Pre-computed Radiance Transfer

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

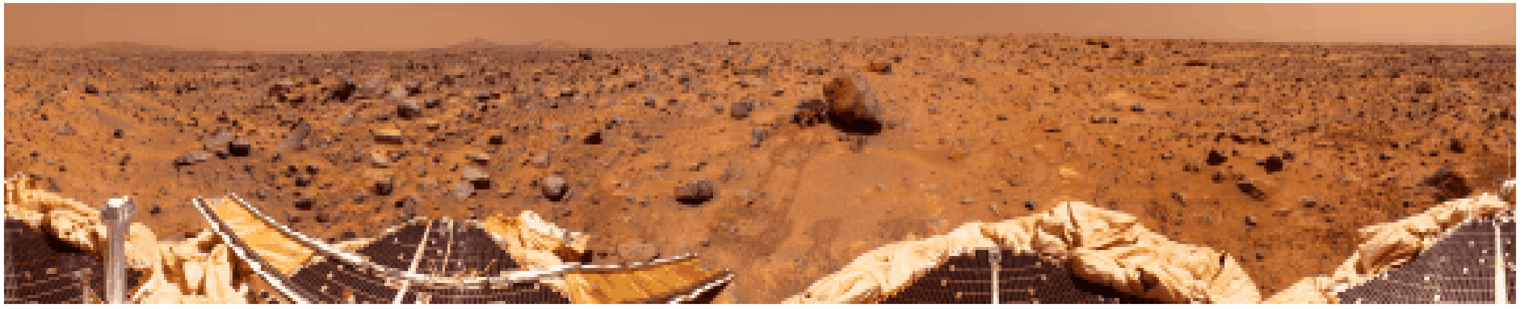
- Real-time rendering with complex lighting, shadows, and possibly GI
- Infeasible – too much computation for too small a time budget

- 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"

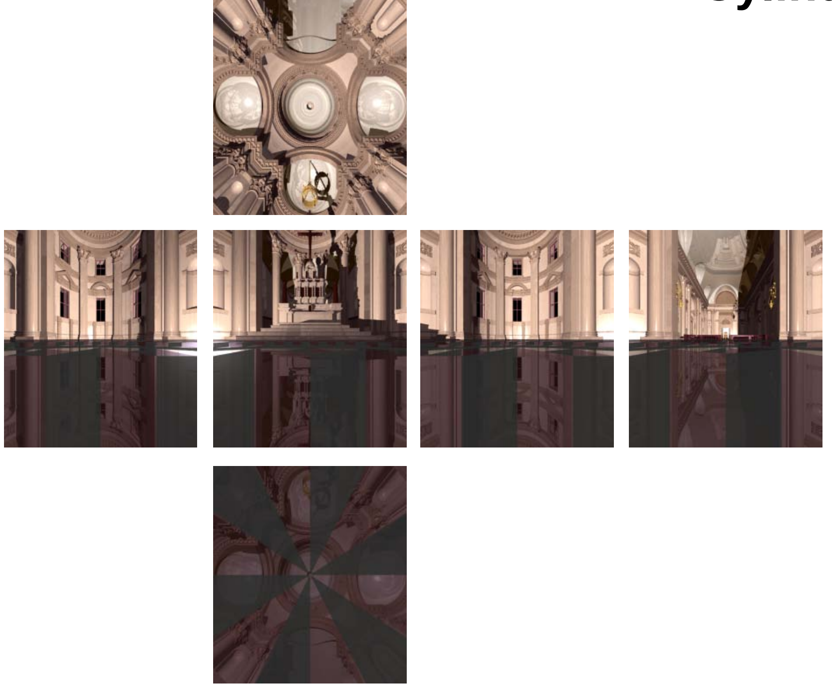
Environment mapping

Miller and Hoffman, 1984
Later, Greene 86, Cabral et al, Debevec 97, …

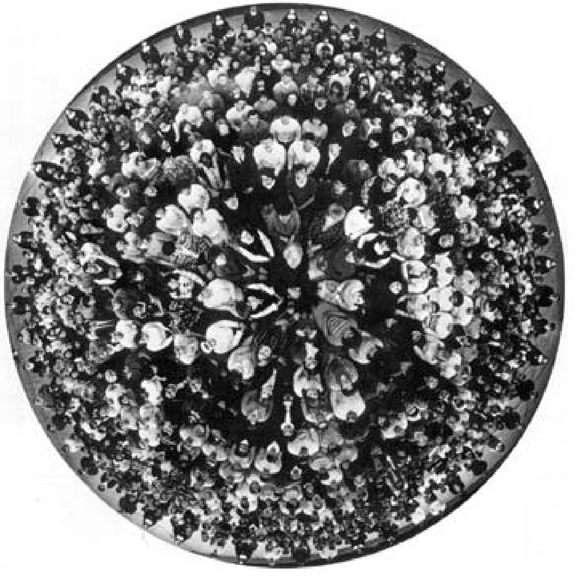
Environment Maps



Cylindrical Panoramas



Cubical Environment Map

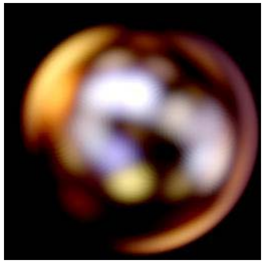


180 degree fisheye
Photo by R. Packo

Assumptions

- Distant illumination
- No shadowing, interreflection

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



- What if the surface is rougher…

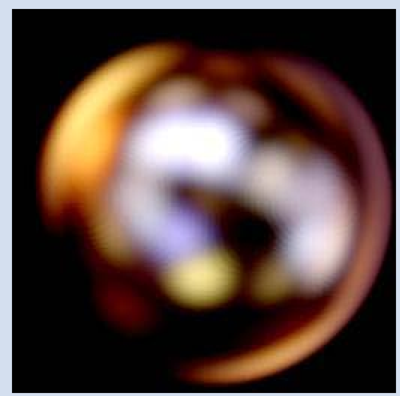
- Or completely diffuse?

