Pre-computed Radiance Transfer II

Jaroslav Křivánek, KSVI, MFF UK

Jaroslav.Krivanek@mff.cuni.cz

Acknowledgement

- Mostly based on Ravi Ramamoorthi's slides available from http://inst.eecs.berkeley.edu/~cs283/fa10

Goal

- Real-time rendering with complex lighting, shadows, and possibly GI
- Infeasible – too much computation for too small a time budget

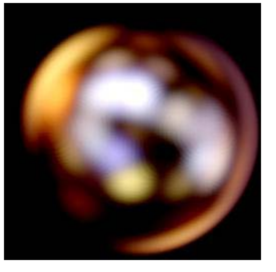
- Approaches
  - Lift some requirements, do specific-purpose tricks
    - Environment mapping, irradiance environment maps
    - SH-based lighting
  - Split the effort
    - Offline pre-computation + real-time image synthesis
    - “Pre-computed radiance transfer”

Assumptions

- Distant illumination
- No shadowing, interreflection

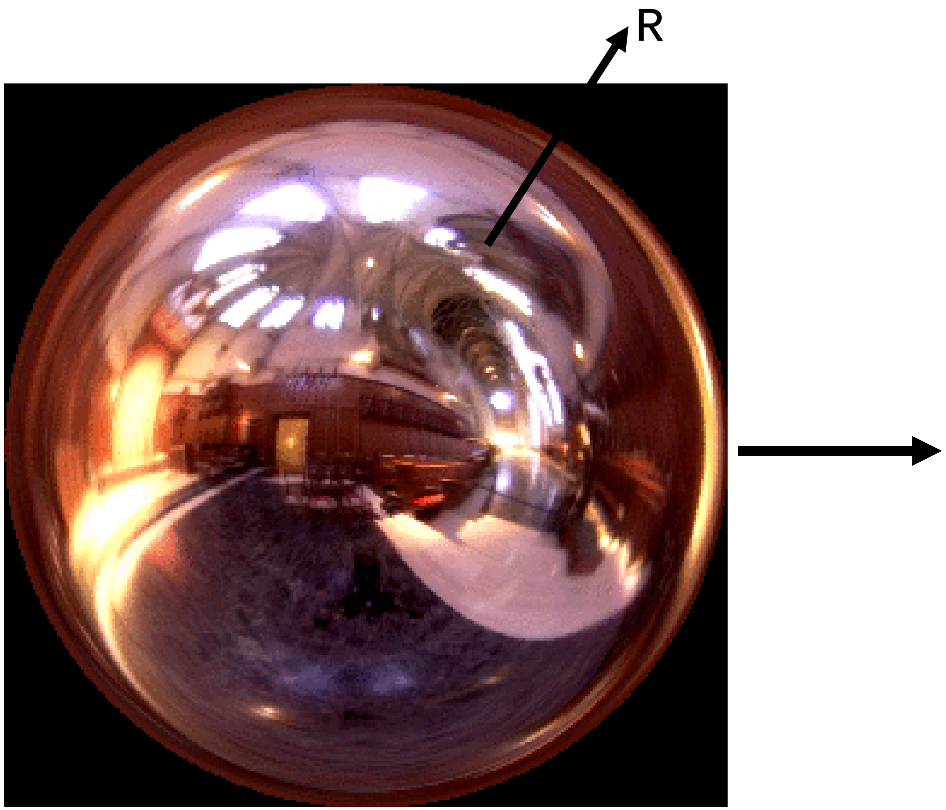
- Mirror surfaces easy
(just a texture look-up)

- What if the surface is rougher...

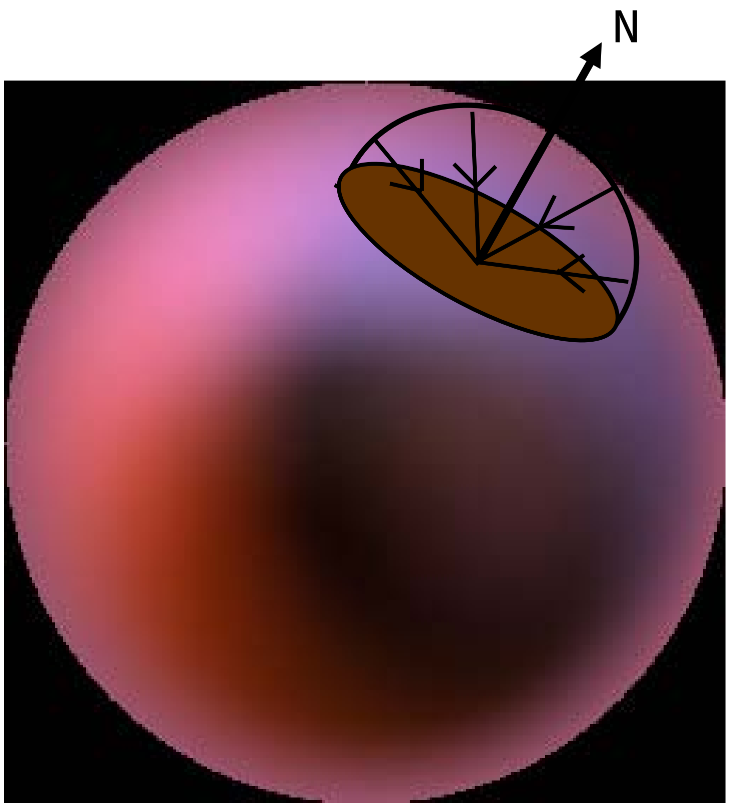


- Or completely diffuse?

SH-based Irradiance Env. Maps



Incident Radiance
(Illumination Environment Map)



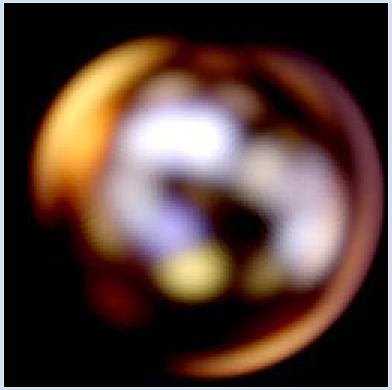
Irradiance Environment Map

Reflection Maps

- Phong model for rough surfaces
  - Illumination function of reflection direction R
- Lambertian diffuse surface
  - Illumination function of surface normal N

Matte Sphere

Chrome Sphere



- Reflection Maps [Miller and Hoffman, 1984]
  - Irradiance (indexed by N) and Phong (indexed by R)

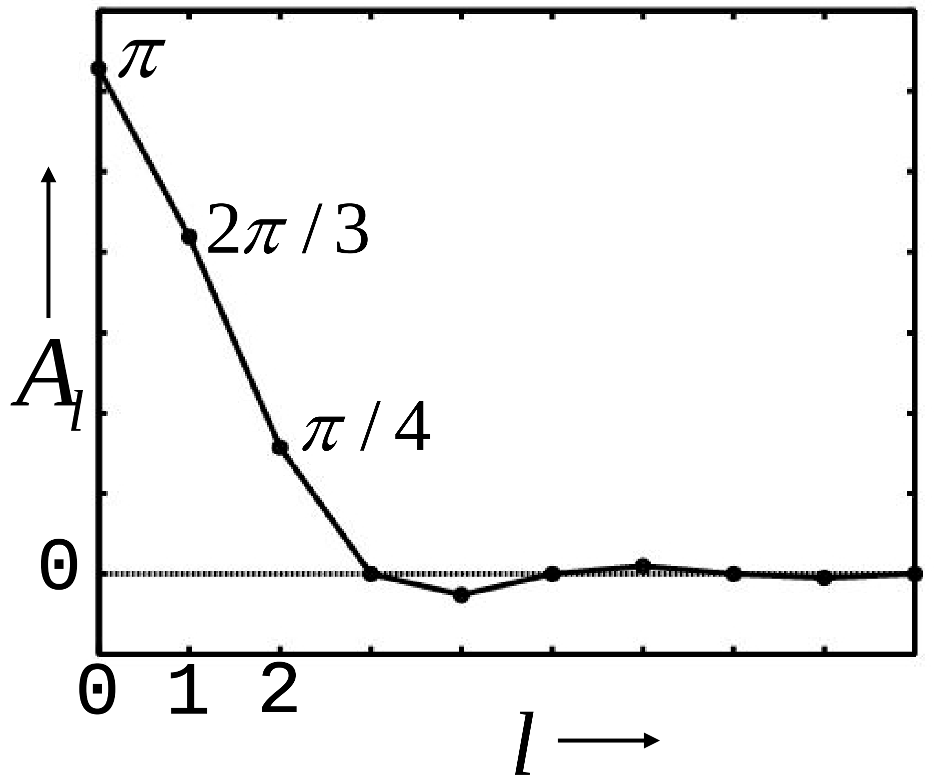
Reflection Maps

- Can’t do dynamic lighting
  - Slow blurring in pre-process

Analytic Irradiance Formula

Lambertian surface acts like low-pass filter

$$E_{lm} = A_l L_{lm}$$



Ramamoorthi and Hanrahan 01
Basri and Jacobs 01

$$A_l = 2\pi \frac{(-1)^{\frac{l}{2}-1}}{(l+2)(l-1)} \left[\frac{l!}{2^l \left(\frac{l}{2}!\right)^2} \right] \quad l \text{ even}$$

