPLs



2. Render with VPLs



Instant radiosity with glossy surfaces



- Large number of VPLs required
  - True even for diffuse scenes
  - Scalability issues

Scalable many-light methods

1. Generate many, many VPLs

2. Pick only the most relevant VPLs for rendering

Scalable many-light methods

- Choosing the right VPLs
  - Per-pixel basis
    - Lightcuts [Walter et al 05/06]
  - Per-image basis
    - Matrix Row Column Sampling [Hašan et al. 07]
  - Somewhere in-between
    - LightSlice [Ou & Pellacini 2011]
    - Importance caching [Georgiev et al. 2012]

Scalable many-light rendering

Lightcuts
Multidimensional Lightcuts

Walter et al., SIGGRAPH 2005/2006

Slides courtesy Bruce Walter:

http://www.graphics.cornell.edu/~bjw/papers.html

Lightcuts

Lightcuts: A Scalable Approach to Illumination

Bruce Walter Sebastian Fernandez Adam Arbree Kavita Bala Michael Donikian Donald P. Greenberg
*Program of Computer Graphics, Cornell University**

Abstract

Lightcuts is a scalable framework for computing realistic illumination. It handles arbitrary geometry, non-diffuse materials, and illumination from a wide variety of sources including point lights, area lights, HDR environment maps, sun/sky models, and indirect illumination. At its core is a new algorithm for accurately approximating illumination from many point lights with a strongly *sublinear* cost. We show how a group of lights can be cheaply approximated while bounding the maximum approximation error. A binary light tree and perceptual metric are then used to adaptively partition the lights into groups to control the error vs. cost tradeoff.

We also introduce reconstruction cuts that exploit spatial coherence to accelerate the generation of anti-aliased images with complex illumination. Results are demonstrated for five complex scenes and show that lightcuts can accurately approximate hundreds of thousands of point lights using only a few hundred shadow rays. Reconstruction cuts can reduce the number of shadow rays to tens.

- http://www.graphics.cornell.edu/~bjw/papers.html

Complex Lighting

- Simulate complex illumination using VPLs
  - Area lights
  - HDR environment maps
  - Sun & sky light
  - Indirect illumination

- Unifies illumination

Area lights + Sun/sky + Indirect

Scalable

- Scalable solution for many point lights
  - Thousands to millions
  - Sub-linear cost

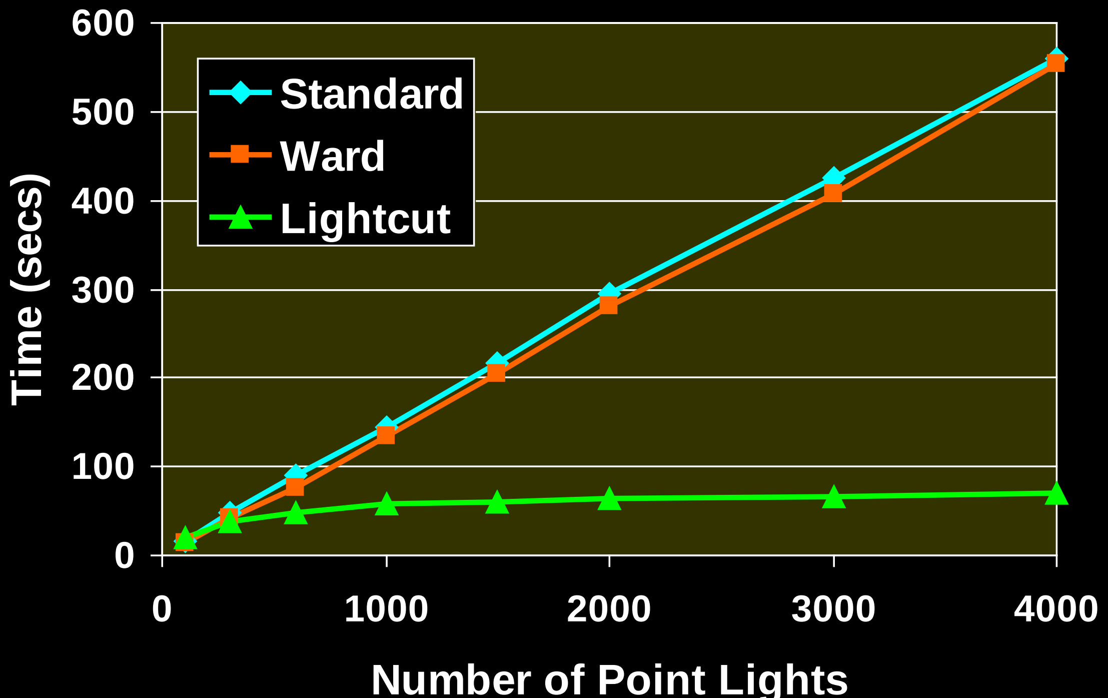
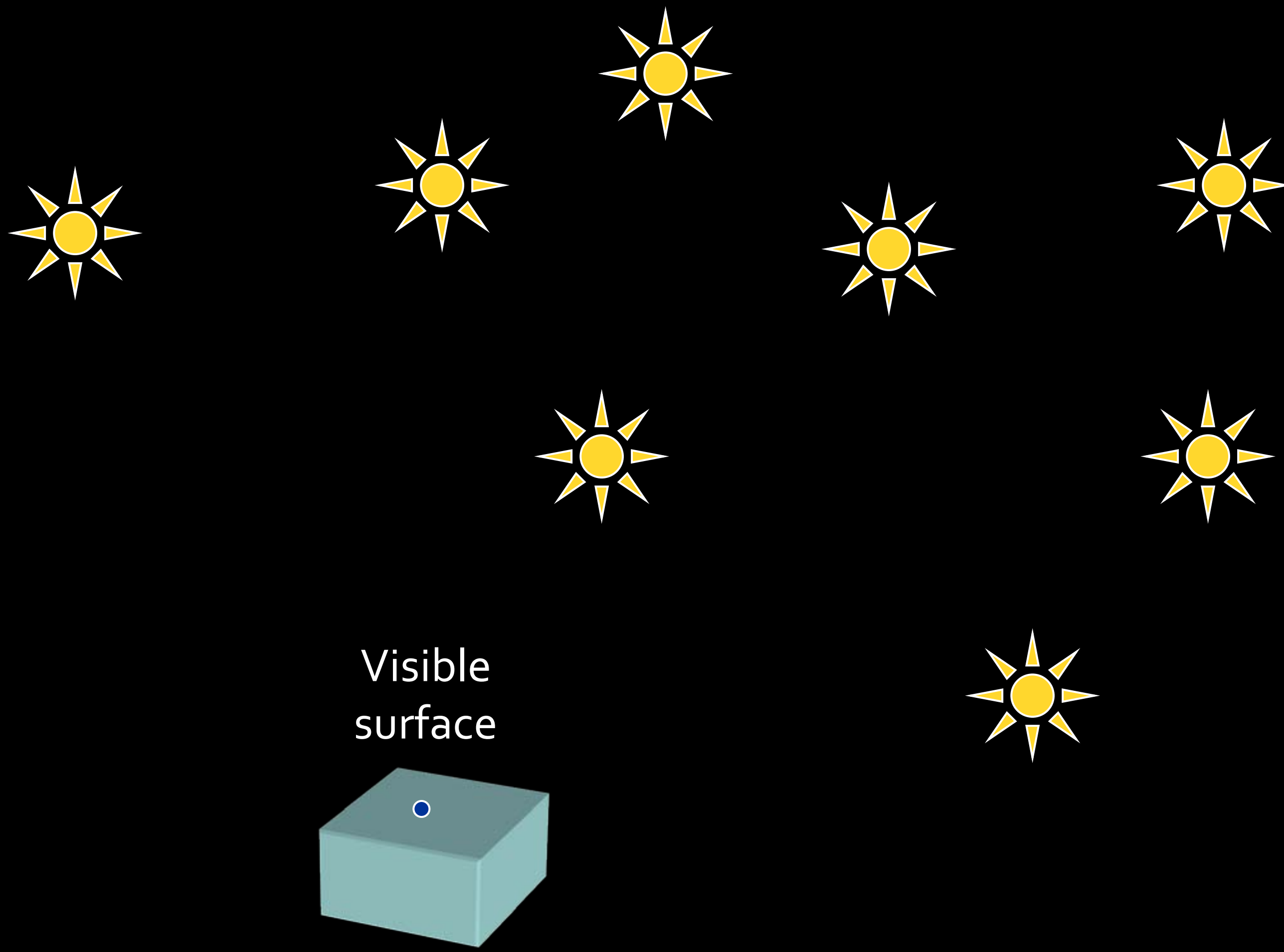
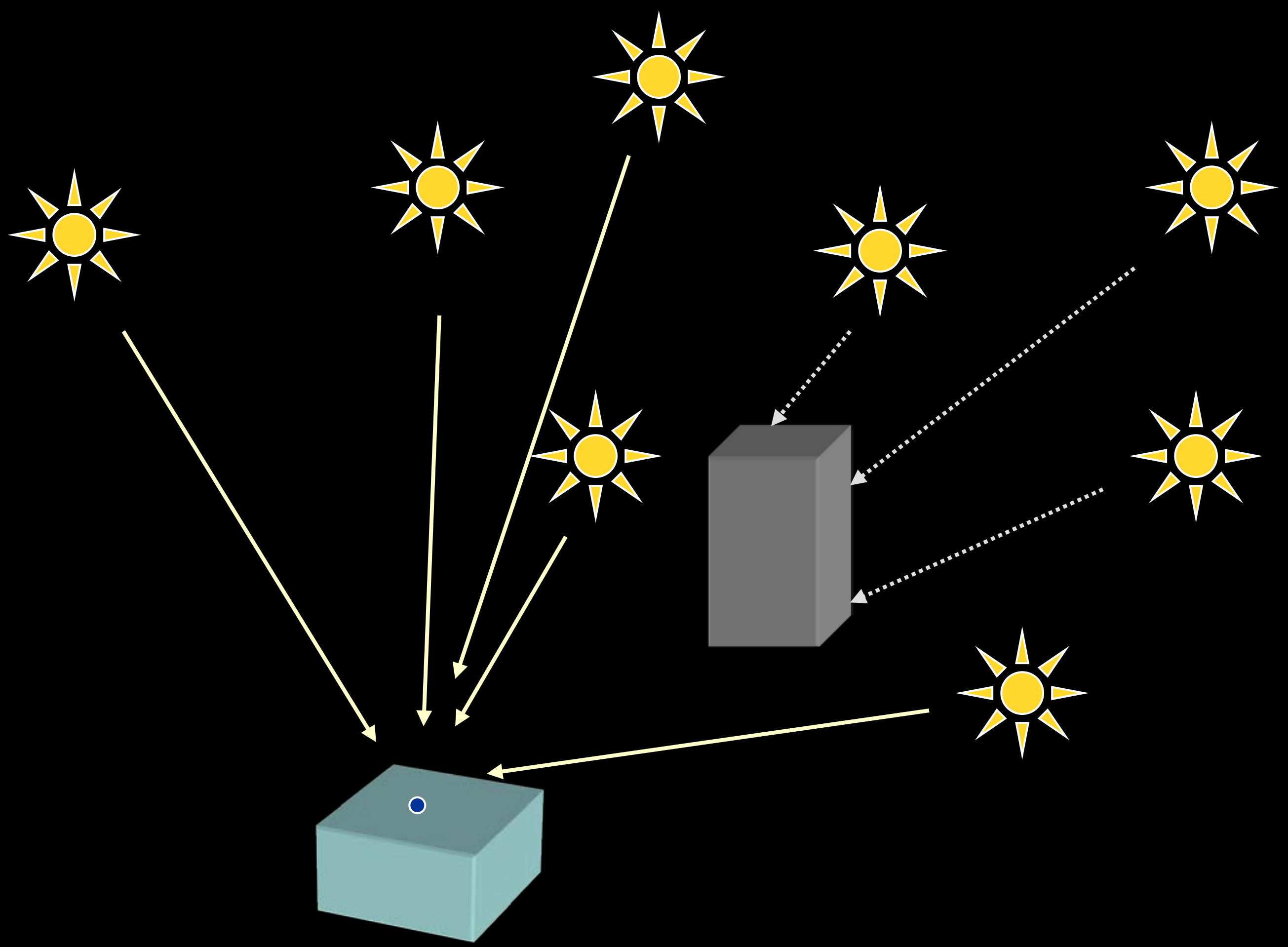


Tableau Scene

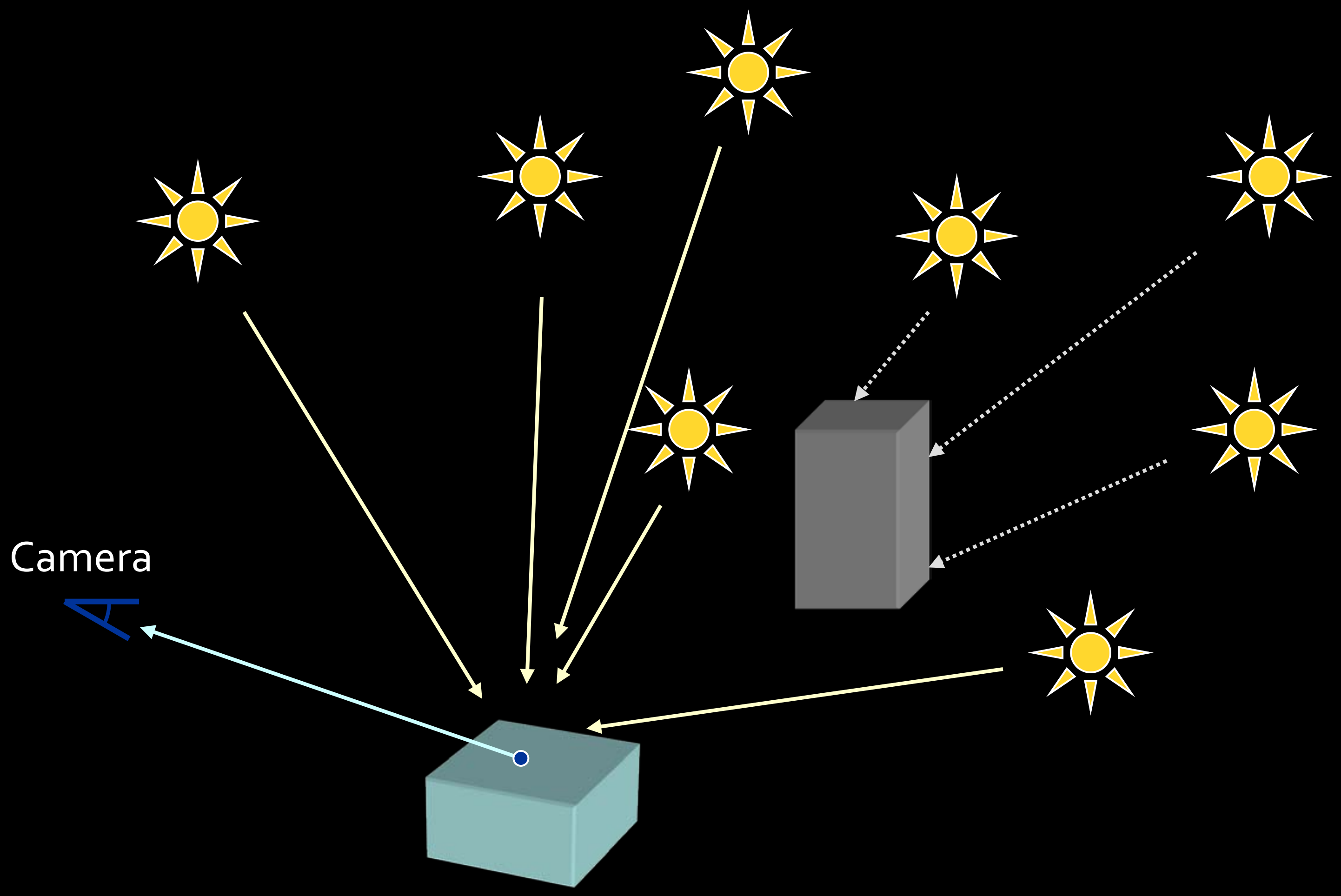
Lightcuts Problem



Lightcuts Problem

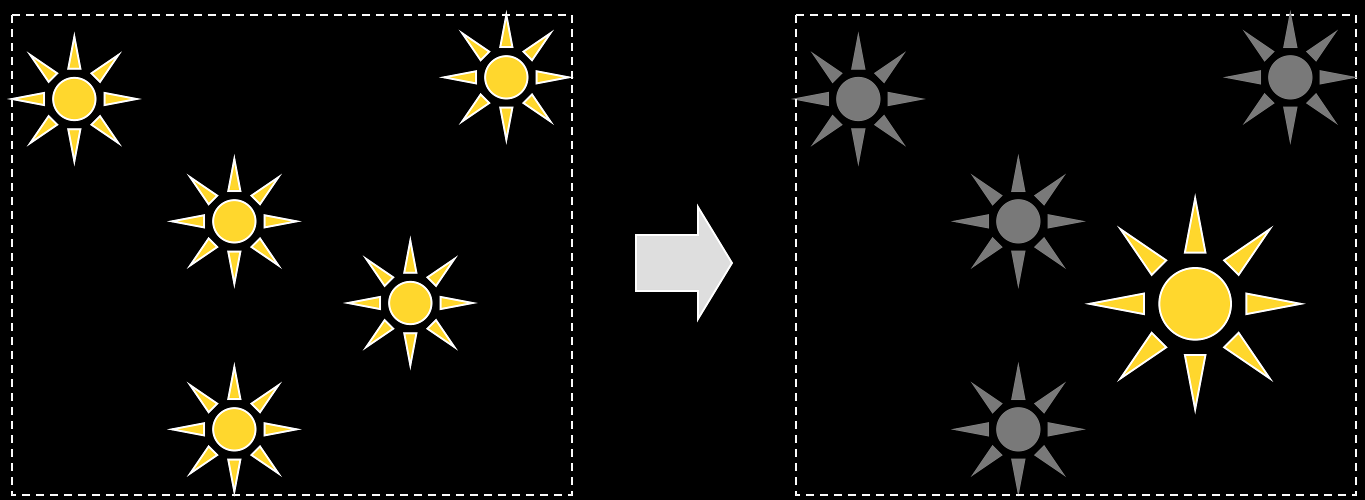


Lightcuts Problem



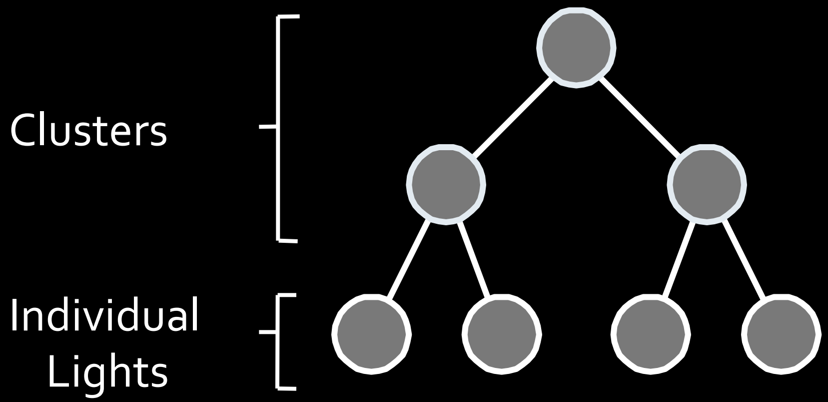
Key Concepts

- Light Cluster
  - Approximate many lights by a single brighter light (the representative light)



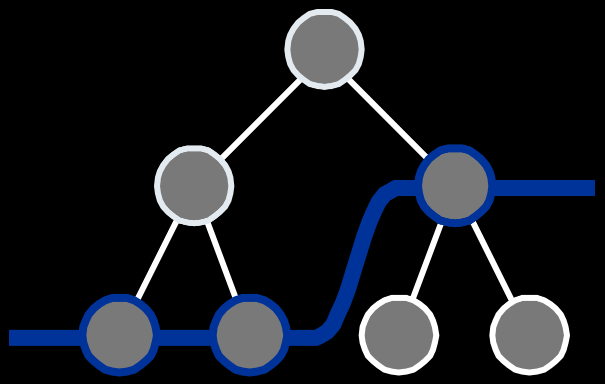
Key Concepts

- Light Cluster
- Light Tree
  - Binary tree of lights and clusters

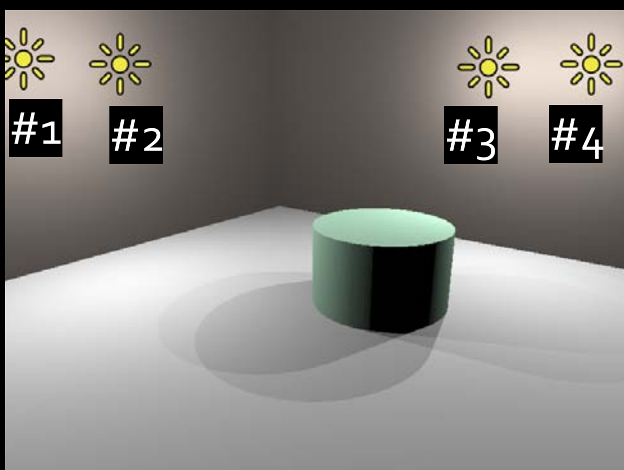


Key Concepts

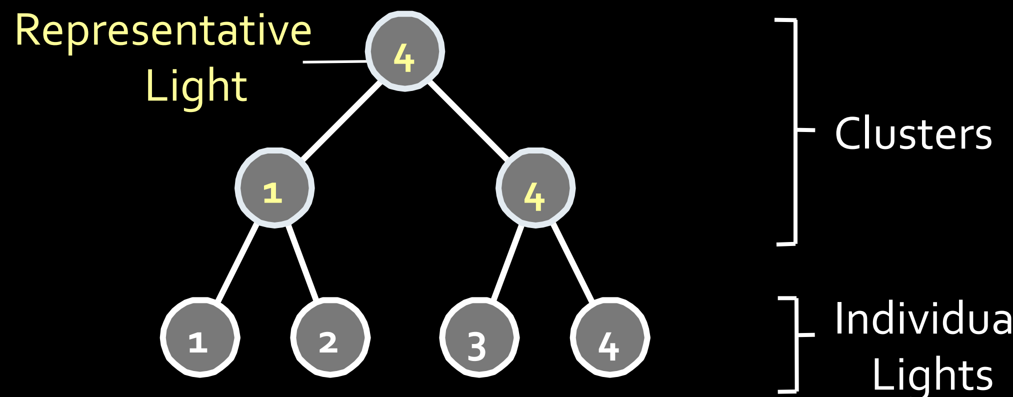
- Light Cluster
- Light Tree
- A Cut
  - A set of nodes that partitions the lights into clusters