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



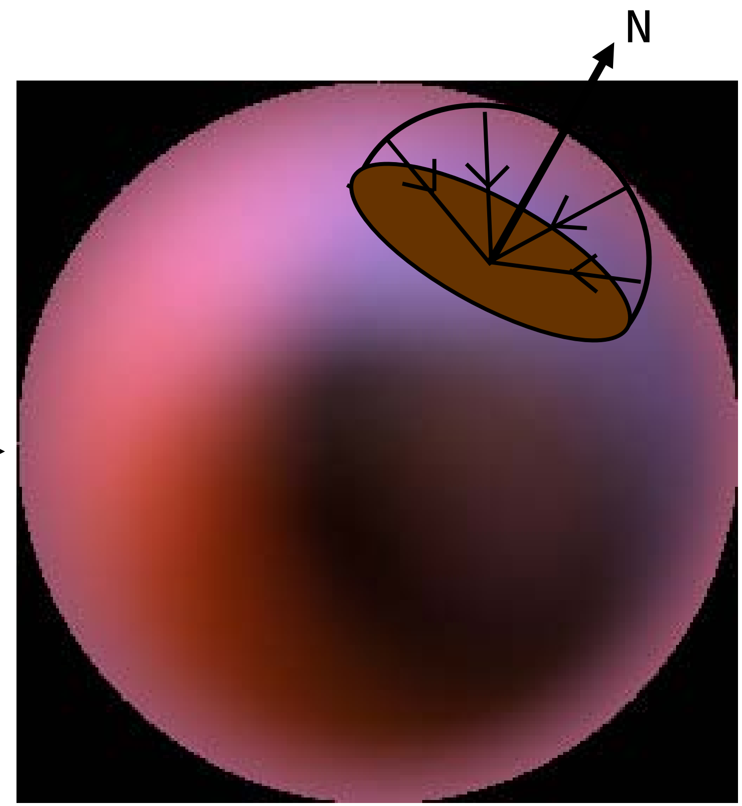
- Reflection Maps [Miller and Hoffman, 1984]
  - Irradiance (indexed by N) and Phong (indexed by R)

Reflection Maps

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

SH-based Irradiance Env. Maps

Incident Radiance
(Illumination Environment Map)

