Precomputed Lighting: Theory and Practice

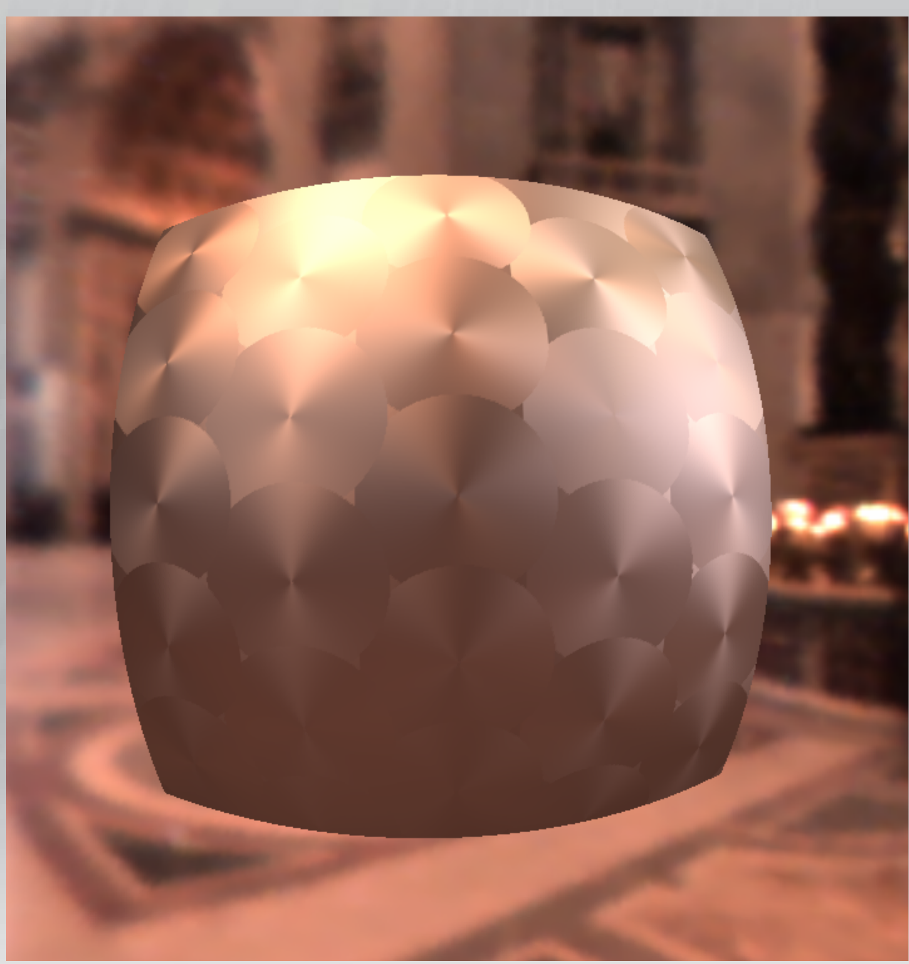
Peter-Pike Sloan
Software Development Engineer
WGGT/D3DX

Challenges in interactive rendering

- Generating realistic images interactively is hard
- Many dimensions of complexity
  - Geometric complexity
  - Material complexity
  - Meso-scale complexity
  - Lighting complexity
  - Transport complexity
  - Synergy
- This talk focuses on techniques that enable more lighting/transport complexity

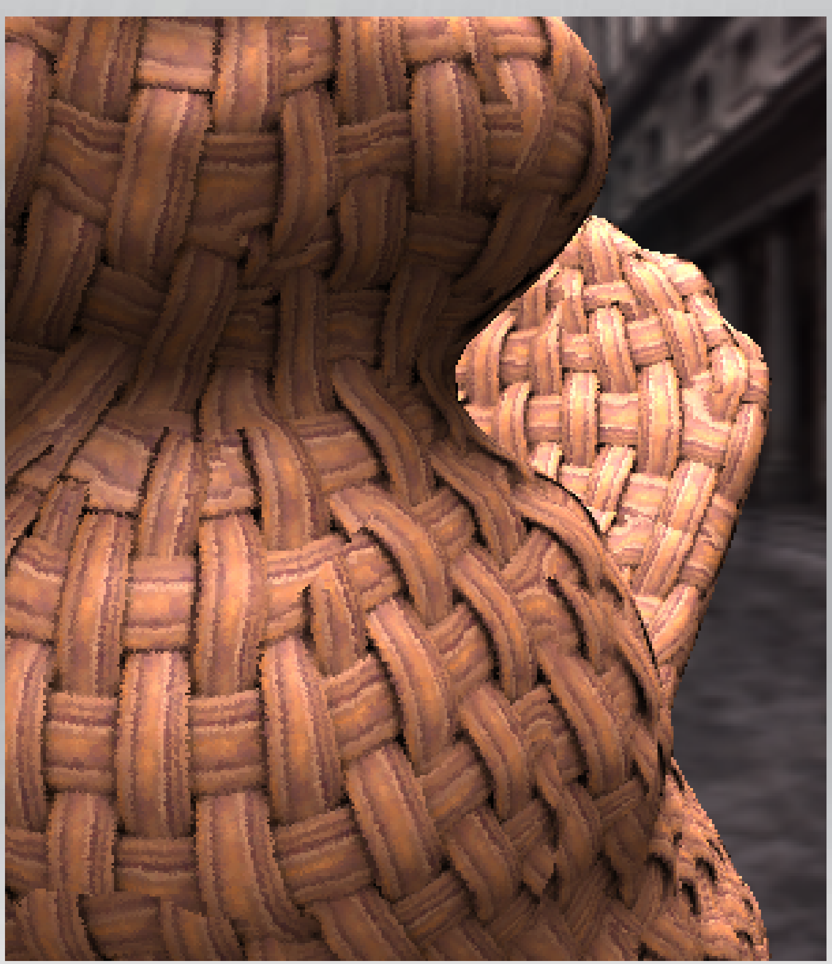
Material Complexity

- Models how light interacts with a surface
  - Assume the “structure” of the material is below the visible scale
  - Simple variation
    - Twist maps