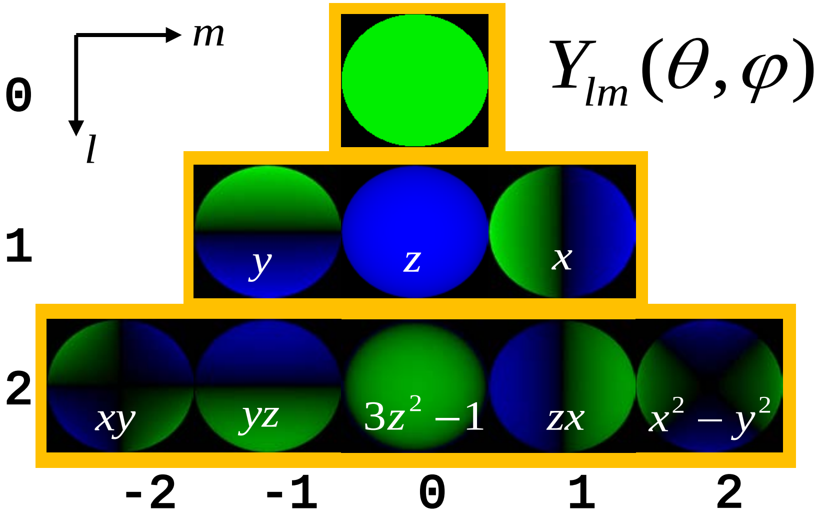
9 Parameter Approximation

Exact image

Order 2
9 terms

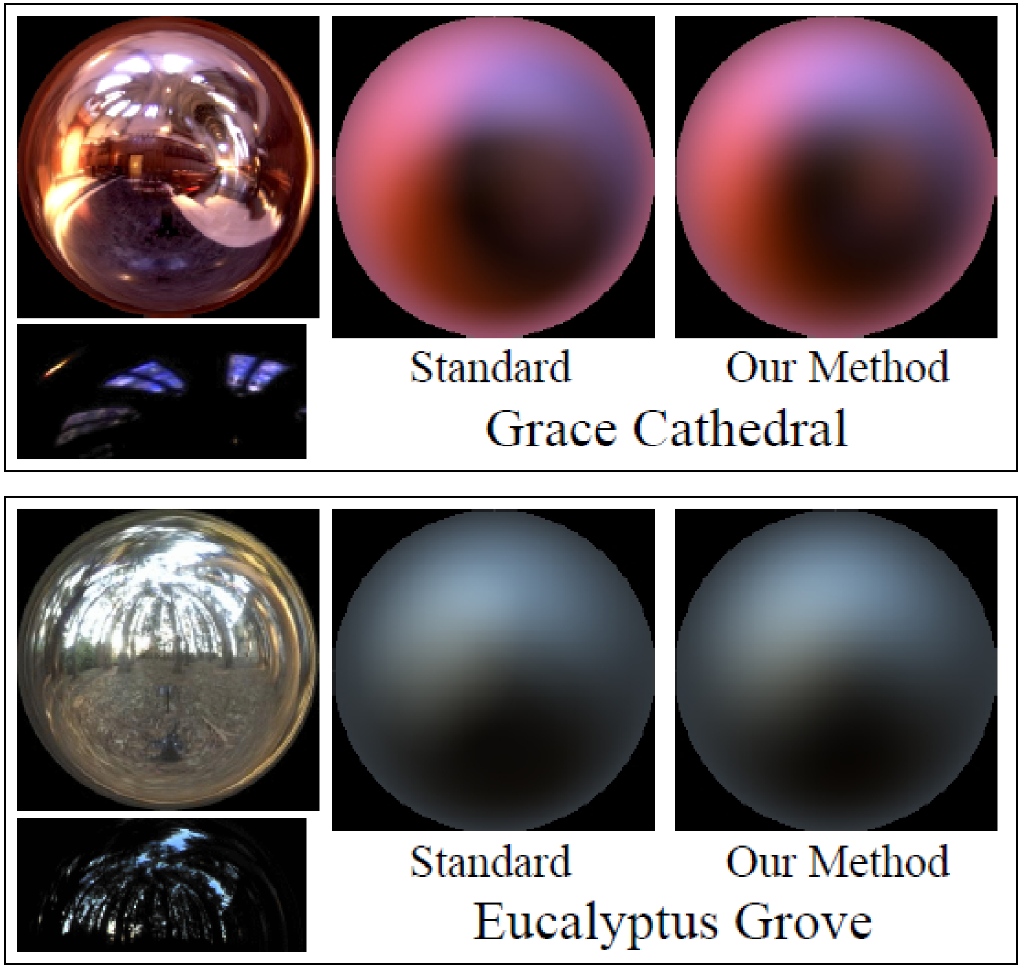
RMS Error = 1%

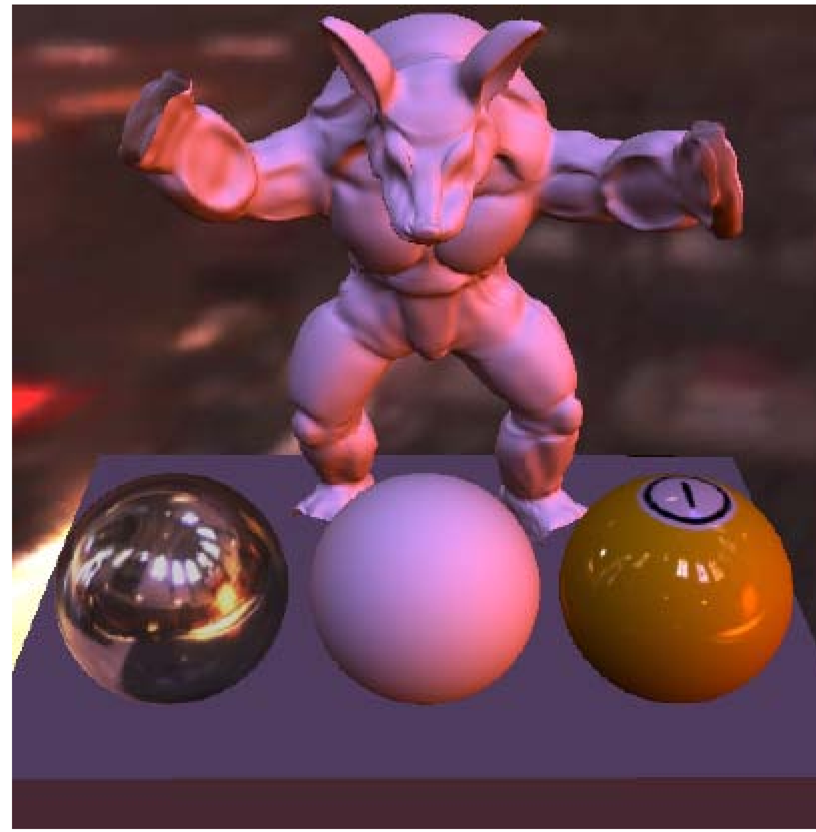
For any illumination, average error < 3% [Basri Jacobs 01]



$Y_{lm}(\theta,\varphi)$

SH-based Irradiance Env. Maps





Images courtesy Ravi Ramamoorthi & Pat Hanrahan

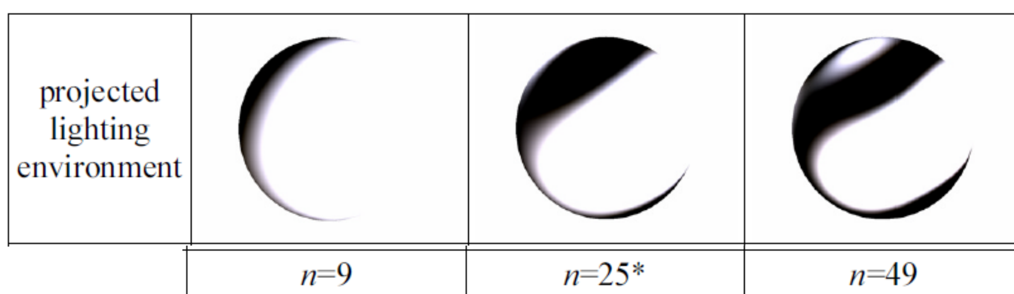
SH-based Arbitrary BRDF Shading 1

- [Kautz et al. 2003]
- Arbitrary, dynamic env. map
- Arbitrary BRDF
- No shadows

(a) point light (b) glossy (c) anisotropic

- SH representation
  - Environment map (one set of coefficients)
  - Scene BRDFs (one coefficient vector for each discretized view direction)

original lighting environment

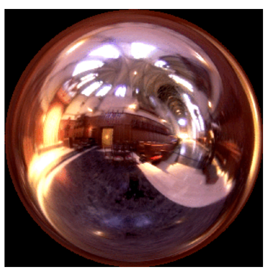


SH-based Arbitrary BRDF Shading 3

- Rendering: for each vertex / pixel, do

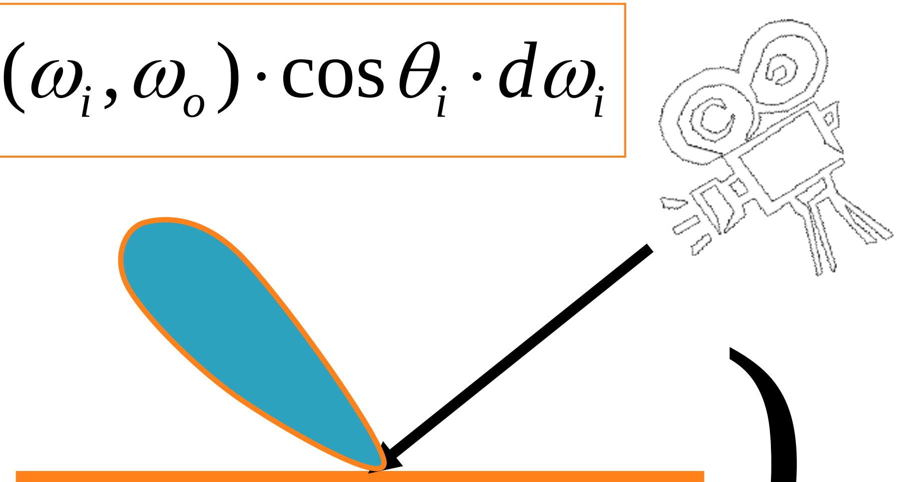
$$L_o(\omega_o) = \int_\Omega L_i(\omega_i) \cdot BRDF(\omega_i, \omega_o) \cdot \cos\theta_i \cdot d\omega_i$$

$\int ($



Environment map

$\bullet$



BRDF

$)$

= coeff. dot product

$$L_o(\omega_o) = \Lambda_{\text{intp}}(\mathbf{p}) \bullet F(\mathbf{p}, \omega_o)$$

SH-based Arbitrary BRDF Shading 4

- BRDF is in local frame
- Environment map in global frame
- Need coordinate frame alignment -> **SH rotation**

- SH closed under rotation
  - rotation matrix
  - Fastest known procedure is the *zxzxz*-decomposition [Kautz et al. 2003]

$$R_{SH} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\ 0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

SH-based Arbitrary BRDF Shading 5

Figure 3: *Brushed metal head in various lighting environments.*

(a) varying exponent *(b) varying anisotropy*

Figure 4: *Spatially-Varying BRDFs.*

Pre-computed Radiance Transfer

Pre-computed Radiance Transfer

- Goal
  - Real-time rendering with complex lighting, shadows, and GI
  - Infeasible – too much computation for too small a time budget
- Approach
  - Precompute (offline) some information (images) of interest
  - Must assume something about scene is constant to do so
  - Thereafter real-time rendering. Often hardware accelerated

Assumptions

- Precomputation
- Static geometry
- Static viewpoint (some techniques)

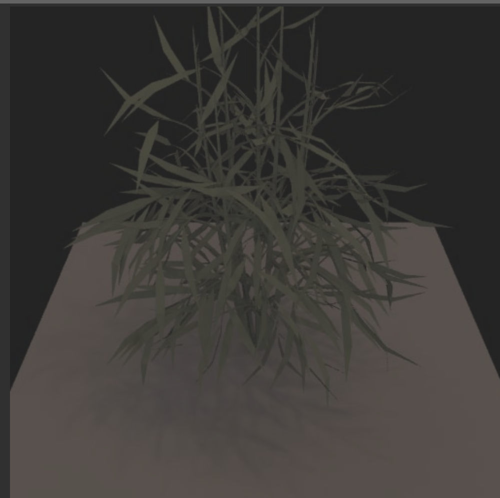
- Real-Time Rendering (relighting)
  - Exploit linearity of light transport

Relighting as a Matrix-Vector Multiply

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_N \end{bmatrix}$$

$$= \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_M \end{bmatrix}$$

Relighting as a Matrix-Vector Multiply



Output Image
(Pixel Vector)

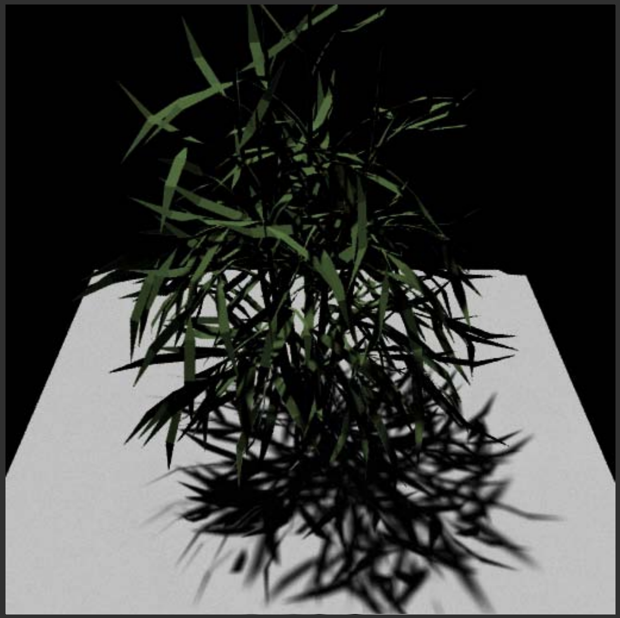
Input Lighting
(Cubemap Vector)

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_N \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_M \end{bmatrix}$$

Precomputed
Transport
Matrix

Matrix Columns (Images)

$$\begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix}$$



Problem Definition

Matrix is Enormous

- 512 x 512 pixel images
- 6 x 64 x 64 cubemap environments

Full matrix-vector multiplication is intractable

- On the order of 10^{10} operations *per frame*

How to relight quickly?