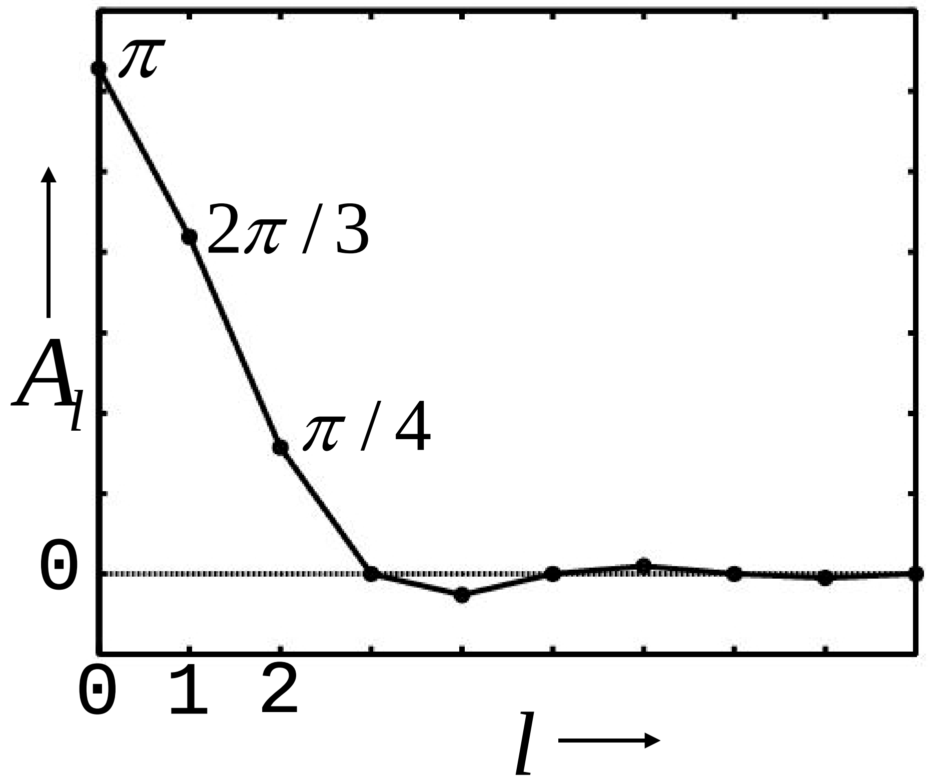


Irradiance Environment Map

Analytic Irradiance Formula

Lambertian surface acts like low-pass filter

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

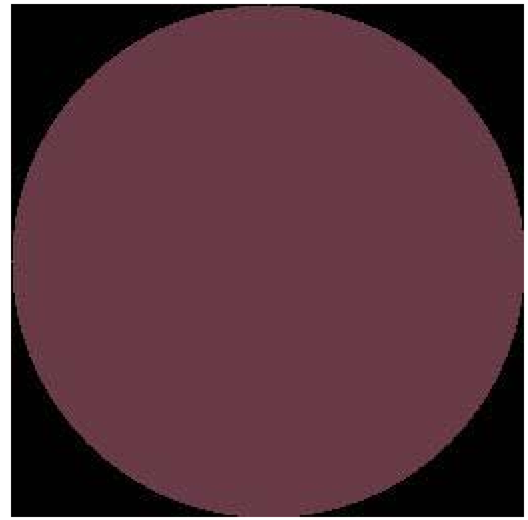


Ramamoorthi and Hanrahan 01
Basri and Jacobs 01

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

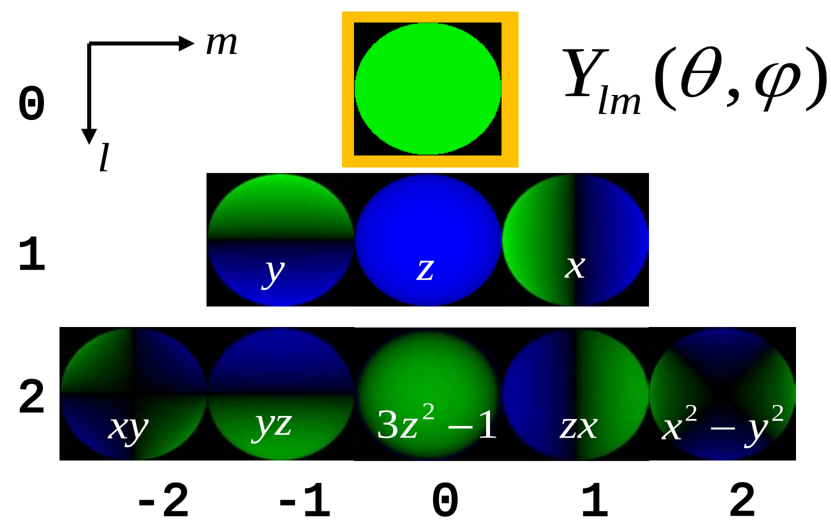
9 Parameter Approximation

Exact image



Order 0
1 term

RMS error = 25 %

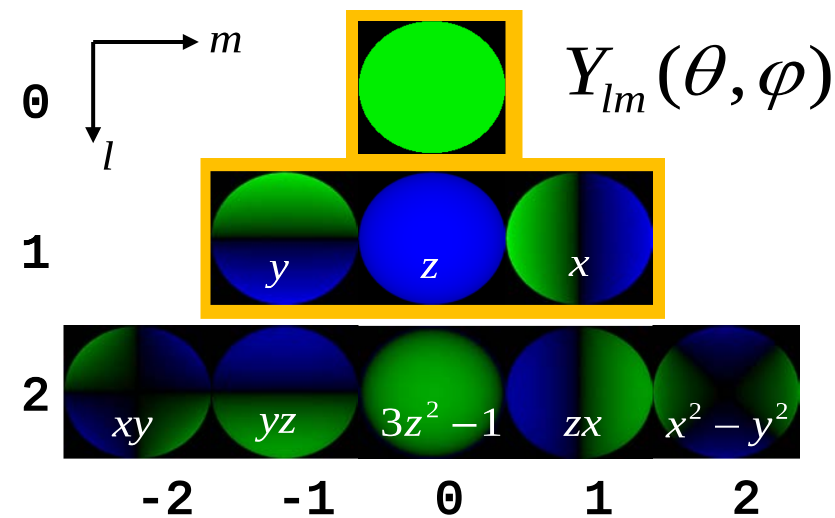


9 Parameter Approximation

Exact image

Order 1
4 terms

RMS Error = 8%



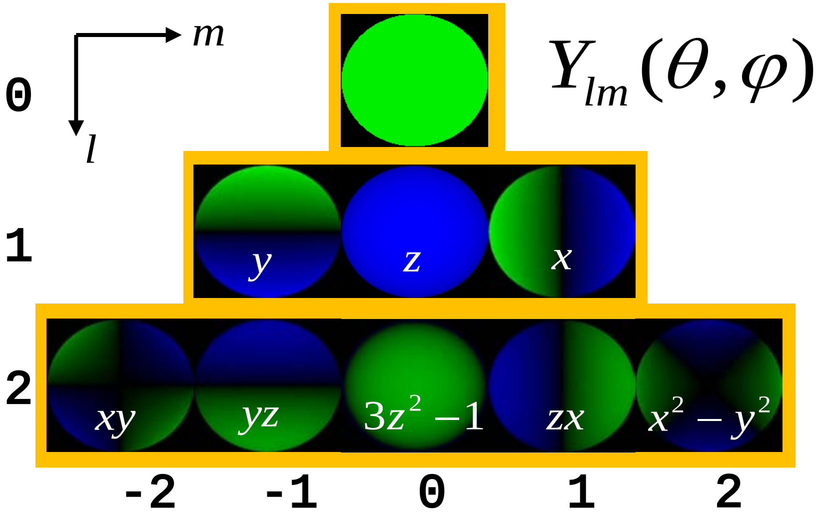