Meso-Scale Complexity

- Variations at a visible scale - not geometry
  - Bump/Roughness maps
  - Parallax Mapping/BTF's extreme examples of this

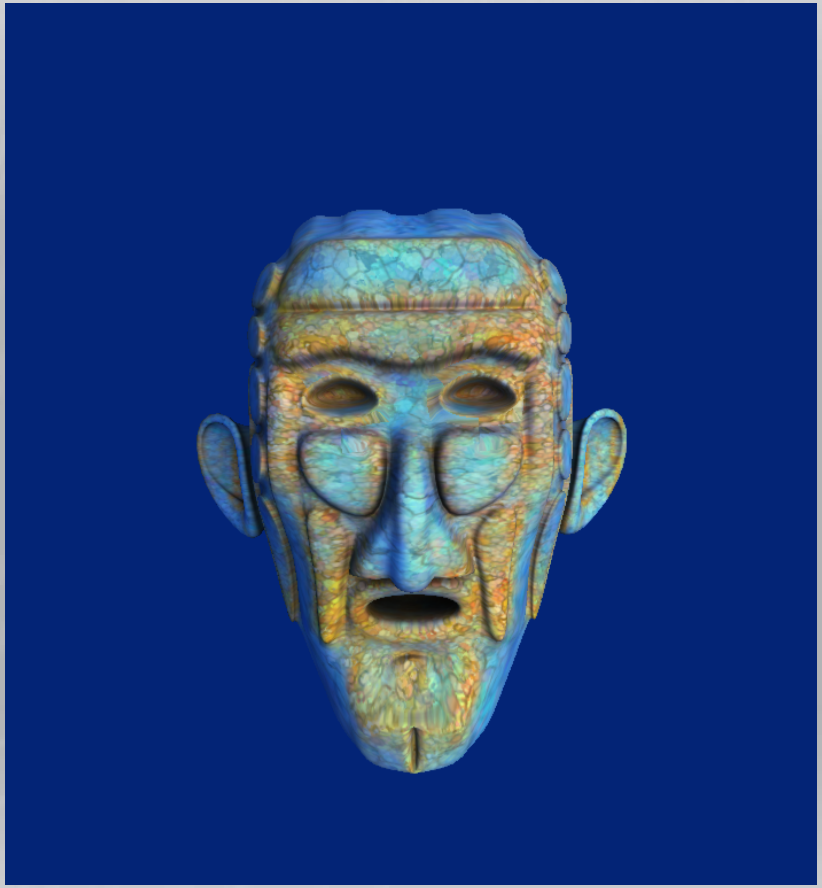


Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - "Smooth" (low frequency) lighting
  - Completely general

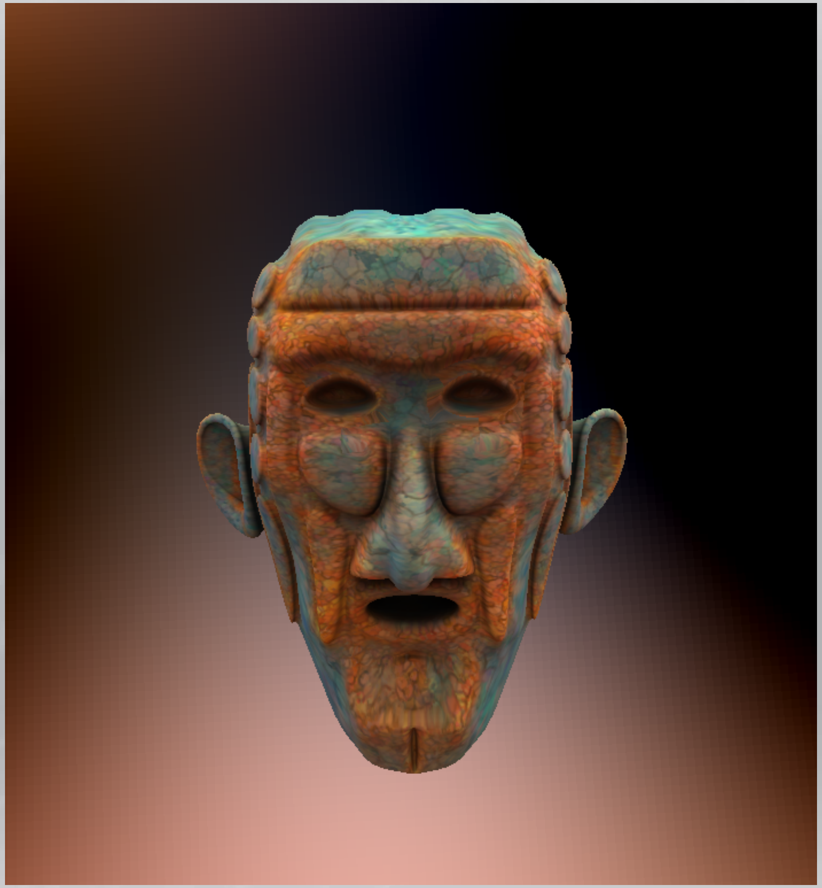
Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - "Smooth" (low frequency) lighting
  - Completely general