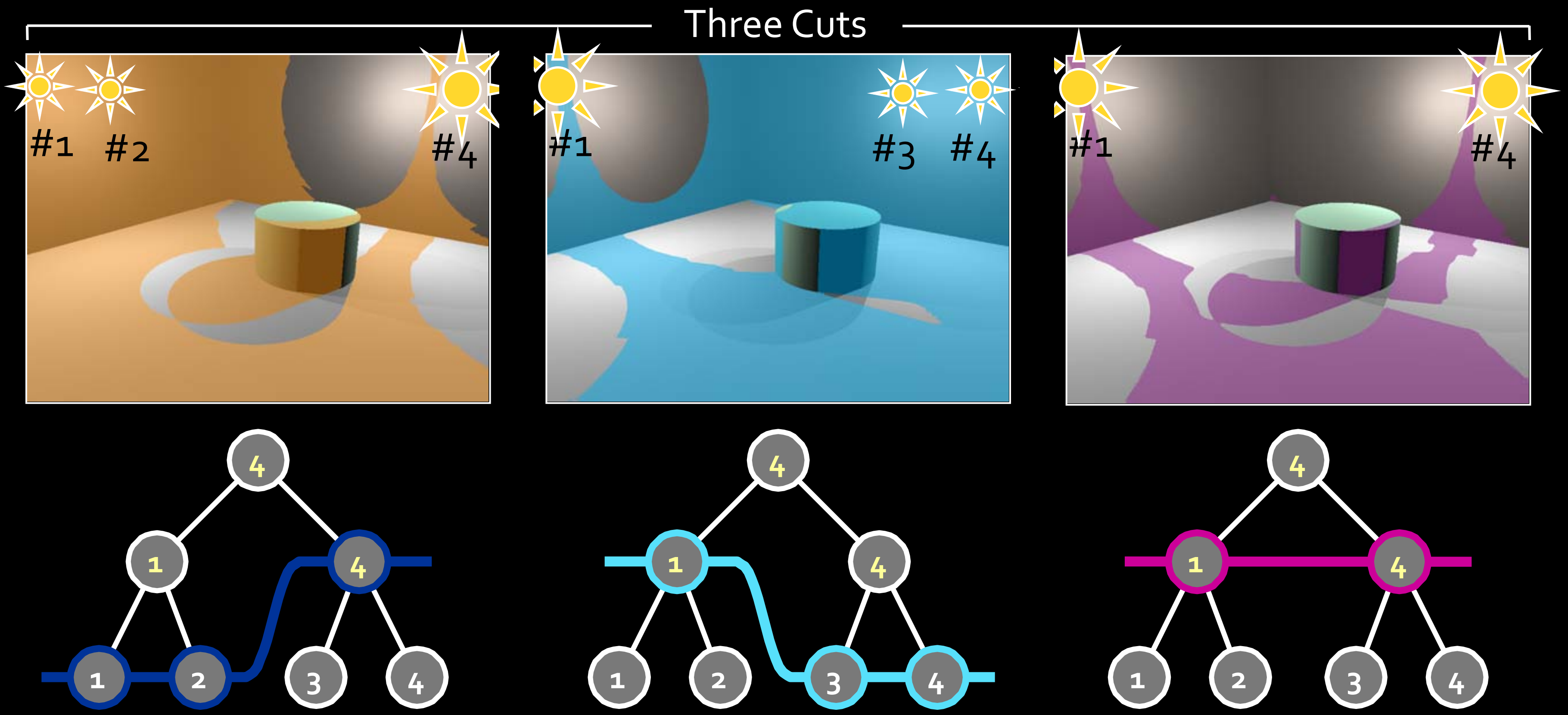
Simple Example



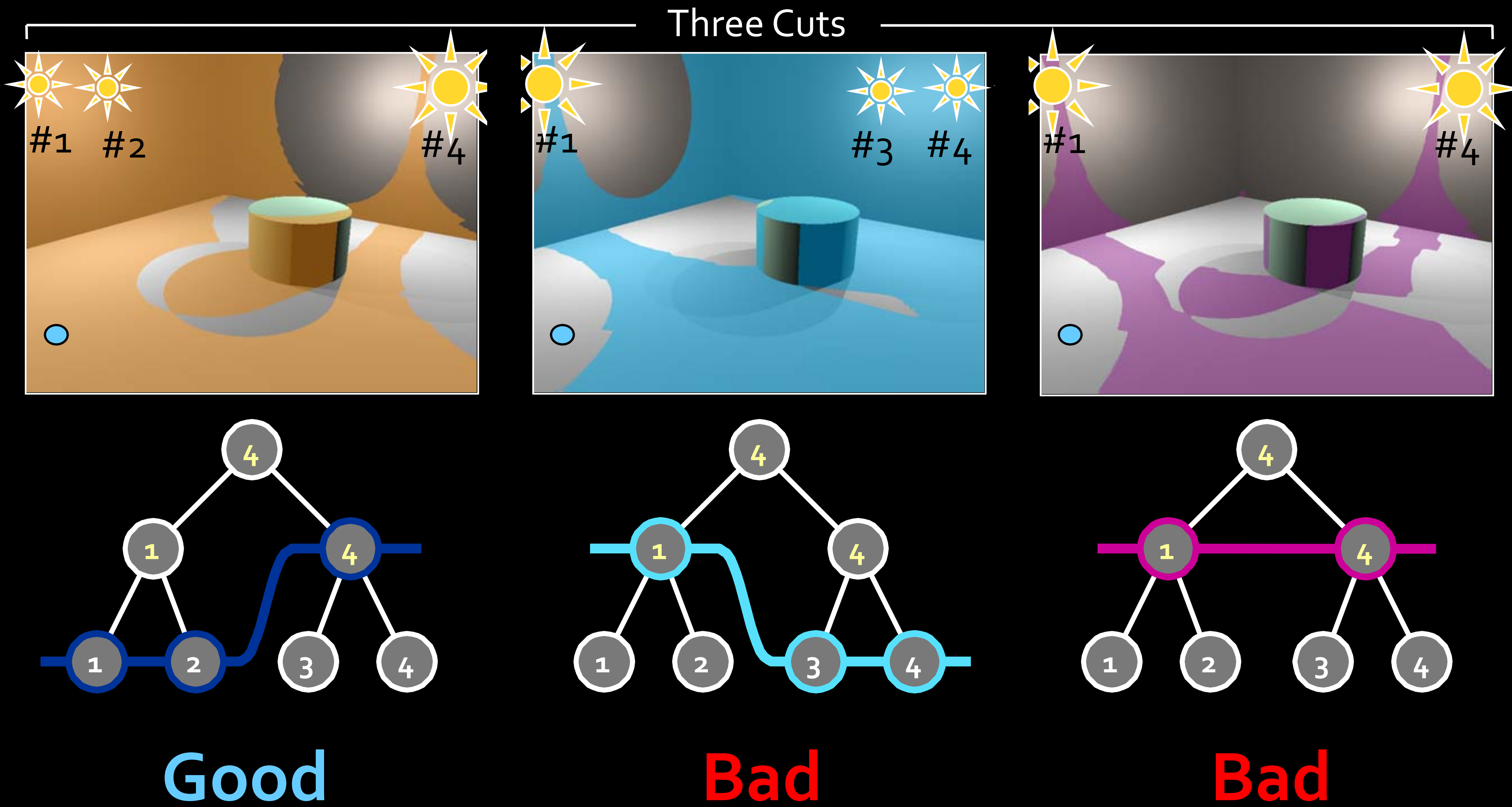
Light Tree



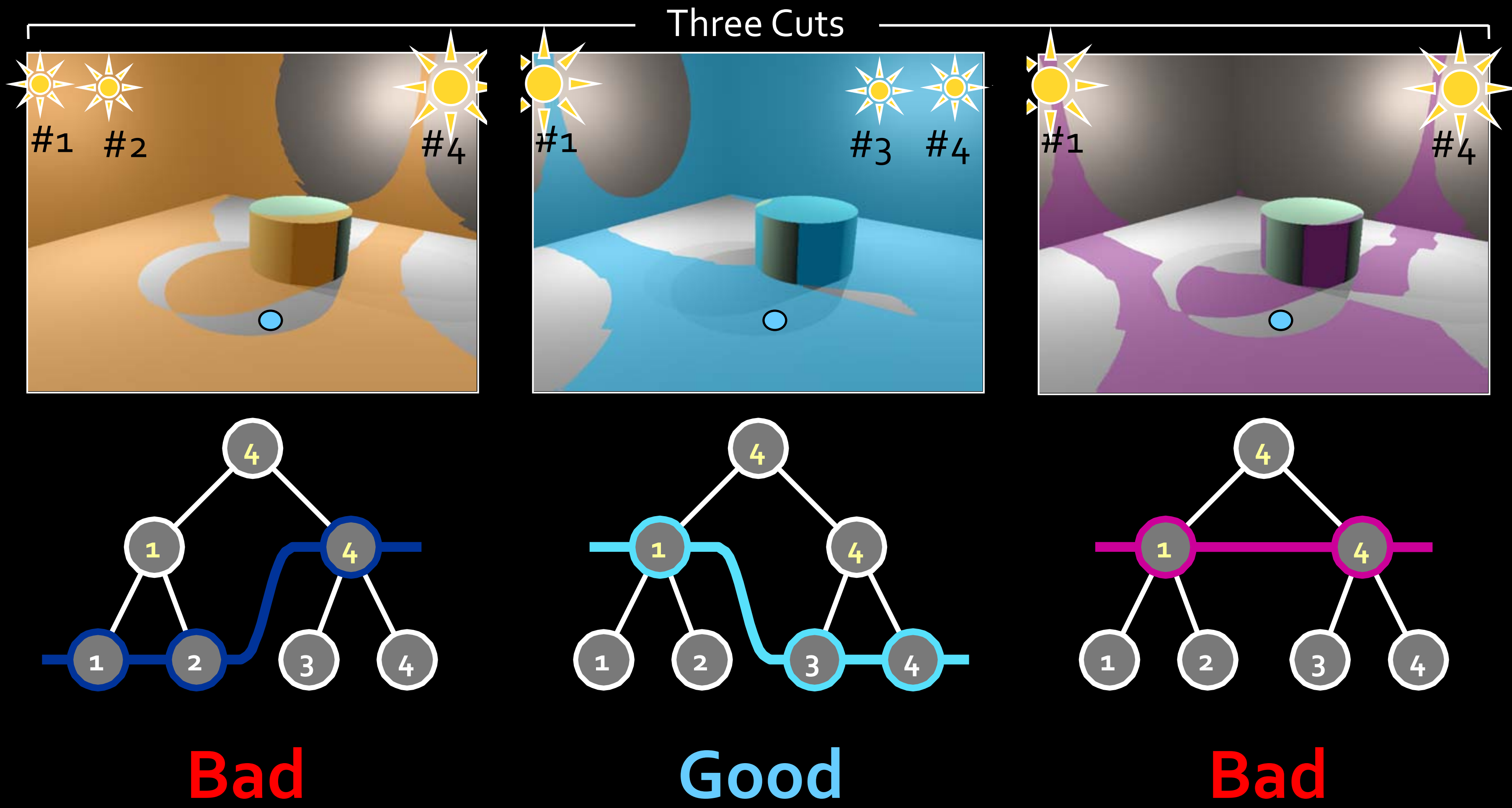
Three Example Cuts



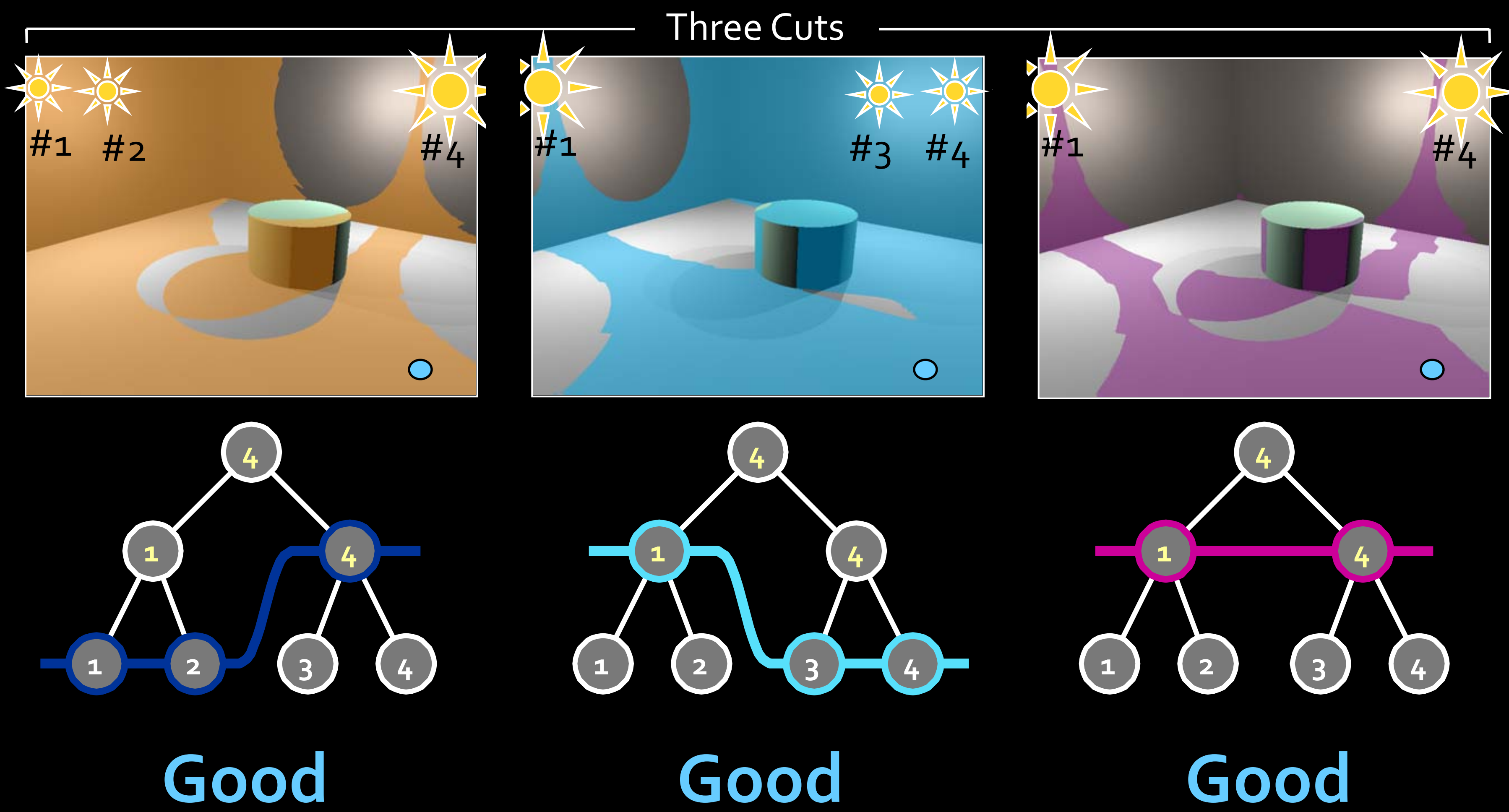
Three Example Cuts



Three Example Cuts



Three Example Cuts

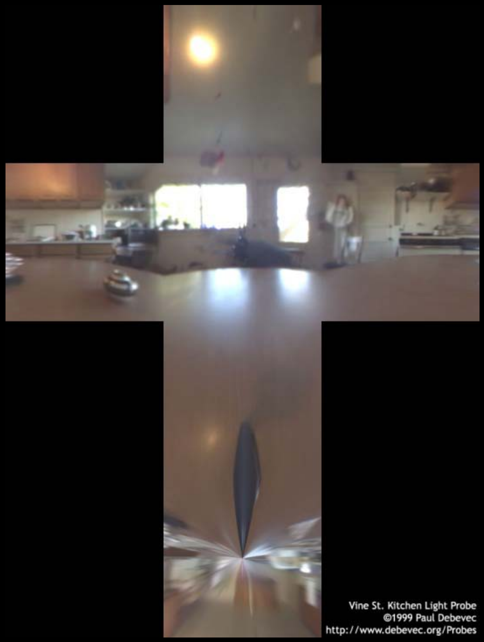


Algorithm Overview

- Pre-process
  - Convert illumination to point lights
  - Build light tree
- For each eye ray
  - Choose a cut to approximate the illumination

Convert Illumination

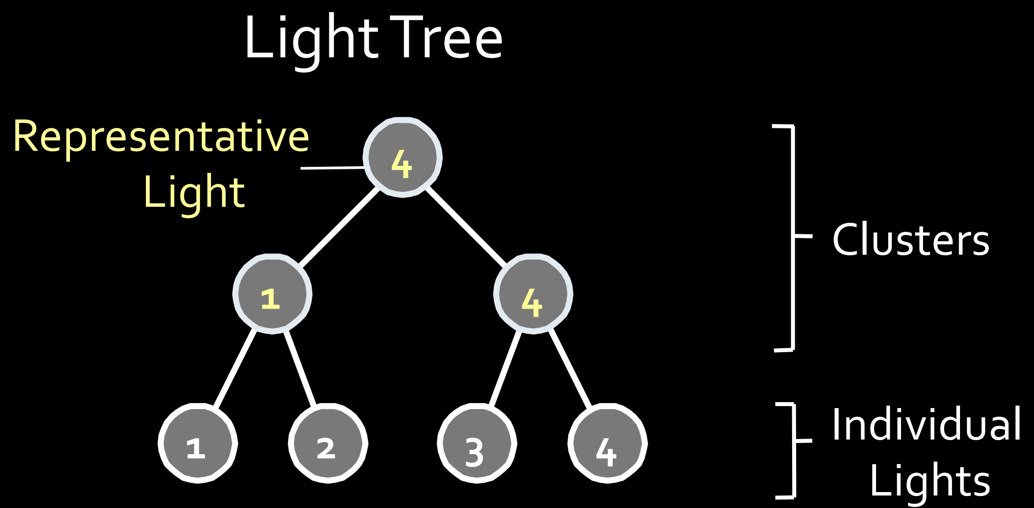
- **HDR environment map**
  – Importance sampling



- **Indirect Illumination**
  – Convert indirect to direct illumination using Instant Radiosity [Keller 97]
    - Caveats: no caustics, clamping, etc.
  – More lights = more indirect detail

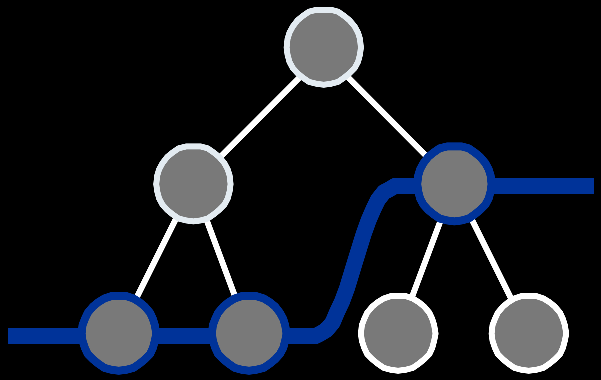
Build light tree

- Cluster spatially close lights with similar orientation



Choose a cut

- Approximate illumination with a bounded error
- Different cut for each pixel



Illumination Equation

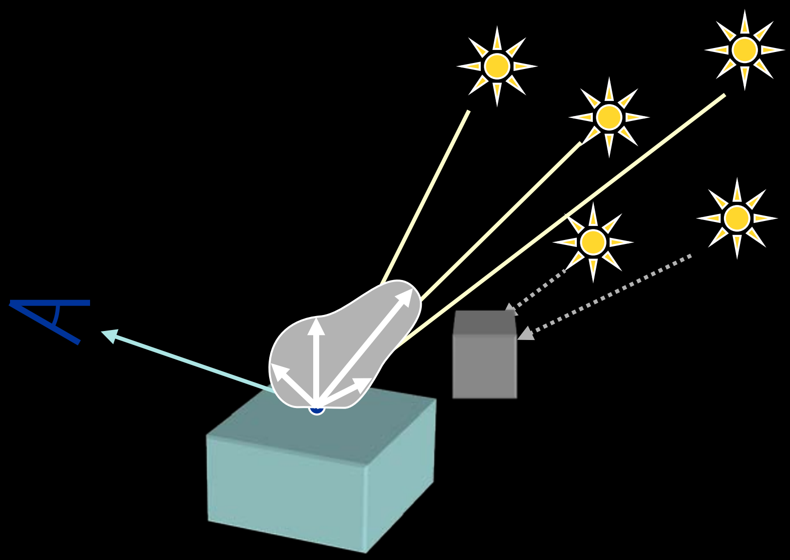
$$\text{result} = \sum_{\text{lights}} M_i \; G_i \; V_i \; I_i$$

Material term

Geometric term

Visibility term

Light intensity

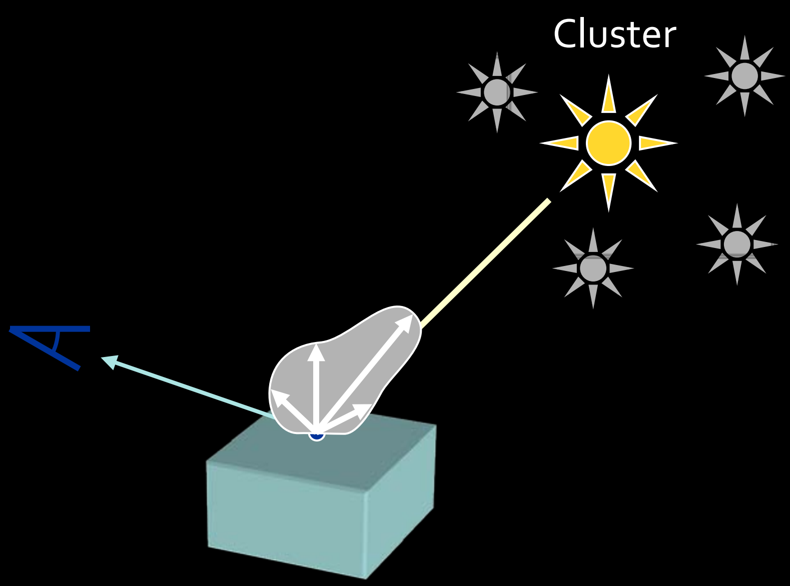


Cluster Approximation

Sum pre-computed during light tree construction

$$\text{result} \approx M_j \; G_j \; V_j \sum_{\text{lights}} I_i$$

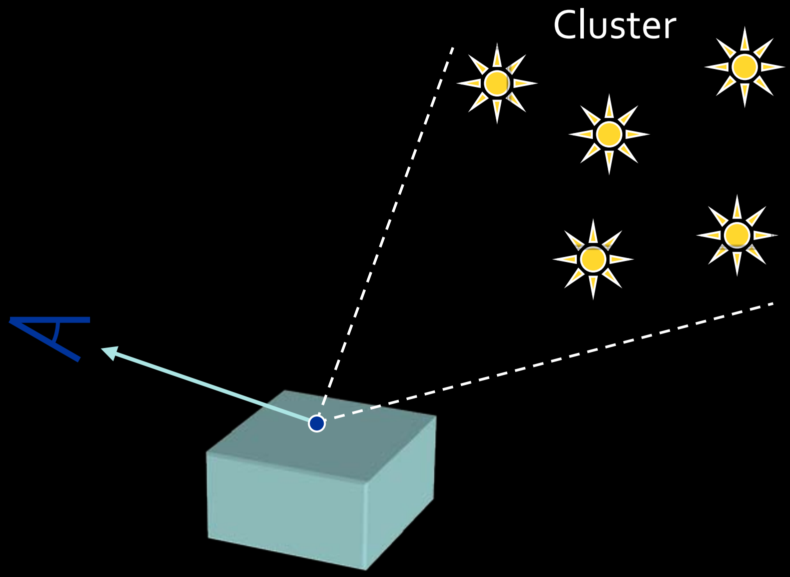
j is the representative light



Cluster Error Bound

$$\text{error} \leq M_{\text{ub}}\, G_{\text{ub}}\, V_{\text{ub}} \sum_{\text{lights}} I_{\text{i}}$$

- Bound each term
  - Visibility <= 1 (trivial)
  - Intensity is known
  - Bound material and geometric terms using cluster bounding volume



ub = upper bound

Perceptual Metric

- Weber's Law
  - Contrast visibility threshold is fixed percentage of signal
  - Used 2% in our results

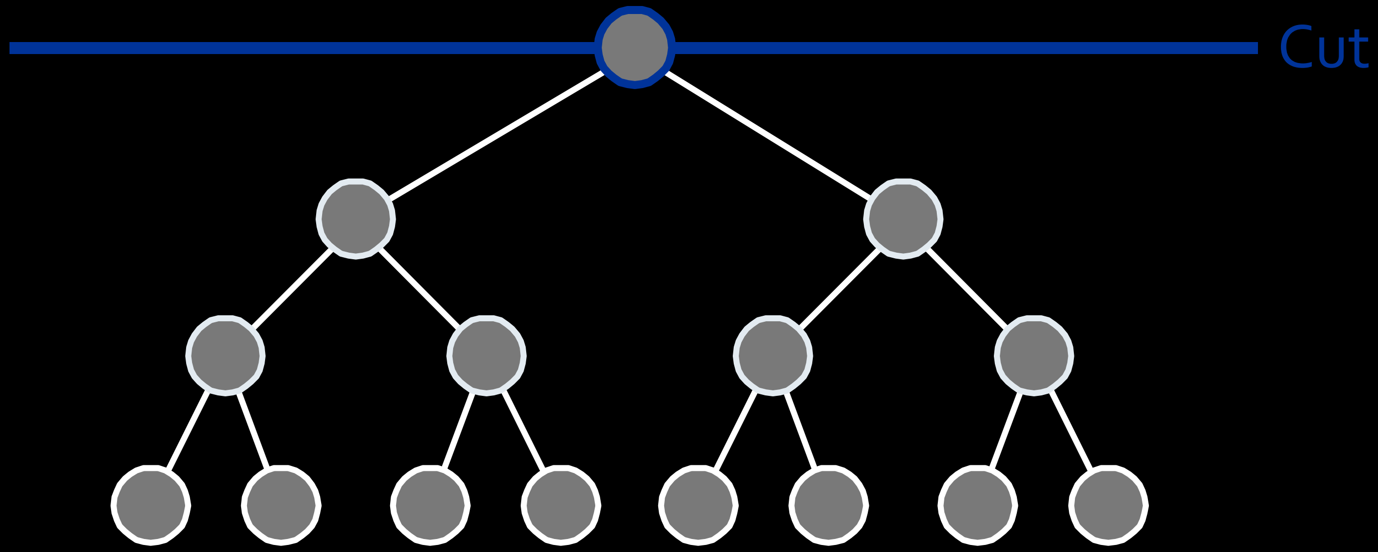
- Ensure each cluster's error < visibility threshold
  - Transitions will not be visible
  - Used to select cut

Perceptual Metric

- **Problem:**
  - We don't know the illumination so we don't know the threshold either
    - (because threshold = 2% illumination)
- **Solution:**
  - As we traverse the tree, gradually improve the illumination estimate.
  - Stop the traversal if the error bound for all cut nodes is below threshold.

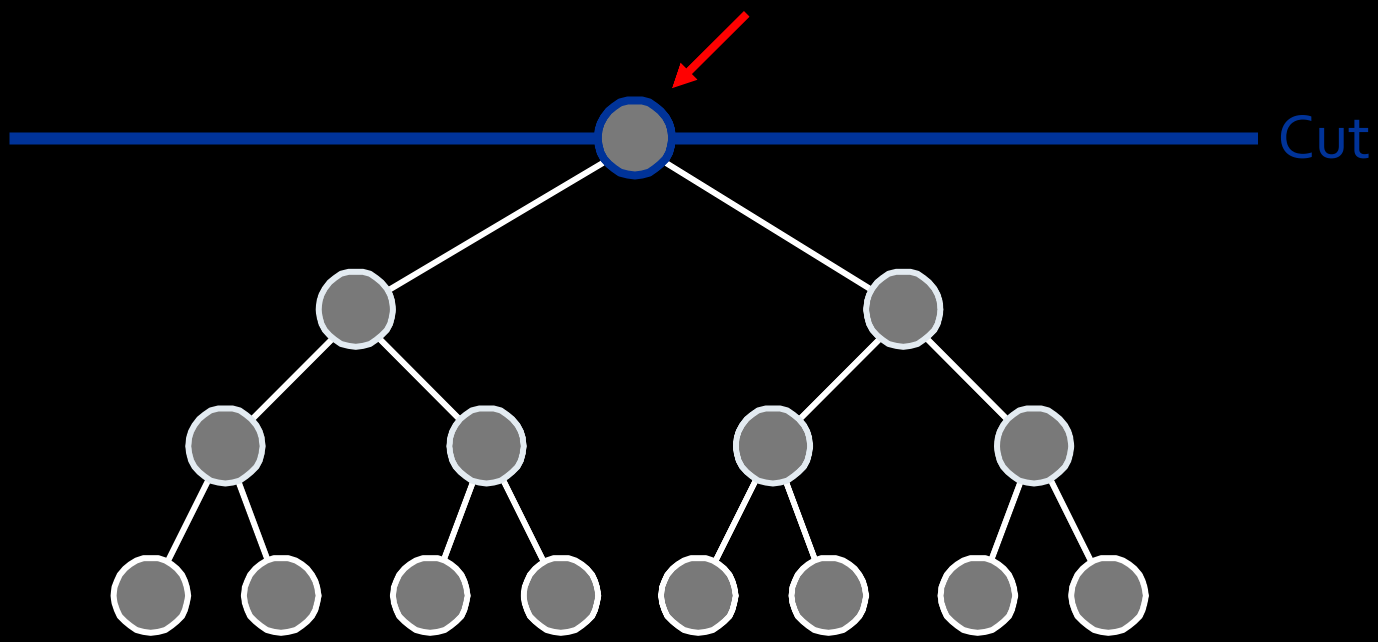
Cut Selection Algorithm

- Start with coarse cut (eg, root node)