9 Parameter Approximation

Exact image

Order 2
9 terms

RMS Error = 1%

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



Real-Time Rendering

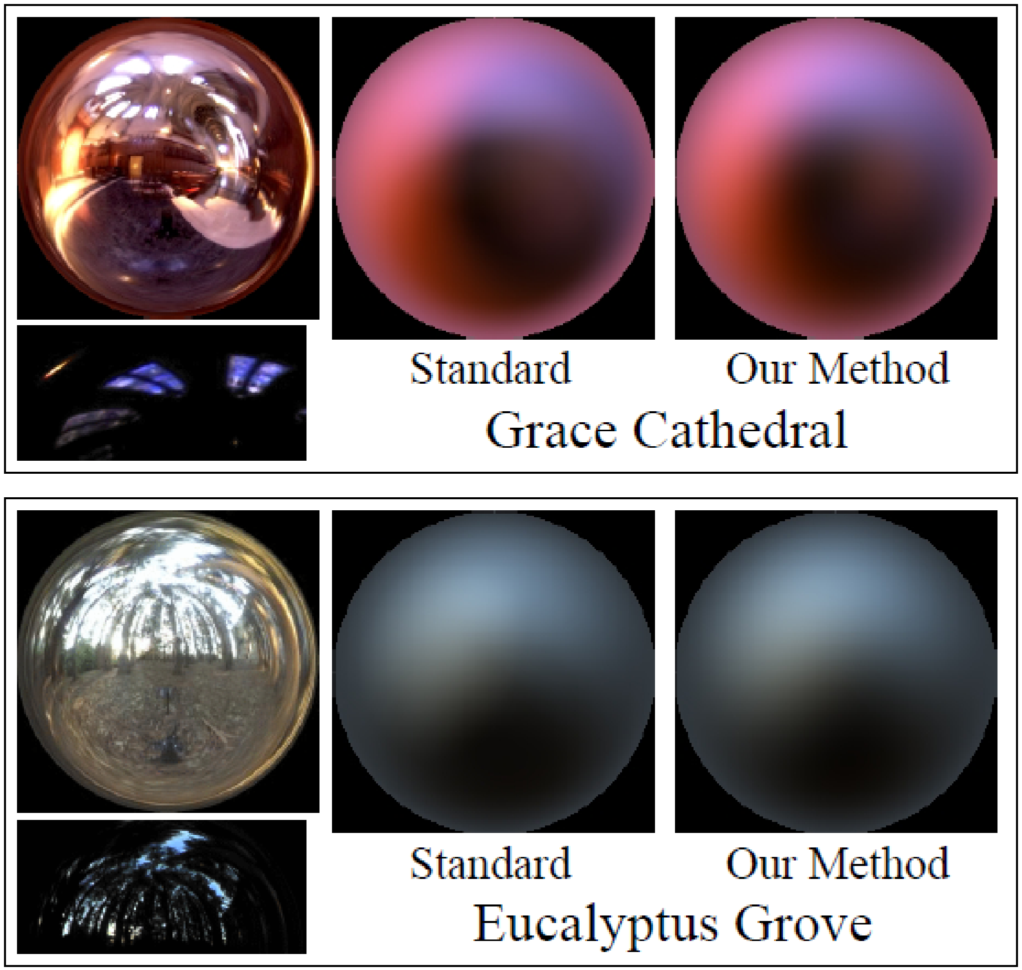
$$\boxed{E(n) = n^t M n}$$

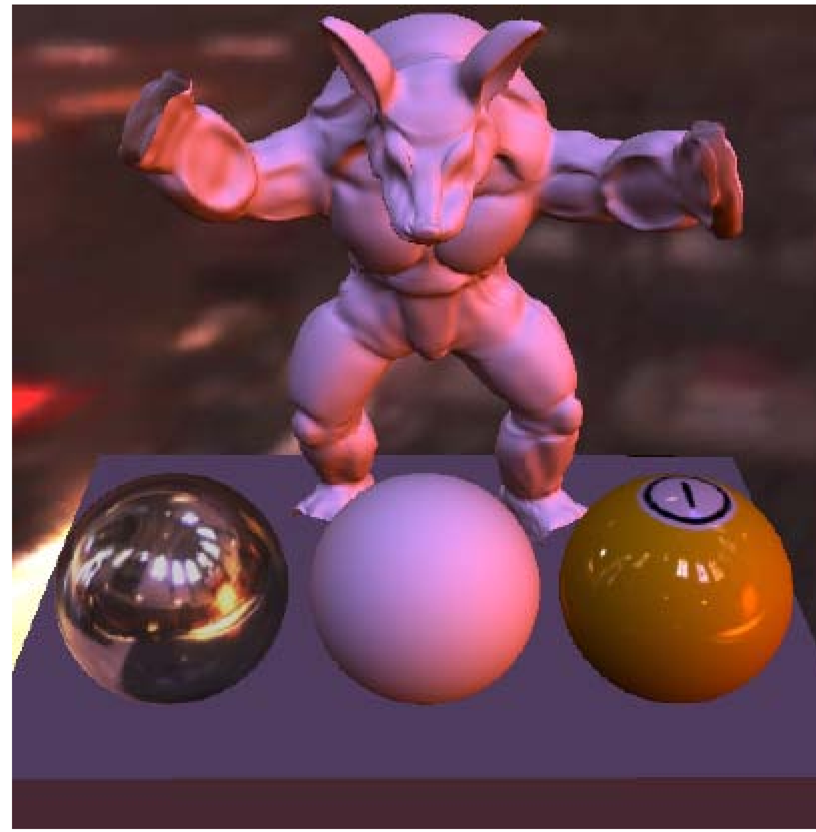
- Simple procedural rendering method (no textures)
  - Requires only matrix-vector multiply and dot-product
  - In software or NVIDIA vertex programming hardware

- Widely used in Games (AMPED for Microsoft Xbox), Movies (Pixar, Framestore CFC, …)

```
surface float1 irradmat (matrix4 M, float3 v) {
        float4 n = {v , 1} ;
        return dot(n , M*n) ;
}
```