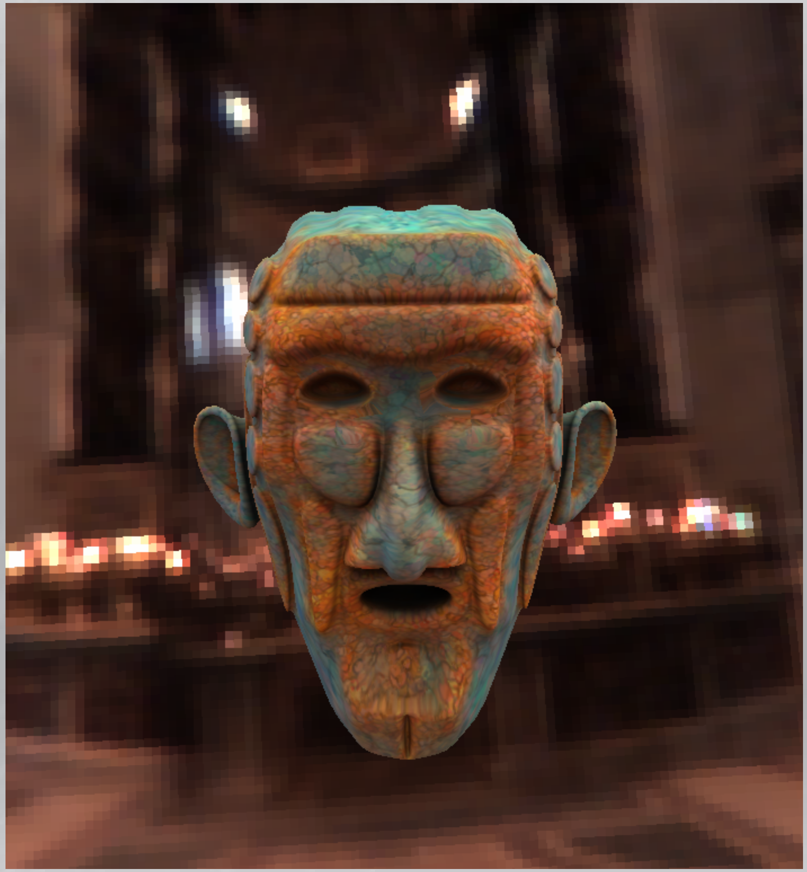
Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - "Smooth" (low frequency) lighting
  - Completely general



Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - “Smooth” (low frequency) lighting
  - Completely general

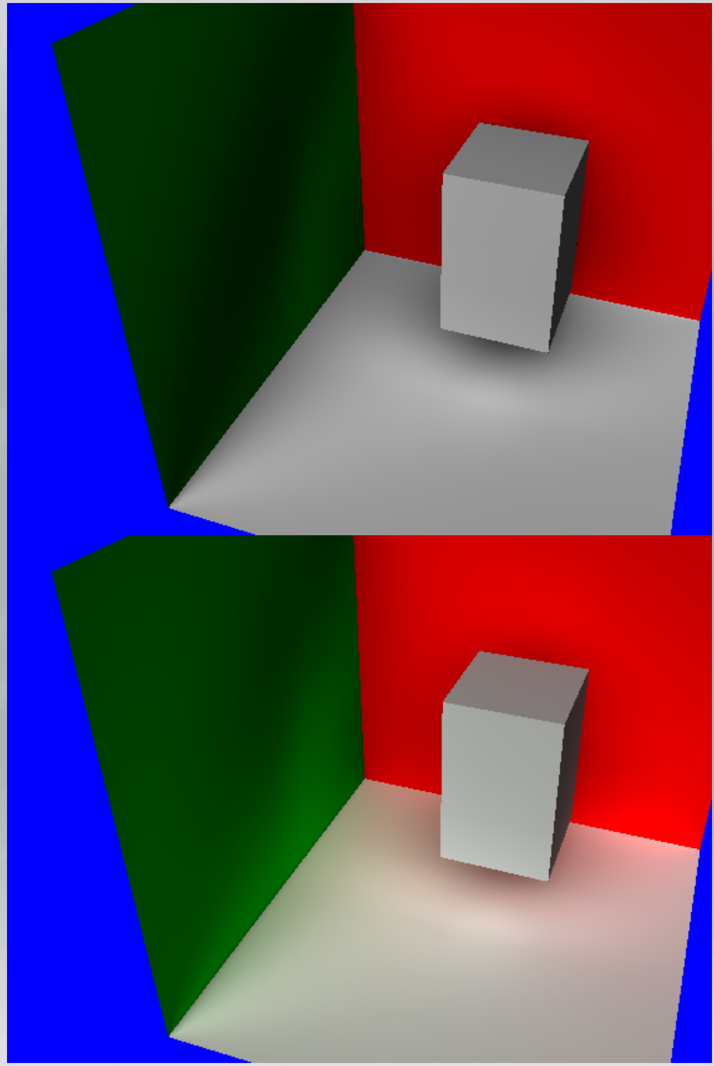


Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)

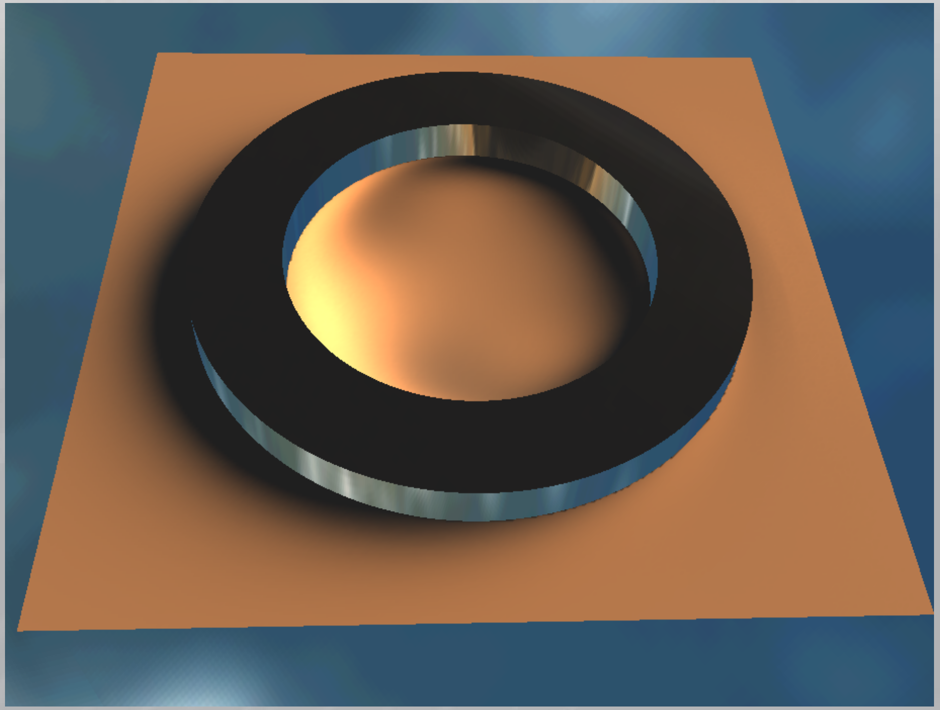
Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)