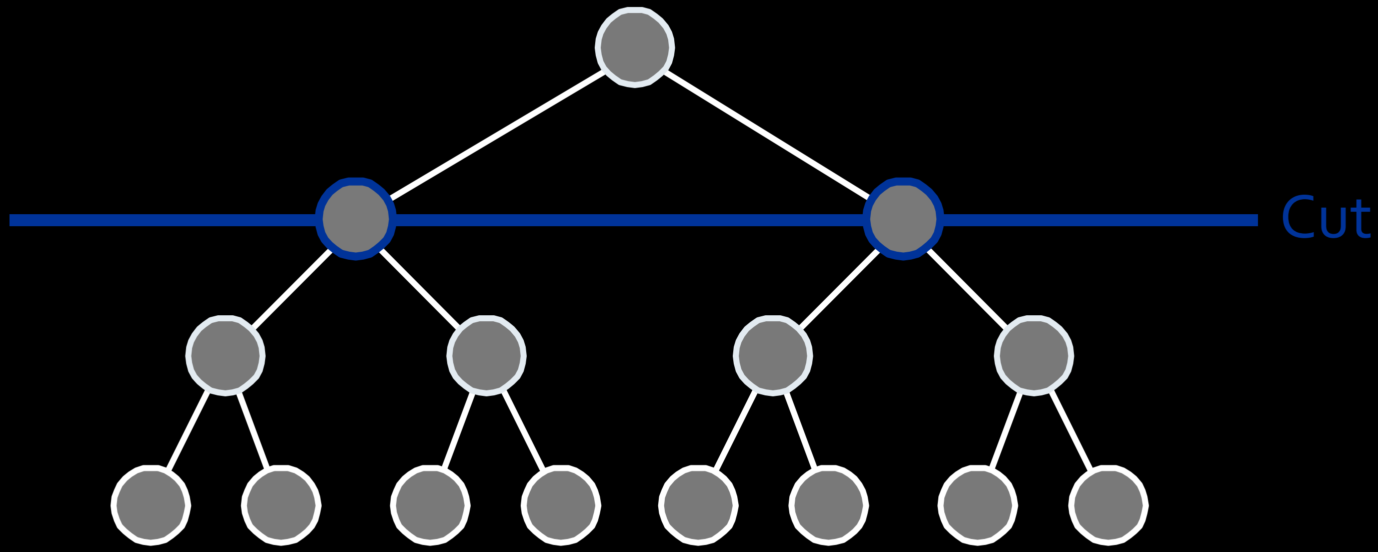
Cut Selection Algorithm

- Select cluster with largest error bound

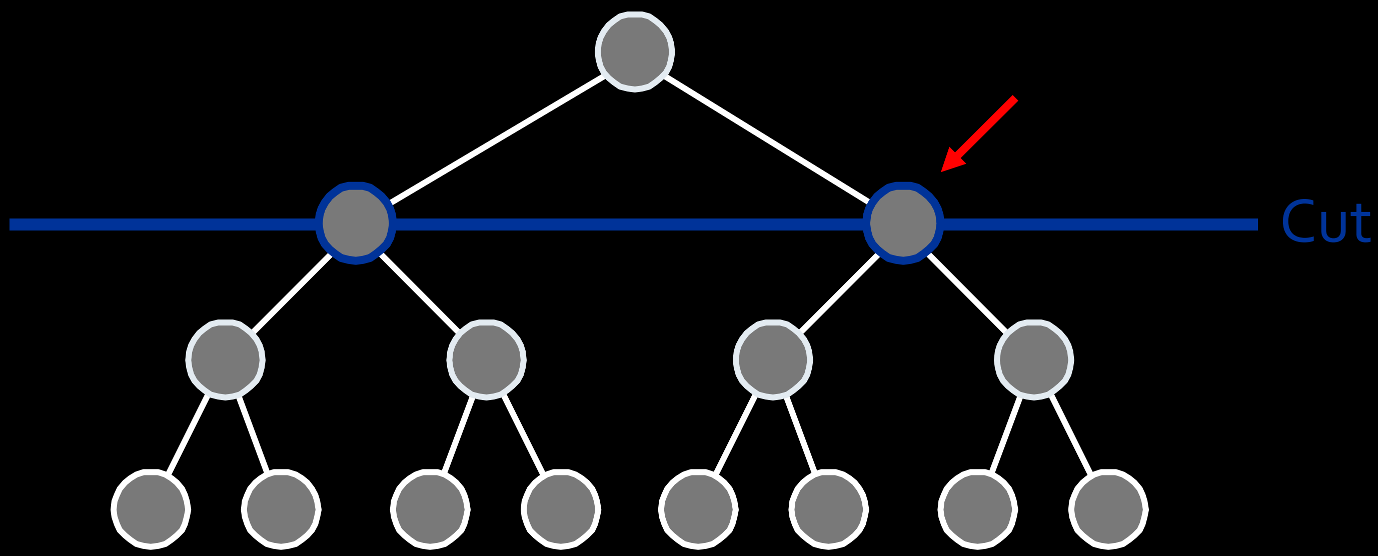


Cut Selection Algorithm

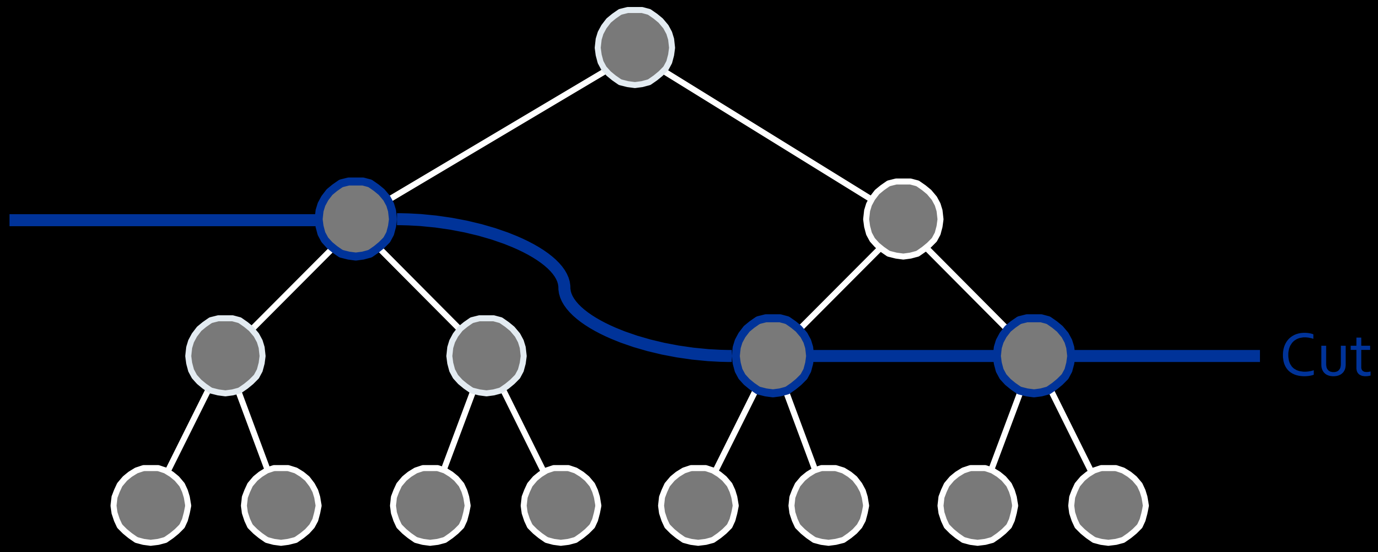
- Refine if error bound > 2% of total



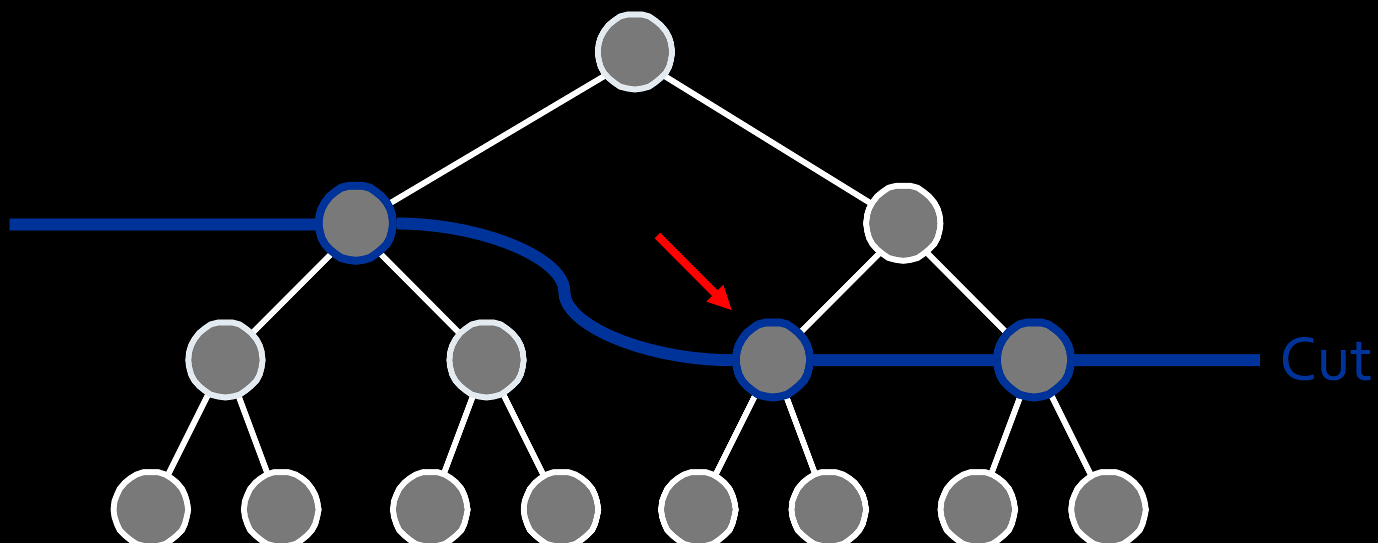
Cut Selection Algorithm



Cut Selection Algorithm

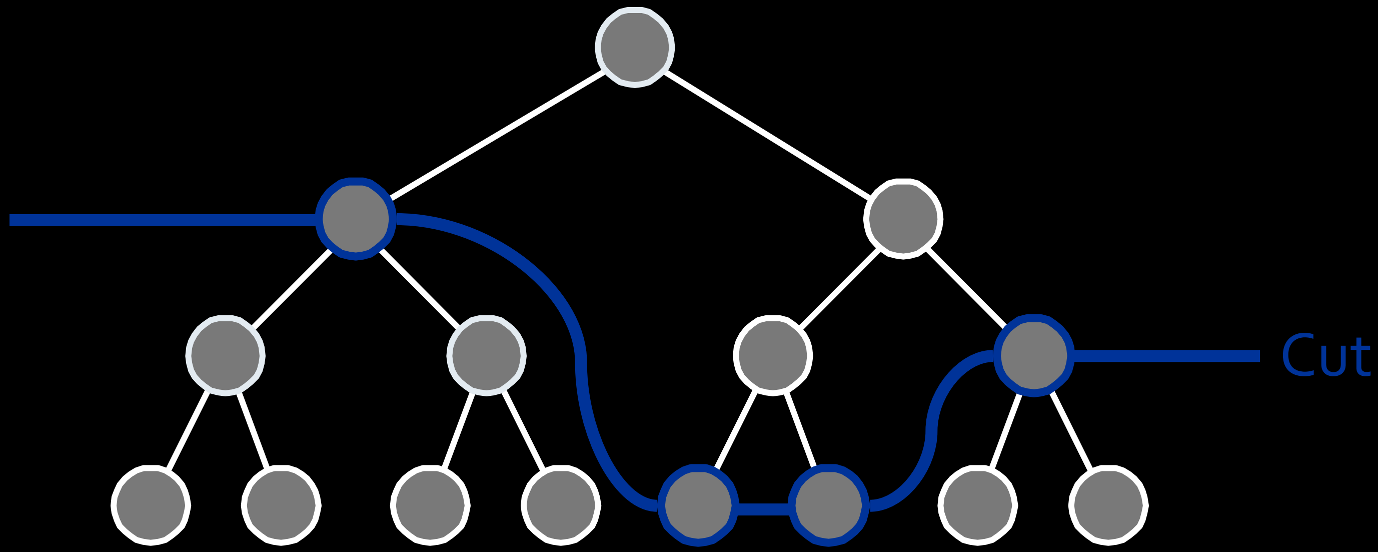


Cut Selection Algorithm



Cut Selection Algorithm

- Repeat until cut obeys 2% threshold



Kitchen, 388K polygons, 4608 lights (72 area sources)

Combined Illumination

Lightcuts 128s

4 608 Lights
(Area lights only)

Avg. 259 shadow rays / pixel

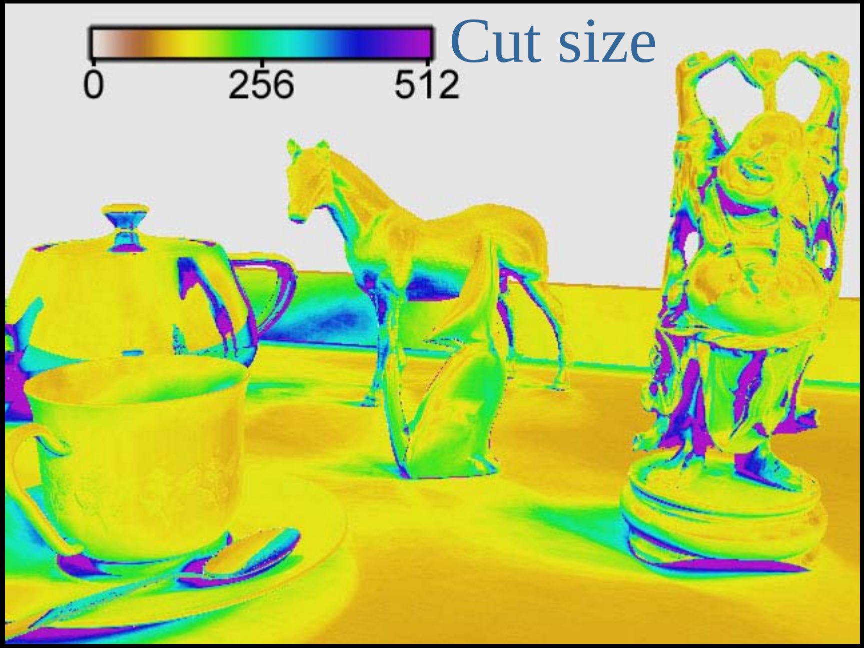
Lightcuts 290s

59 672 Lights
(Area + Sun/sky + Indirect)

Avg. 478 shadow rays / pixel
(only 54 to area lights)


Lightcuts


Reference

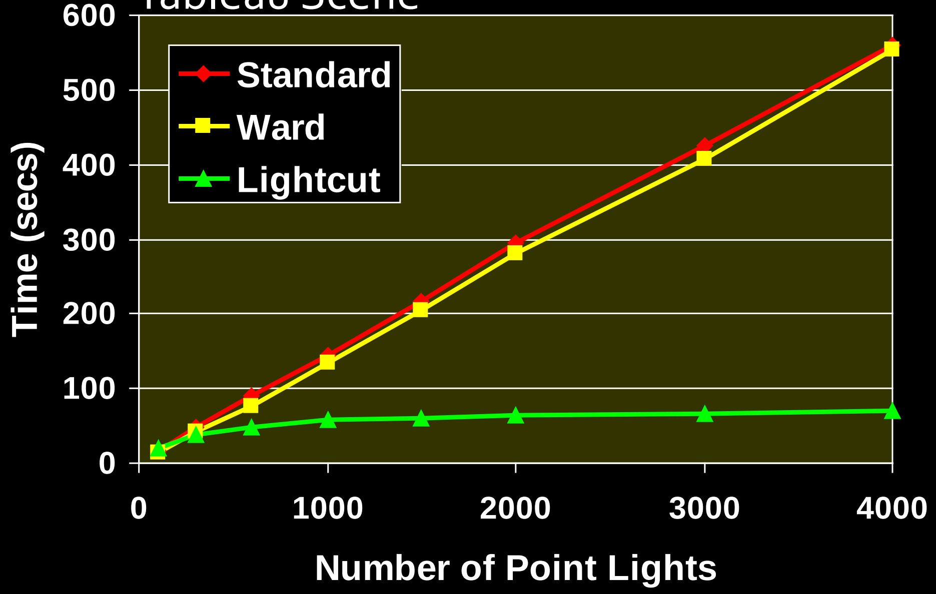

Cut size
0
256
512


Error x 16

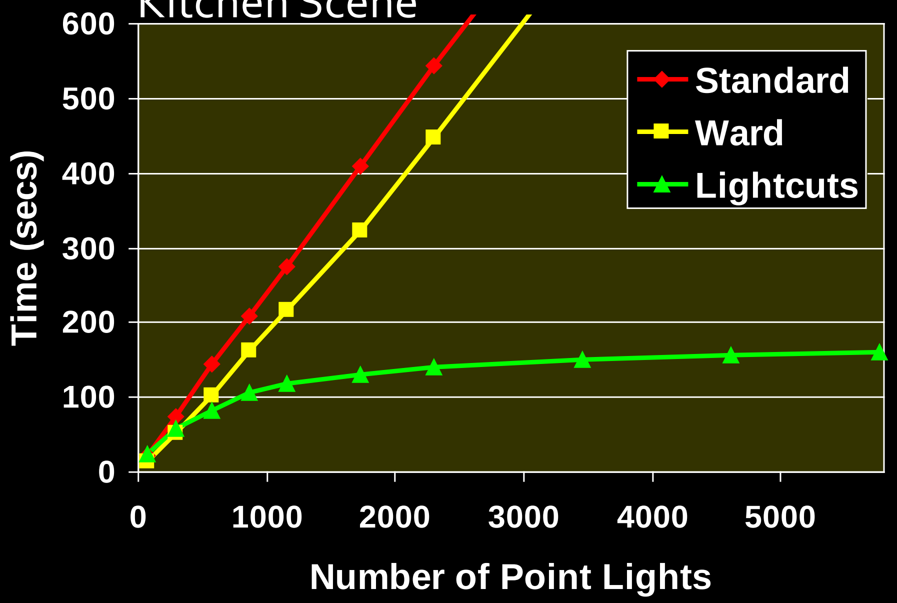
Scalable

- Scalable solution for many point lights
  - Thousands to millions
  - Sub-linear cost

Tableau Scene



Kitchen Scene



Why does it work so well?

- Data-driven stratification & importance sampling
- Stratification
  - Clustering of similar lights in the light tree
- Importance sampling
  - Subdividing clusters with high contribution

Main issue

Bigscreen Model

Cut Size (False Color)

- Problem: Large cuts in dark areas

Lightcuts Recap

- **Key ingredients**
  - Upper bound on error
  - Refinement of the highest-error nodes first

Multidimensional Lightcuts

Multidimensional Lightcuts

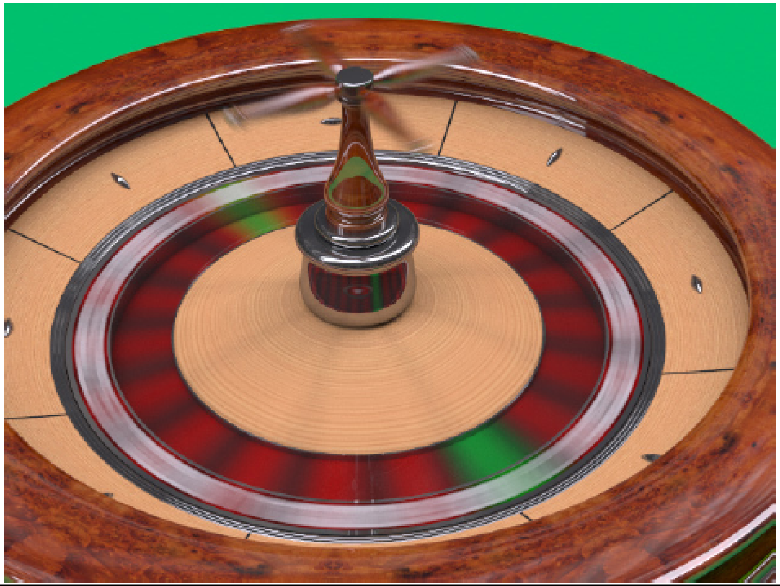
Bruce Walter Adam Arbree Kavita Bala Donald P. Greenberg
Cornell University*

Abstract

Multidimensional lightcuts is a new scalable method for efficiently rendering rich visual effects such as motion blur, participating media, depth of field, and spatial anti-aliasing in complex scenes. It introduces a flexible, general rendering framework that unifies the handling of such effects by discretizing the integrals into large sets of gather and light points and adaptively approximating the sum of all possible gather-light pair interactions.

We create an implicit hierarchy, the product graph, over the gather-light pairs to rapidly and accurately approximate the contribution from hundreds of millions of pairs per pixel while only evaluating a tiny fraction (e.g., 200–1,000). We build upon the techniques of the prior Lightcuts method for complex illumination at a point, however, by considering the complete pixel integrals, we achieve much greater efficiency and scalability.

Our example results demonstrate efficient handling of volume scat-



http://www.graphics.cornell.edu/~bjw/papers.html

Problem

- Simulate complex, expensive phenomena
  - Complex illumination
  - Anti-aliasing
  - Motion blur
  - Participating media
  - Depth of field

$$\text{Pixel} = \int_{\text{Time}} \int_{\text{Pixel Area}} \int_{\text{Lights}} L(\mathbf{x},\omega)\ldots$$

Problem

- Simulate complex, expensive phenomena
  - Complex illumination
  - Anti-aliasing
  - Motion blur
  - Participating media
  - Depth of field

$$\text{Pixel} = \int_{\text{Volume}} \int_{\text{Time}} \int_{\text{Pixel Area}} \int_{\text{Lights}} L(\mathbf{x}, \omega) \ldots$$

Problem

- Simulate complex, expensive phenomena
  - Complex illumination
  - Anti-aliasing
  - Motion blur
  - Participating media
  - Depth of field

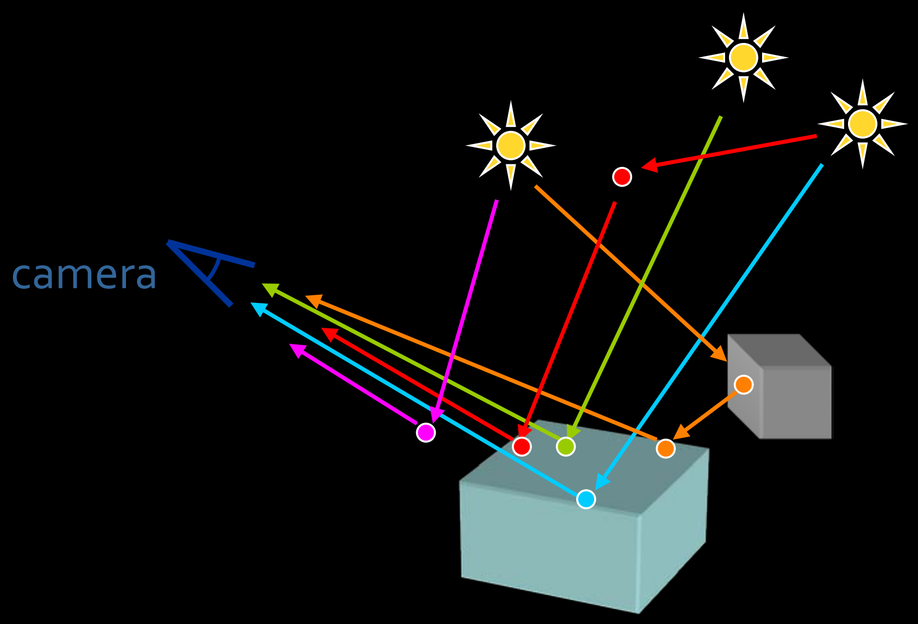
$$\text{Pixel} = \int\limits_{\text{Aperture}} \int\limits_{\text{Volume}} \int\limits_{\text{Time}} \int\limits_{\text{Pixel Area}} \int\limits_{\text{Lights}} L(\mathbf{x},\omega)\ldots$$

Problem

- Complex integrals over multiple dimensions

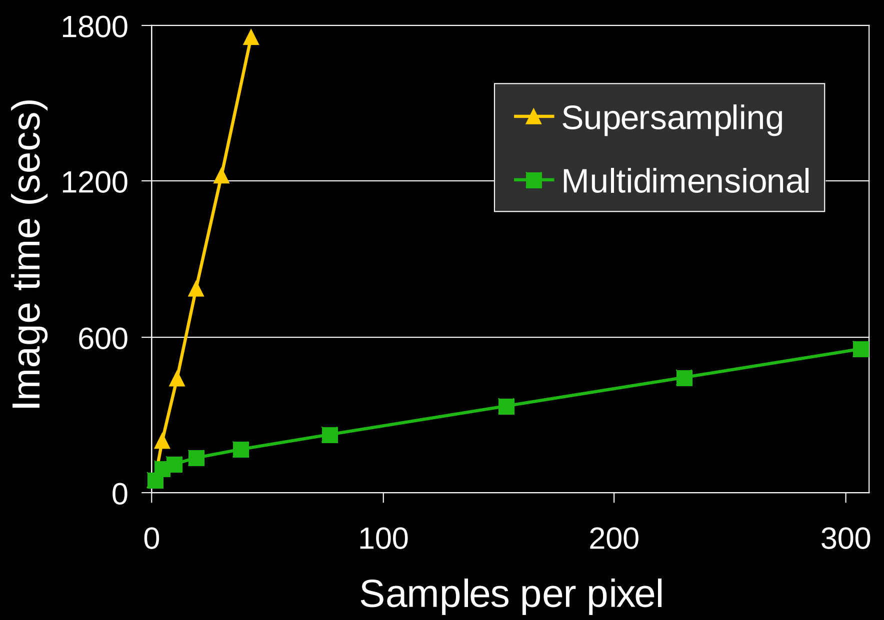
$$\text{Pixel} = \int_{\text{Aperture}} \int_{\text{Volume}} \int_{\text{Time}} \int_{\substack{\text{Pixel}\\\text{Area}}} \int_{\text{Lights}} L(\mathbf{x},\omega)\ldots$$

  - Requires many samples



Multidimensional Lightcuts

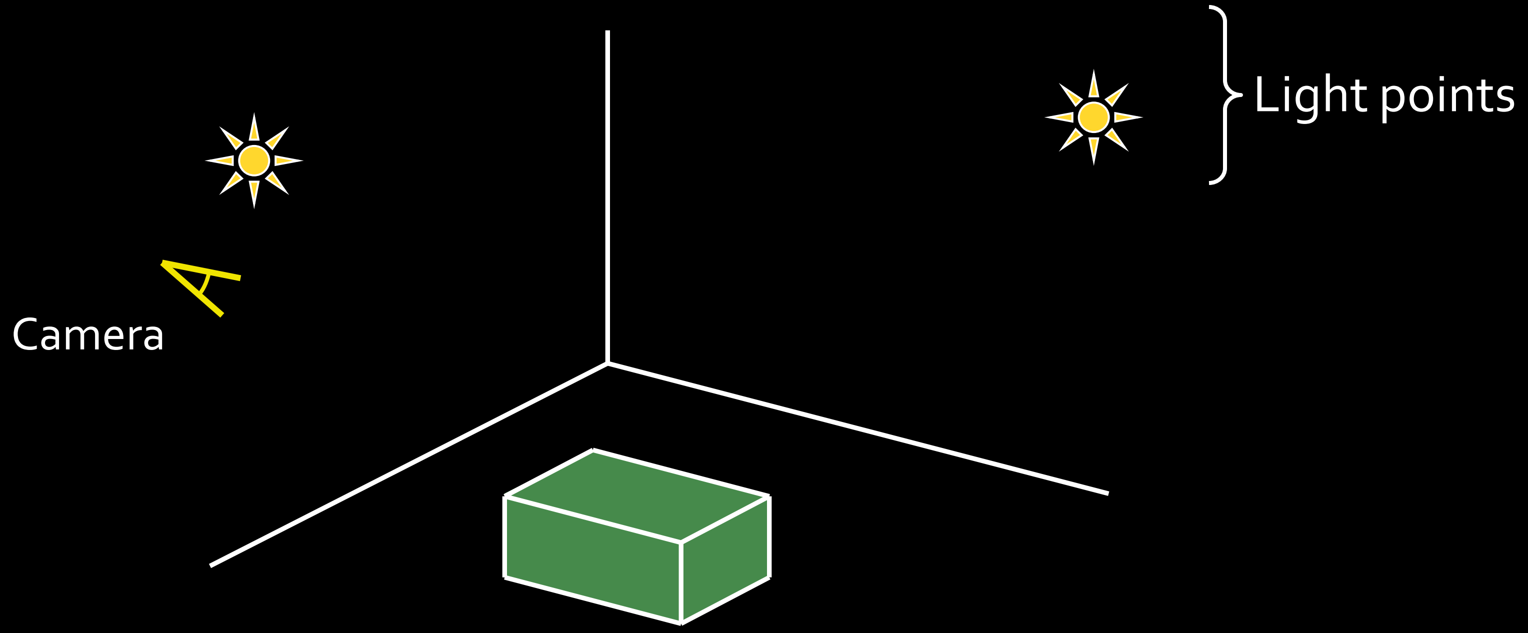
- Solves all integrals simultaneously
- Accurate
- Scalable




Direct only (relative cost 1x)
Direct+Indirect (1.3x)
Direct+Indirect+Volume (1.8x)
Direct+Indirect+Volume+Motion (2.2x)

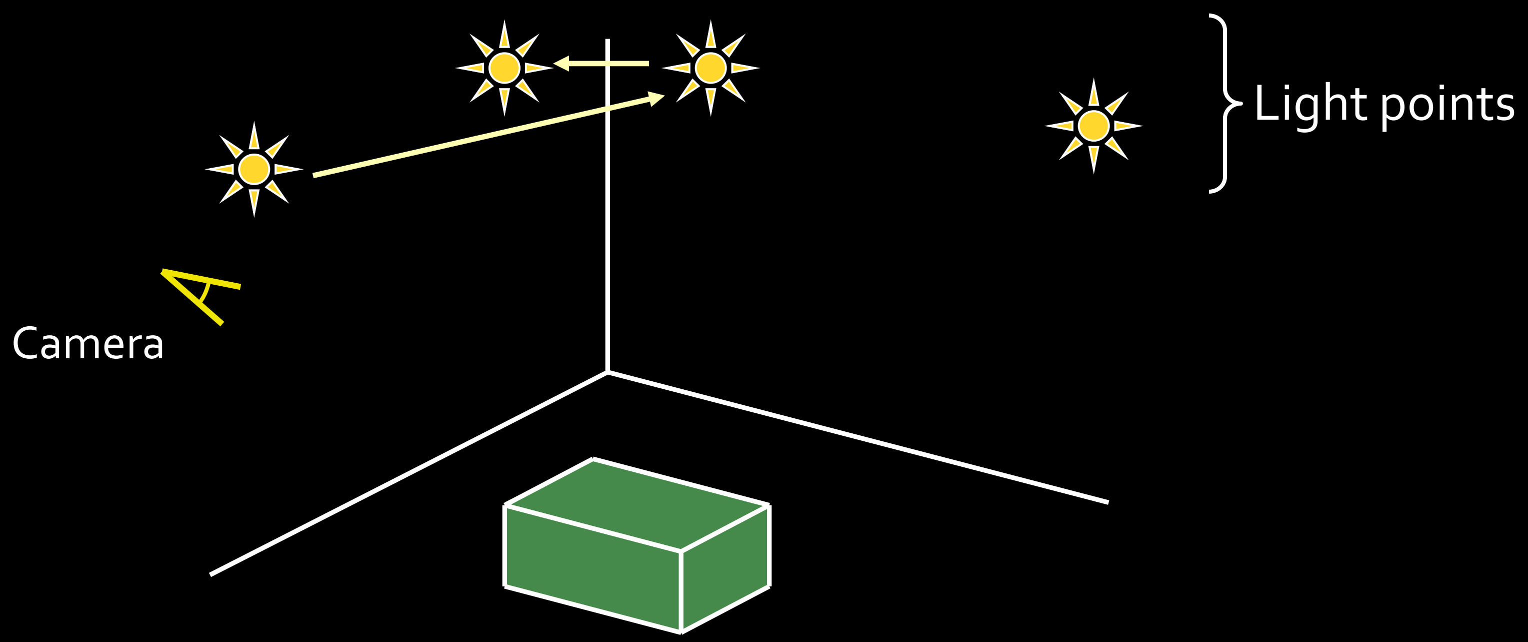
Point Sets

- Discretize full integral into 2 point sets
  - Light points ($\mathbf{L}$)
  - Gather points ($\mathbf{G}$)