SH-based Irradiance Env. Maps





Images courtesy Ravi Ramamoorthi & Pat Hanrahan

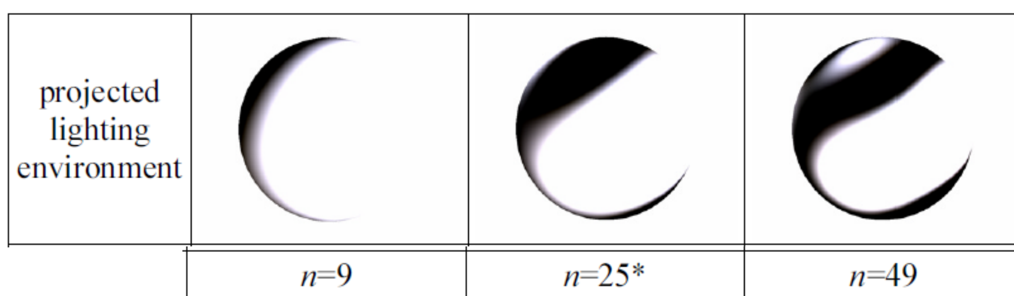
SH-based Arbitrary BRDF Shading 1

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

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

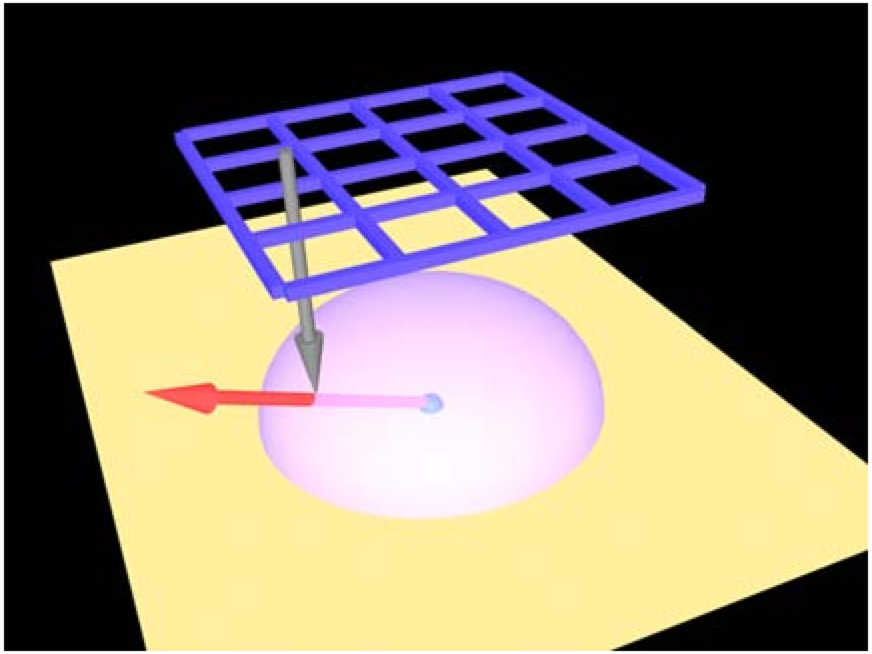
- SH representation
  - Environment map (one set of coefficients)
  - Scene BRDFs (one coefficient vector for each discretized view direction)

original lighting environment



SH-based Arbitrary BRDF Shading 2

- BRDF Representation
  - BRDF coefficient vector for a given ω_o, looked up from a texture (use e.g. paraboloid mapping to map ω_o to a texture coordinate)
  - BRDF coefficients pre-computed for all scene BRDFs (SH projection)

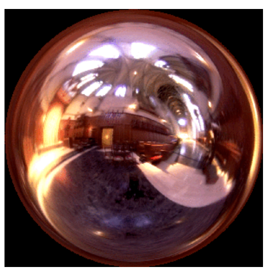


SH-based Arbitrary BRDF Shading 3

- Rendering: for each vertex / pixel, do

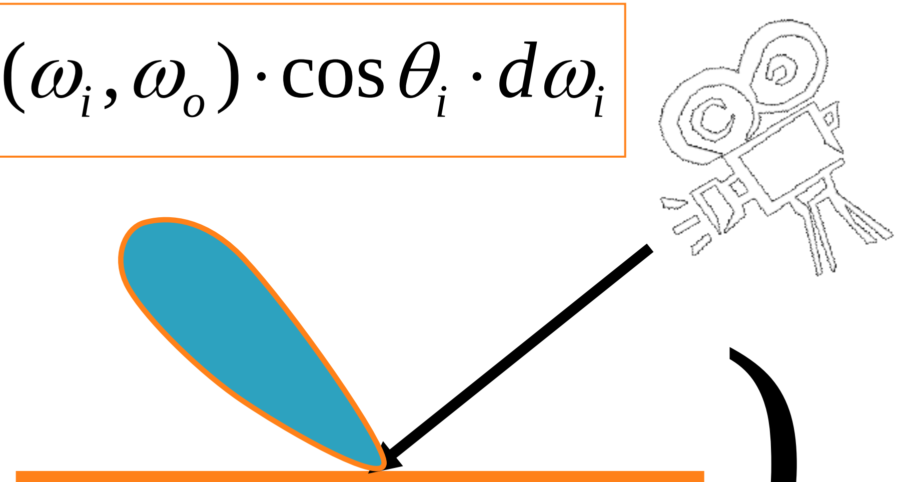
$$L_o(\omega_o) = \int_\Omega L_i(\omega_i) \cdot BRDF(\omega_i, \omega_o) \cdot \cos\theta_i \cdot d\omega_i$$

$\int \Big($



Environment map

$\bullet$



BRDF

$\Big)$

= 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.*

Environment Map Summary

- Very popular for interactive rendering
- Extensions handle complex materials
- Shadows with precomputed transfer

- But cannot directly combine with shadow maps
- Limited to distant lighting assumption

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

Simple Example – Daytime Relighting

- Analyze precomputed images of scene

Jensen 2000

- Synthesize relit images from precomputed data

Simple Example – Daytime Relighting

- Analyze precomputed images of scene

Jensen 2000

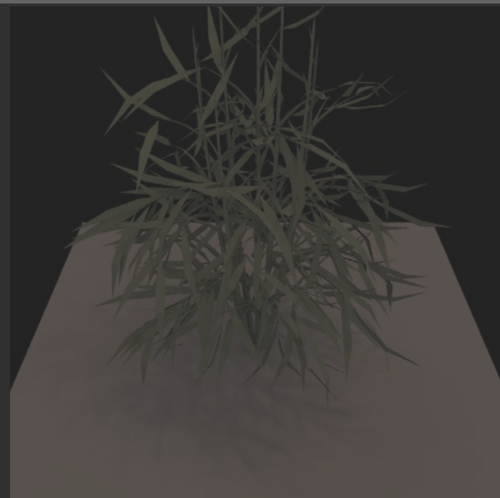
- Synthesize relit images from precomputed data