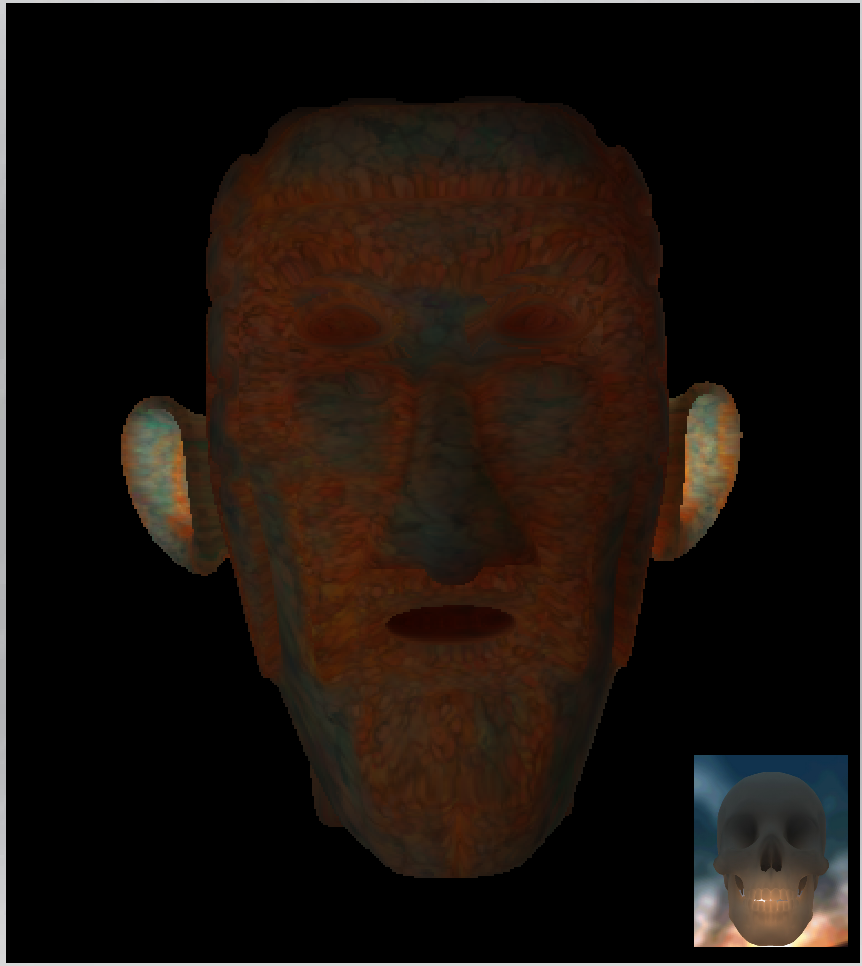
Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)



Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)



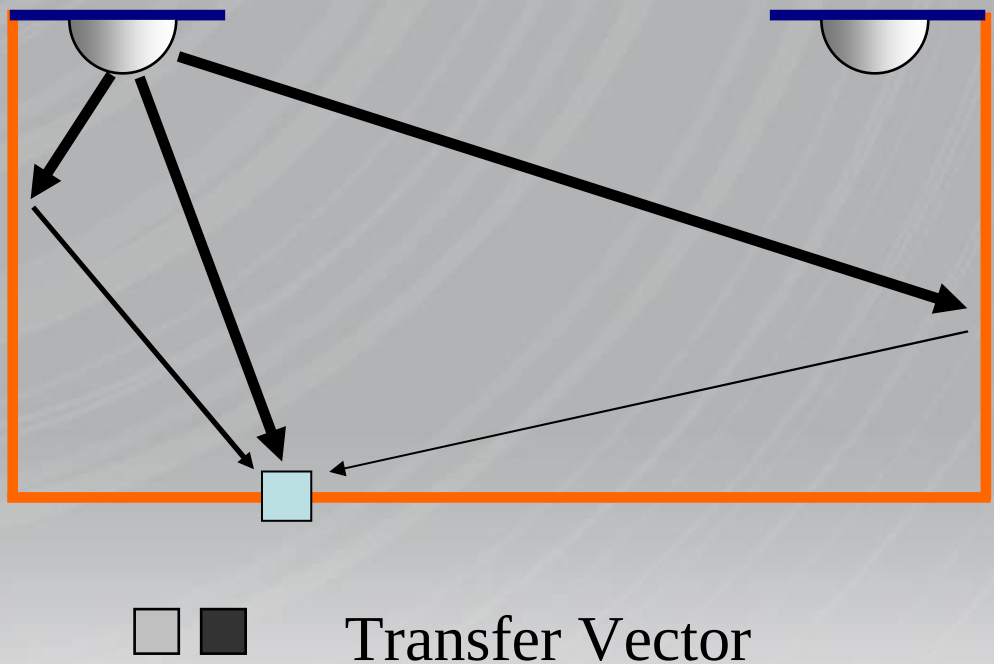
Some Of All Of This

- Real scenes have all of these forms of complexity
- Extreme realism on one is not necessarily that interesting
  - Incredible material models that are completely homogenous and lit by a single directional light
  - Great lighting environments for diffuse surfaces with no shadows