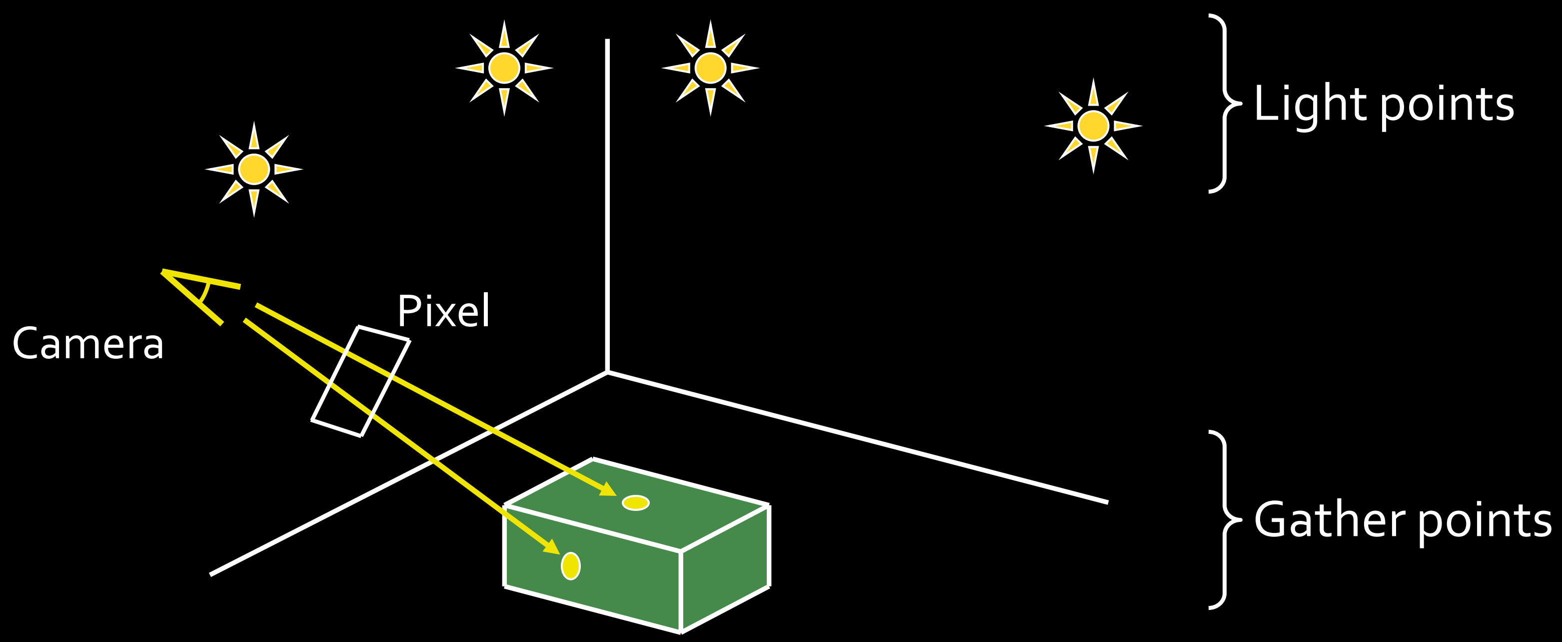
Point Sets

- Discretize full integral into 2 point sets
  - Light points ($\mathbf{L}$)
  - Gather points ($\mathbf{G}$)



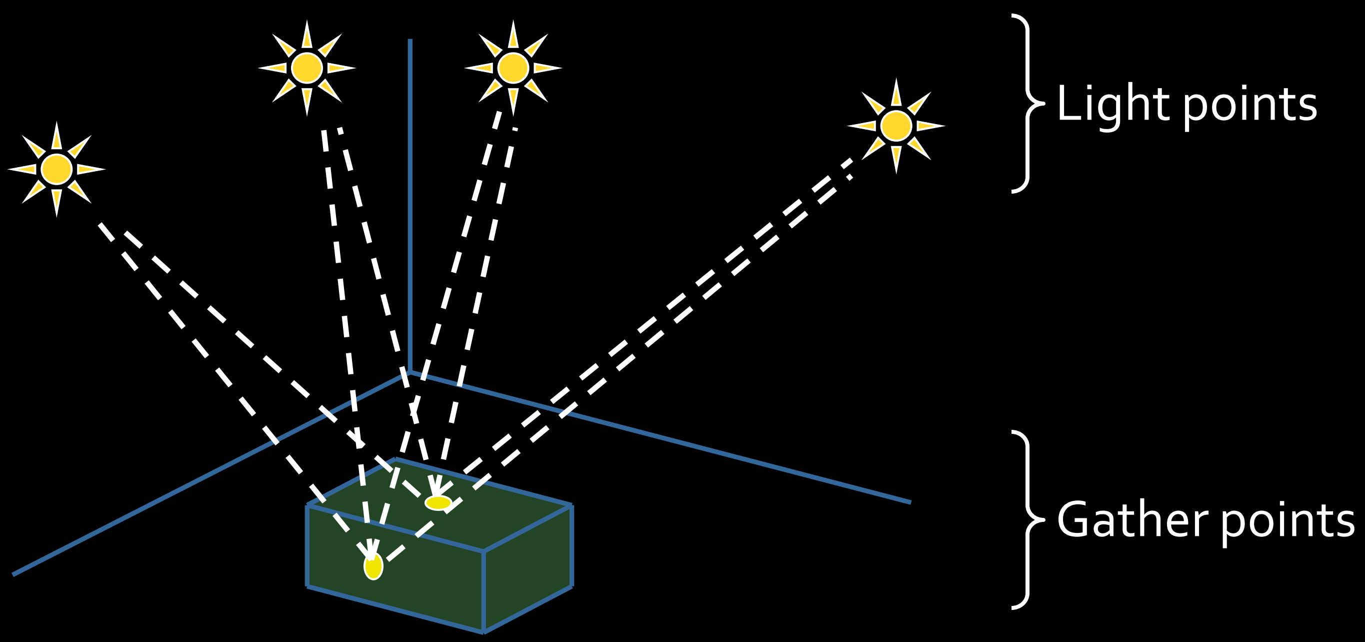
Point Sets

- Discretize full integral into 2 point sets
  - Light points ($\mathbf{L}$)
  - Gather points ($\mathbf{G}$)



Point Sets

- Discretize full integral into 2 point sets
  - Light points (**L**)
  - Gather points (**G**)



Discrete Equation

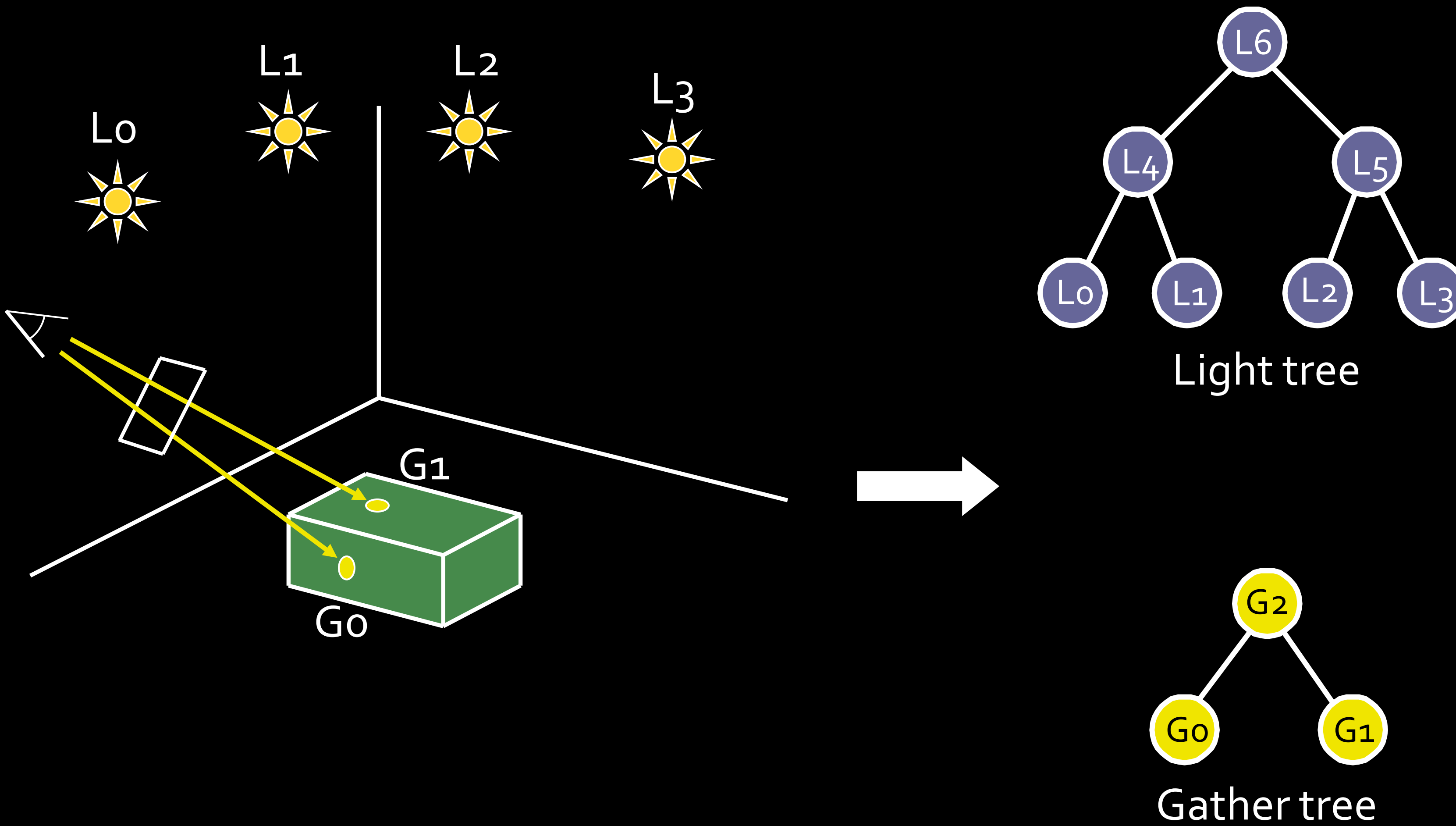
- Sum over all pairs of gather and light points
  - Can be billions of pairs per pixel

$$\text{Pixel} = \sum_{(j,i) \in \mathbf{G} \times \mathbf{L}} \underbrace{S_j}_{\text{Gather strength}} \; \underbrace{M_{ji}}_{\text{Material term}} \; \underbrace{G_{ji}}_{\text{Geometric term}} \; \underbrace{V_{ji}}_{\text{Visibility term}} \; \underbrace{I_i}_{\text{Light intensity}}$$

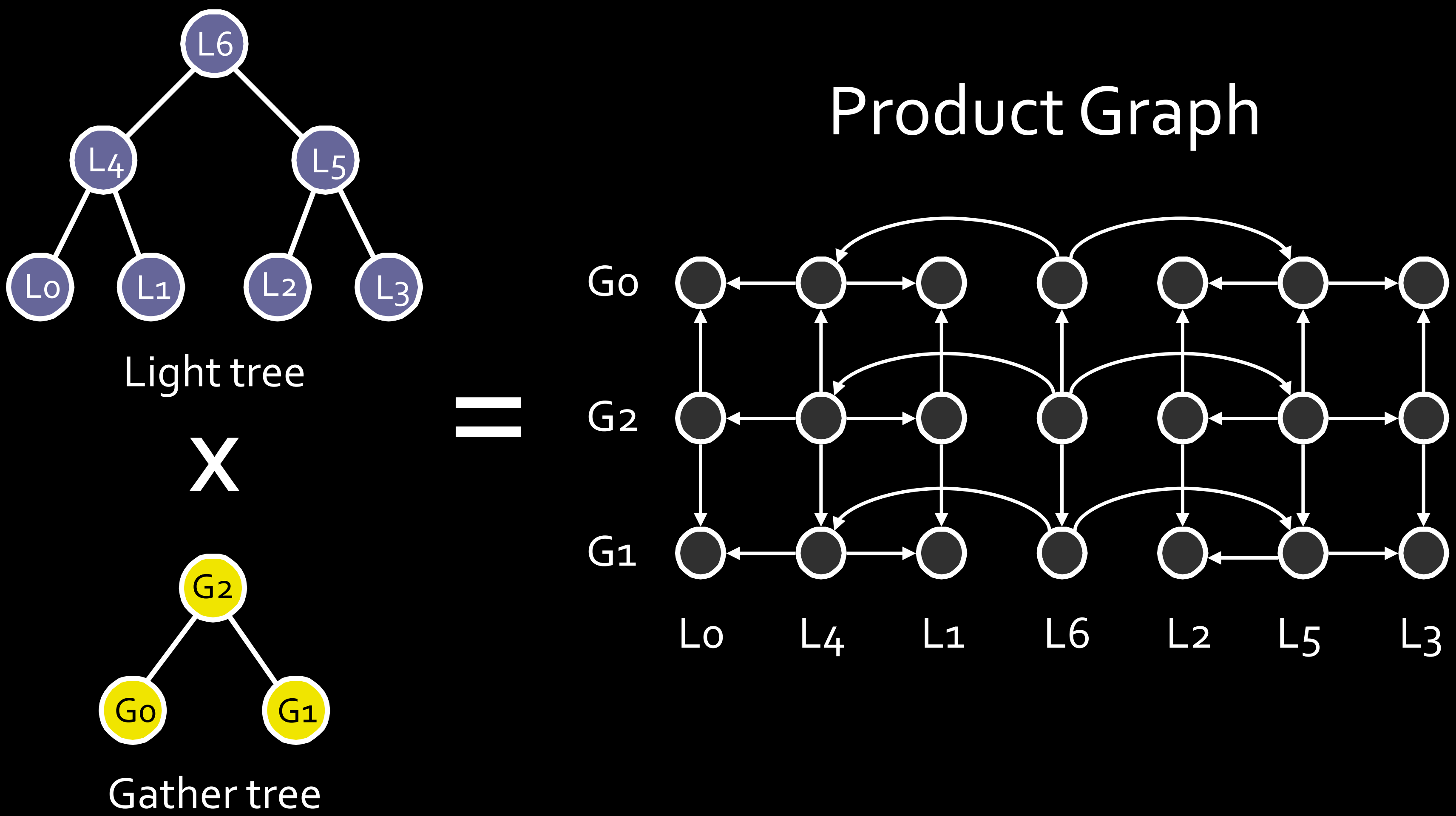
Product Graph

- Explicit hierarchy would be too expensive
  - Up to billions of pairs per pixel
- Use implicit hierarchy
  - Cartesian product of two trees (gather & light)