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)

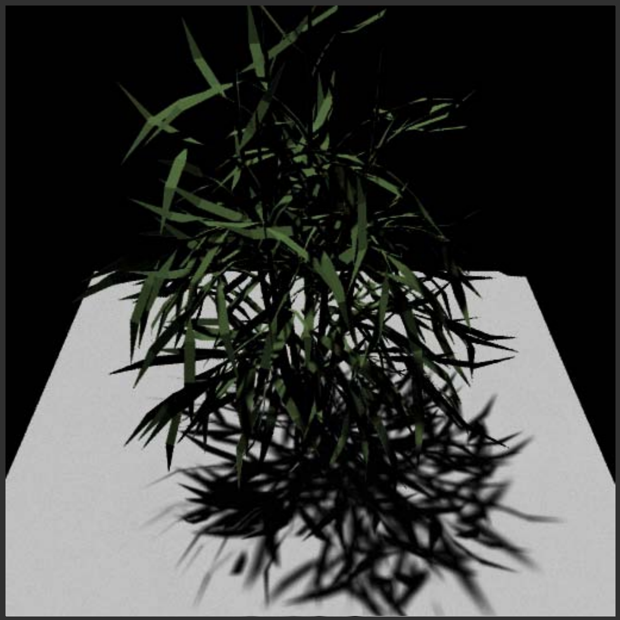
$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_N \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix} \begin{bmatrix} L_1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

Input Lighting
(Cubemap Vector)

Precomputed
Transport
Matrix

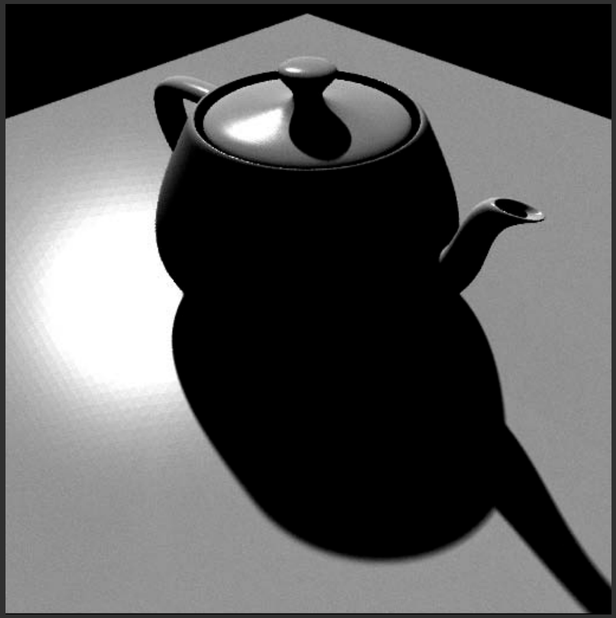
Matrix Columns (Images)

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



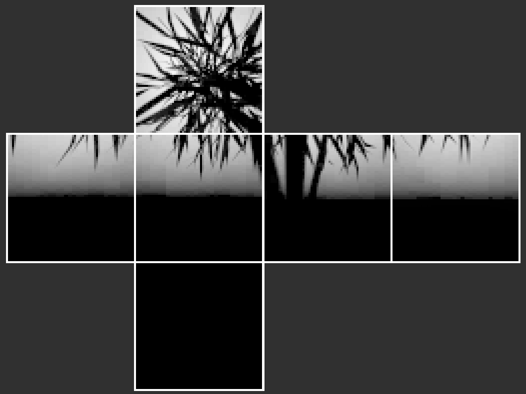
Precompute: Ray-Trace Image Cols

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



Precompute 2: Rasterize Matrix Rows

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



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*

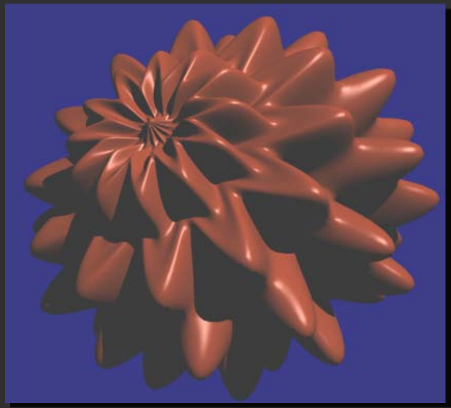
How to relight quickly?

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
  - Factored BRDFs
  - Triple Product Integrals