Outline

- ***Compression methods***
  - ***Spherical harmonics-based PRT [Sloan et al. 02]***
  - (Local) factorization and PCA
  - Non-linear wavelet approximation
- Changing view as well as lighting
  - Clustered PCA
  - Triple Product Integrals
- Handling Local Lighting
  - Direct-to-Indirect Transfer

SH-based PRT

- Better light integration and transport
  - dynamic, env. lights
  - self-shadowing
  - interreflections
- For diffuse and glossy surfaces
- At real-time rates
- Sloan et al. 02

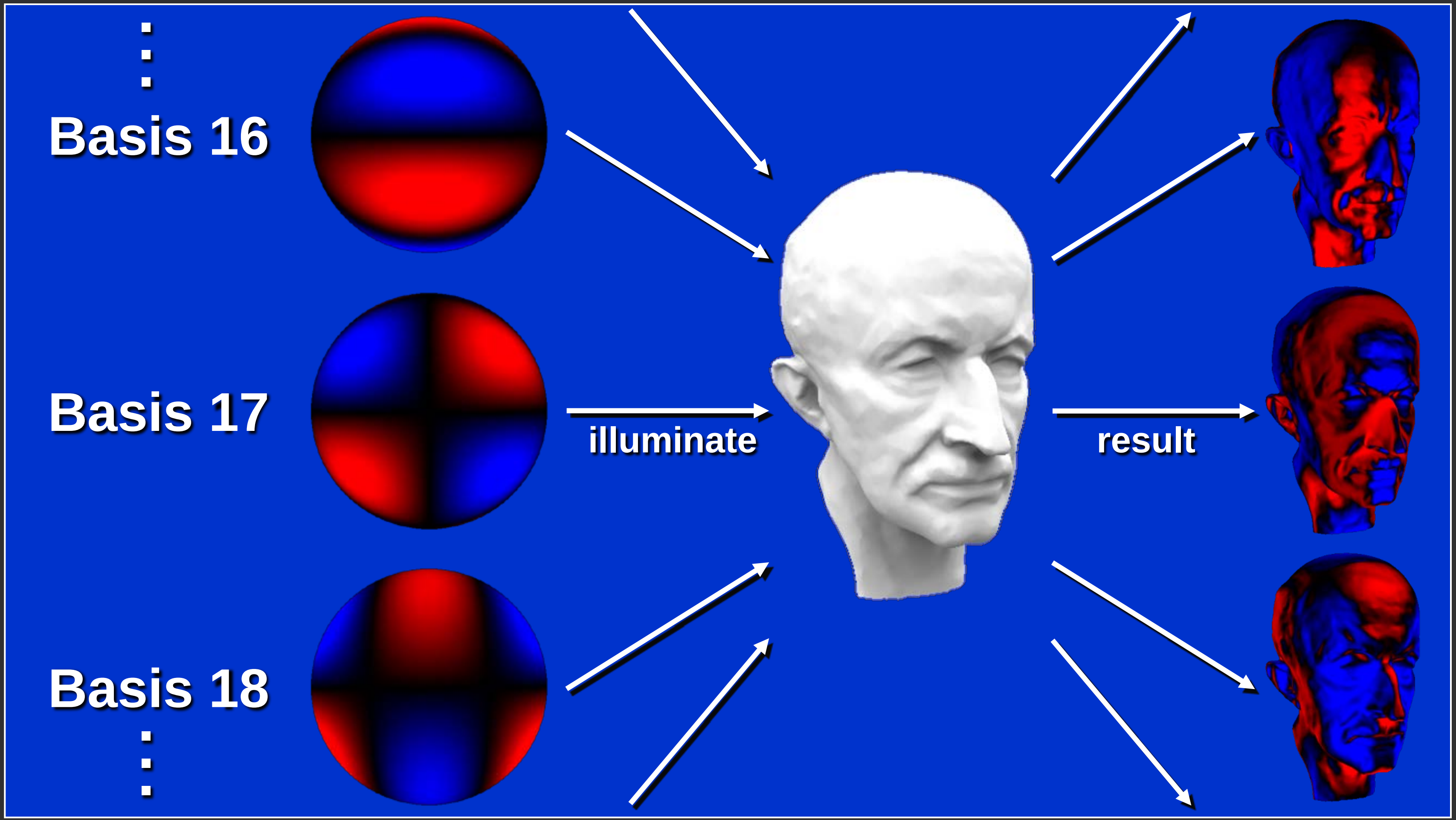
point light

Env. light

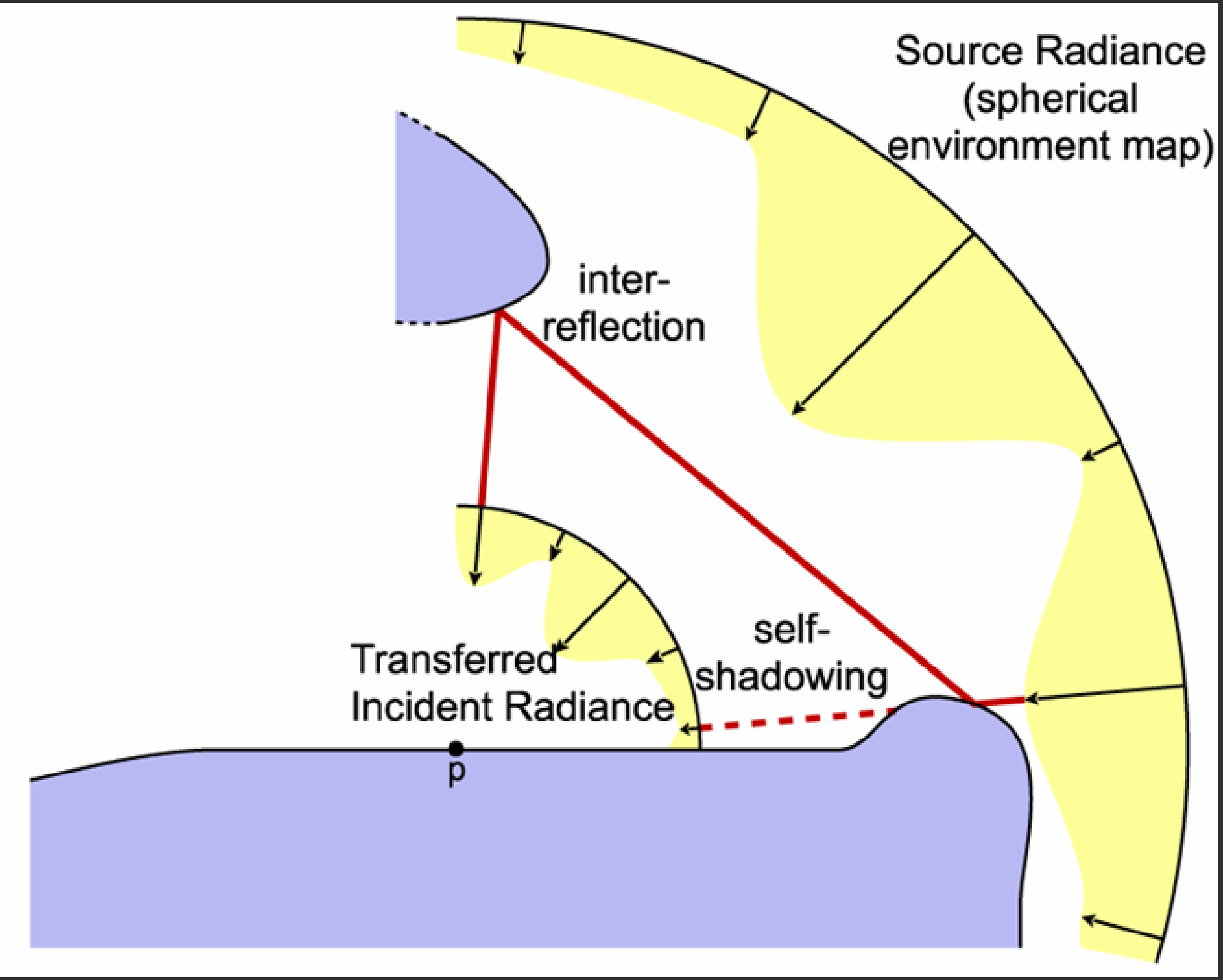
Env. lighting, no shadows

Env. lighting, shadows

SH-based PRT: Idea



PRT Terminology



Relation to a Matrix-Vector Multiply

a) SH coefficients of transferred radiance
b) Irradiance
(per vertex)

$$= \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_M \end{bmatrix}$$

SH coefficients of EM (source radiance)

Idea of SH-based PRT

- The *L* vector is projected onto low-frequency components (say 25). Size greatly reduced.
- Hence, only 25 matrix columns
- But each pixel/vertex still treated separately
  - One RGB value per pixel/vertex:
    - Diffuse shading / arbitrary BRDF shading w/ fixed view direction
  - SH coefficients of transferred radiance (25 RGB values per pixel/vertex for order 4 SH)
    - Arbitrary BRDF shading w/ variable view direction
- Good technique (becoming common in games) but useful only for broad low-frequency lighting

Diffuse Transfer Results

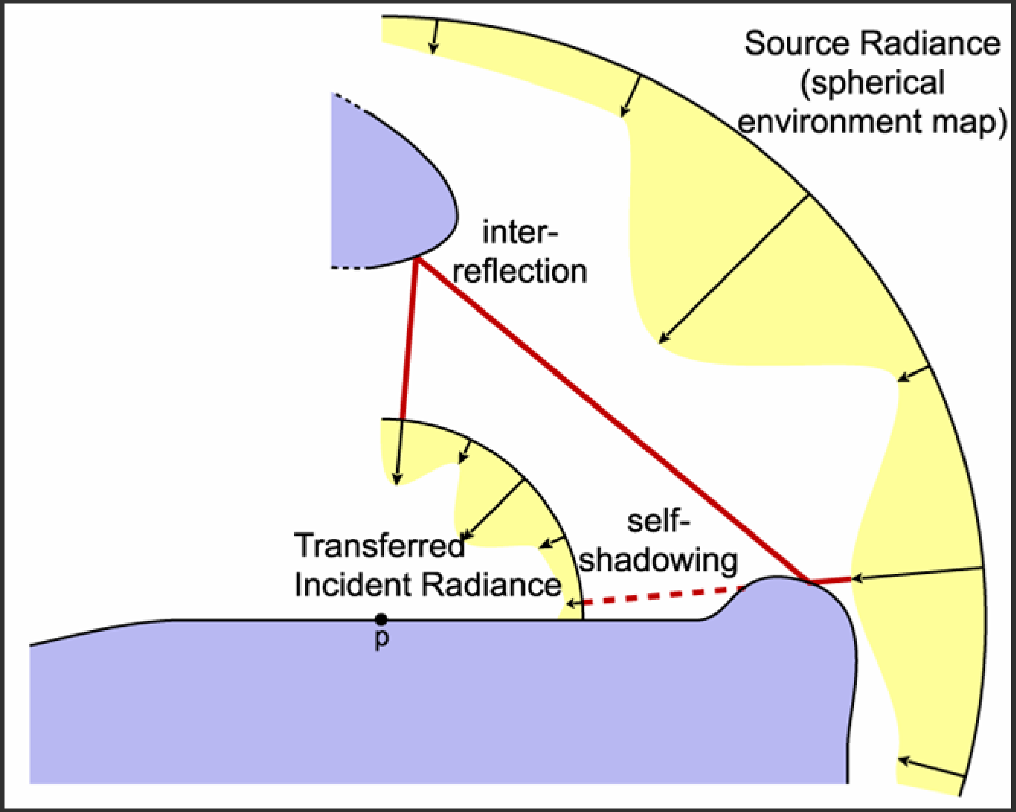
No Shadows/Inter

Shadows

Shadows+Inter

SH-based PRT with Arbitrary BRDFs

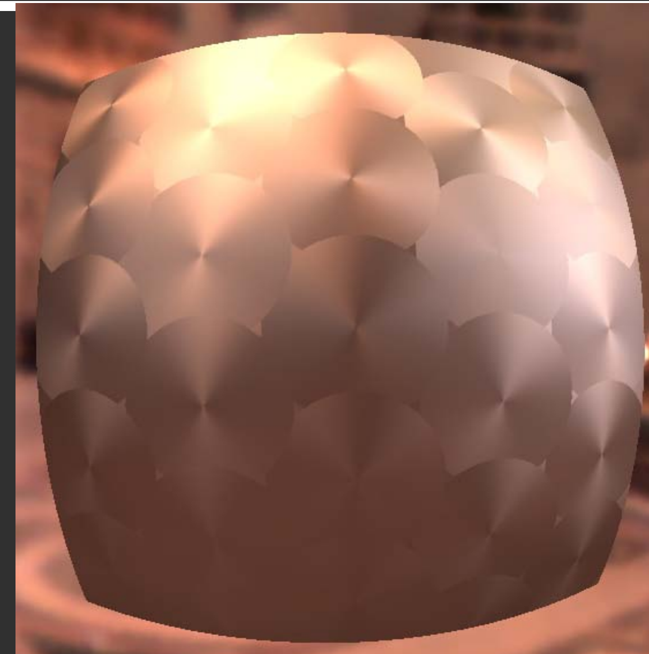
- Combine with Kautz et al. 03
- Transfer matrix turns SH env. map into SH transferred radiance
- Kautz et al. 03 is applied to transferred radiance