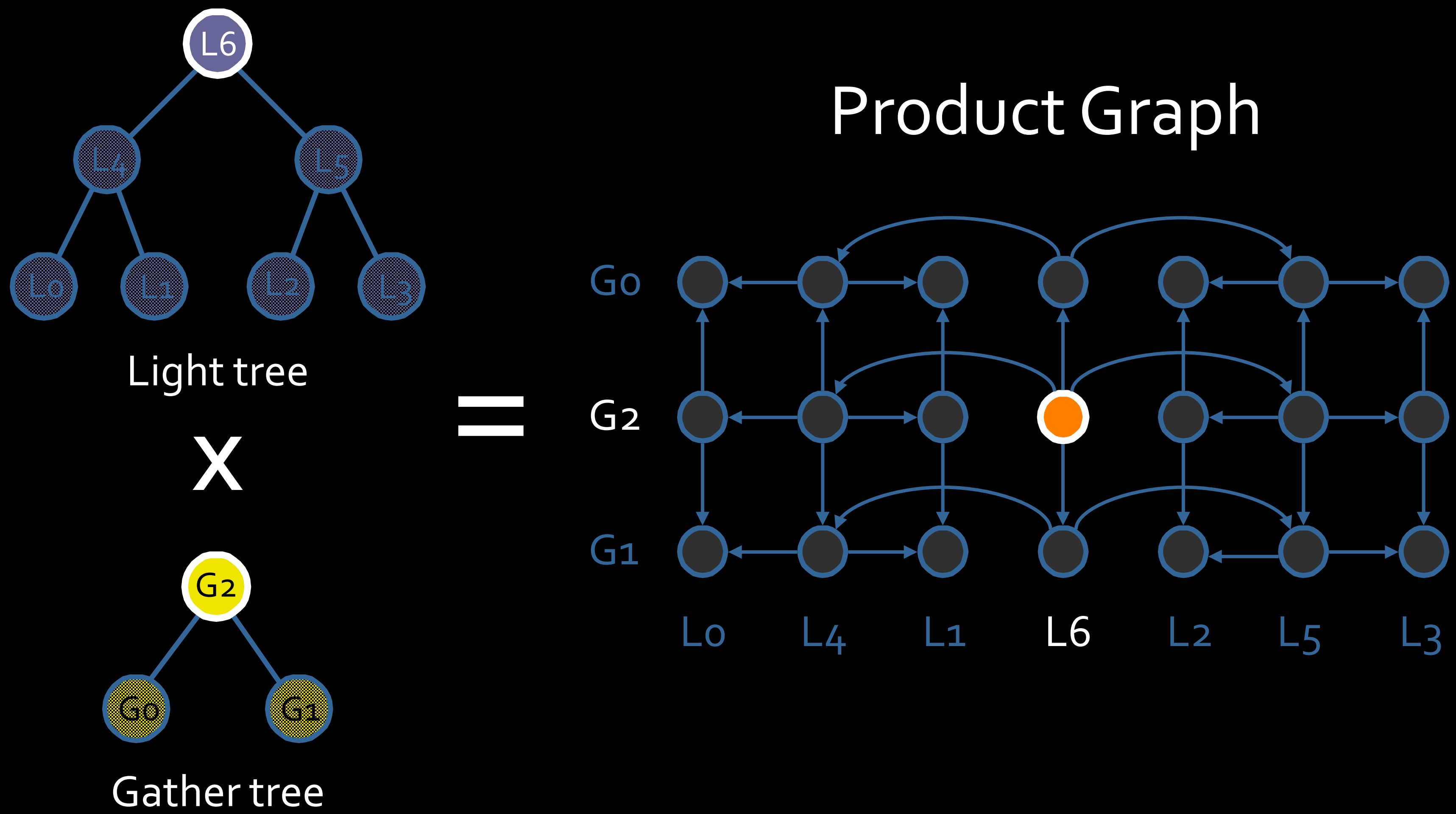
Product Graph



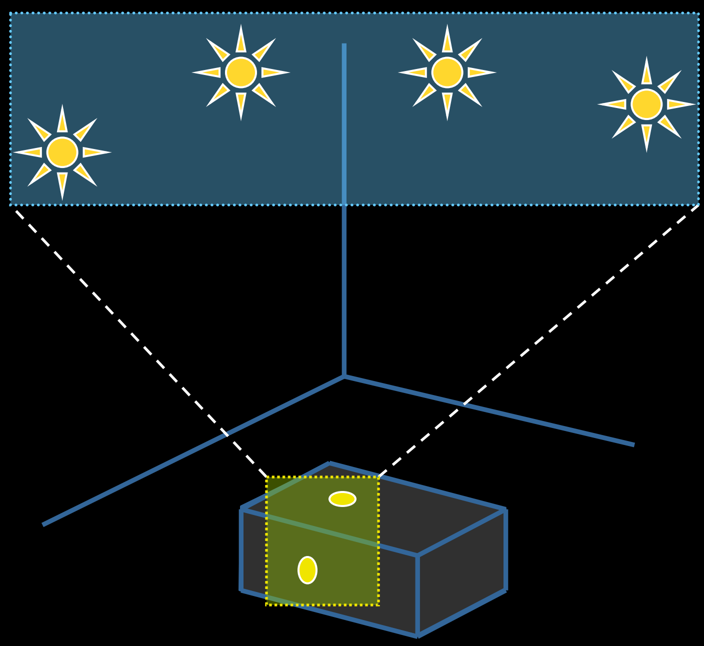
Product Graph

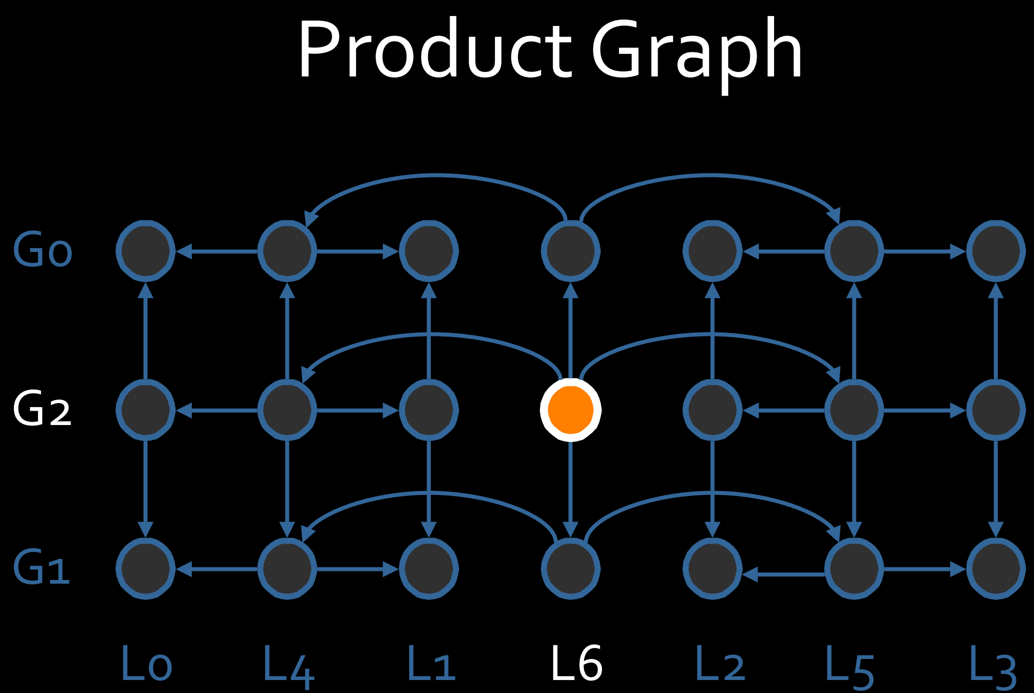


Product Graph

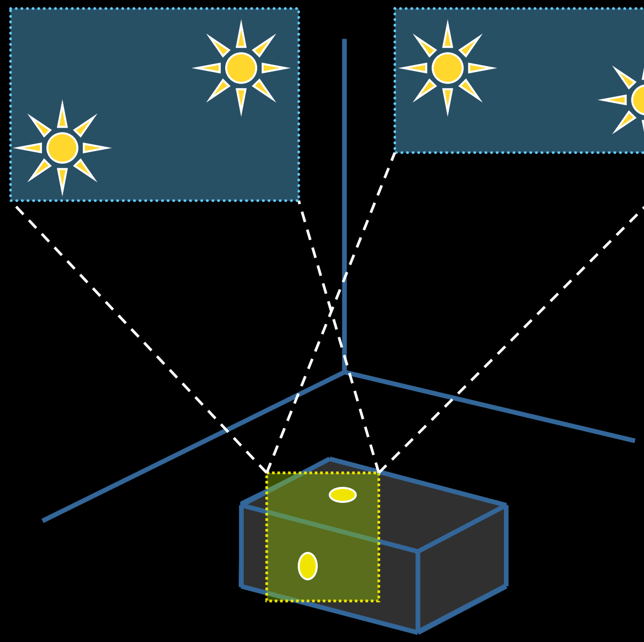


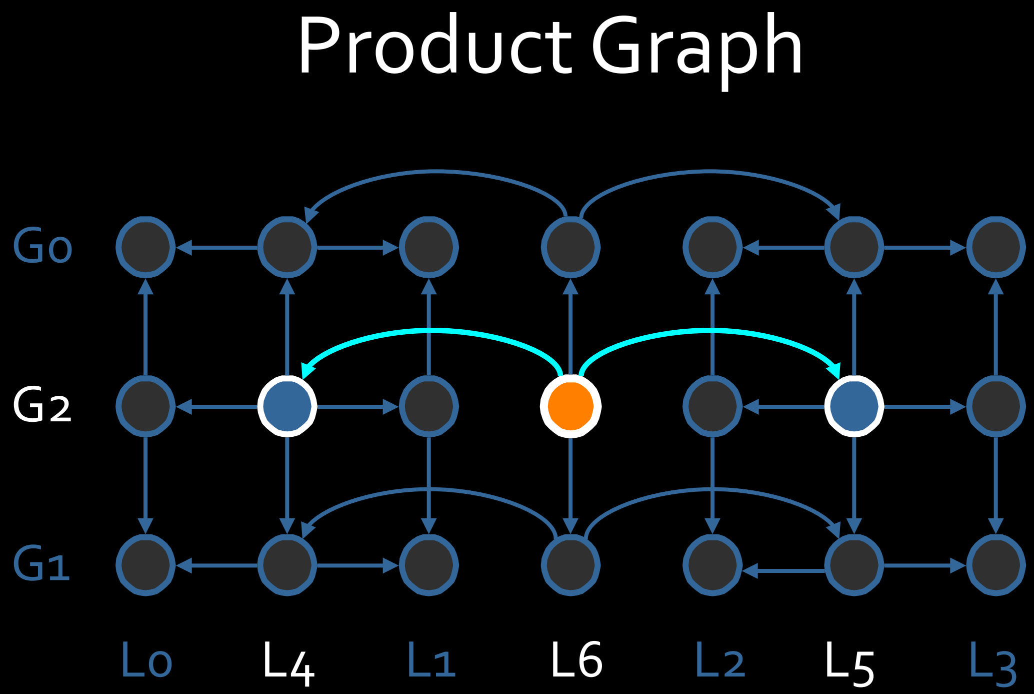
Product Graph



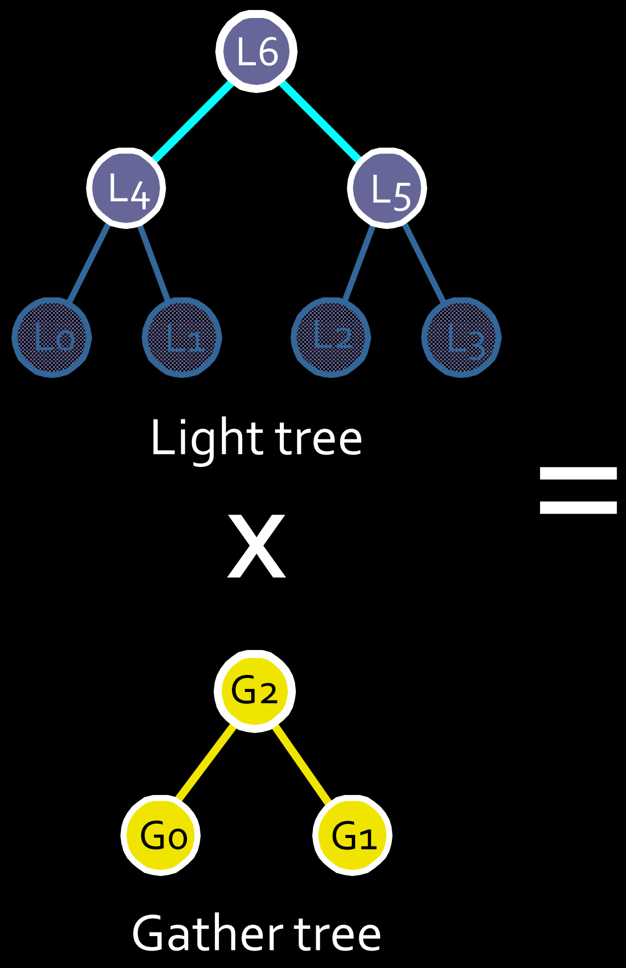


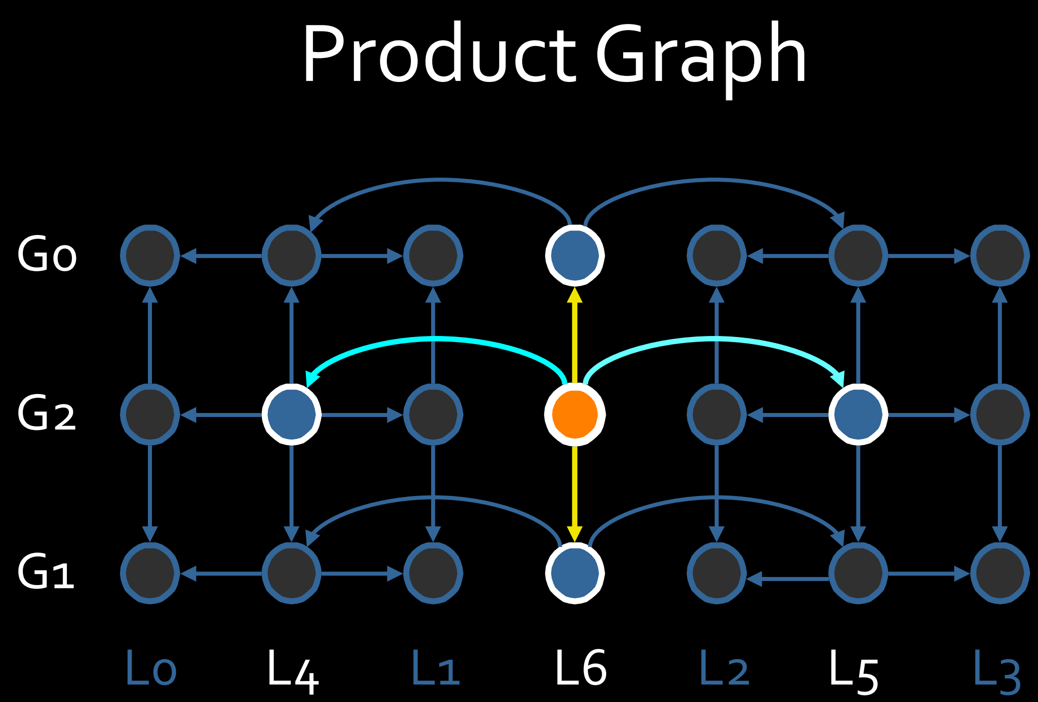
Product Graph



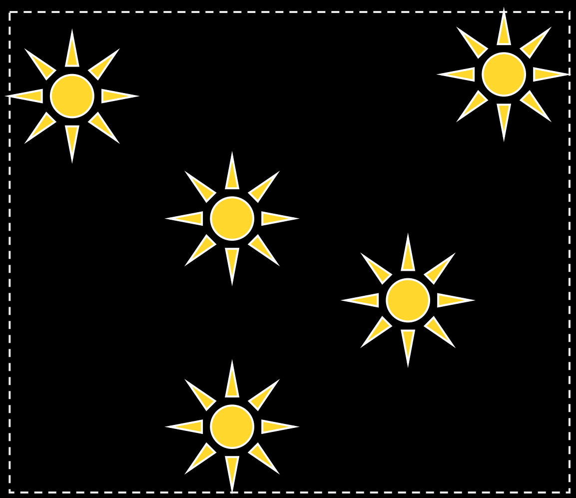


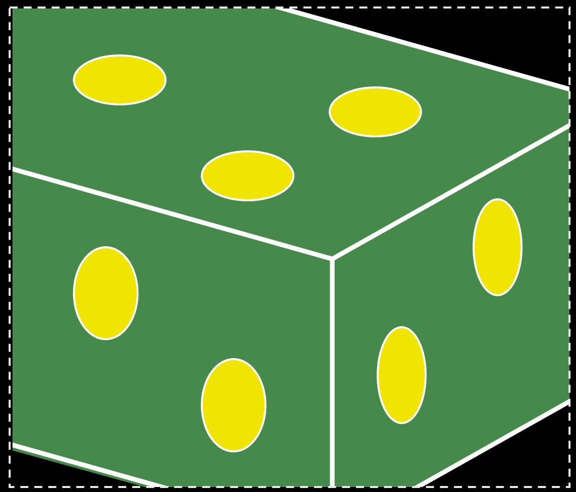
Product Graph



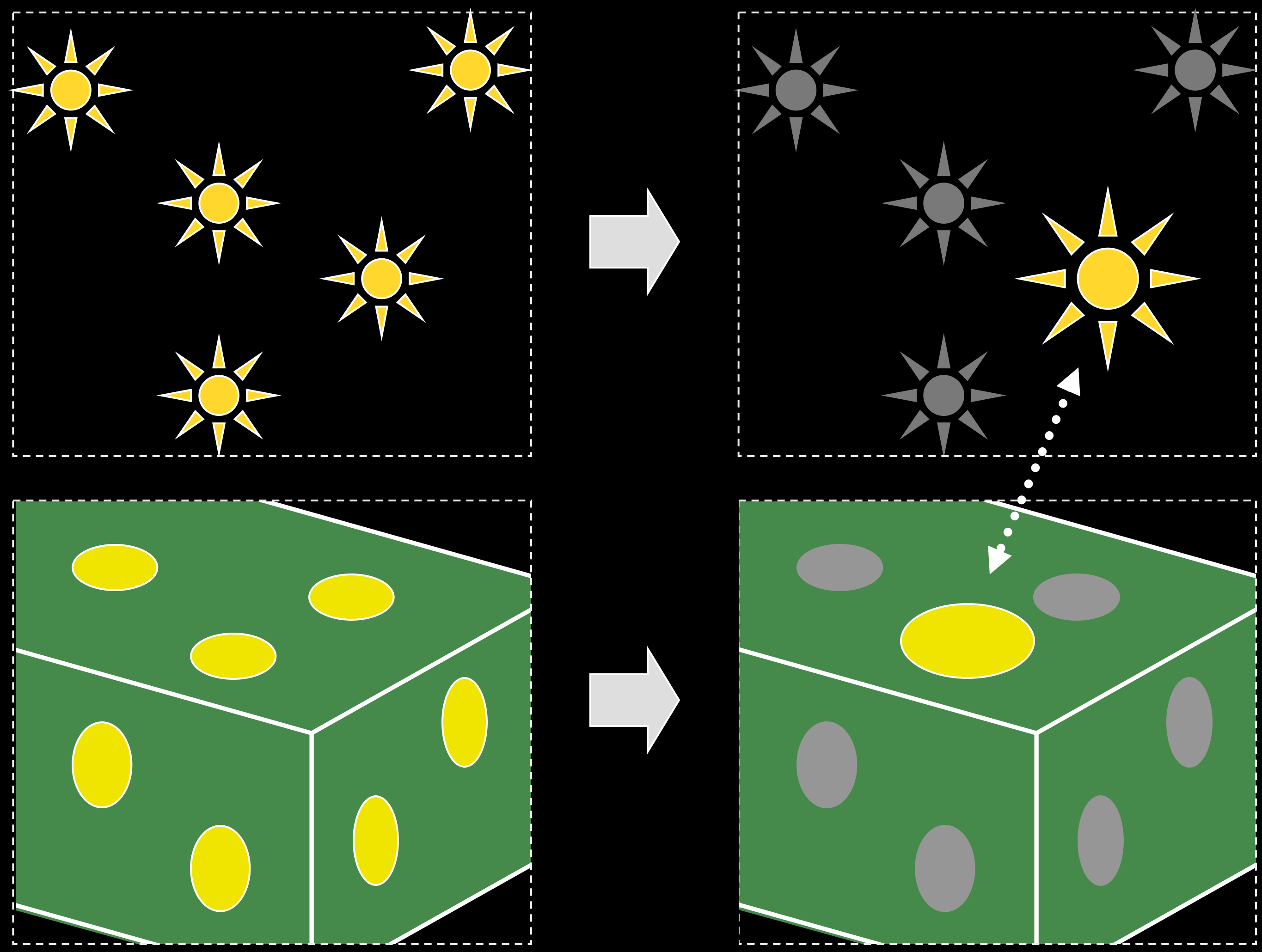


Cluster Representatives



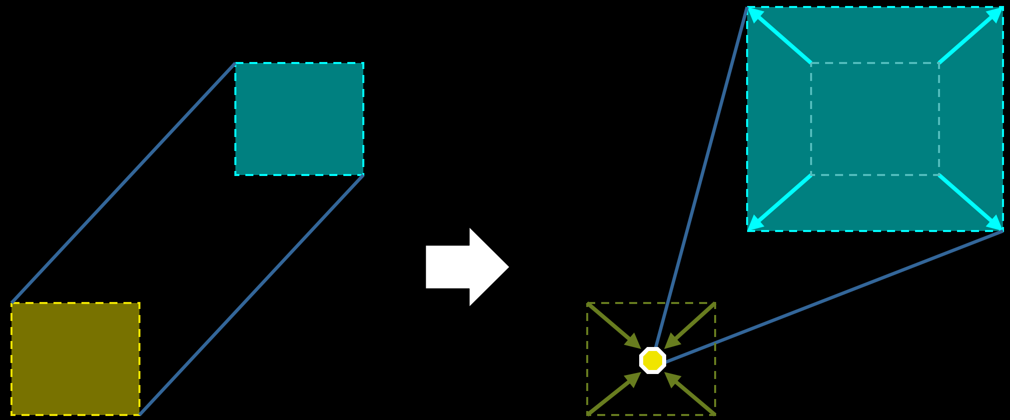


Cluster Representatives

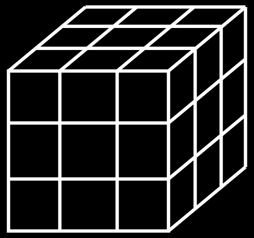


Error Bounds

- Collapse cluster-cluster interactions to point-cluster
  - Minkowski sums
  - Reuse bounds from Lightcuts



- Compute maximum over multiple BRDFs
  - Rasterize into cube-maps



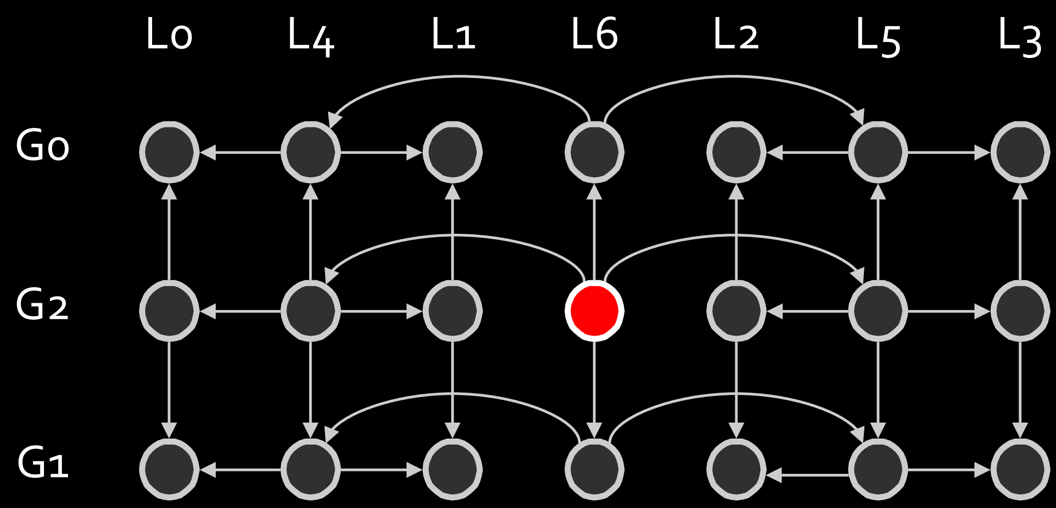
- More details in the paper

Algorithm Summary

- Once per image
  - Create lights and light tree
- For each pixel
  - Create gather points and gather tree for pixel
  - Adaptively refine clusters in product graph until all cluster errors < perceptual metric

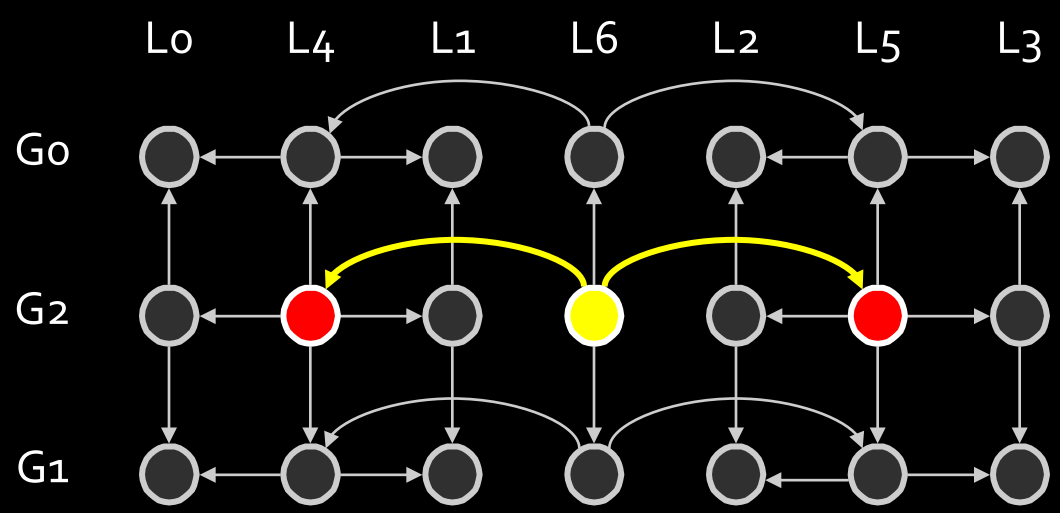
Scalability

- Start with a coarse cut
  - Eg, source node of product graph



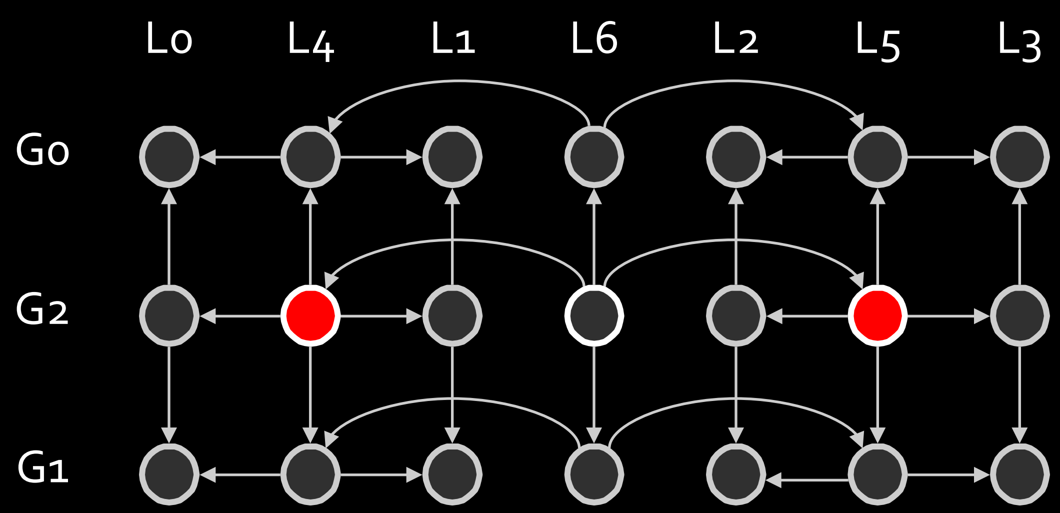
Scalability

- Choose node with largest error bound & refine
  - In gather or light tree



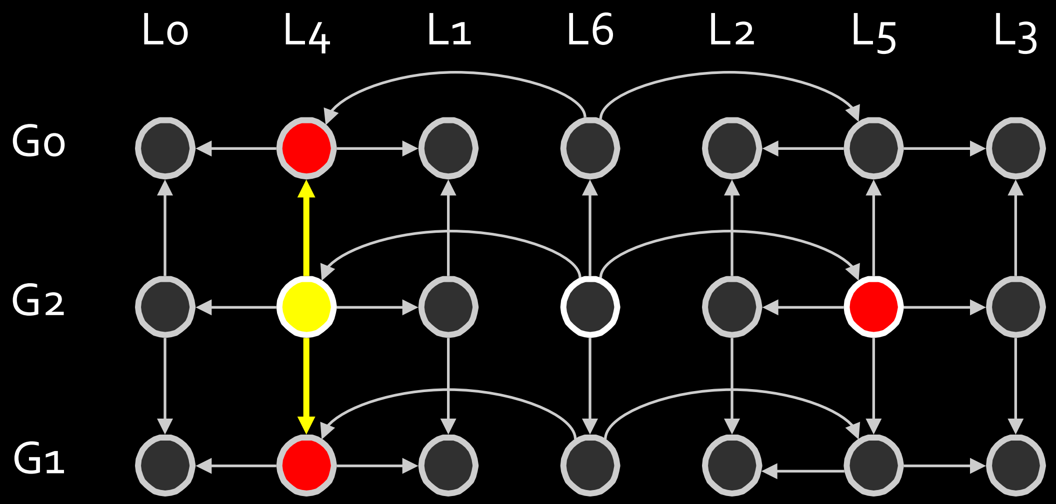
Scalability

- Choose node with largest error bound & refine
  - In gather or light tree



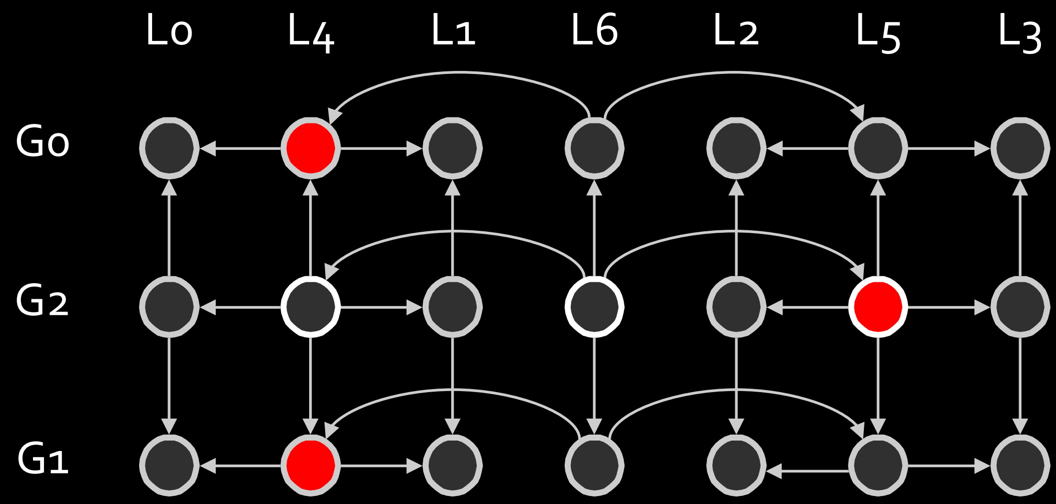
Scalability

- Repeat process



Algorithm summary

- Until all clusters errors < perceptual metric
  - 2% of pixel value (Weber's law)



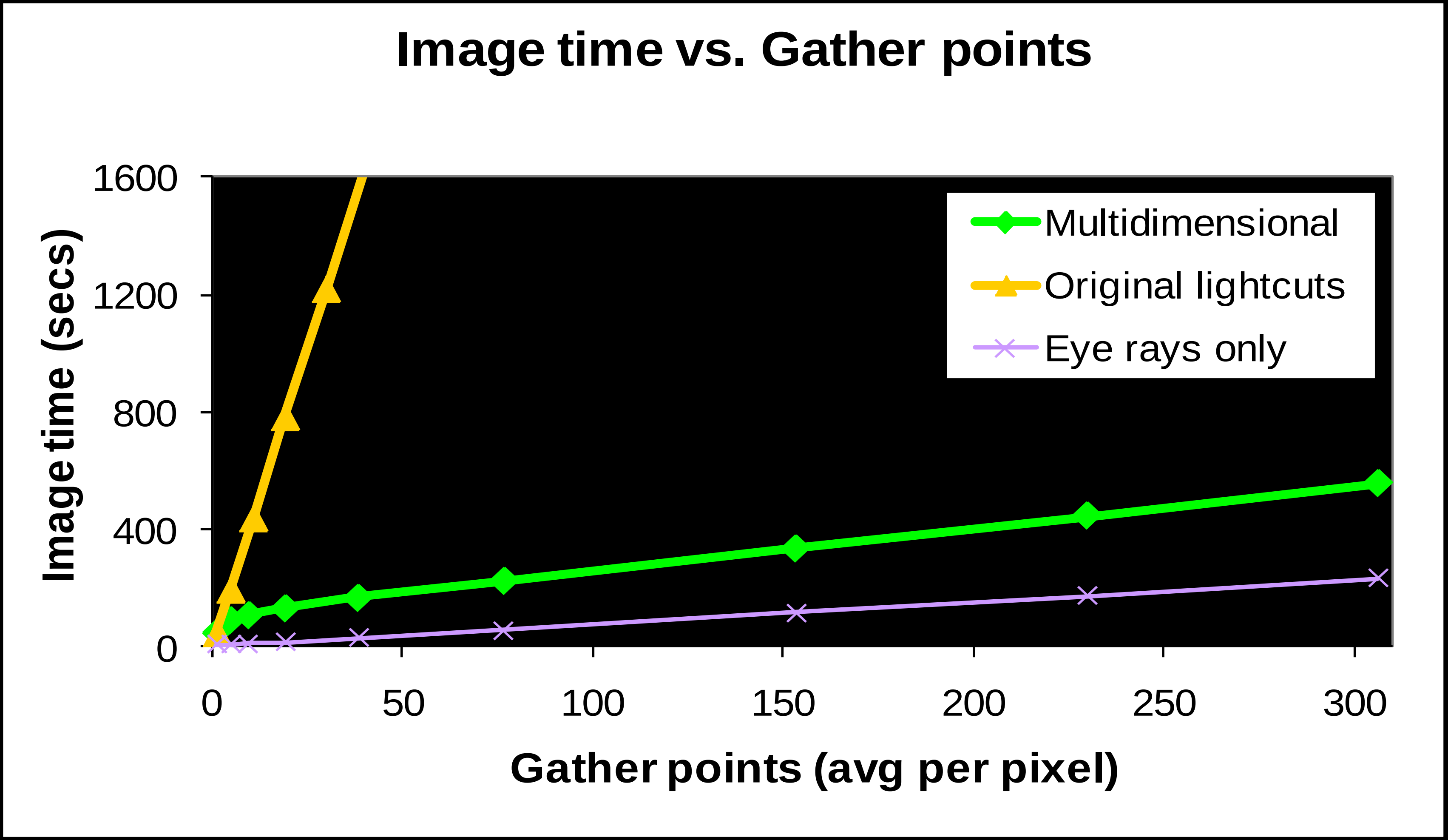
Results

- Limitations
  - Some types of paths not included
    - Eg, caustics
  - Prototype only supports diffuse, Phong, and Ward materials and isotropic media

Roulette

7,047,430 Pairs per pixel   Time 590 secs
Avg cut size 174 (0.002%)

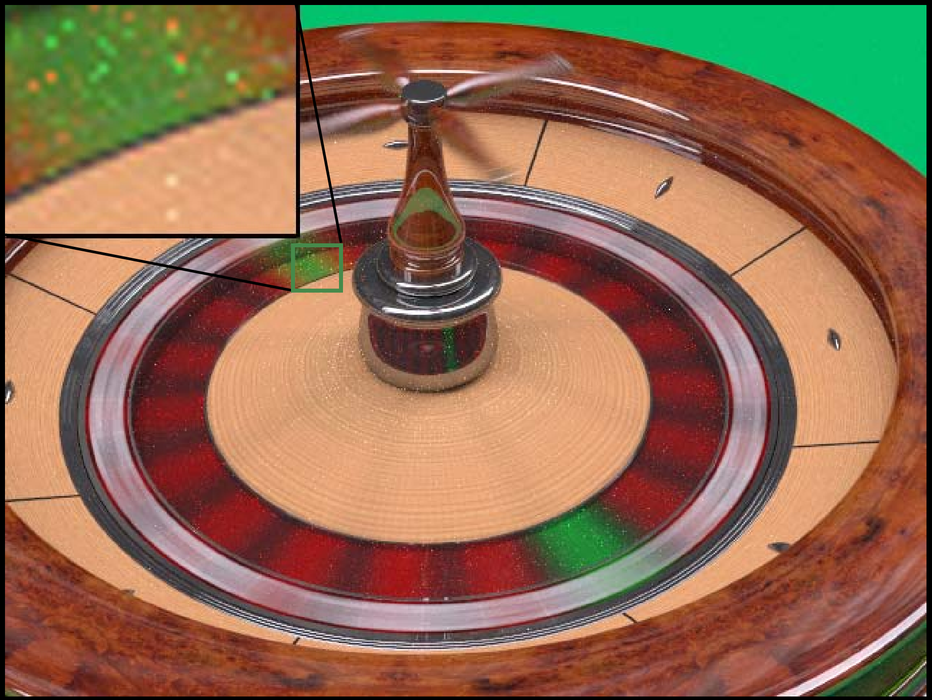
Scalability



Metropolis Comparison

Zoomed insets

Our result
Time 9.8min



Metropolis
Time 148min (15x)
Visible noise
5% brighter (caustics etc.)

Kitchen

5,518,900 Pairs per pixel Time 705 secs
Avg cut size 936 (0.017%)

180 Gather points X 13,000 Lights = 234,000 Pairs per pixel

Avg cut size 447 (0.19%)

Scalable many-light rendering

Matrix Row-Column sampling

Hašan et al., SIGGRAPH 2007

Slides courtesy Miloš Hašan:

http://www.cs.cornell.edu/~mhasan/

Matrix Row-Column sampling

Matrix Row-Column Sampling for the Many-Light Problem

Miloš Hašan*
Cornell University

Fabio Pellacini
Dartmouth College

Kavita Bala
Cornell University

2.2m triangles: 300 rows, 900 columns, 16.9 s | 388k triangles: 432 rows, 864 columns, 13.5 s | 869k triangles: 100 rows, 200 columns, 3.8 s

Figure 1: *In the above images, over 1.9 million surface samples are shaded from over 100 thousand point lights in a few seconds. This is achieved by sampling a few hundred rows and columns from the large unknown matrix of surface-light interactions.*

- http://miloshasan.net/
- Designed specifically for GPU rendering (shadow mapping)

Improving Scalability and Performance

Brute force

(100k VPLs)

10 min

13 min

20 min

↓ ↓ ↓

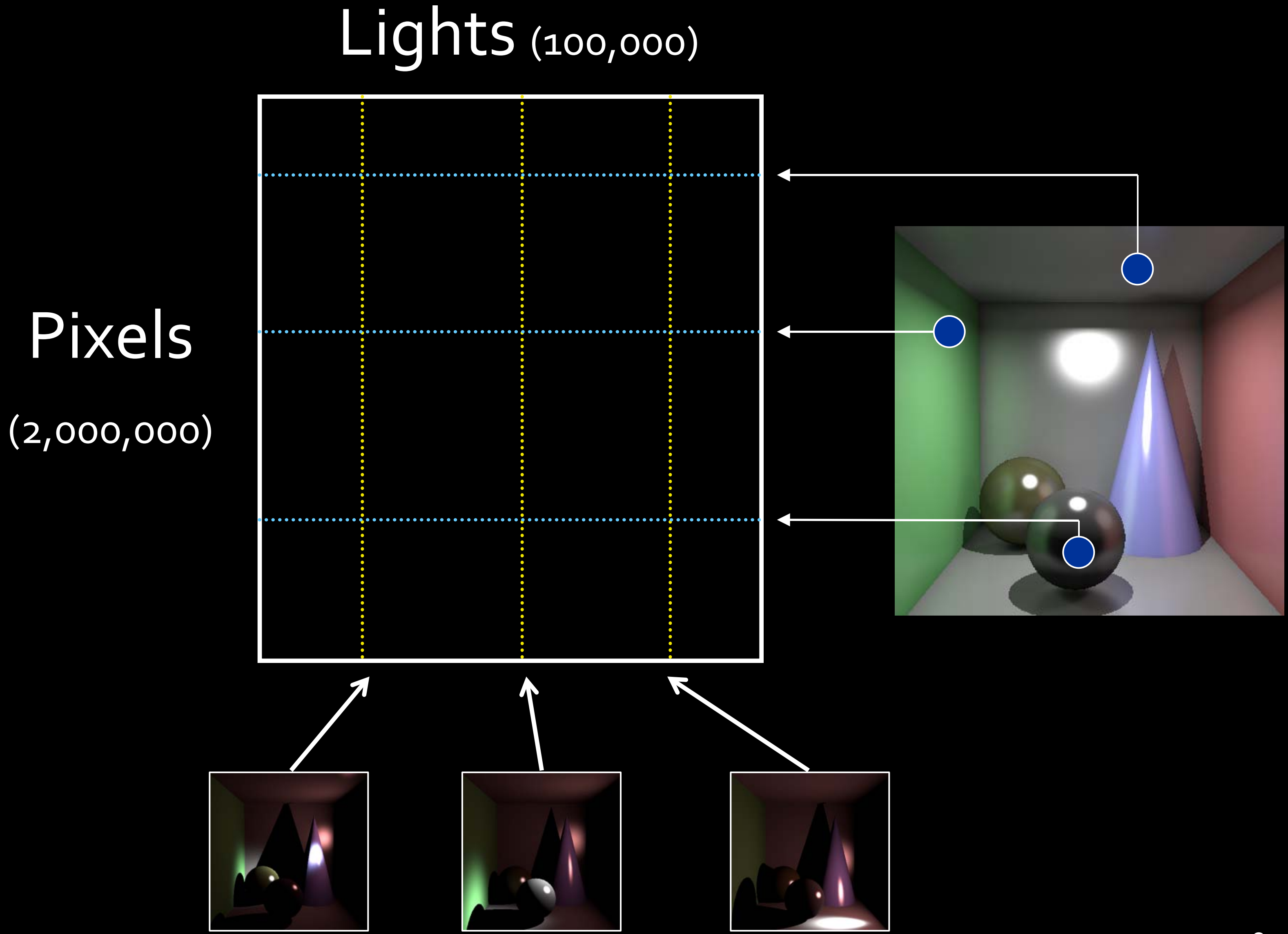
MRCS
(our result)