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

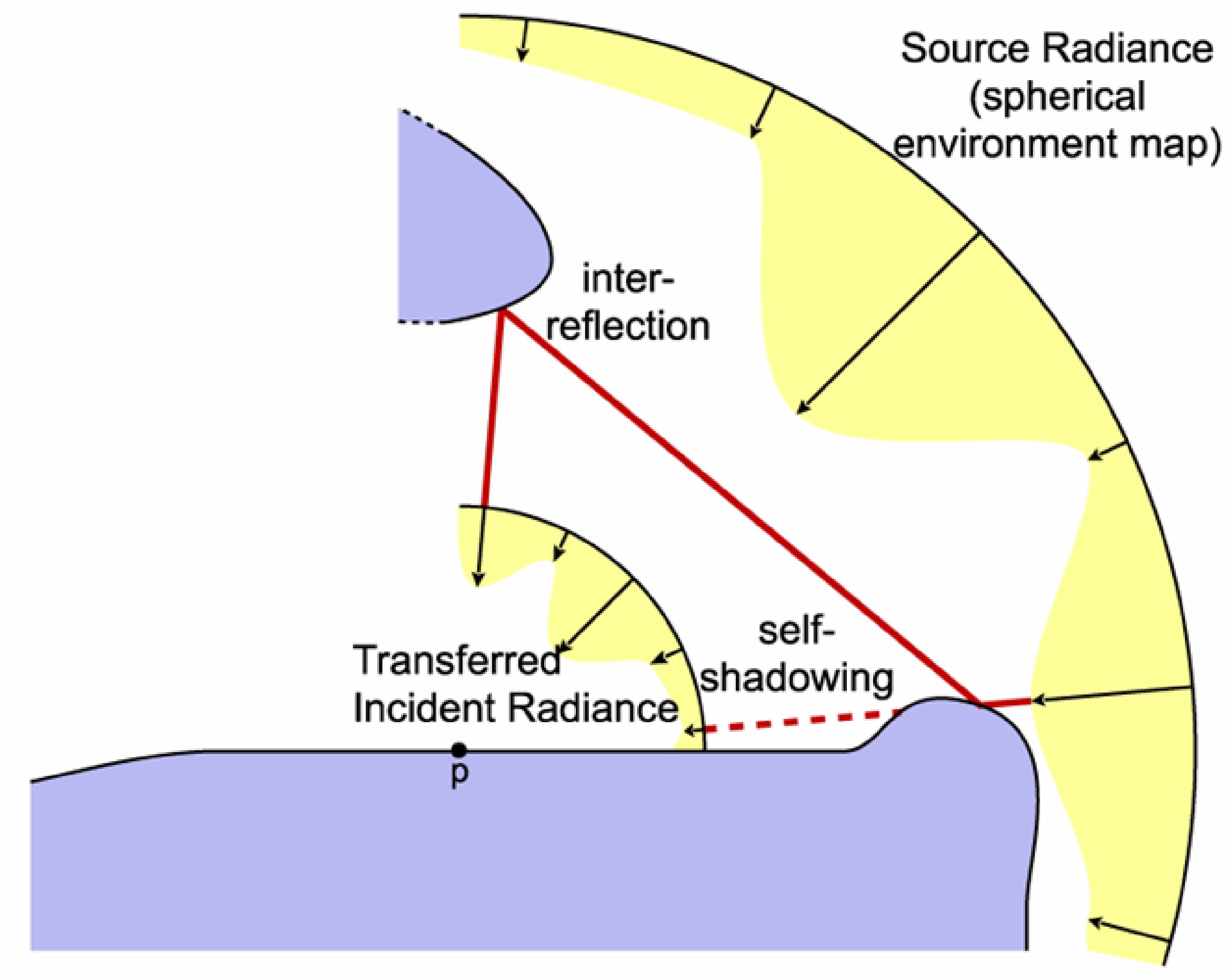

point light


Env. light

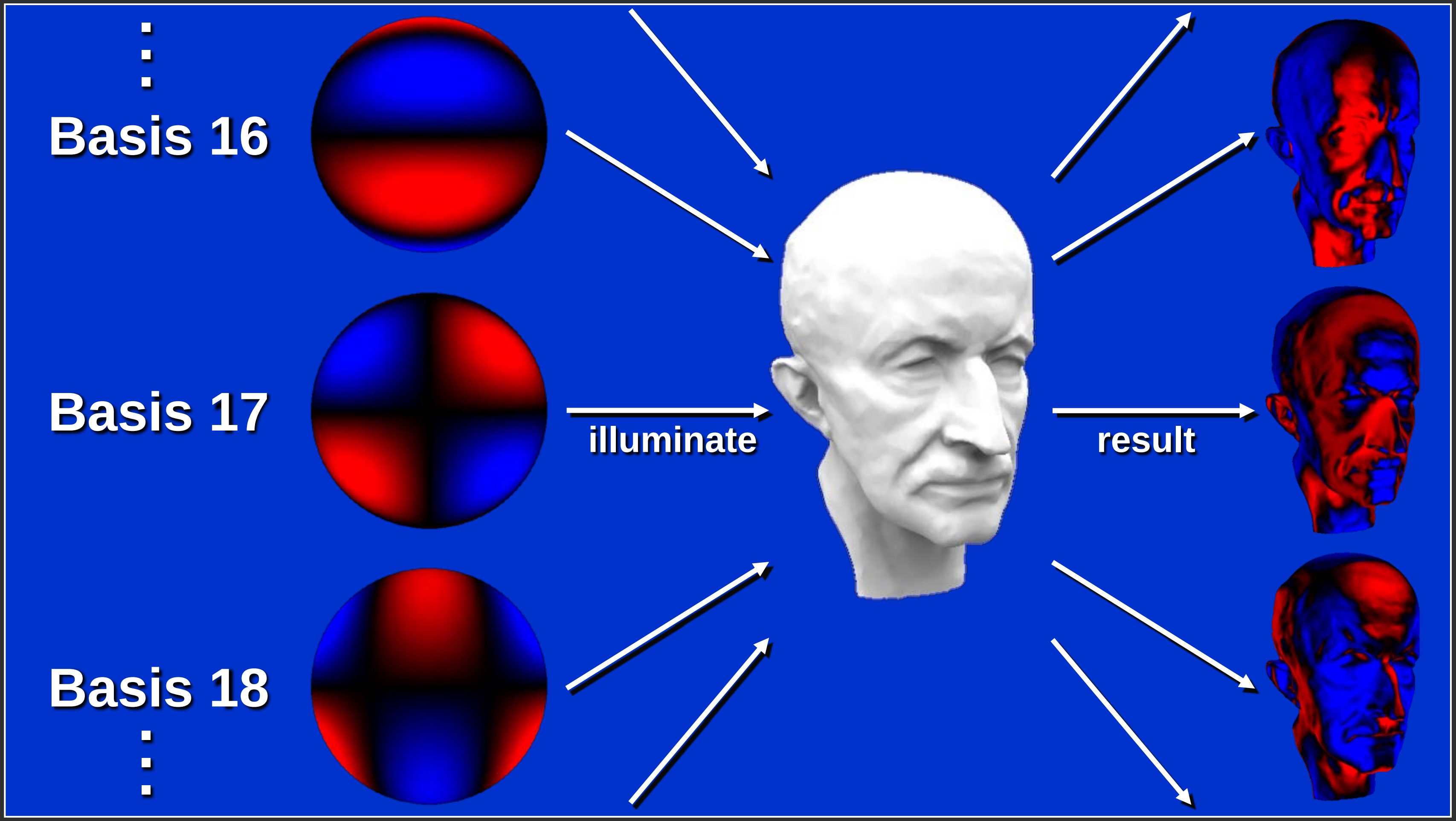

Env. lighting, no shadows


Env. lighting, shadows

PRT Terminology



SH-based PRT: Idea



Relation to a Matrix-Vector Multiply

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

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

Idea of SH-based PRT

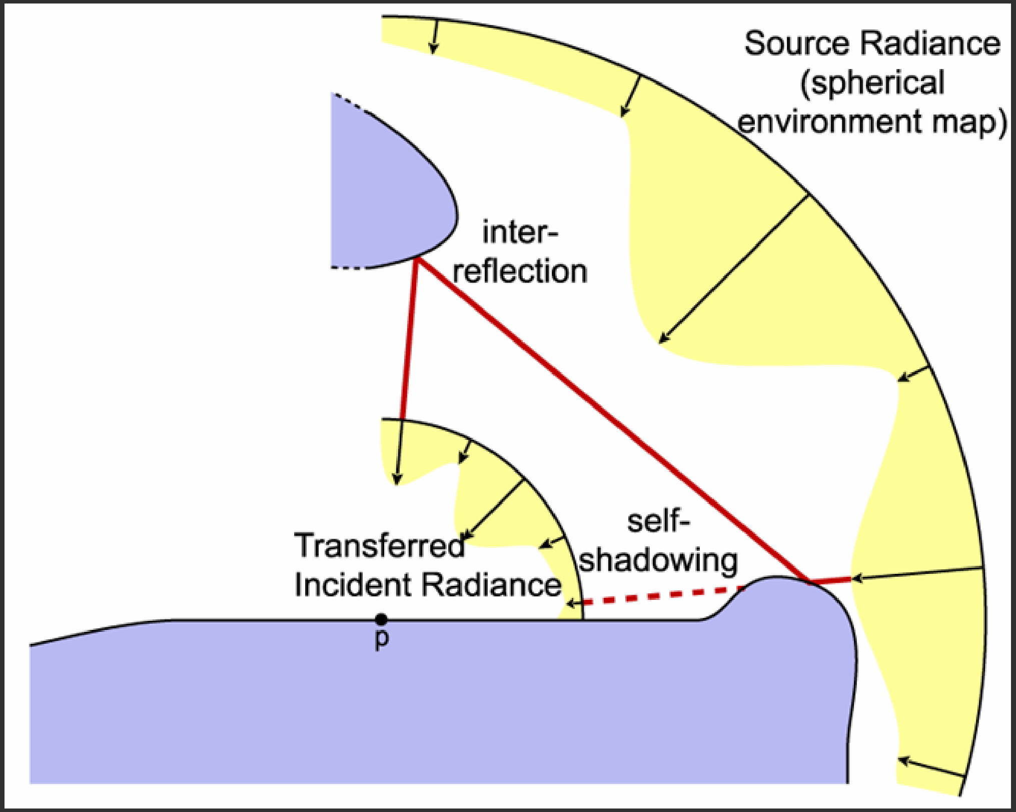
- The *L* vector is projected onto low-frequency components (say 25). Size greatly reduced.
- Hence, only 25 matrix columns