Precomputed Radiance Transfer (PRT)

- Models an object/scenes response to lighting expressed in a given basis
  - Can model arbitrary transport complexity
- Factors things into two steps
  - Off-line transport simulation that is independent of specific lighting environment
  - Simple run time component that depends on specific lighting environment

General Scene Response



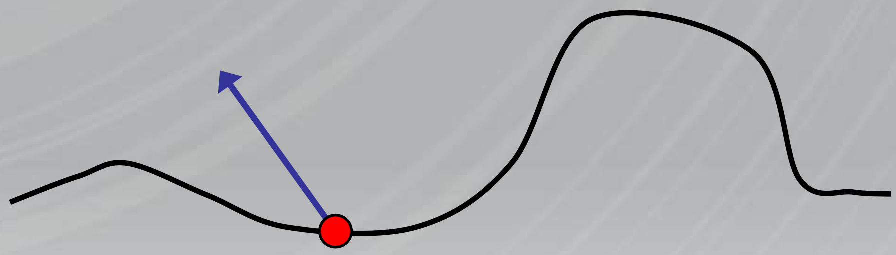
Transfer Vector

Rendering Equation

$$L\left(p \to \vec{d}\right) = L_e\left(p \to \vec{d}\right) + \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L\left(p \leftarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

Rendering Equation

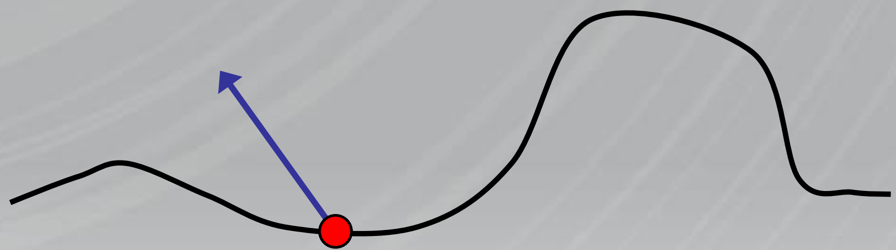
$$\boxed{L\left(p \to \vec{d}\right)} = L_e\left(p \to \vec{d}\right) + \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L\left(p \leftarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$



Radiance leaving point ***p*** in direction ***d***

Rendering Equation

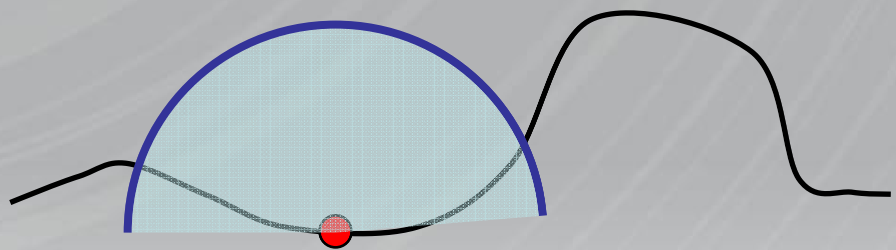
$$L\left(p \to \vec{d}\right) = \boxed{L_e\left(p \to \vec{d}\right)} + \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L\left(p \leftarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$



Radiance emitted from point ***p*** in direction ***d***

Rendering Equation

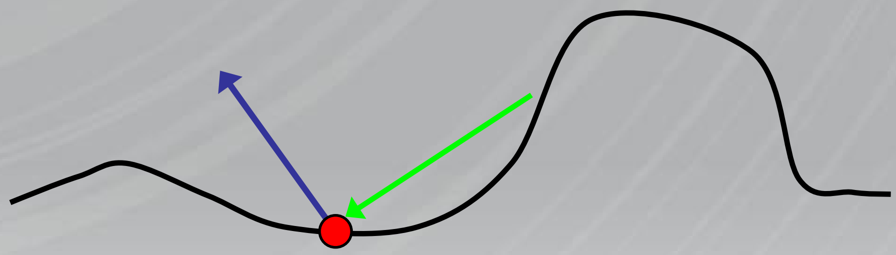
$$L\left(p \rightarrow \vec{d}\right) = L_e\left(p \rightarrow \vec{d}\right) + \int_{\Omega} f_r\left(p, \vec{s} \rightarrow \vec{d}\right) L\left(p \leftarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$



Integral over directions ***s*** on the hemisphere around ***p***

Rendering Equation

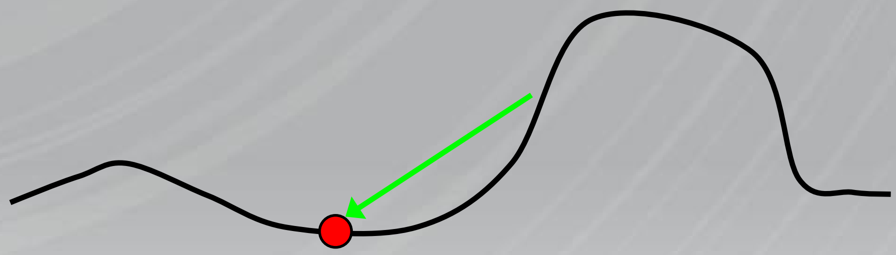
$$L\left(p \to \vec{d}\right) = L_e\left(p \to \vec{d}\right) + \int_{\Omega} \boxed{f_r\left(p, \vec{s} \to \vec{d}\right)} L\left(p \leftarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$



BRDF at point ***p*** evaluated for incident direction ***s*** in outgoing direction ***d***