Arbitrary BRDF Results

Anisotropic BRDFs

Other BRDFs



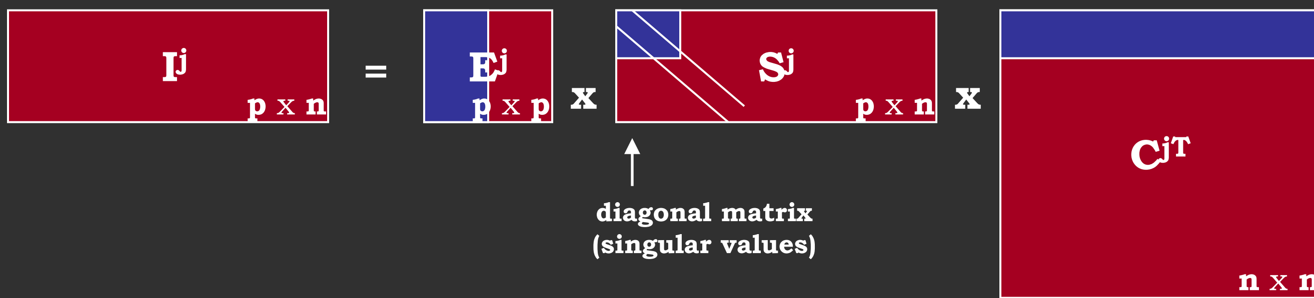
Spatially Varying

Outline

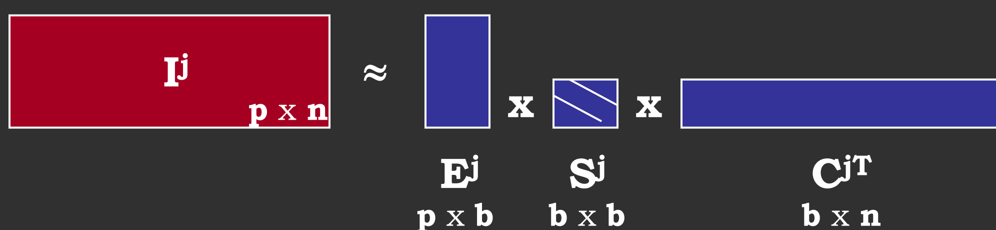
- ***Compression methods***
  - Spherical harmonics-based PRT [Sloan et al. 02]
  - ***(Local) factorization and PCA***
  - Non-linear wavelet approximation
- Changing view as well as lighting
  - Clustered PCA
  - Triple Product Integrals
- Handling Local Lighting
  - Direct-to-Indirect Transfer

PCA or SVD factorization

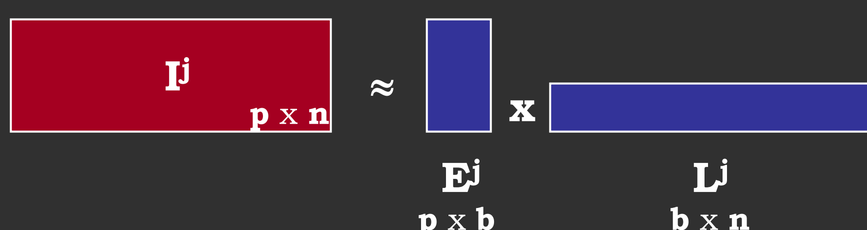
- SVD:



- Applying Rank **b**:



- Absorbing $\mathbf{S^j}$ values into $\mathbf{C^{iT}}$:



Idea of Compression

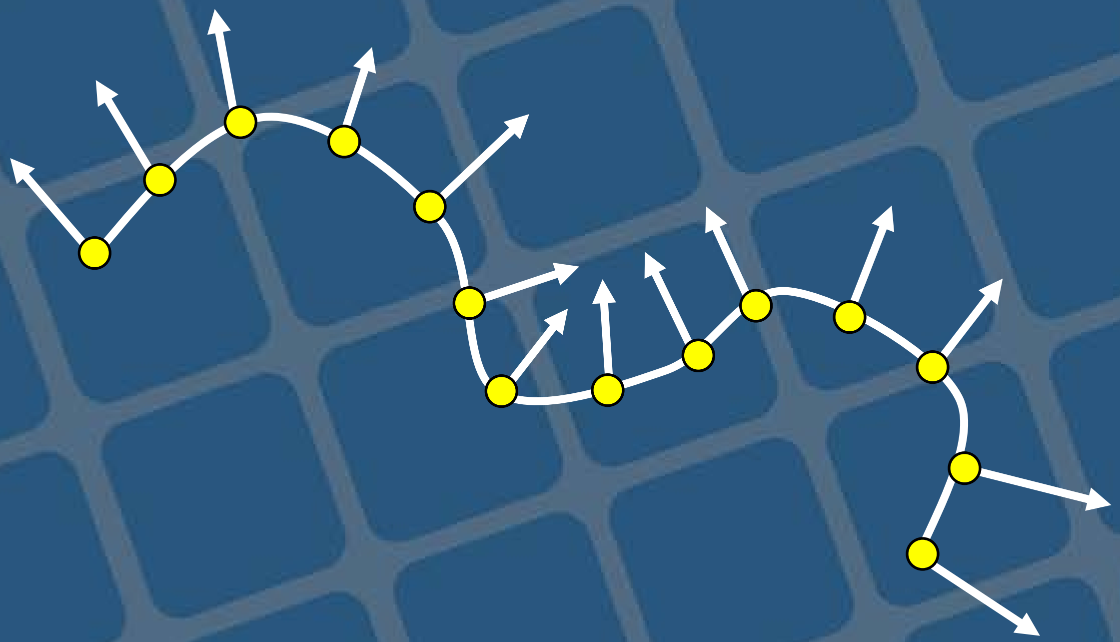
- Represent matrix (rather than light vector) compactly
- Can be (and is) combined with SH light vector
- Useful in broad contexts.
  - BRDF factorization for real-time rendering (reduce 4D BRDF to 2D texture maps) McCool et al. 01 etc
  - Surface Light field factorization for real-time rendering (4D to 2D maps) Chen et al. 02, Nishino et al. 01
  - BTF (Bidirectional Texture Function) compression
- **Not too useful for general precomput. relighting**
  - **Transport matrix not low-dimensional!!**

Local or Clustered PCA

- Exploit local coherence (in say 16x16 pixel blocks)
  - Idea: light transport is locally low-dimensional.
  - Even though globally complex
  - See Mahajan et al. 07 for theoretical analysis
- Clustered PCA [Sloan et al. 2003]
  - Combines two widely used compression techniques: Vector Quantization or VQ and Principal Component Analysis