3.8 sec

13.5 sec

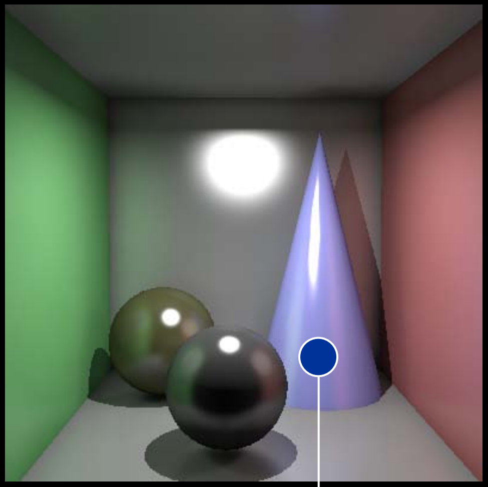
16.9 sec

A Matrix Interpretation

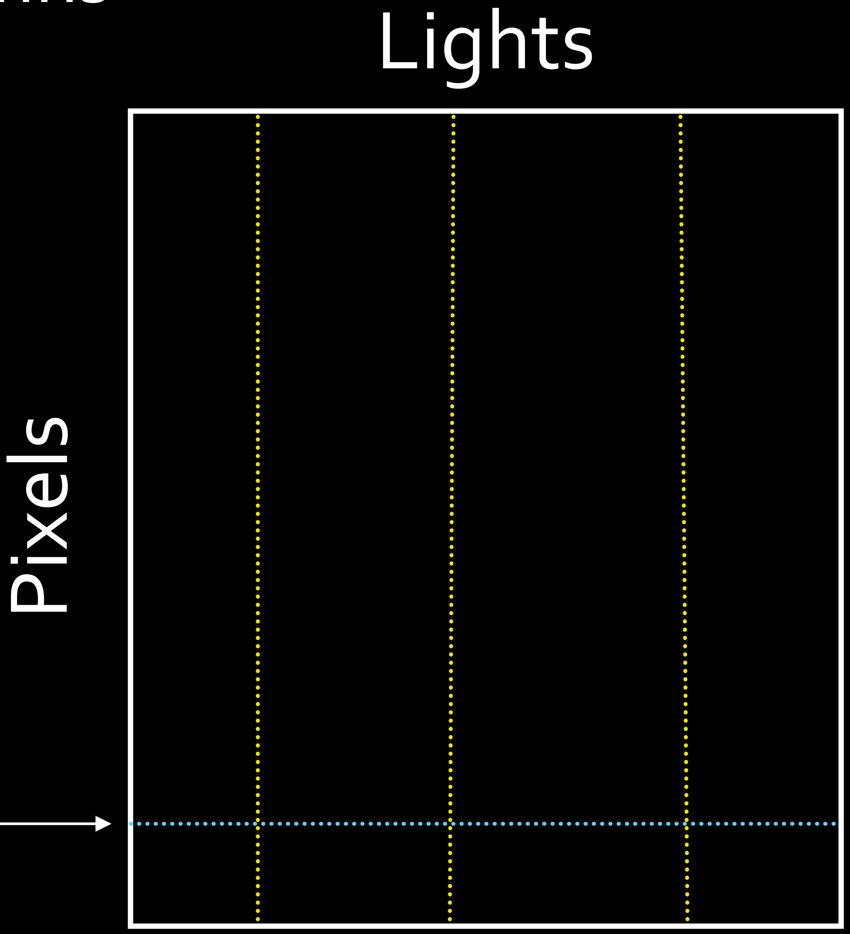


Problem Statement

- Compute sum of columns



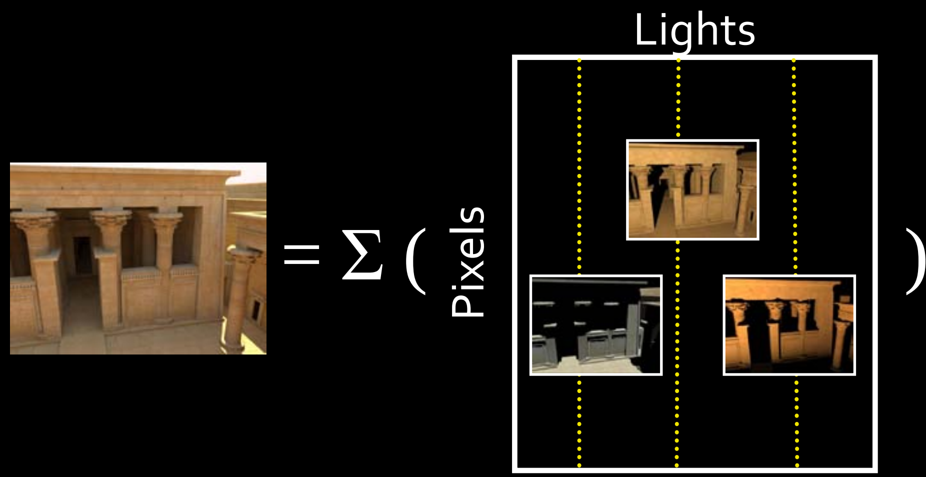
$= \Sigma \Big($



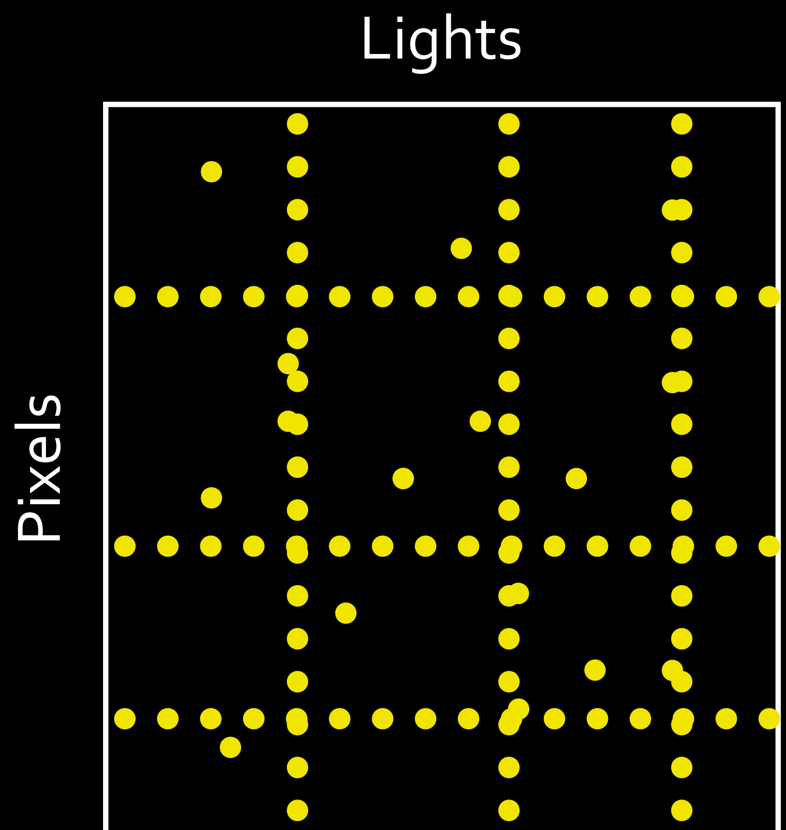
$\Big)$

Low-Rank Assumption

- Column space is (close to) low-dimensional



Ray-tracing vs Shadow Mapping

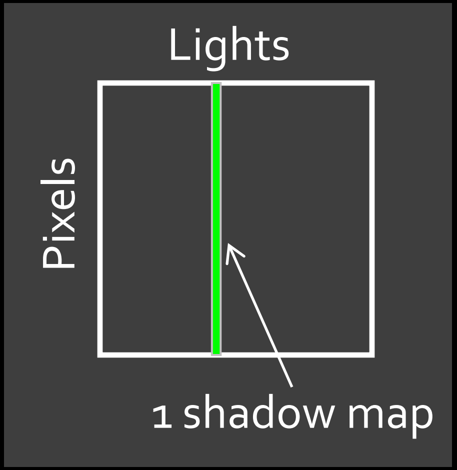


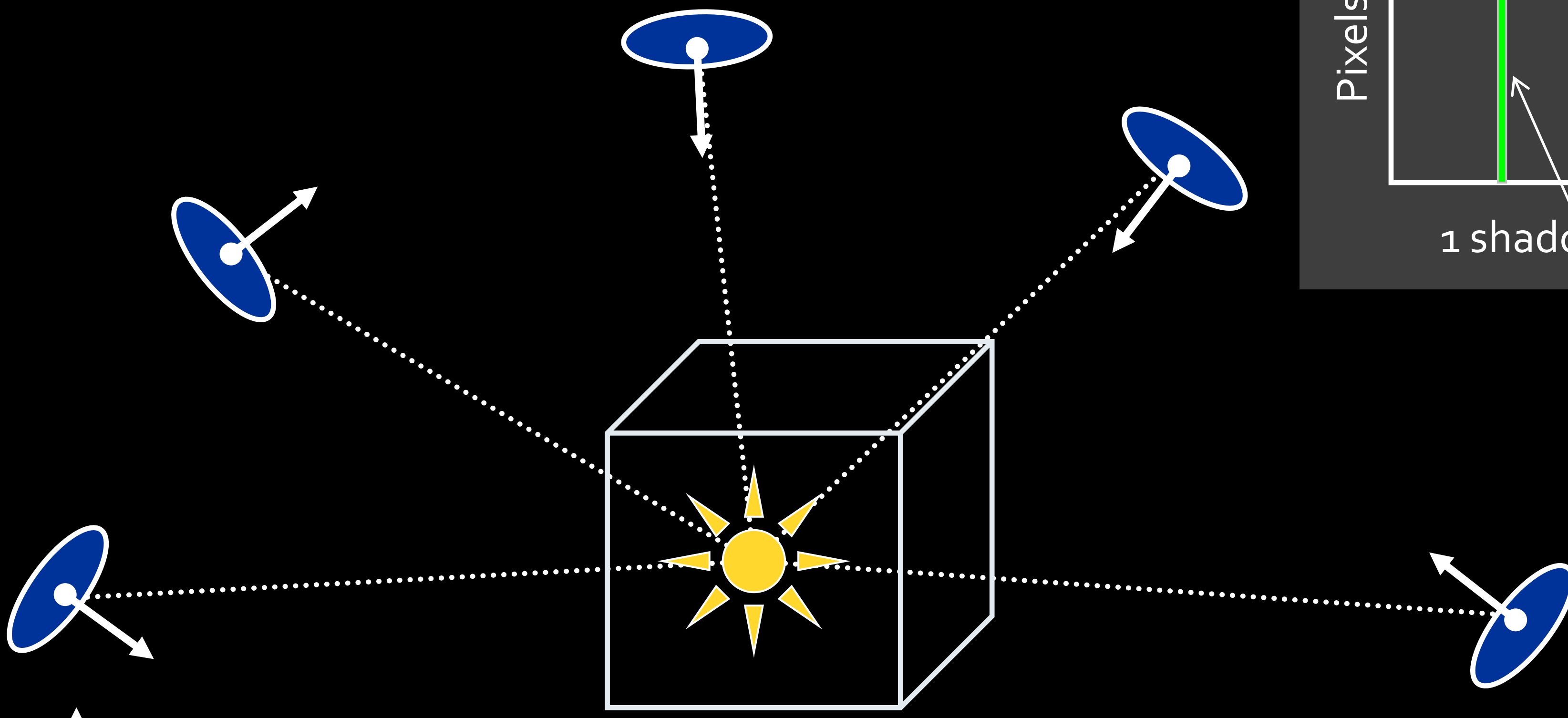
Point-to-many-points visibility: Shadow-mapping

Point-to-point visibility: Ray-tracing

Computing Column Visibility

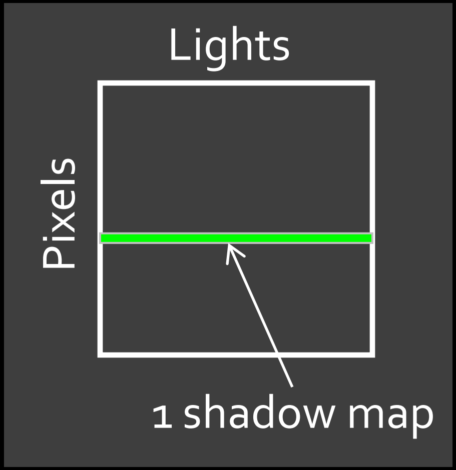
- Regular Shadow Mapping

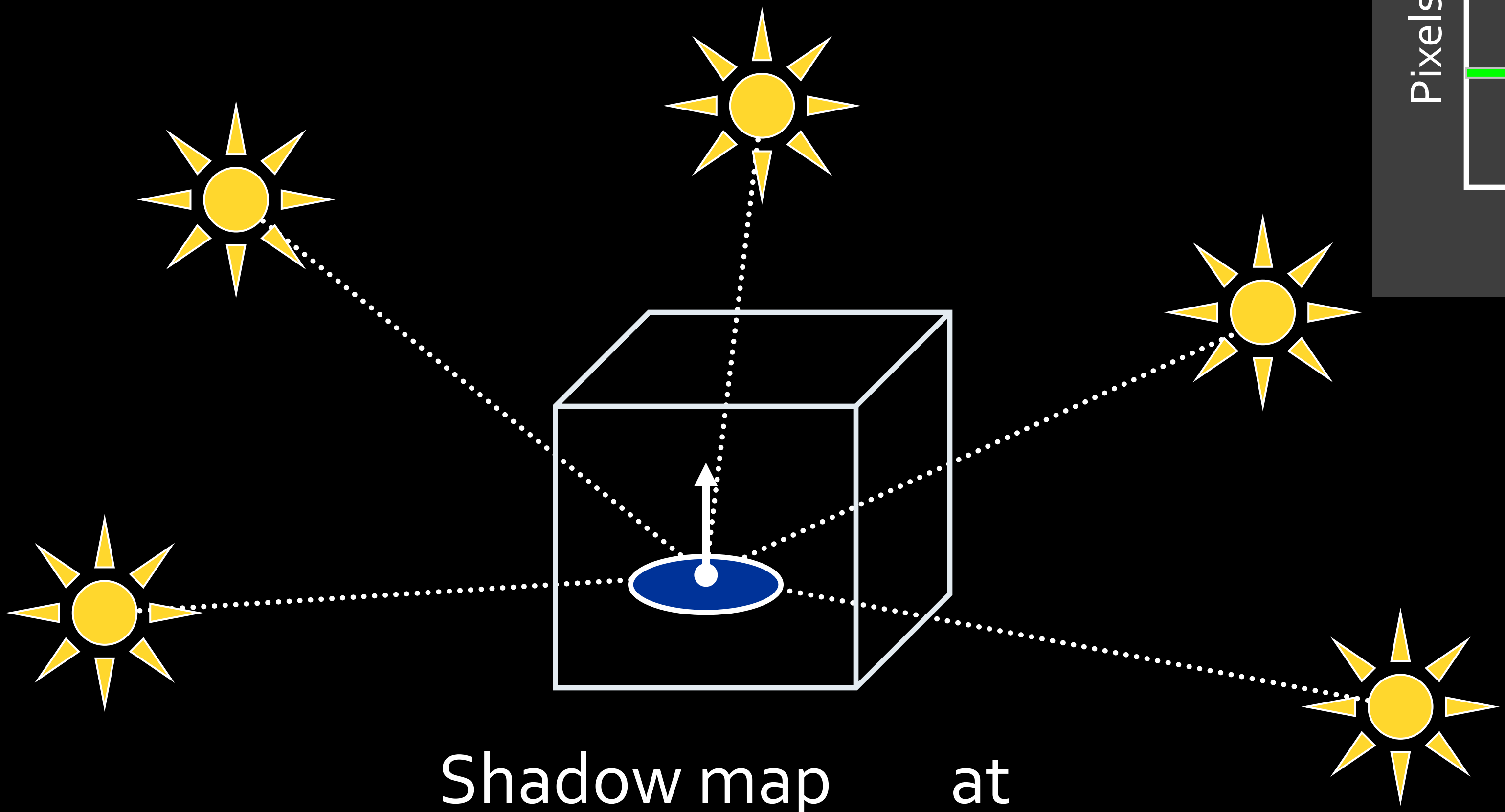




Row-Column Duality

- Rows: Also Shadow Mapping!

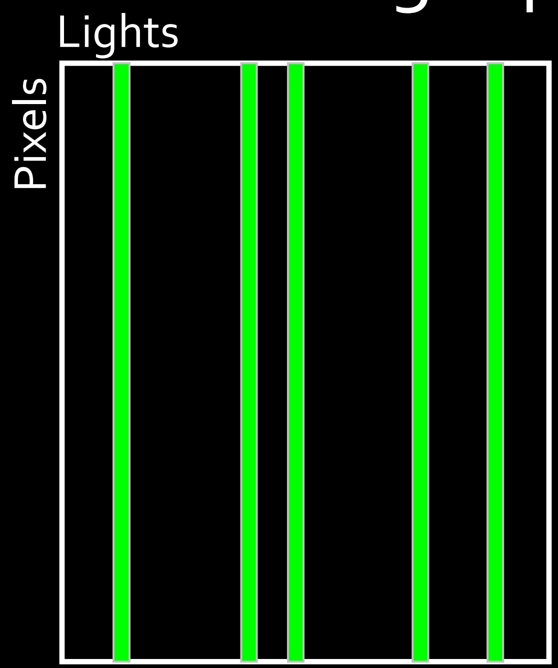




Shadow map at sample position

Image as a Weighted Column Sum

- The following is possible:

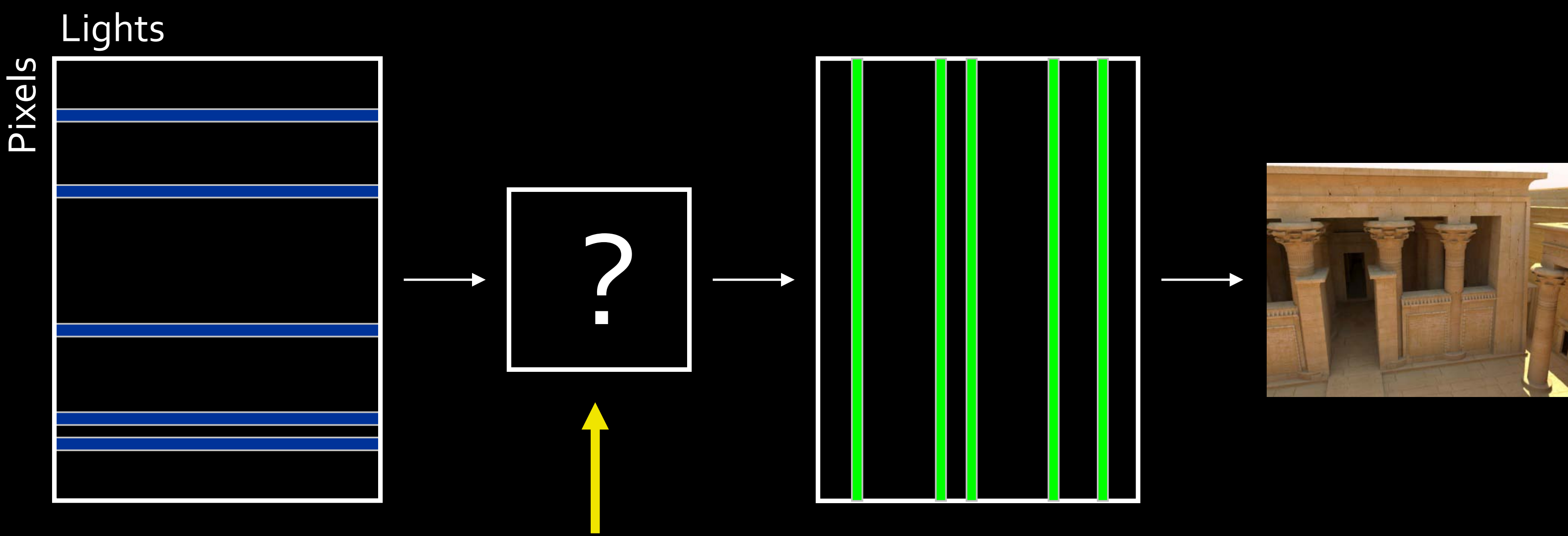


Compute small
subset of columns
(i.e. pick some lights)

compute
weighted sum

Use rows to choose a good set of columns (=lights)

The Row-Column Sampling Idea



compute rows

(i.e. pick pixels, compute contrib from ALL lights for each)

choose columns (=lights) and weights

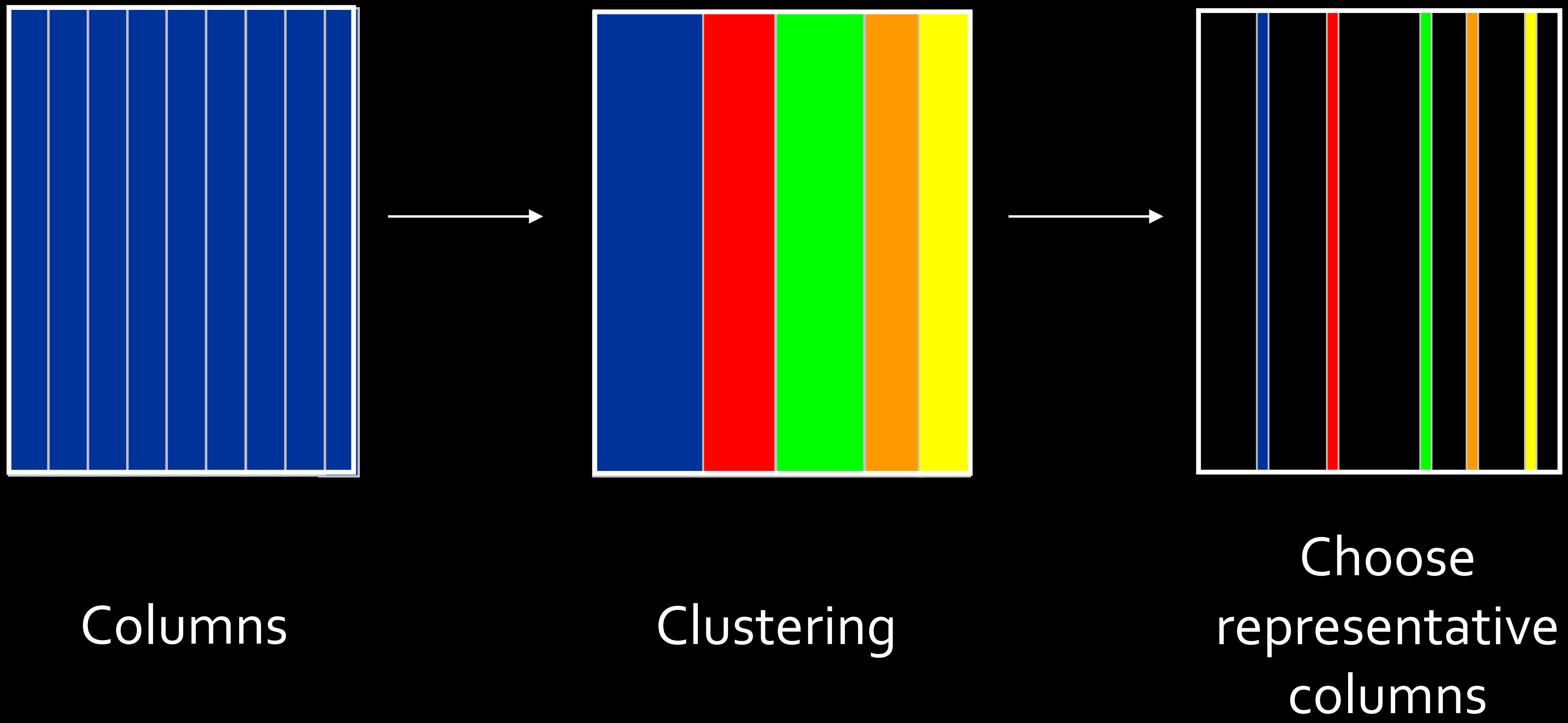
how to choose columns and weights?

compute columns

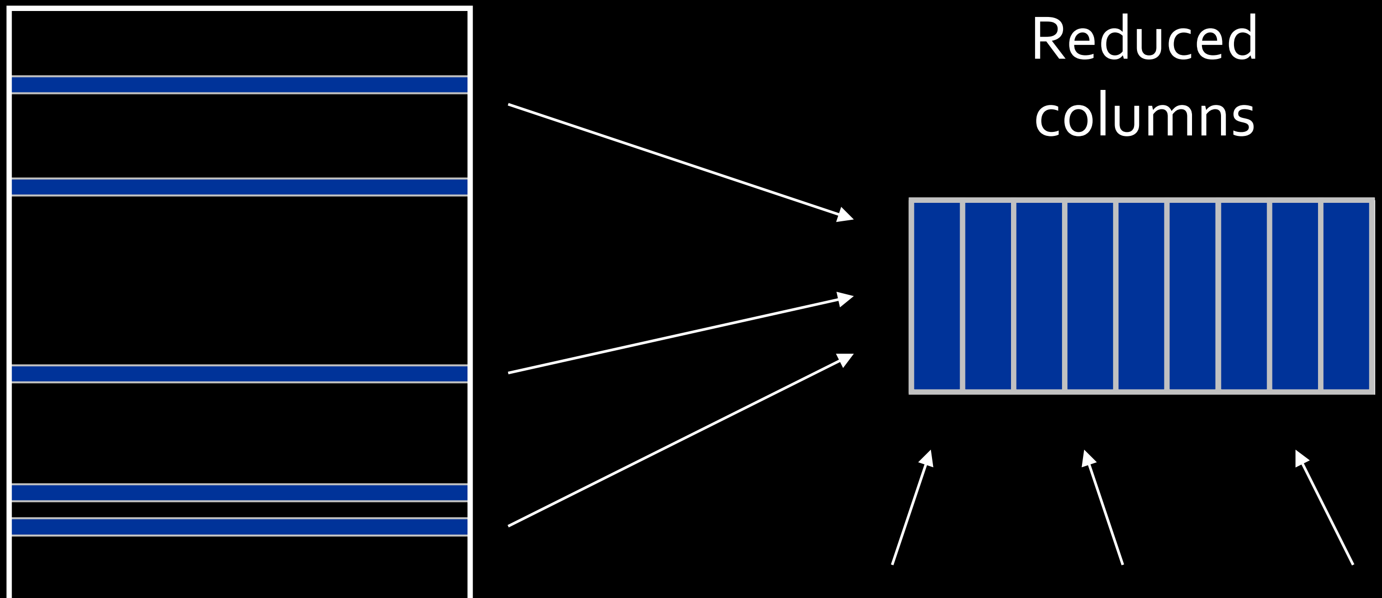
(i.e. for the selected lights, compute contribution to all pixels)

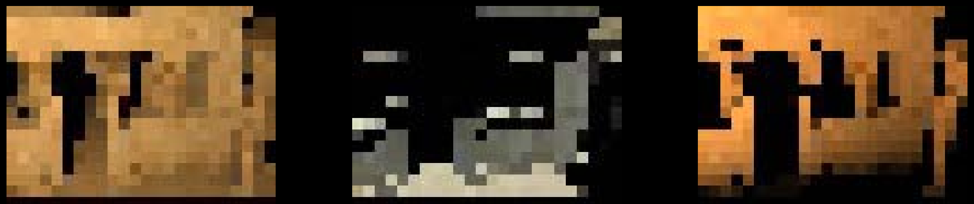
weighted sum

Clustering Approach



Reduced Matrix





Monte Carlo Estimator

- Algorithm:
  1. Cluster reduced columns
  2. Choose a representative in each cluster, with probability proportional to weight
  3. Approximate other columns in cluster by (scaled) representative
- This is an unbiased Monte Carlo estimator (of the sum of matrix columns)
- Which clustering minimizes its variance?