Rendering Equation

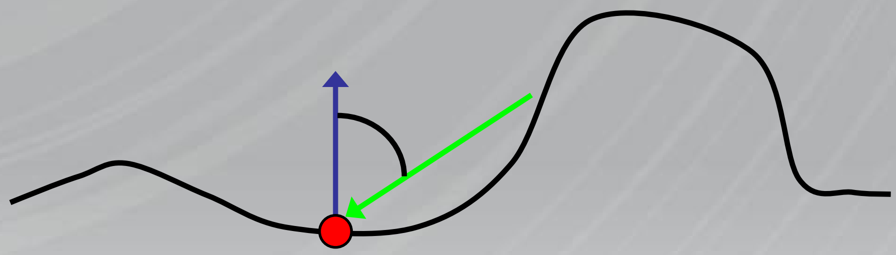
$$L\left(p\to\vec{d}\right)=L_e\left(p\to\vec{d}\right)+\int_{\Omega} f_r\left(p,\vec{s}\to\vec{d}\right)L\left(p\leftarrow\vec{s}\right)H_{N_p}\left(-\vec{s}\right)ds$$



Radiance arriving at point ***p*** from direction ***s*** (also LHS)

Rendering Equation

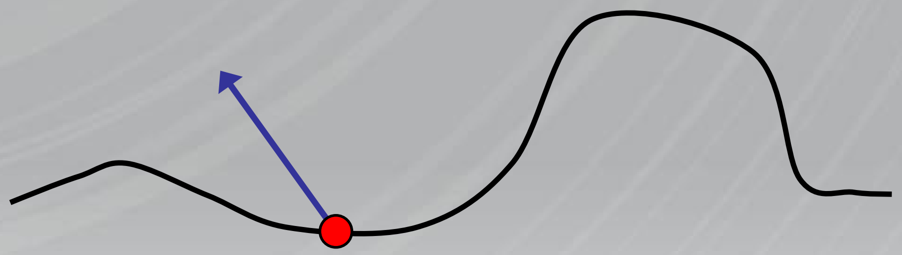
$$L\left(p \to \vec{d}\right) = L_e\left(p \to \vec{d}\right) + \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L\left(p \leftarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$



Lamberts law – cosine between normal and -s = dot(Np, -s)

Neumann Expansion

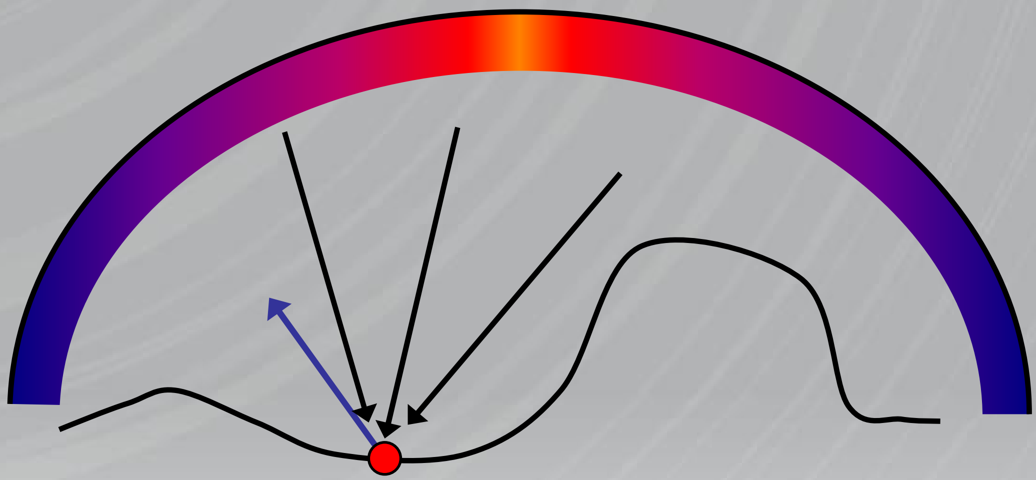
$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$



Exit radiance expressed as infinite series

Neumann Expansion

$$L\left(p \to \vec{d}\right) = \boxed{L_0\left(p \to \vec{d}\right)} + L_1\left(p \to \vec{d}\right) + \cdots$$

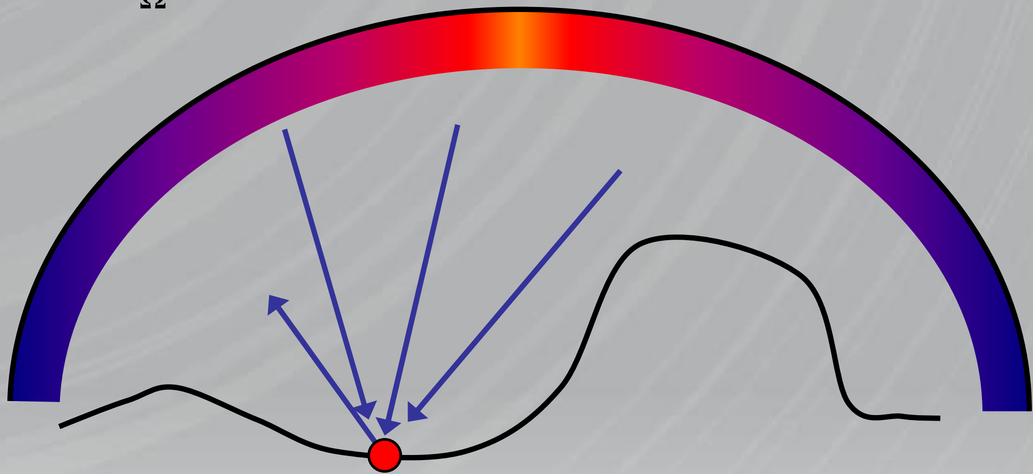


Direct lighting arriving at point ***p*** – from distant environment

Neumann Expansion

$$L\left(p \rightarrow \vec{d}\right) = L_0\left(p \rightarrow \vec{d}\right) + L_1\left(p \rightarrow \vec{d}\right) + \cdots$$

$$L_0\left(p \rightarrow \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \rightarrow \vec{d}\right) L_S\left(p \leftarrow \vec{s}\right) V\left(p \rightarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