Compression Example

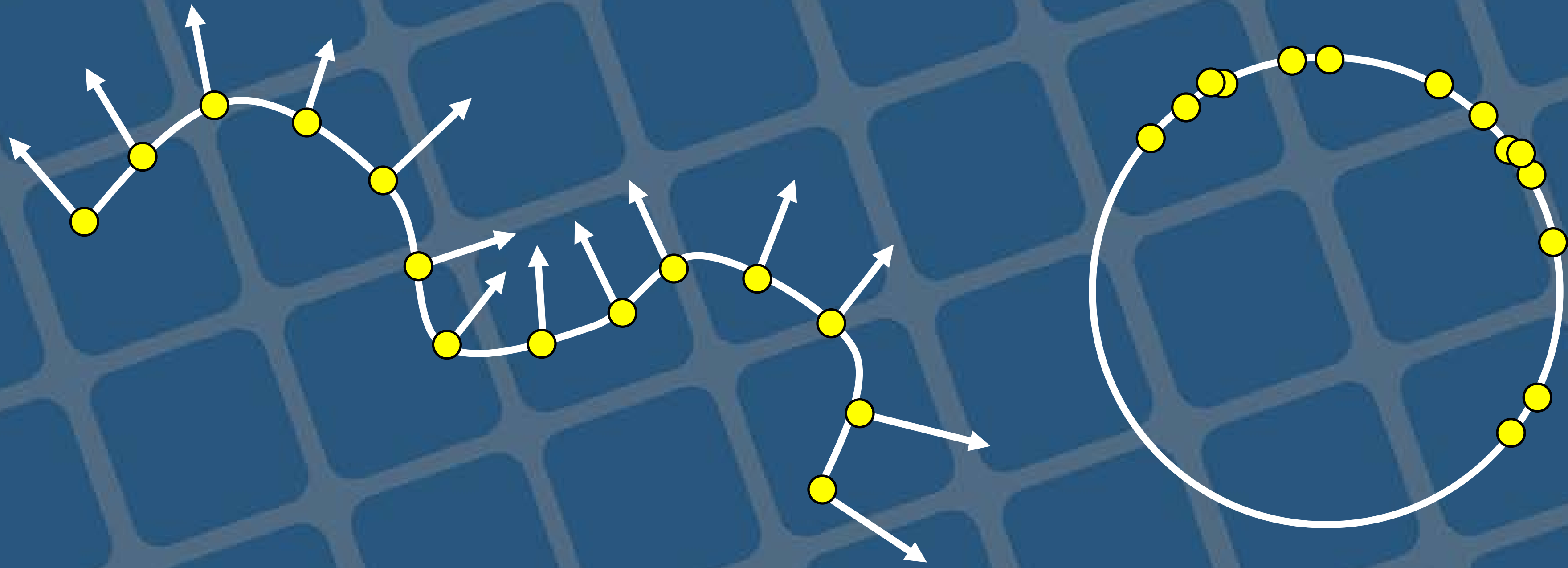
Surface is curve, signal is normal



Following couple of slides courtesy P.-P. Sloan

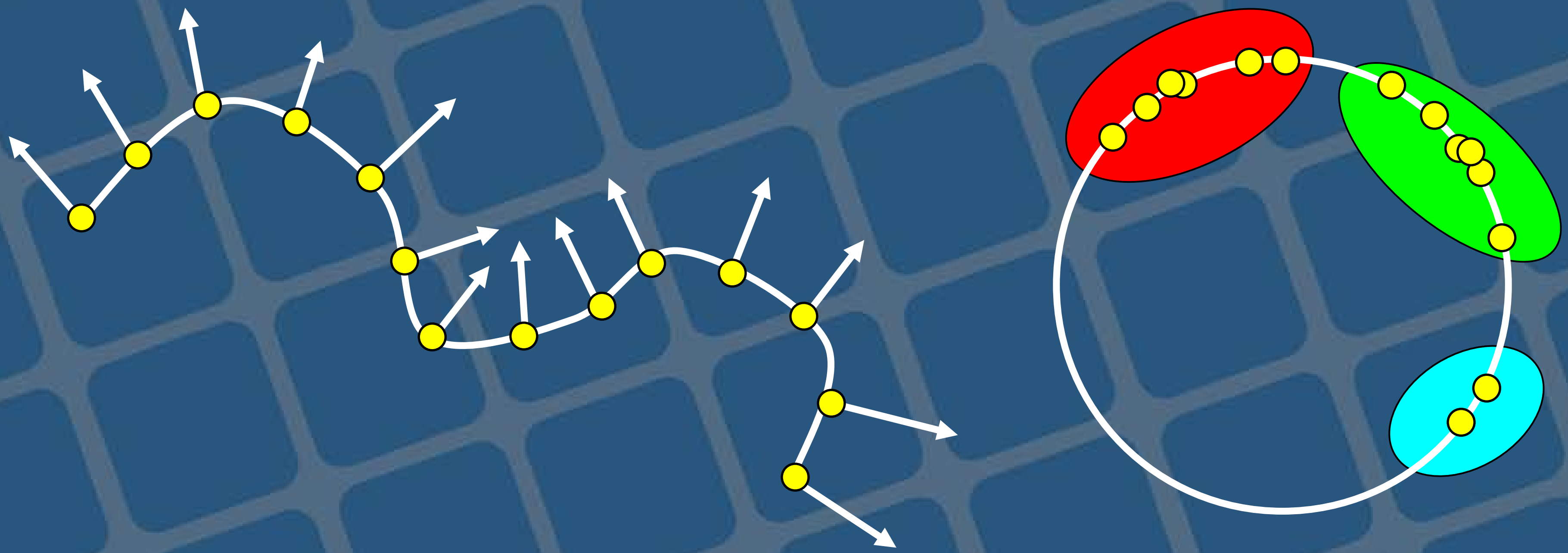
Compression Example

Signal Space



VQ

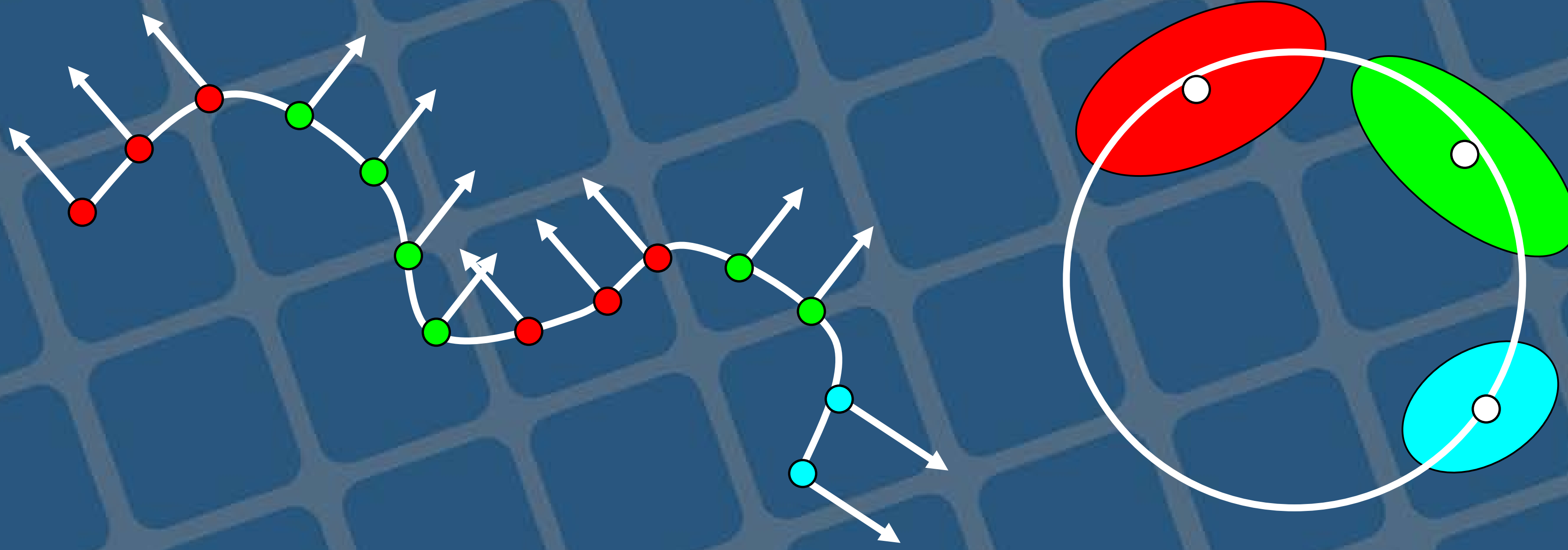
Cluster normals



VQ

Replace samples with cluster mean

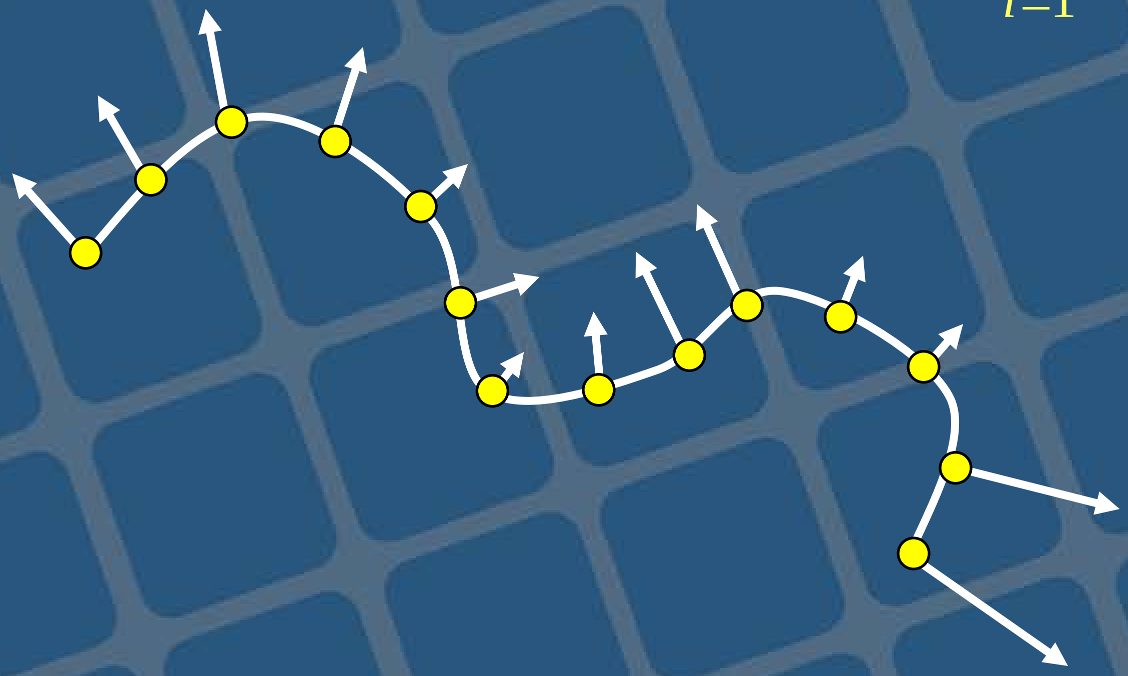
$$\mathbf{M}_p \approx \tilde{\mathbf{M}}_p = \mathbf{M}_{C_p}$$

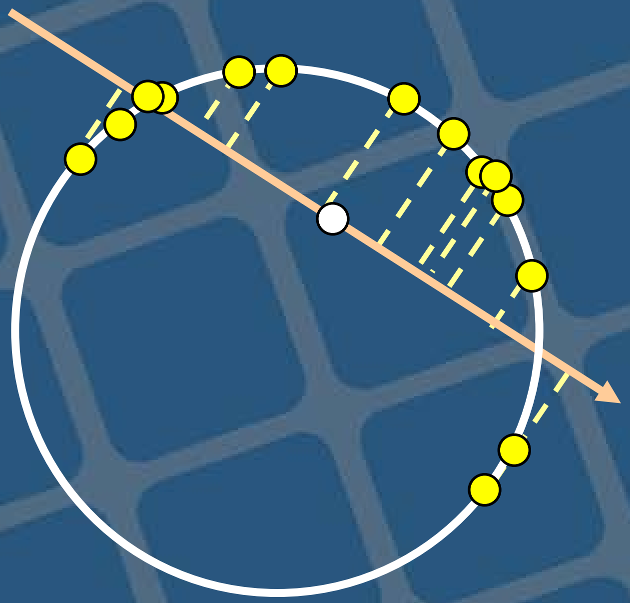


PCA

Replace samples with mean + linear combination

$$\mathbf{M}_p \approx \tilde{\mathbf{M}}_p = \mathbf{M}^0 + \sum_{i=1}^{N} w_p^i \mathbf{M}^i$$

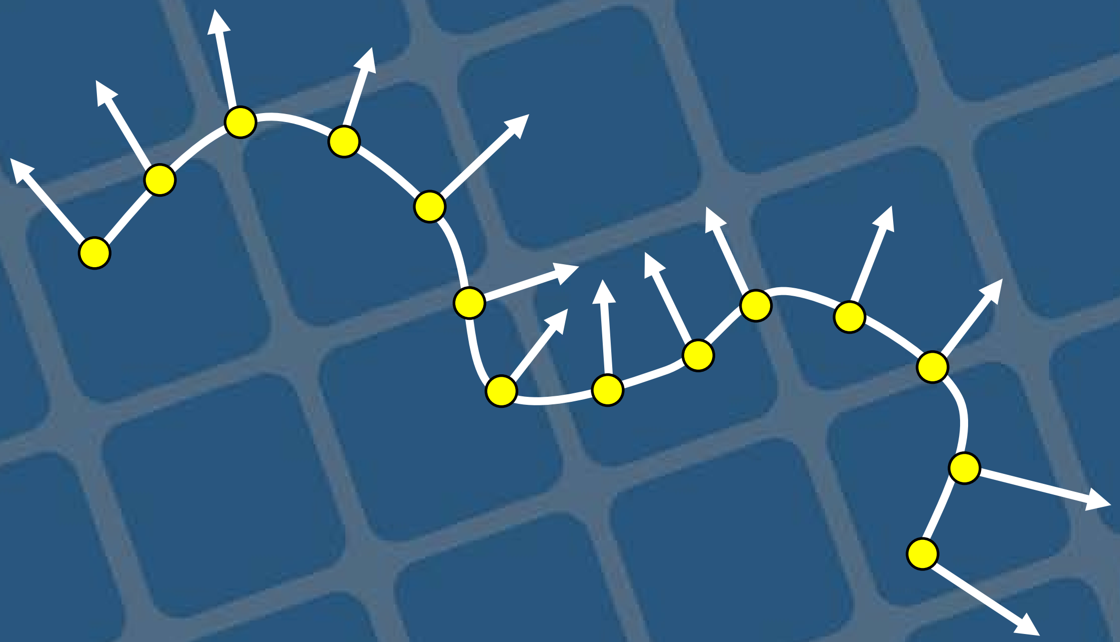


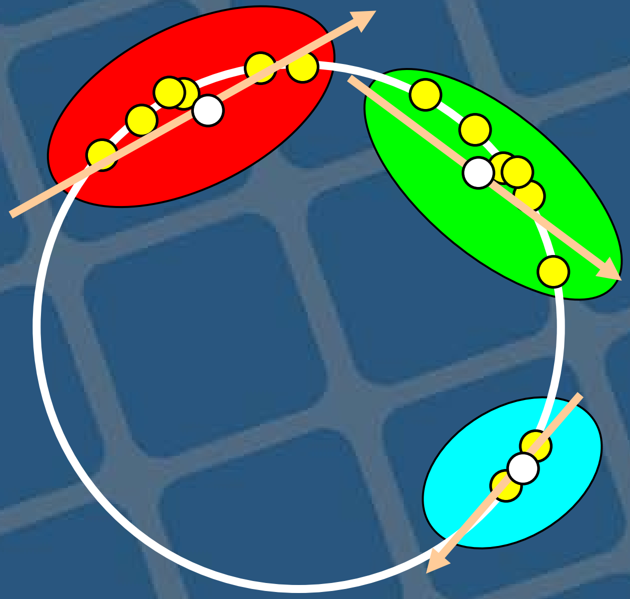


CPCA

Compute a linear subspace in each cluster

$$\mathbf{M}_p \approx \tilde{\mathbf{M}}_p = \mathbf{M}^0_{C_p} + \sum_{i=1}^{N} w^i_p \mathbf{M}^i_{C_p}$$