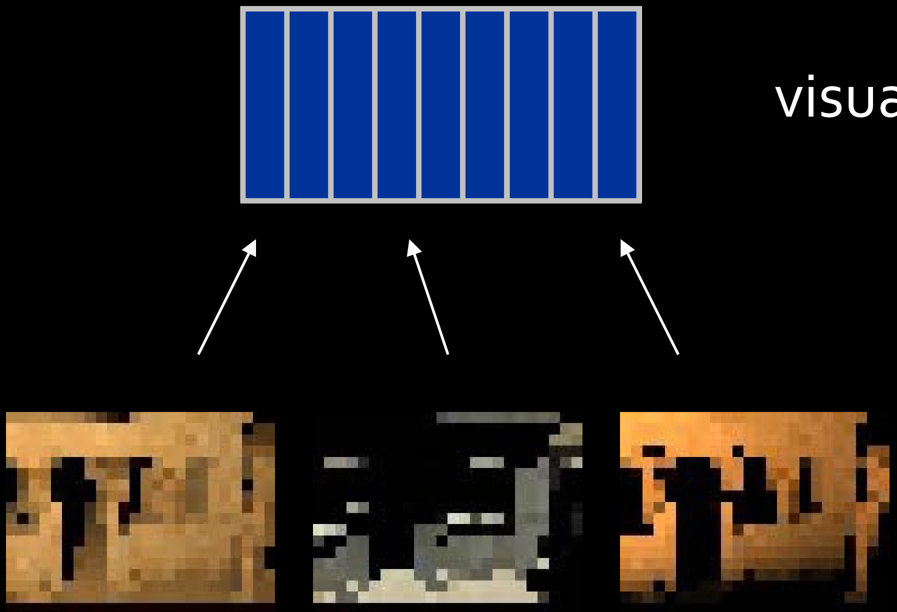
Weights and Information Vectors

- Weights w_i
  - Norms of reduced columns
  - Represent the "energy" of the light

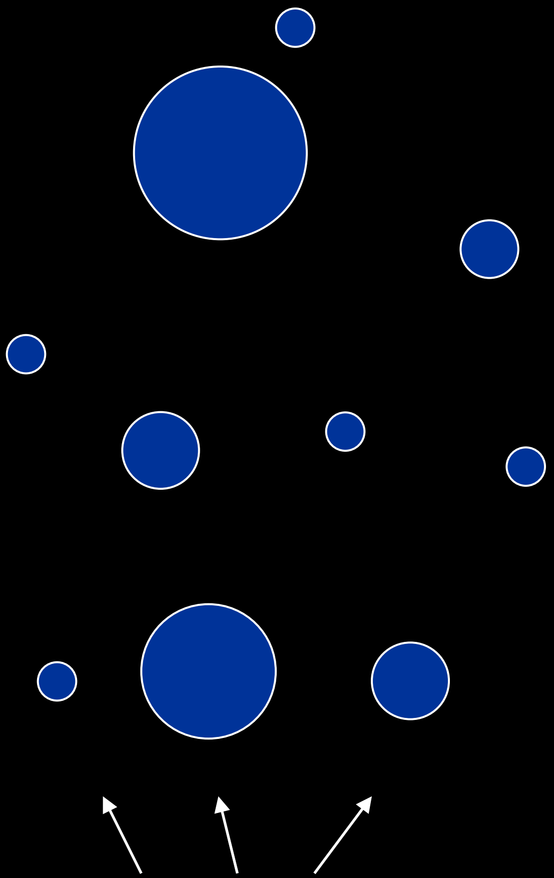
- Information vectors x_i
  - Normalized reduced columns
  - Represent the "kind" of light's contribution

Visualizing the Reduced Columns

Reduced columns:
vectors in high-
dimensional space



visualize as …



radius = weight

position = information vector

The Clustering Objective

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

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

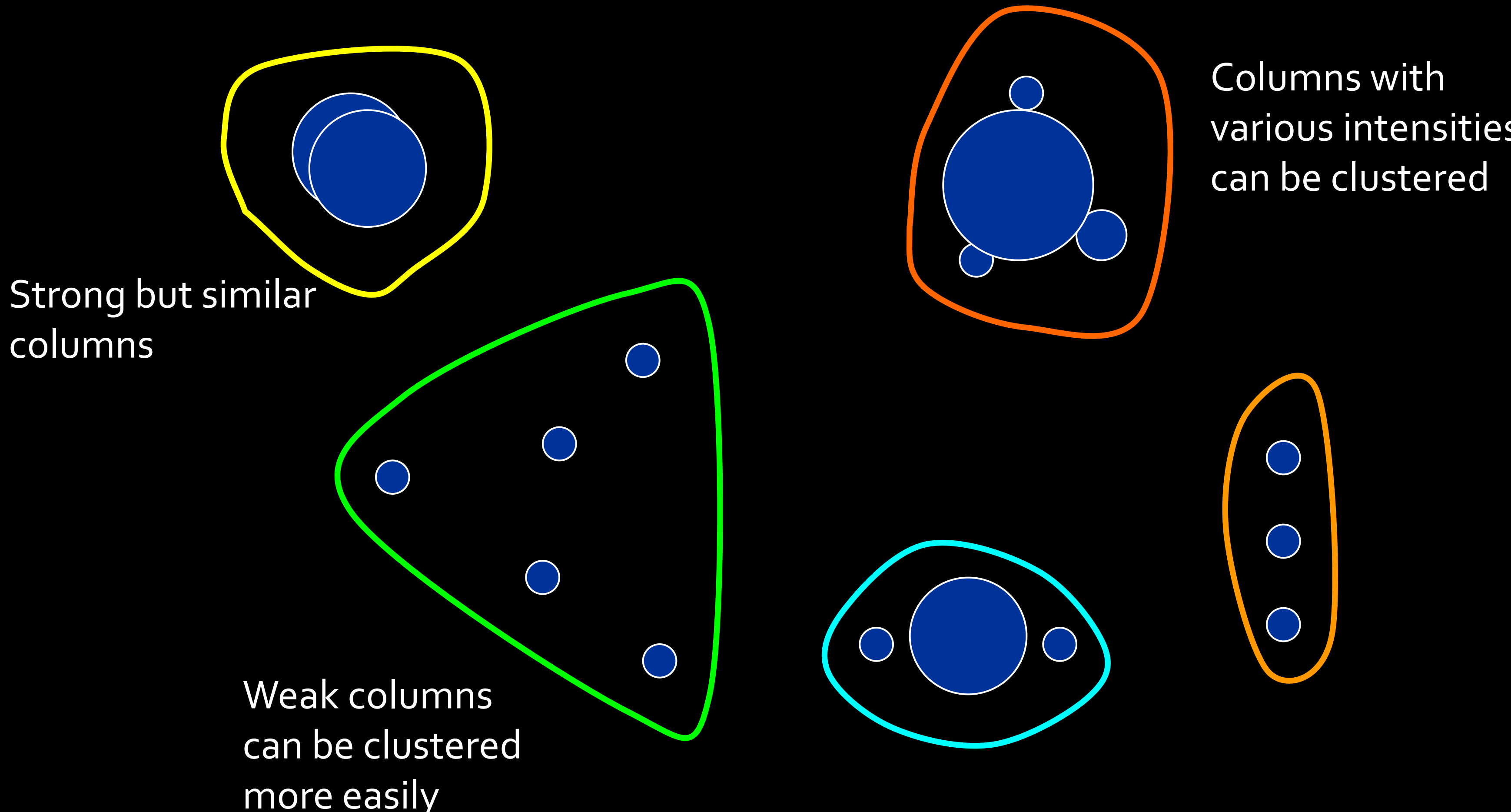
cost of a cluster

sum over all pairs in it

weights

squared distance between information vectors

Clustering Illustration

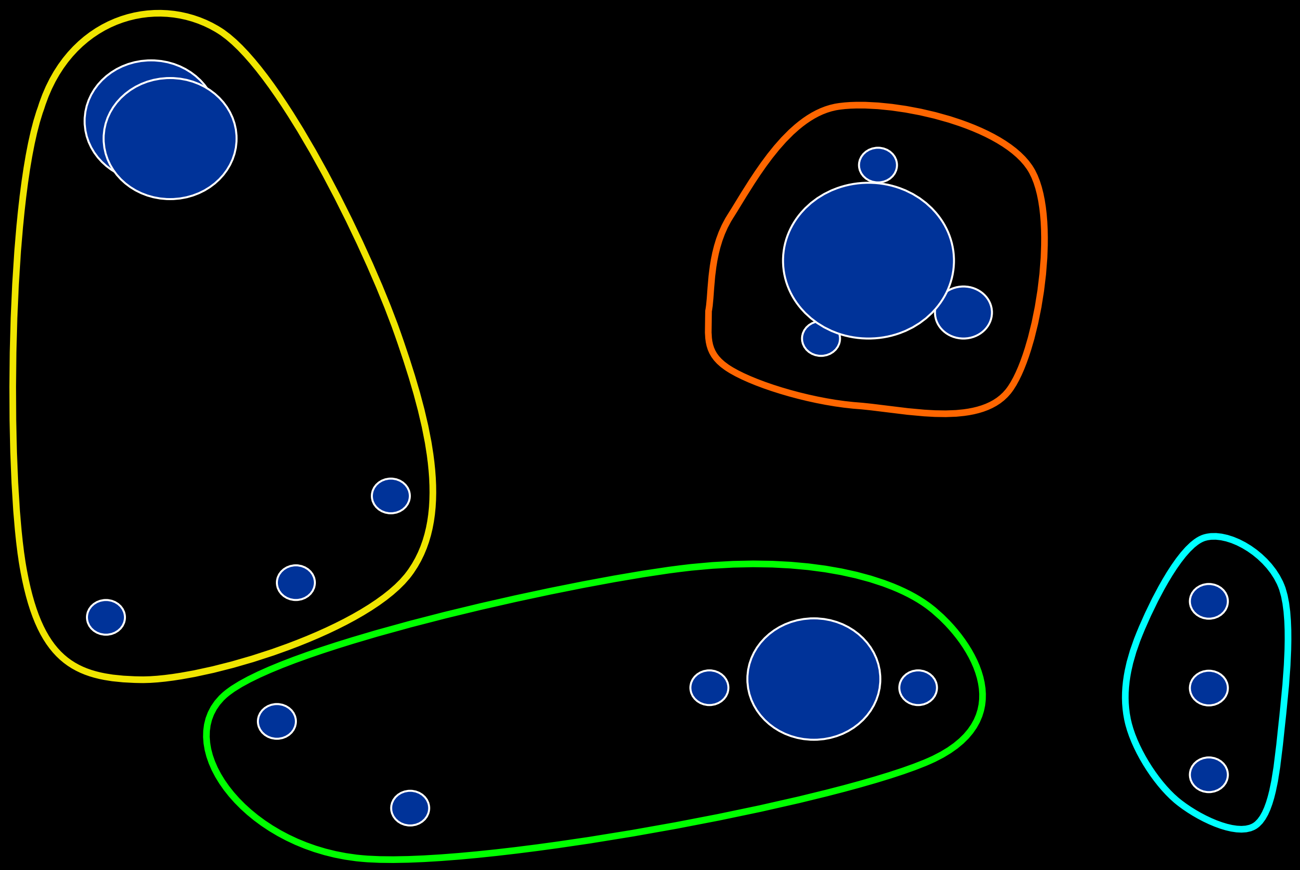


$$cost(C) = \sum_{i,j \in C} w_i \; w_j \; \|\mathbf{x}_i - \mathbf{x}_j\|^2$$

How to minimize?

- Problem is NP-hard
- Not much previous research
- Should handle large input:
  - 100,000 points
  - 1000 clusters
- We introduce 2 heuristics:
  - Random sampling
  - Divide & conquer

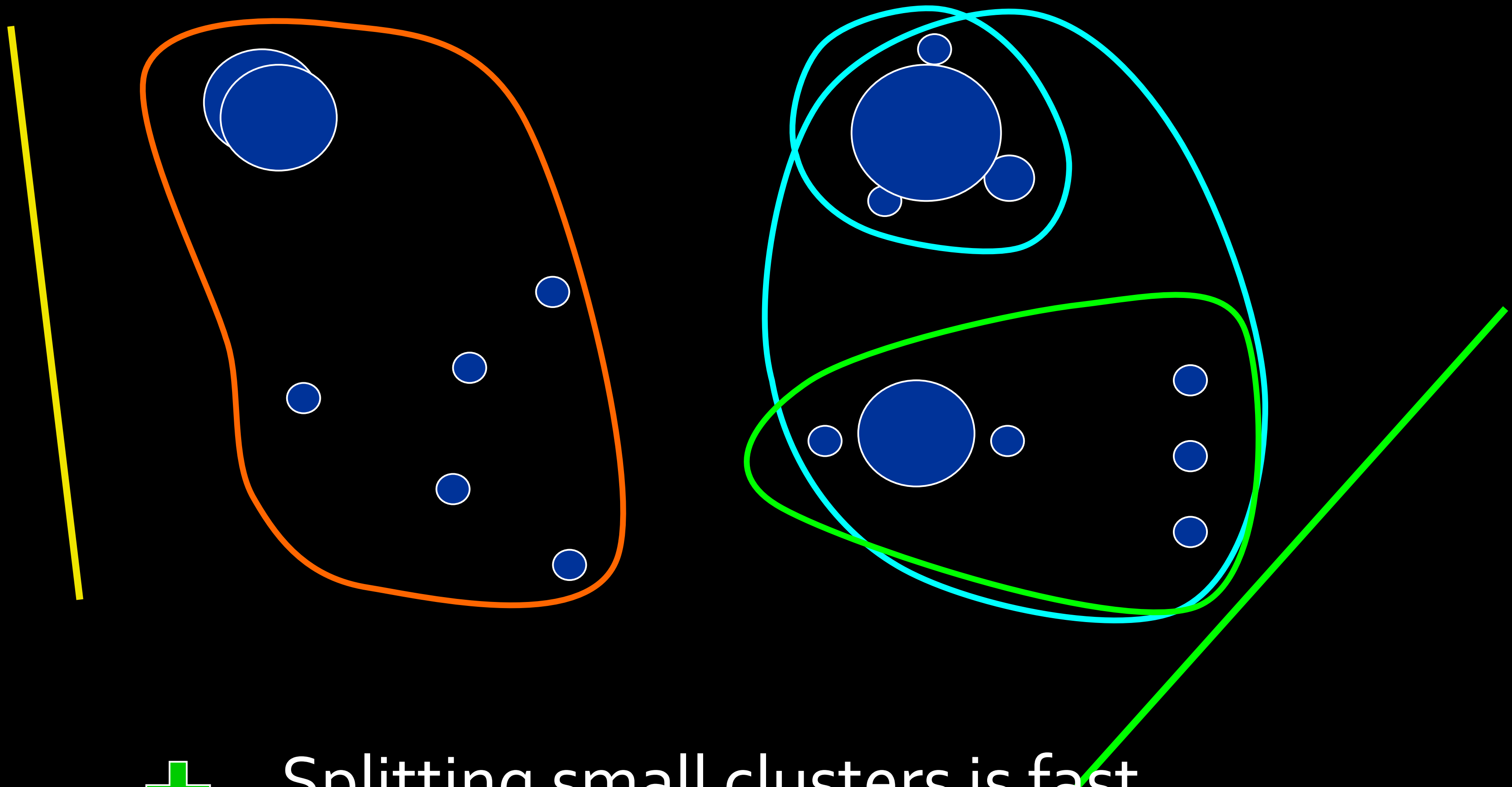
Clustering by Random Sampling



+ Very fast (use optimized BLAS)

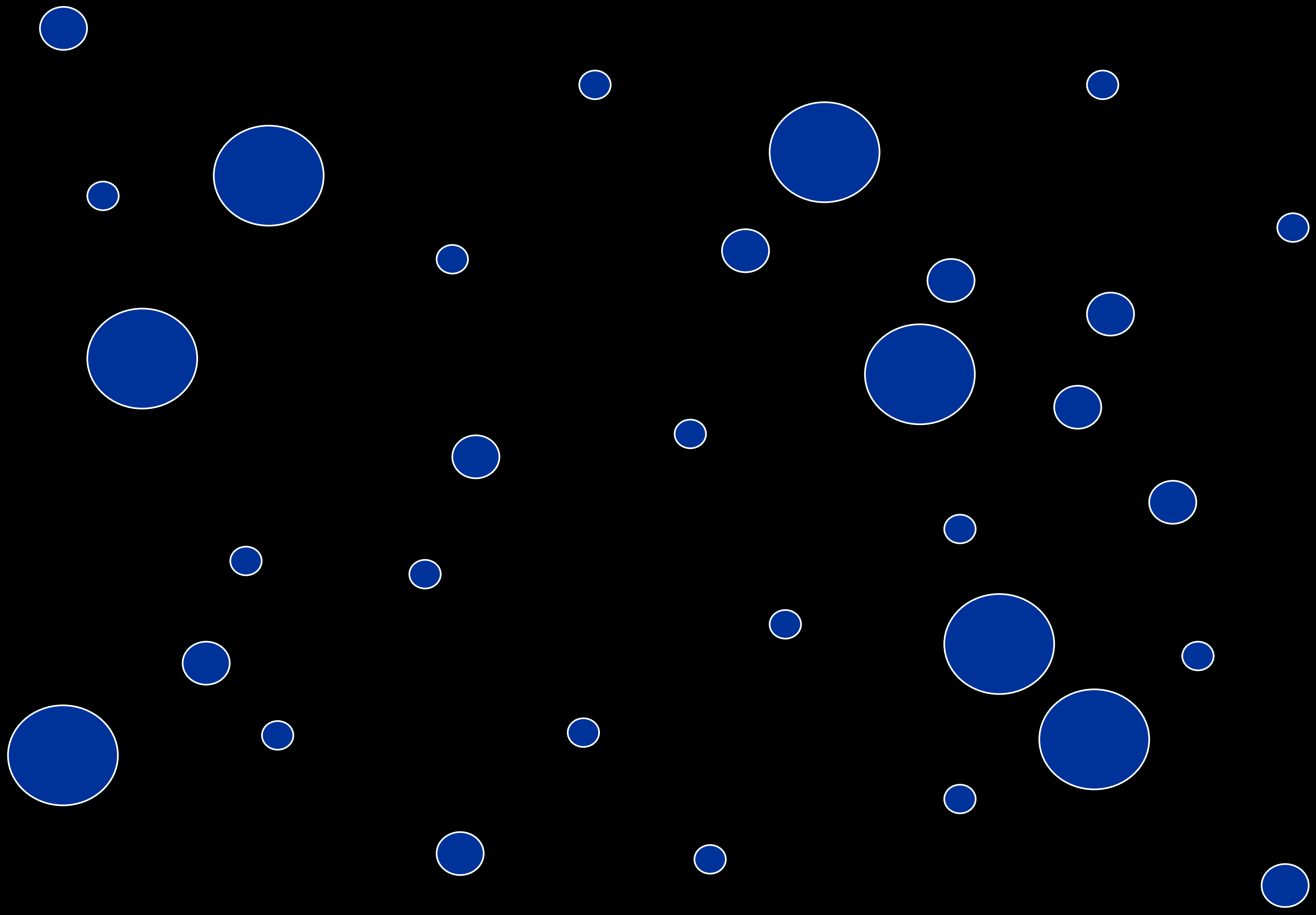
− Some clusters might be too small / large

Clustering by Divide & Conquer

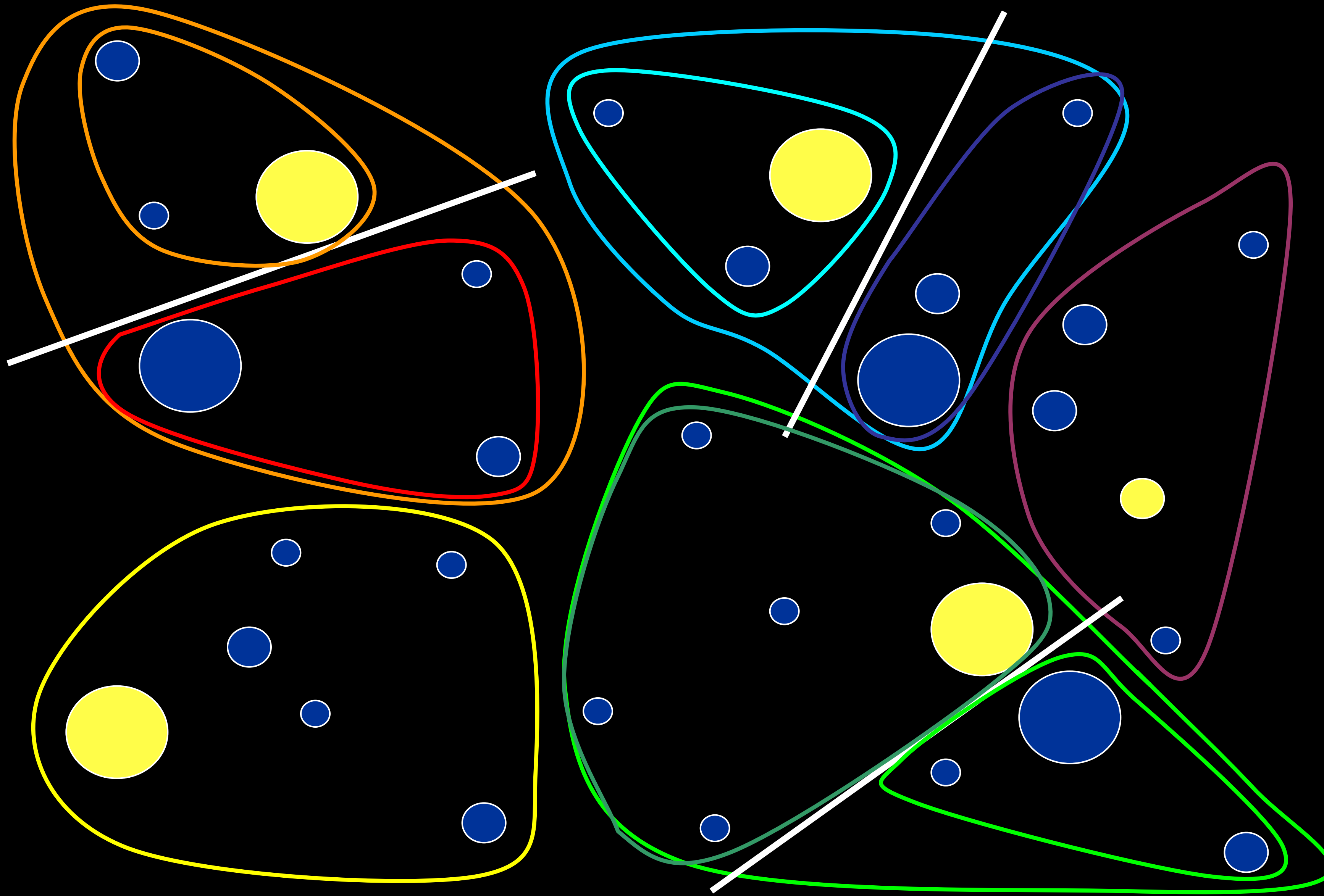


- Splitting small clusters is fast
- Splitting large clusters is slow

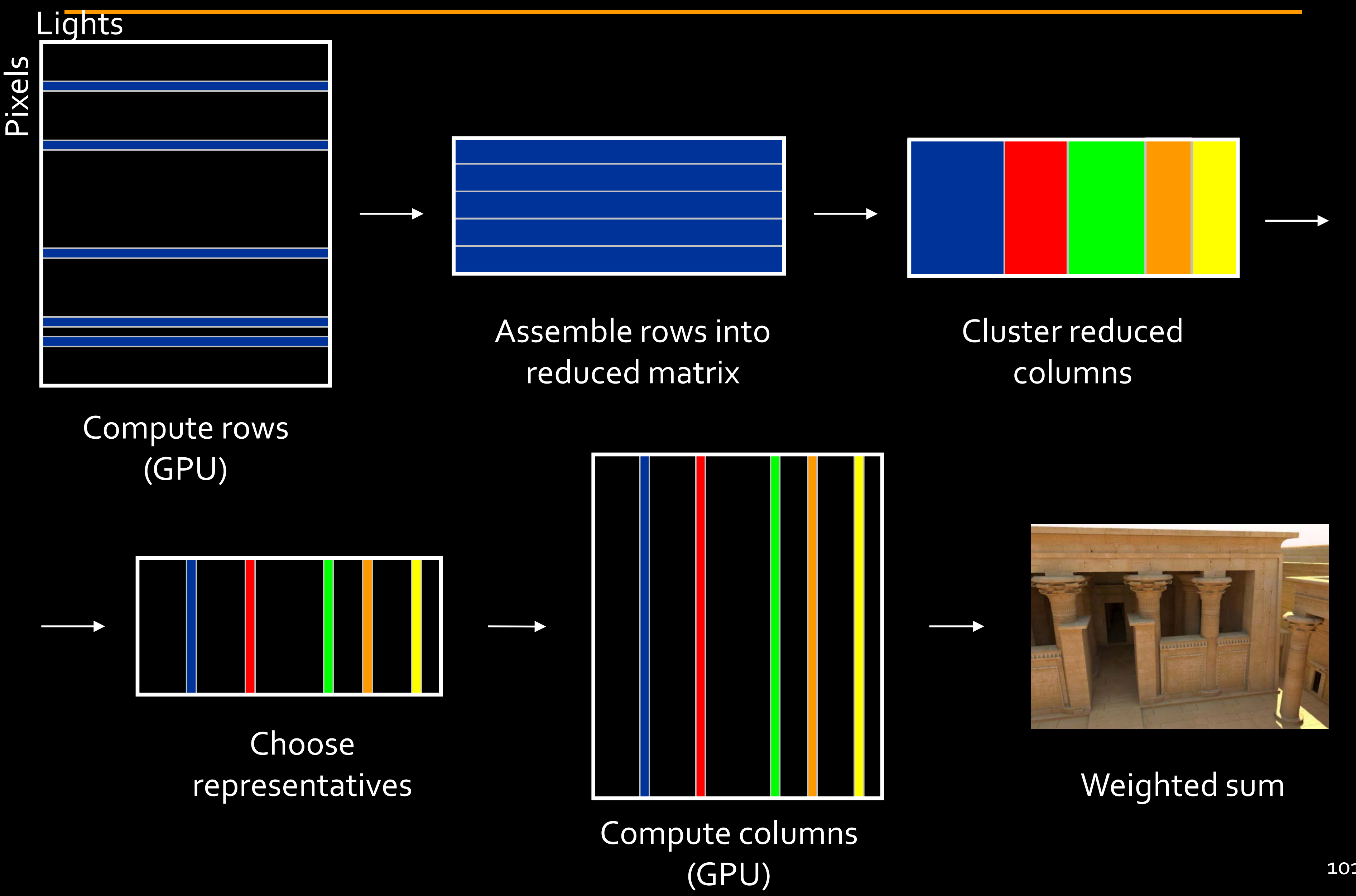
Combined Clustering Algorithm



Combined Clustering Algorithm



Full Algorithm



Example: Temple

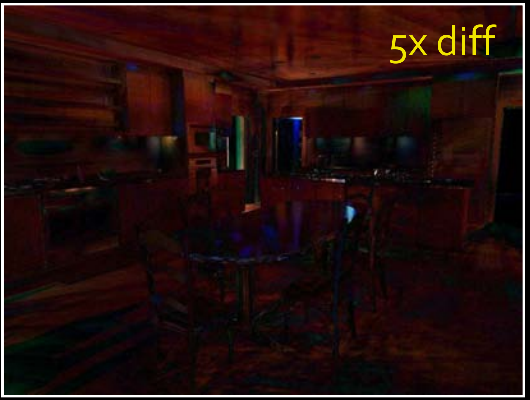
- 2.1m polygons
- Mostly indirect & sky illumination
- Indirect shadows

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

Reference: 20 min
(using all 100k lights)

Example: Kitchen

- 388k polygons
- Mostly indirect illumination
- Glossy surfaces
- Indirect shadows

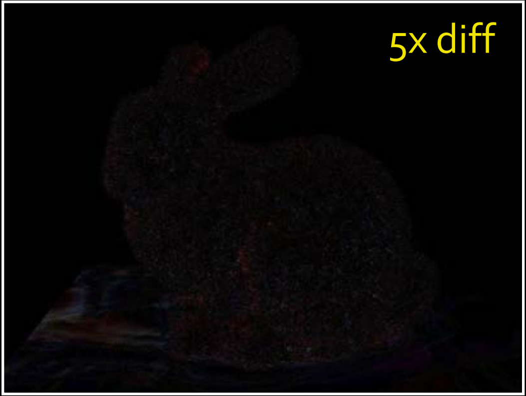


Our result: 13.5 sec
(432 rows + 864 columns)

Reference: 13 min
(using all 100k lights)

Example: Bunny

- 869k polygons
- Incoherent geometry
- High-frequency lighting
- Kajiya-Kay hair shader



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

Reference: 10 min
(using all 100k lights)

Effect of exploration

#VPLs = 250k, # cols = 10k

rows = 300
28 sec

Too few VPLs

rows = 900
35 sec

Comparison

Lightcuts (2M VPLs)
30 sec

MRCS (250k VPLs)
35 sec

Why does it work so well?