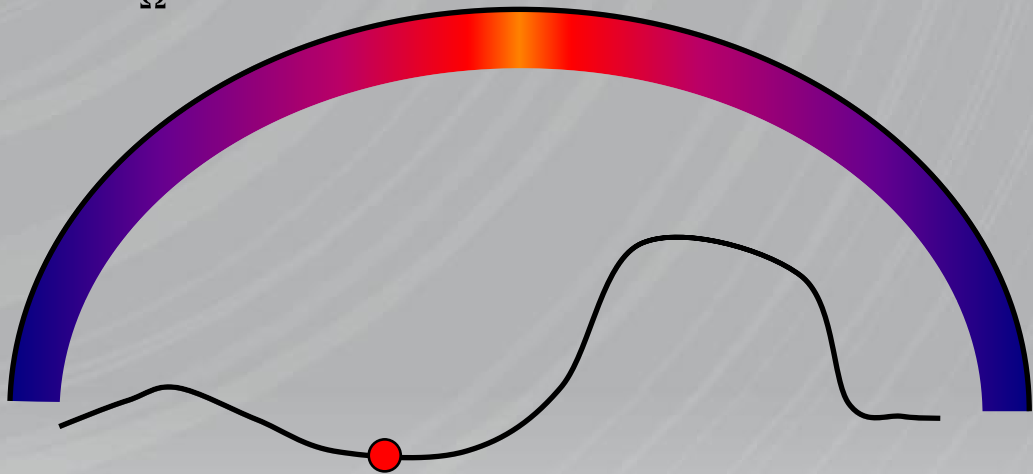


Direct lighting arriving at point ***p*** – from distant environment

Neumann Expansion

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_0\left(p \to \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L_S\left(p \leftarrow \vec{s}\right) V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

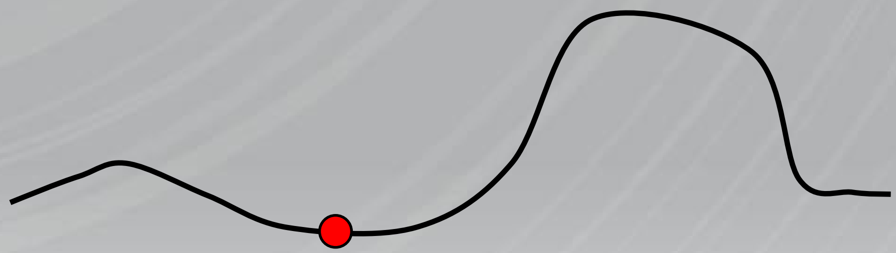


Source Radiance – distant lighting environment

Neumann Expansion

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_0\left(p \to \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L_S\left(p \leftarrow \vec{s}\right) V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

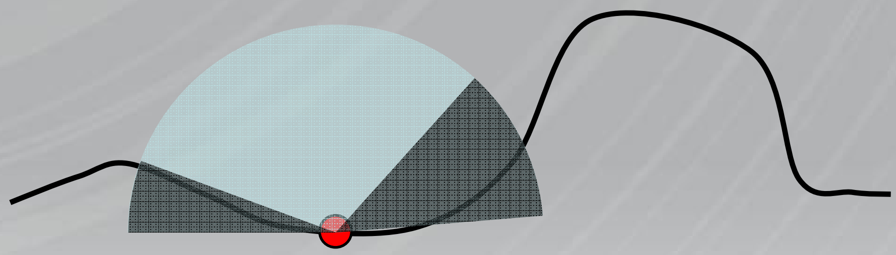


Visibility function - binary

Neumann Expansion

$$L\left(p \rightarrow \vec{d}\right) = L_0\left(p \rightarrow \vec{d}\right) + L_1\left(p \rightarrow \vec{d}\right) + \cdots$$

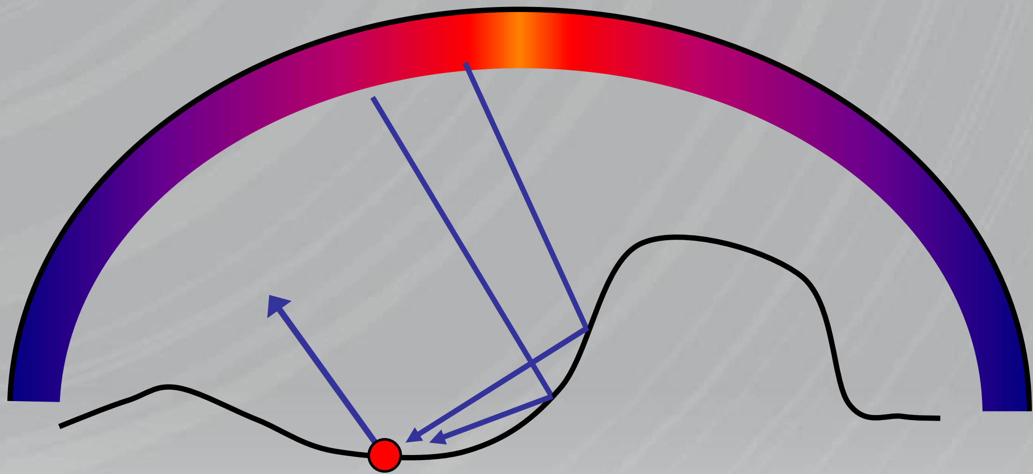
$$L_0\left(p \rightarrow \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \rightarrow \vec{d}\right) L_S\left(p \leftarrow \vec{s}\right) V\left(p \rightarrow \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$



Visibility function - binary

Neumann Expansion

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

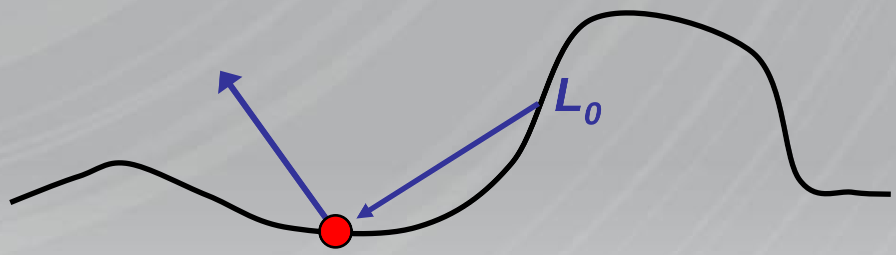


All paths from source that take 1 bounce

Neumann Expansion

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_1\left(p \to \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L_0\left(p \leftarrow \vec{s}\right)\left(1 - V\left(p \to \vec{s}\right)\right) H_{N_p}\left(-\vec{s}\right) ds$$