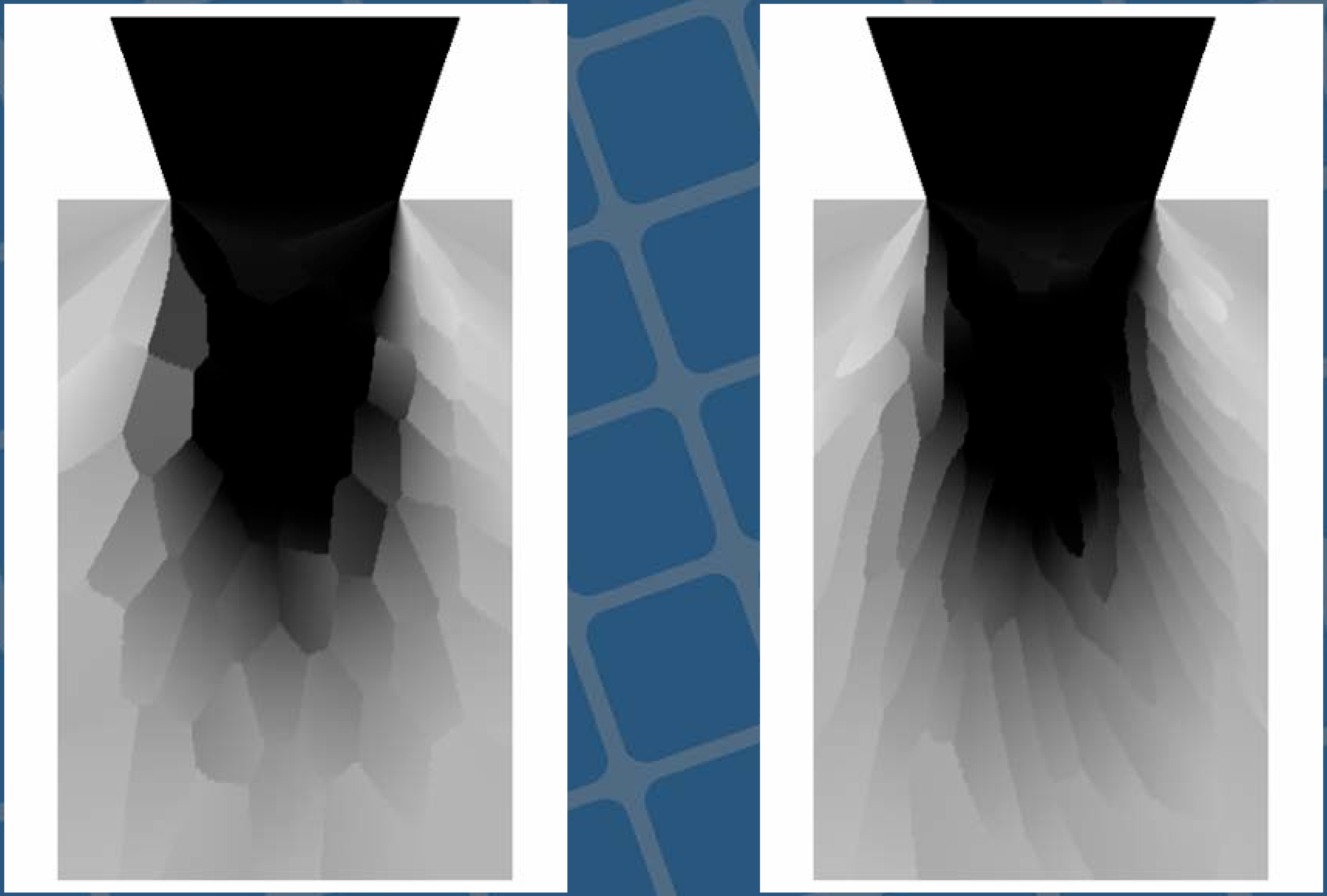




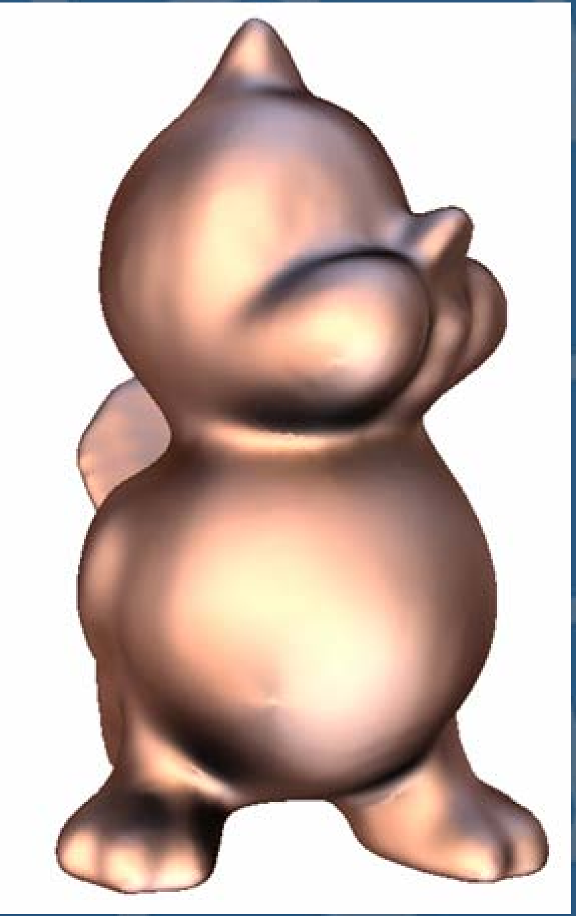
CPCA

- Clusters with low dimensional affine models
- How should clustering be done?
  - *k*-means clustering
- Static PCA
  - VQ, followed by one-time per-cluster PCA
  - optimizes for piecewise-constant reconstruction
- Iterative PCA
  - PCA in the inner loop, slower to compute
  - optimizes for piecewise-affine reconstruction

Static vs. Iterative



Equal Rendering Cost



VQ PCA CPCA

Outline

- ***Compression methods***
  - Spherical harmonics-based PRT [Sloan et al. 02]
  - (Local) factorization and PCA
  - ***Non-linear wavelet approximation***
- Changing view as well as lighting
  - Clustered PCA
  - Triple Product Integrals
- Handling Local Lighting
  - Direct-to-Indirect Transfer

Sparse Matrix-Vector Multiplication

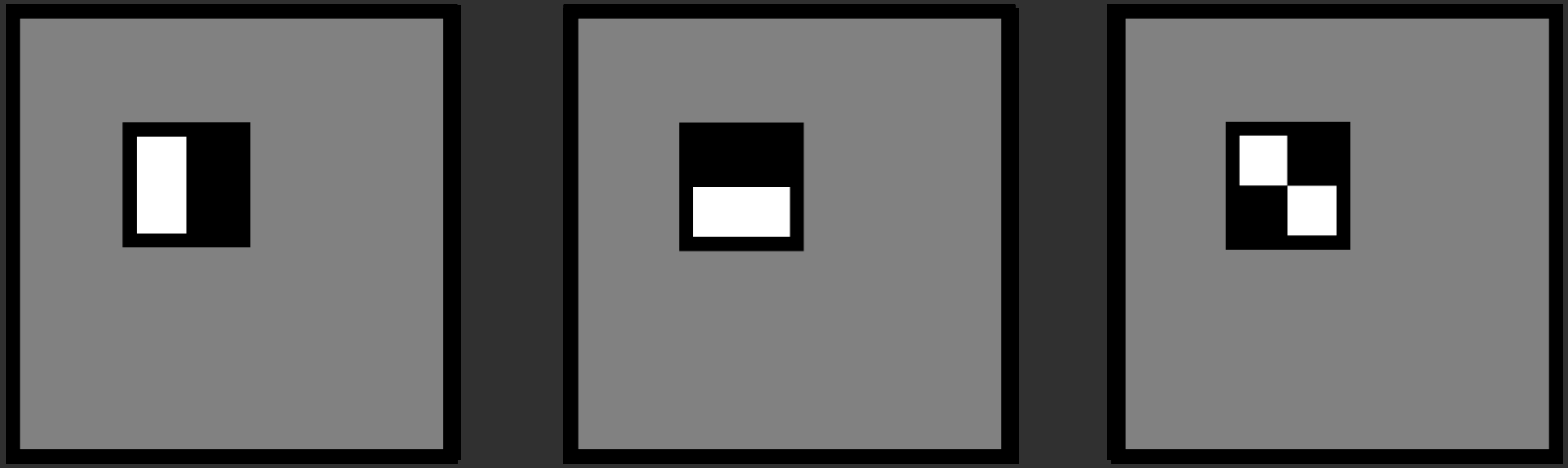
Choose data representations with mostly zeroes

Vector: Use *non-linear wavelet approximation* on lighting

Matrix: Wavelet-encode transport rows

$$\begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_M \end{bmatrix}$$

Haar Wavelet Basis



Non-linear Wavelet Approximation

Wavelets provide dual space / frequency locality

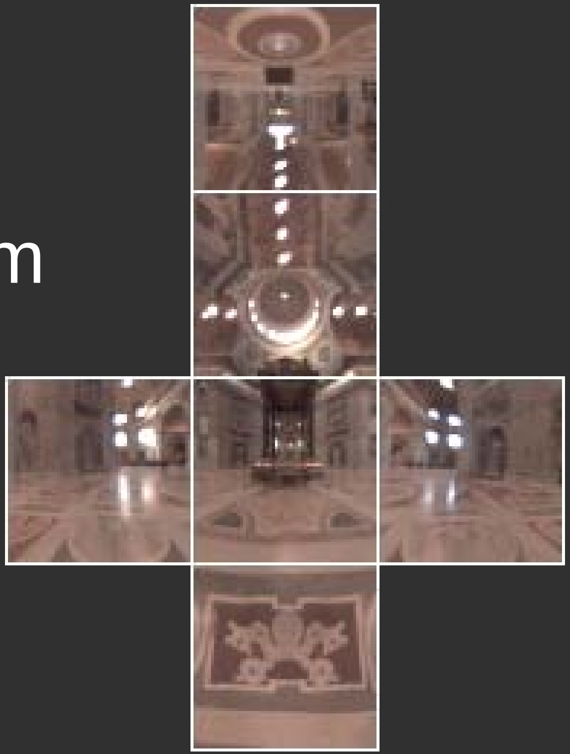
- Large wavelets capture low frequency area lighting
- Small wavelets capture high frequency compact features

Non-linear Approximation

- Use a dynamic set of approximating functions (*depends on each frame's lighting*)
- By contrast, linear approx. uses fixed set of basis functions (like 25 lowest frequency spherical harmonics)
- We choose 10's - 100's from a basis of 24,576 wavelets (64x64x6)