- Data-driven stratification & importance sampling
  - Same reason as for Lightcuts
- Stratification
  - Split clusters with dissimilar lights
- Importance sampling
  - Split clusters with high-contribution lights

Comparison to Lightcuts

- Advantage
  - Takes visibility into account in light selection (reduced matrix contains the full light contributions)
- Drawback
  - Impossible to capture localized effects
    - Low likelihood of getting the right row sample
    - Global light selection

LightSlice

LightSlice: Matrix Slice Sampling for the Many-Lights Problem

Jiawei Ou*
Dartmouth College

Fabio Pellacini†
Dartmouth College

SIGGRAPH ASIA 2011 HONG KONG

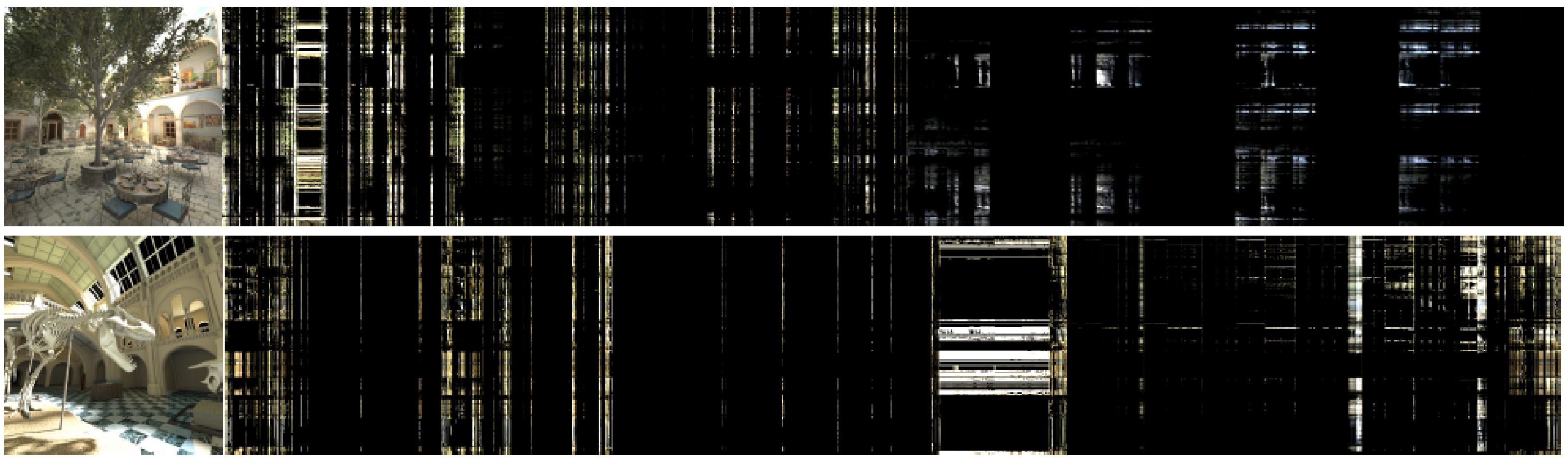


Figure 1: *The light transport matrices of two complex scenes (subsampled from the original). Note the existence of repeating patterns and large areas of near black in the matrices. Our algorithm,* LightSlice*, effectively exploits these typical structures of light transport matrices, and efficiently solves the many-lights problem by seeking locally optimized light clustering for each slice of the light trasnport matrix.*

http://www.cs.dartmouth.edu/~fabio/publication.php?id=lightslice11

LightSlice: Idea

- Get the better from Lightcuts and MRCS
  - Lightcuts: Localized selection of relevant lights
  - MRCS: Take visibility into account in light selection

LightSlice: The Algorithm

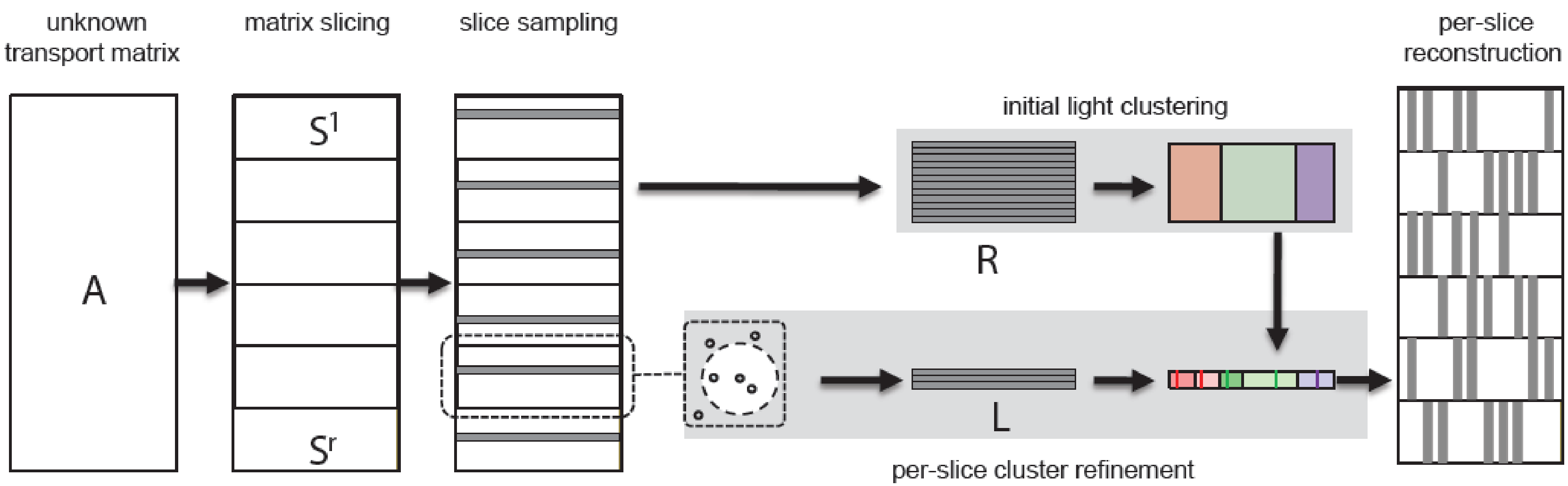
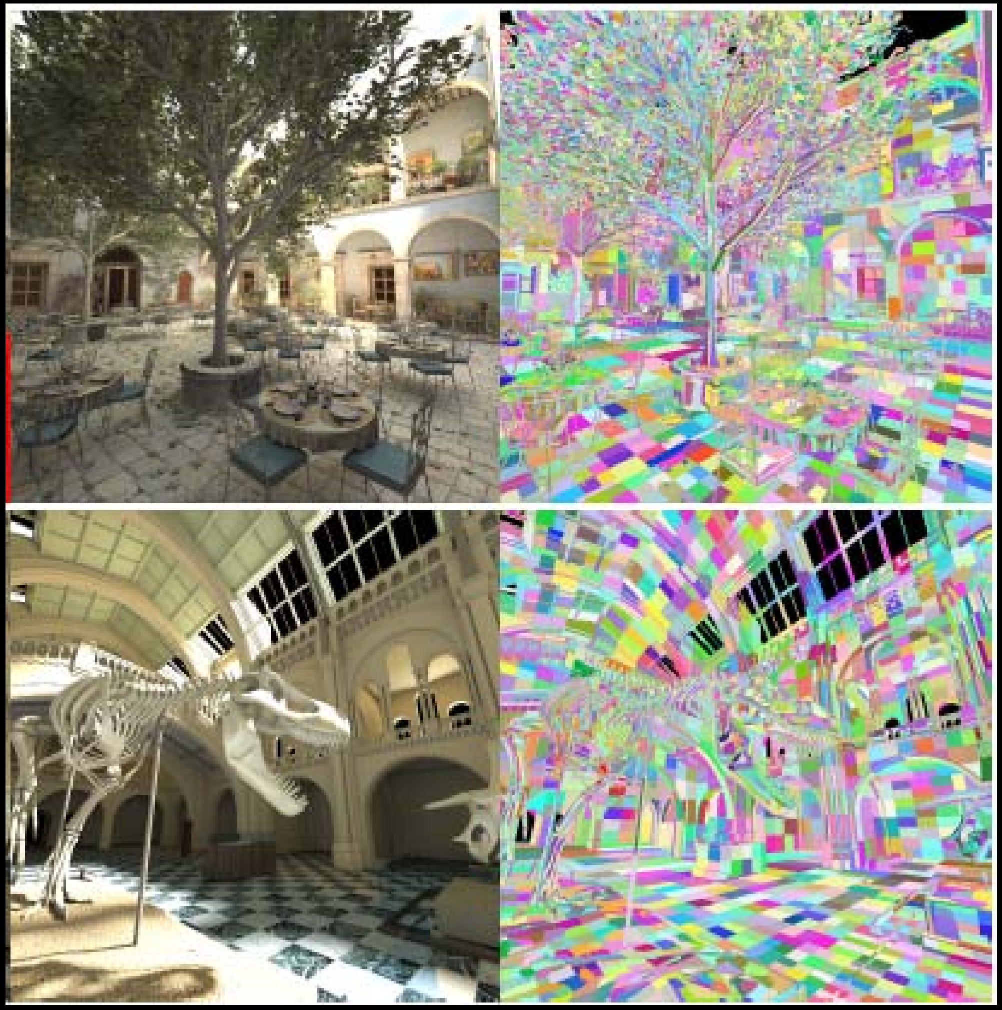


Figure 2: *Algorithm overview: starting from an unknown light transport matrix, first we determine matrix slices by clustering surface samples based on the geometric proximity. For each of these slices, a representative sample point is chosen and the corresponding row of A is computed. These sampled rows form a reduced matrix R on which an initial light clustering is performed to capture the global structure of A. For each slice, we then refine the initial light clusters based on the neighboring slices to effectively capture local lighting effects. Finally, we render each slice by choosing representative columns (lights).*

- Cannot use shadow maps for row/column sampling
- Clustering borrowed from the original MRCS paper

LightSlice: Matrix Slices

Image | **Matrix slice visualization**



LightSlice: Results

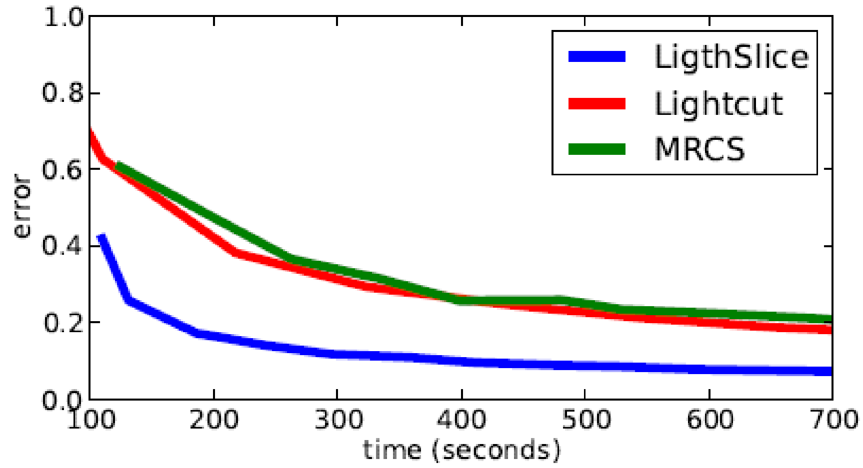


Figure 4: *Average relative error vs. time plot for each algorithm rendering the Sanmiguel scene. The images are rendered using the same number of rows and slices as reported in Table 2 while varying the number of columns or maximum cut size. The result shows that LightSlice is able to reduce error quicker than the other two VPL methods.*

- Not really a fair comparison to MRCS (uses ray traced visibility)

Importance Caching

EUROGRAPHICS 2012 / P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (2012), Number 2

Importance Caching for Complex Illumination

Iliyan Georgiev[†1] Jaroslav Křivánek[‡2] Stefan Popov[†1] Philipp Slusallek[†1,3]

[1]Saarland University and Intel VCI, Saarbrücken [2]Charles University, Prague [3]DFKI, Saarbrücken

- Localized light (VPL) selection taking visibility into account

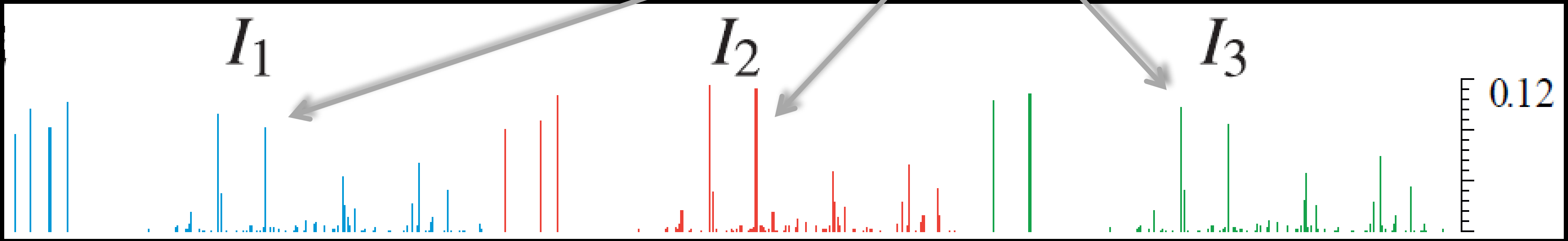
Importance Caching: Algorithm

1. Importance caching
   - Pick image pixels (2000 - 3000)
   - For each such pixel, compute and cache contributions from all lights (i.e. matrix row)

2. Image rendering
   - For each image pixel
     - Collect nearby importance records
     - Pick a VPL proportional to the cached contributions

Cached light importance

Contributions to
the record location
of individual lights



Importance record 1 Importance record 2 Importance record 3

Possible problems

OK | **Occlusion** | **Geometry factor** | **Irrelevant**

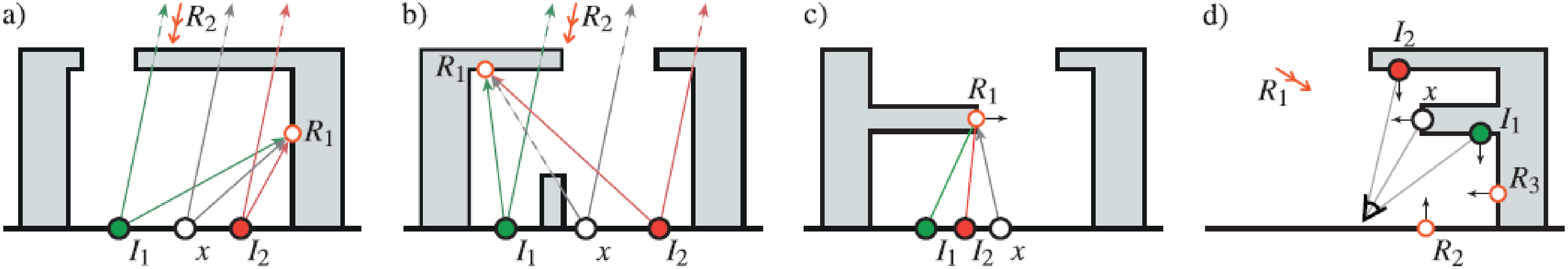
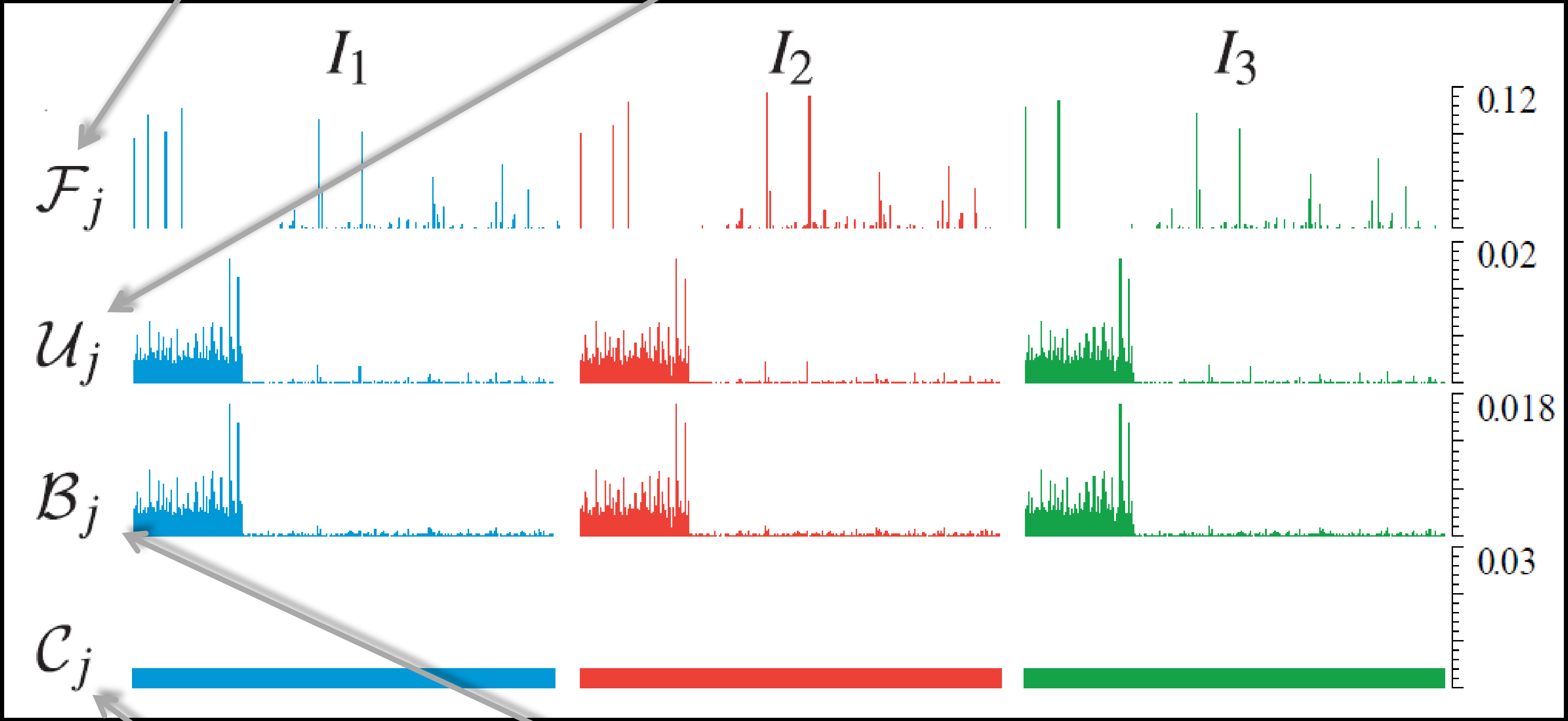


Figure 2: *Four illumination conditions, encountered when reusing information from importance records (IRs) I_1 and I_2 at shading point x. At each IR we define four distributions, designed to discover VPL contributions under a different condition. a) In the case of smooth illumination in the local neighborhood, full contribution sampling ($\mathcal{F}$) can achieve close proportionality to the integrand. b) Unoccluded contribution sampling ($\mathcal{U}$) is robust to VPL contribution changes due to varying occlusion with position. c) Bounded contribution sampling ($\mathcal{B}$) in addition discovers new contributions due to orientation changes. d) Conservative uniform sampling ($\mathcal{C}$) handles situations where the IR importance information is irrelevant at the shading point x.*

Solution: Multiple importances

Full light contribution

Unoccluded contribution



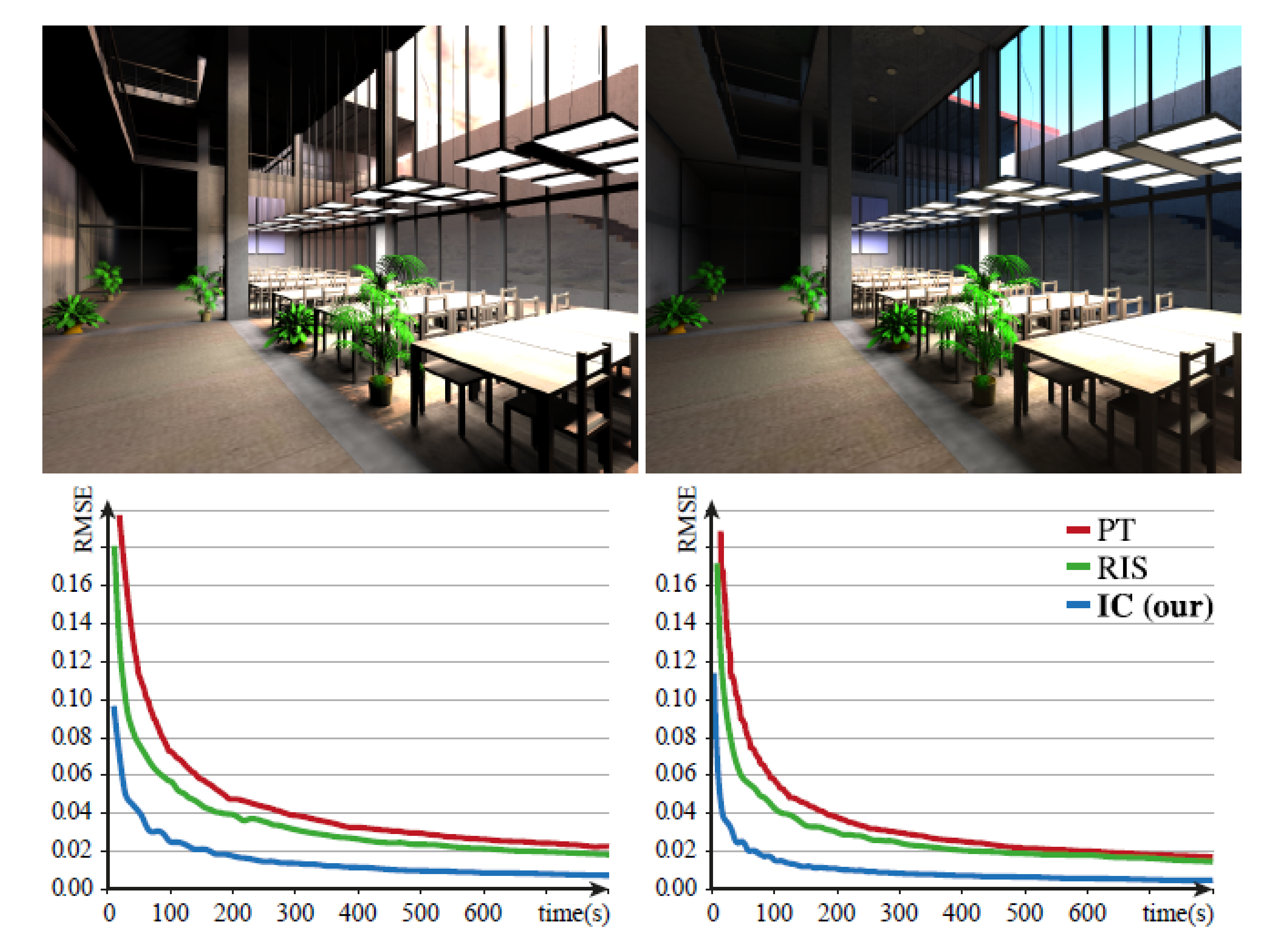
Uniform distribution

Unoccluded contribution w/ bounded G-term

Distribution combination

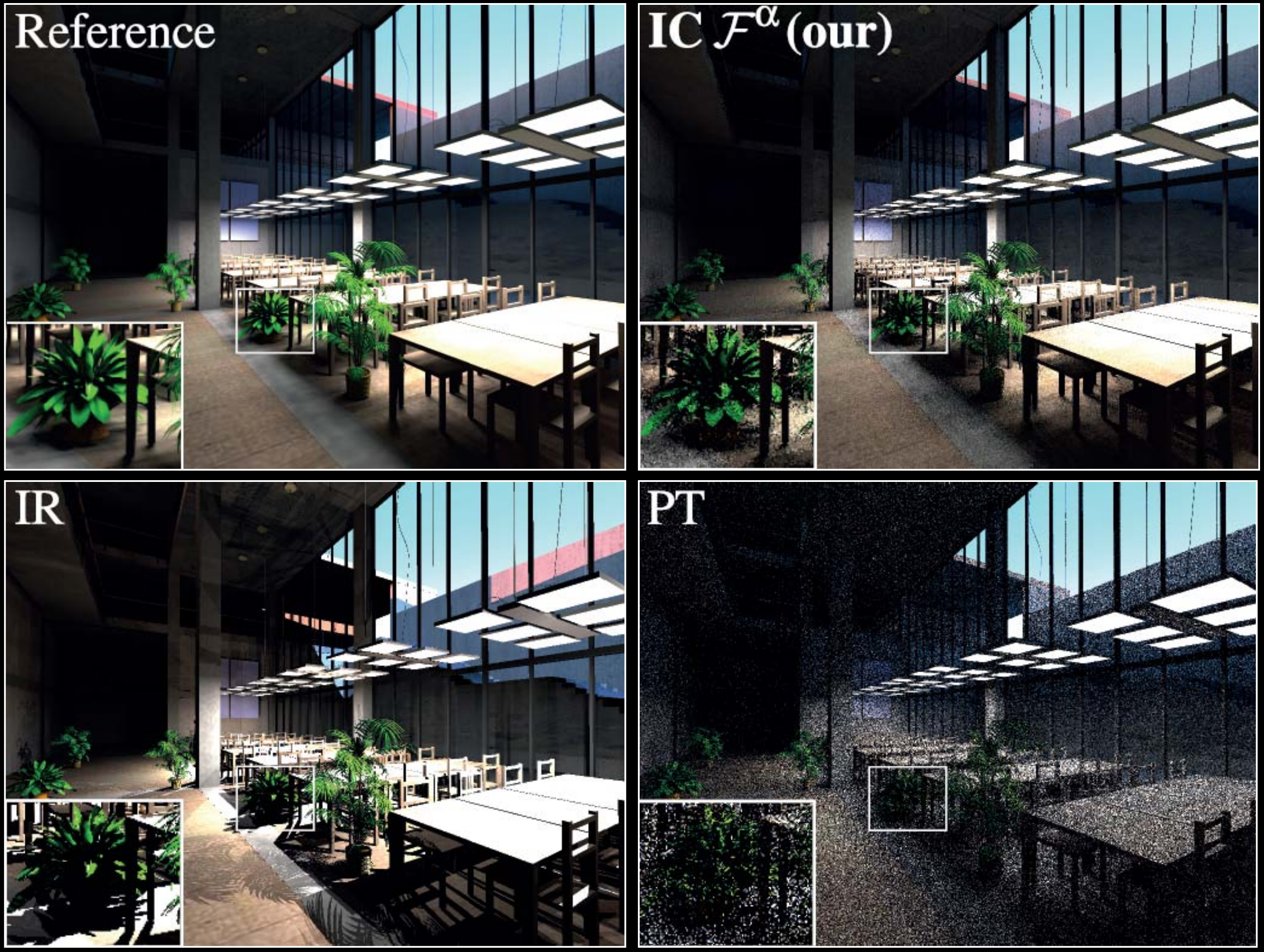
- Multiple importance sampling (MIS)
  - See CG III slides
    http://cgg.mff.cuni.cz/~jaroslav/teaching/2011-pg3/slides/krivanek-07-npgr010-2011-mc2.pptx
- New combination heuristic proposed in the paper

Results



RMSE
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
100
200
300
400
500
600
time(s)
PT
RIS
IC (our)

Results: A 2-second rendering



Importance Caching: Limitations

- Fewer initial VPLs than any of the previous algorithms (up to 10k)
  - Because of memory limitations
  - Must run in iterations if more VPLs needed

- No data-driven stratification

Take-home message

- Data-driven importance sampling may be **dangerous**
  - Non-zero contribution may be sampled with very low, or even zero probability
  - Being conservative is safer, but can reduce the advantage of data-driven IS
  - For more information see [Owen & Zhou 2000]
    http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.2813

How ‘bout Lightcuts and MRCS?

- MRCS
  - In theory suffers from the same problem, but
  - light importance averaged over full columns, so it works ok
  - BTW: Why do you think there’s global clustering in LightSlice ☺

- Lightcuts
  - Has upper bounds so it knows which lights can be safely skipped

Scalable many-light methods

- Lightcuts [Walter et al 05/06]
- Matrix Row Column Sampling [Hašan et al. 07]
- LightSlice [Ou & Pellacini 2011]
- Importance caching [Georgiev et al. 2012]

Improved VPL distribution

- [Segovia et al. 07]: Metropolis instant radiosity
  - http://www710.univ-lyon1.fr/~jciehl/bsegovia/bat710/public_html/papers/mir.html
    Use Metropolis sampling to guide the VPL where they can contribute to the image

- [Georgiev & Slussalek 2010]
  - http://www.iliyan.com/p/publications.html
  - Use rejection sampling to reject VPLs that do not contribute significantly