- But each pixel/vertex still treated separately
  - One RGB value per pixel/vertex:
    - diffuse shading / arbitrary BRDF shading w/ fixed view direction
  - SH coefficients of transferred radiance (25 RGB values per pixel/vertex)
    - 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

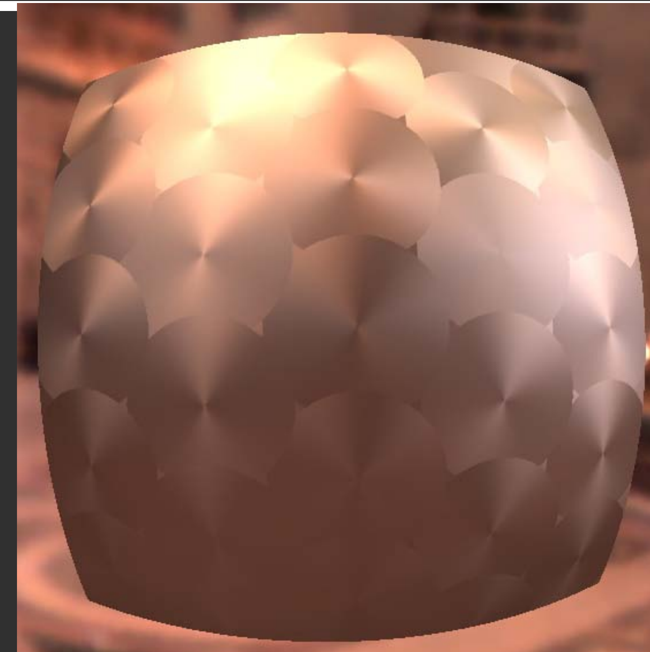
- Combine with Kautz et al. 03
- Transfer matrix turns SH env. map into SH transferred radiance
- Kautz et al. 03 is applied to transferred radiance



Arbitrary BRDF Results

Anisotropic BRDFs

Other BRDFs



Spatially Varying