Non-linear Wavelet Light Approximation

Wavelet Transform



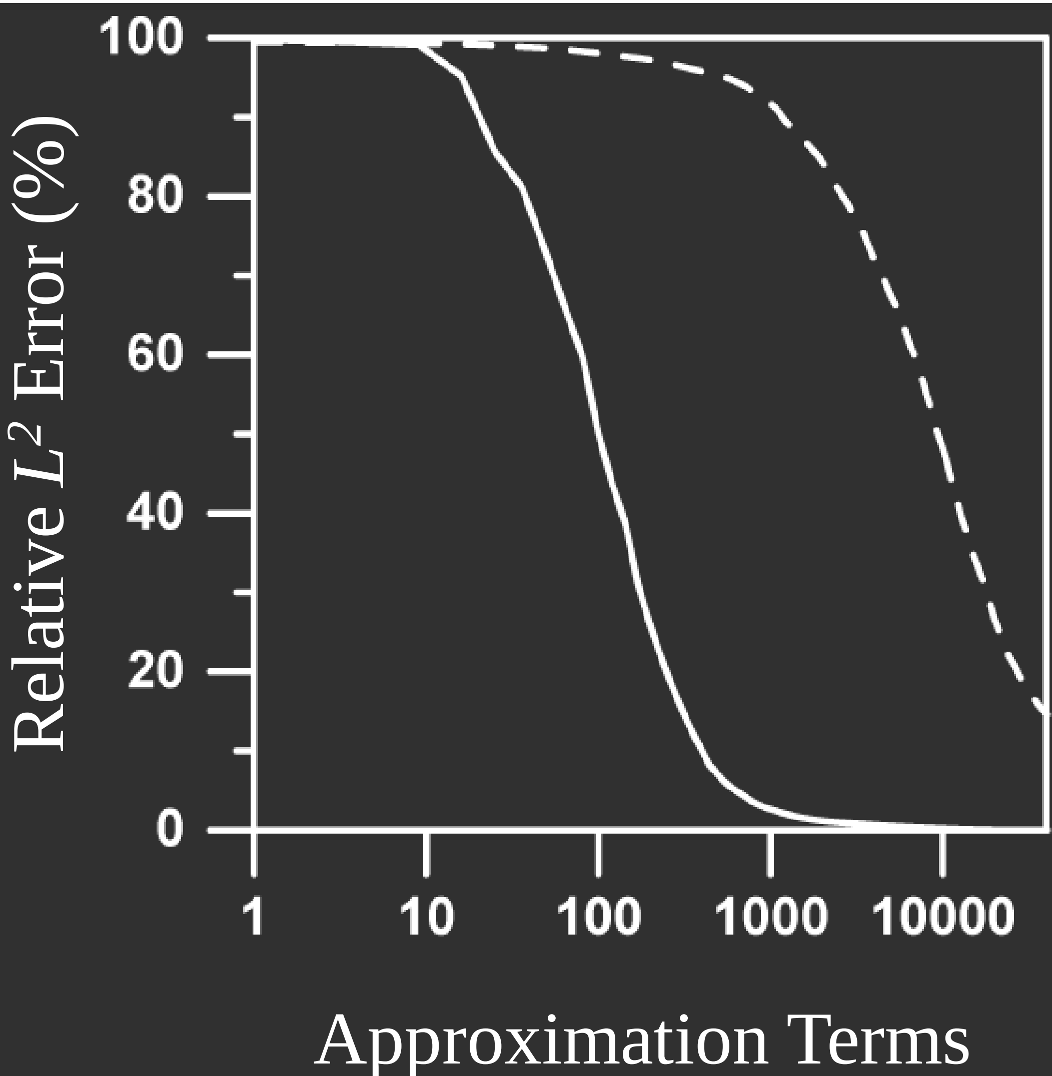
Non-linear Wavelet Light Approximation

$$\begin{bmatrix} 0 \\ L_2 \\ 0 \\ 0 \\ 0 \\ L_6 \\ \vdots \\ 0 \end{bmatrix}$$

Non-linear Approximation

Retain 0.1% – 1% terms

Error in Lighting: St Peter's Basilica



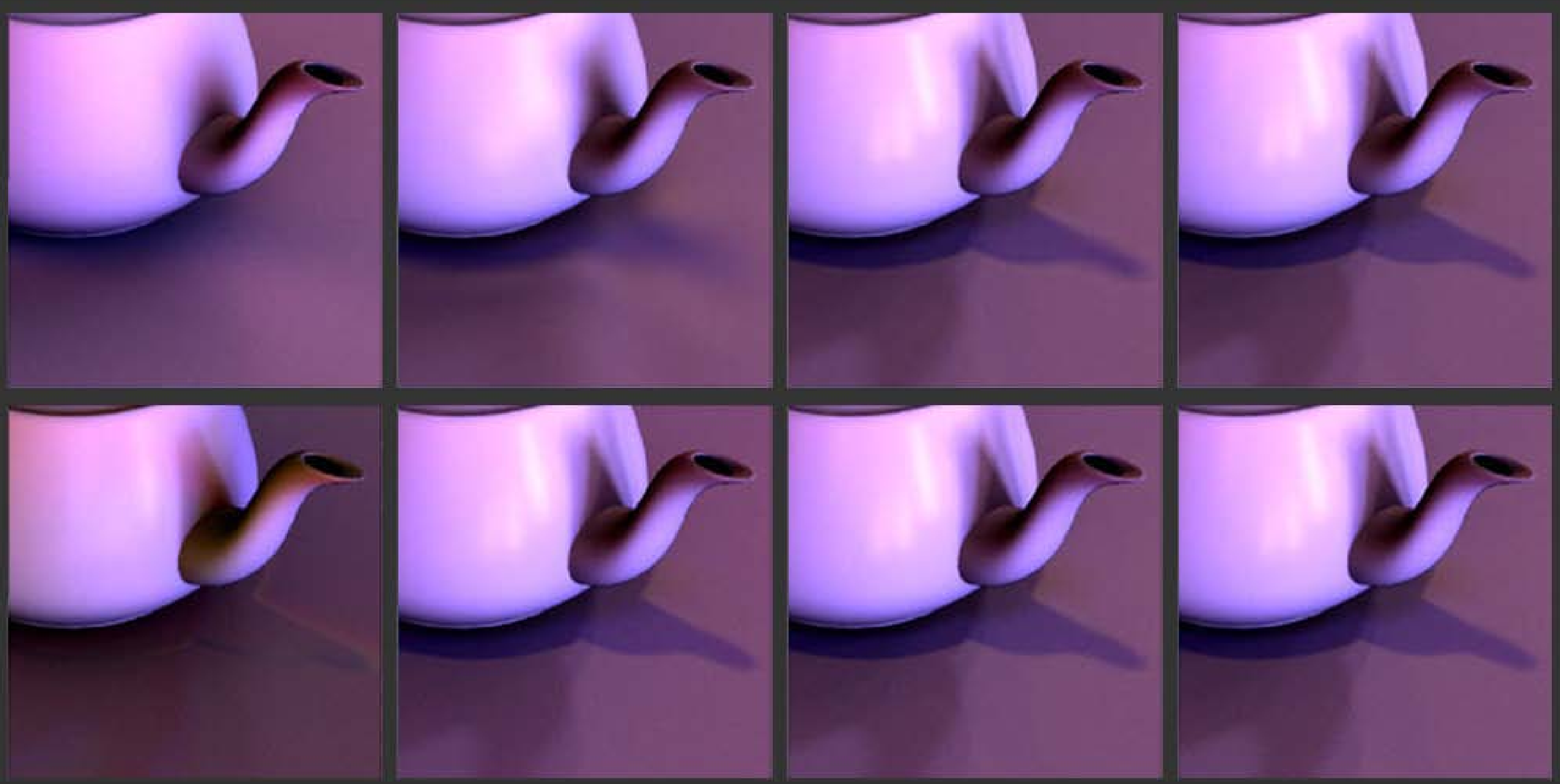
Sph. Harmonics

Non-linear Wavelets

Ng, Ramamoorthi, Hanrahan 03

Output Image Comparison

Top: Linear Spherical Harmonic Approximation
Bottom: Non-linear Wavelet Approximation



25 200 2,000 20,000

Outline

- Compression methods
  - Spherical harmonics-based PRT [Sloan et al. 02]
  - (Local) factorization and PCA
  - Non-linear wavelet approximation
- ***Changing view as well as lighting***
  - ***Clustered PCA***
  - Triple Product Integrals
- Handling Local Lighting
  - Direct-to-Indirect Transfer

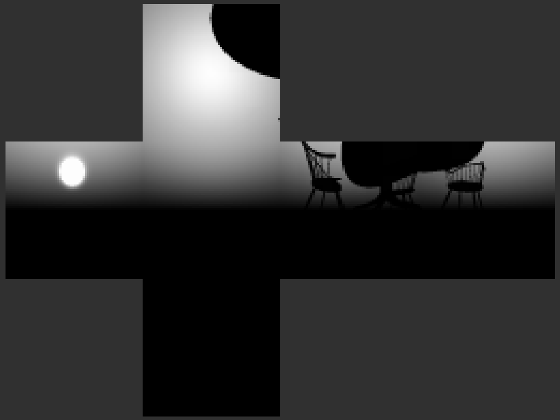
SH + Clustered PCA

- Described earlier (combine Sloan 03 with Kautz 03)
  - Use low-frequency source light and transferred light variation (Order 5 spherical harmonic = 25 for both; total = 25*25=625)
  - 625 element vector for each vertex
  - Apply CPCA directly (Sloan et al. 2003)
  - Does not easily scale to high-frequency lighting
    - Really cubic complexity (number of vertices, illumination directions or harmonics, and view directions or harmonics)
  - Practical real-time method on GPU

Outline

- Compression methods
  - Spherical harmonics-based PRT [Sloan et al. 02]
  - (Local) factorization and PCA
  - Non-linear wavelet approximation
- ***Changing view as well as lighting***
  - Clustered PCA
  - ***Triple Product Integrals***
- Handling Local Lighting
  - Direct-to-Indirect Transfer

Problem Characterization



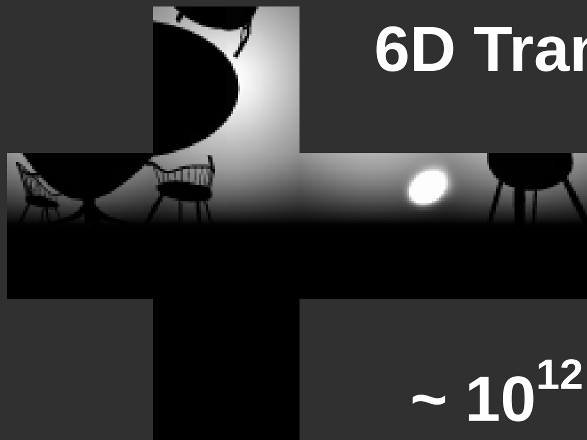
6D Precomputation Space

- Distant Lighting (2D)
- View (2D)
- Rigid Geometry (2D)

With ~ 100 samples per dimension

~ 10^{12} samples total!! : Intractable computation, rendering

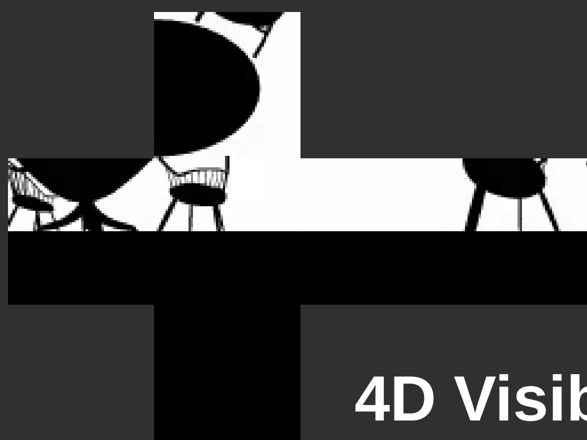
Factorization Approach



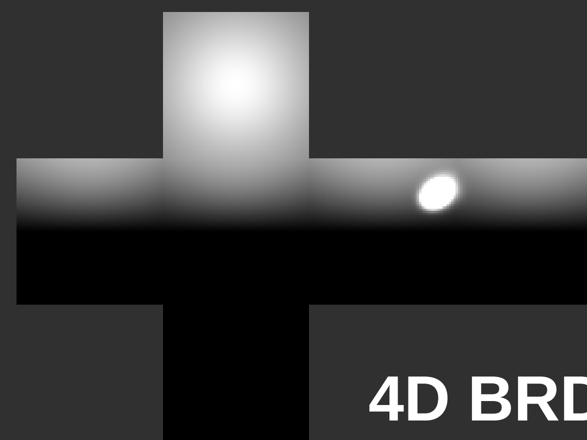
6D Transport

=

~ 10^{12} samples



*



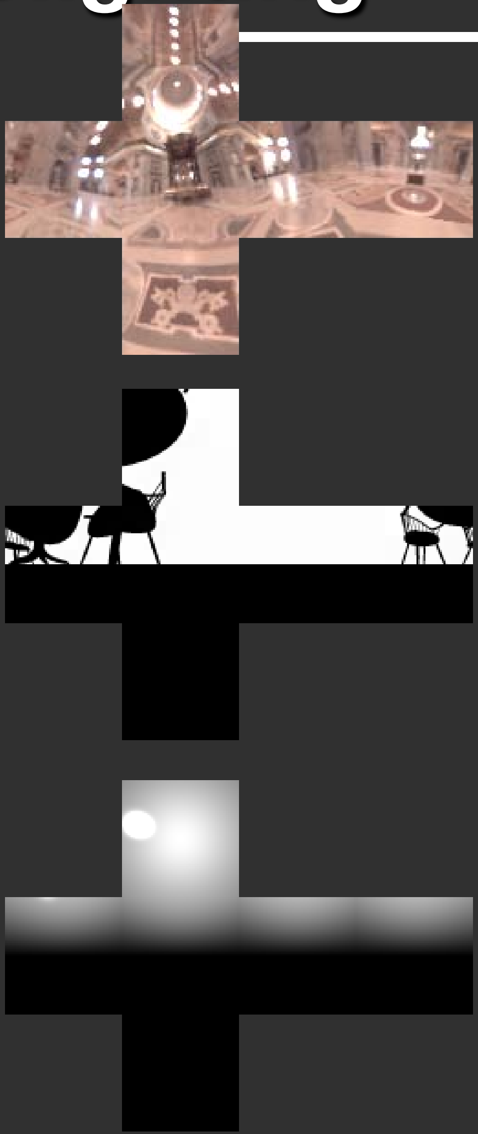
4D Visibility

4D BRDF

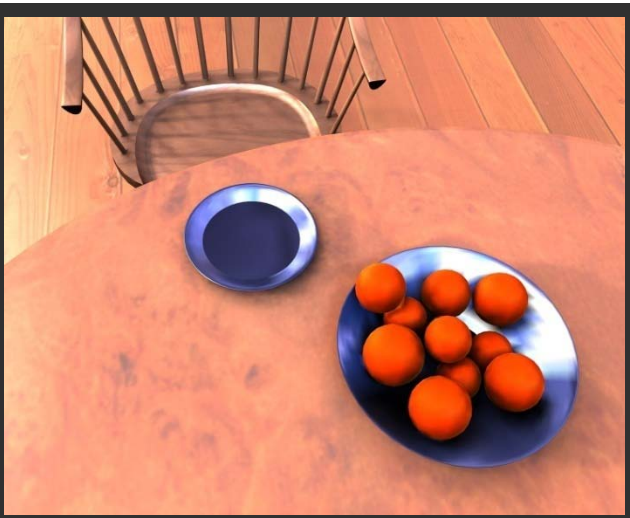
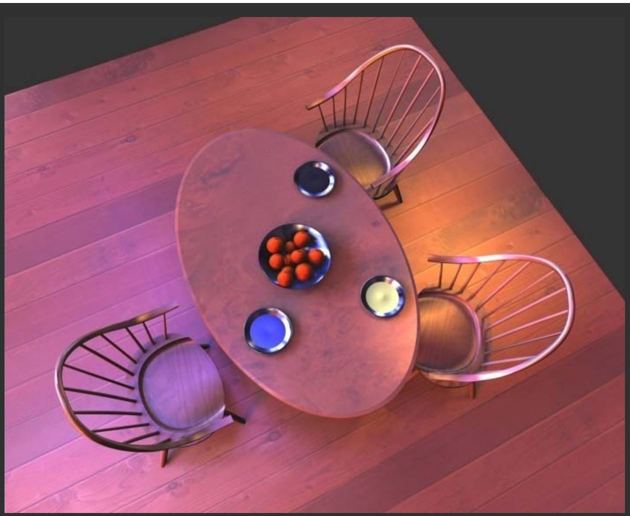
~ 10^{8} samples

~ 10^{8} samples

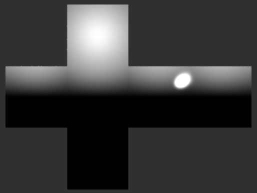
Triple Product Integral Relighting



Relit Images (3-5 sec/frame)

Triple Product Integrals

$$
\begin{aligned}
B &= \int_{S^2} L(\omega)\, V(\omega)\, \tilde{\rho}(\omega)\, d\omega \\
&= \int_{S^2} \left(\sum_i L_i \Psi_i(\omega) \right) \left(\sum_j V_j \Psi_j(\omega) \right) \left(\sum_k \tilde{\rho}_k \Psi_k(\omega) \right) d\omega \\
&= \sum_i \sum_j \sum_k L_i\, V_j\, \tilde{\rho}_k \int_{S^2} \Psi_i(\omega) \Psi_j(\omega) \Psi_k(\omega)\, d\omega \\
&= \sum_i \sum_j \sum_k L_i\, V_j\, \tilde{\rho}_k\, C_{ijk}
\end{aligned}
$$

Basis Requirements

$$B = \sum_i \sum_j \sum_k L_i \, V_j \, \tilde{\rho}_k \, C_{ijk}$$

1. Need few non-zero “tripling” coefficients

$$C_{ijk} = \int_{S^2} \Psi_i(\omega)\Psi_j(\omega)\Psi_k(\omega) \, d\omega$$

2. Need sparse basis coefficients

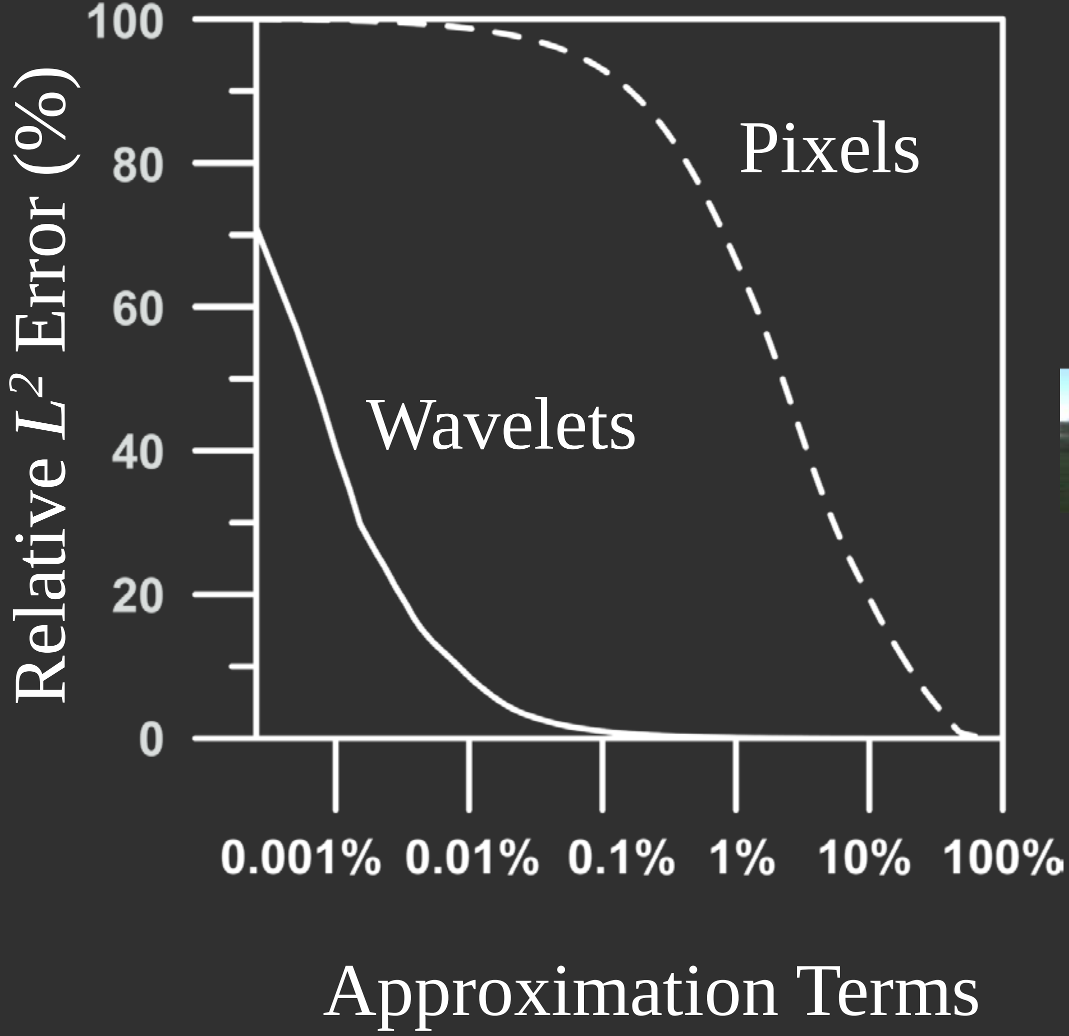
$$L_i, \; V_j, \; \tilde{\rho}_k$$

1. Number Non-Zero Tripling Coeffs

$$C_{ijk} = \int_{S^2} \Psi_i(\omega)\Psi_j(\omega)\Psi_k(\omega)\, d\omega$$

Basis Choice	Number Non-Zero C_{ijk}
General (e.g. PCA)	$O(N^3)$
Sph. Harmonics	$O(N^{5/2})$
Haar Wavelets	$O(N \log N)$

2. Sparsity in Light Approx.



Summary of Wavelet Results

- Derive direct *O(N log N)* triple product algorithm
- Dynamic programming can eliminate *log N* term
- Final complexity linear in number of retained basis coefficients

Outline

- Compression methods
  - Spherical harmonics-based PRT [Sloan et al. 02]
  - (Local) factorization and PCA
  - Non-linear wavelet approximation
- Changing view as well as lighting
  - Clustered PCA
  - Triple Product Integrals
- ***Handling Local Lighting***
  - ***Direct-to-Indirect Transfer***

Direct-to-Indirect Transfer

- Lighting non-linear w.r.t. light source parameters (position, orientation etc.)
- Indirect is a linear function of direct illumination
  - Direct can be computed in real-time on GPU
  - Transfer of direct to indirect is pre-computed
- Hašan et al. 06
  - Fixed view – cinematic relighting with GI

DTIT: Matrix-Vector Multiply

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_N \end{bmatrix}$$

$$= \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_M \end{bmatrix}$$

Direct illumination on a set of samples distributed on scene surfaces

Compression: Matrix rows in Wavelet basis

DTIT: Demo

Summary

- Really a big data compression and signal-processing problem
- Apply many standard methods
  - PCA, wavelet, spherical harmonic, factor compression
- And invent new ones
  - VQPCA, wavelet triple products
- Guided by and gives insights into properties of illumination, reflectance, visibility
  - How many terms enough? How much sparsity?