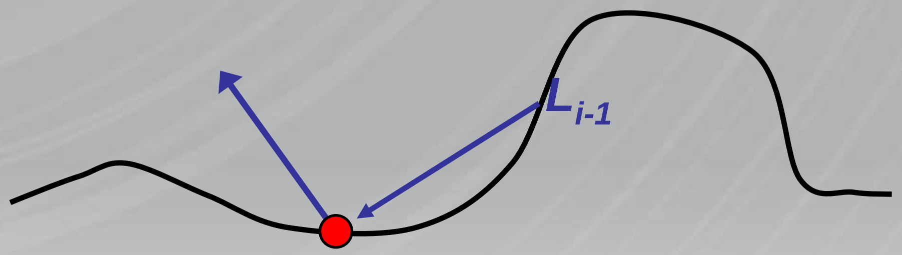


All paths from source that take 1 bounce

Neumann Expansion

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_i\left(p \to \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) \boxed{L_{i-1}\left(p \leftarrow \vec{s}\right)} \left(1 - V\left(p \to \vec{s}\right)\right) H_{N_p}\left(-\vec{s}\right) ds$$

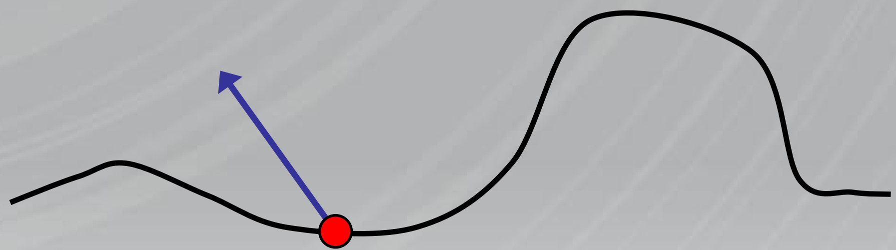


All paths from source that take i bounces

Diffuse PRT

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_0\left(p \to \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) L_S\left(p \leftarrow \vec{s}\right) V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

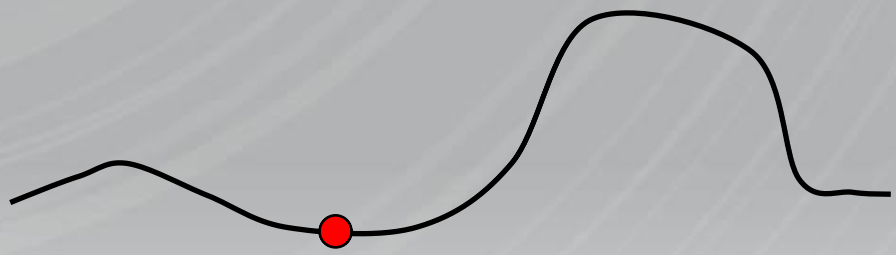


Diffuse PRT

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$\boxed{L_0\left(p \to \vec{d}\right)} = \int_\Omega f_r\left(p, \vec{s} \to \vec{d}\right) L_S\left(p \leftarrow \vec{s}\right) V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

$$\boxed{L_0\left(p\right)} = \frac{\rho_d}{\pi} \int_\Omega L_S\left(-\vec{s}\right) V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

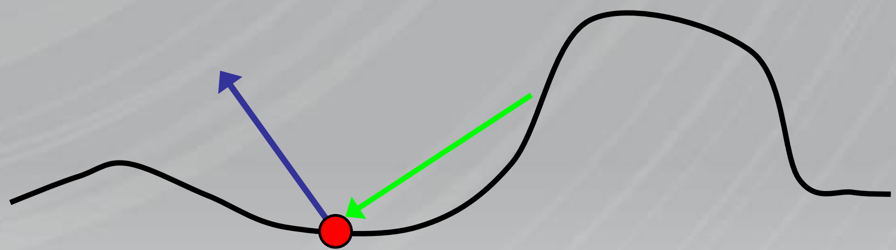


Diffuse PRT

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_0\left(p \to \vec{d}\right) = \int_\Omega \boxed{f_r\left(p, \vec{s} \to \vec{d}\right)} L_S\left(p \leftarrow \vec{s}\right) V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

$$L_0\left(p\right) = \boxed{\frac{\rho_d}{\pi}} \int_\Omega L_S\left(-\vec{s}\right) V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

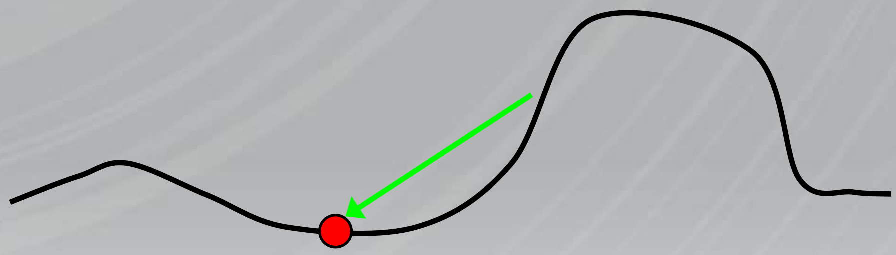


Diffuse PRT

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L_0\left(p \to \vec{d}\right) = \int_{\Omega} f_r\left(p, \vec{s} \to \vec{d}\right) \boxed{L_S\left(p \leftarrow \vec{s}\right)} V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$

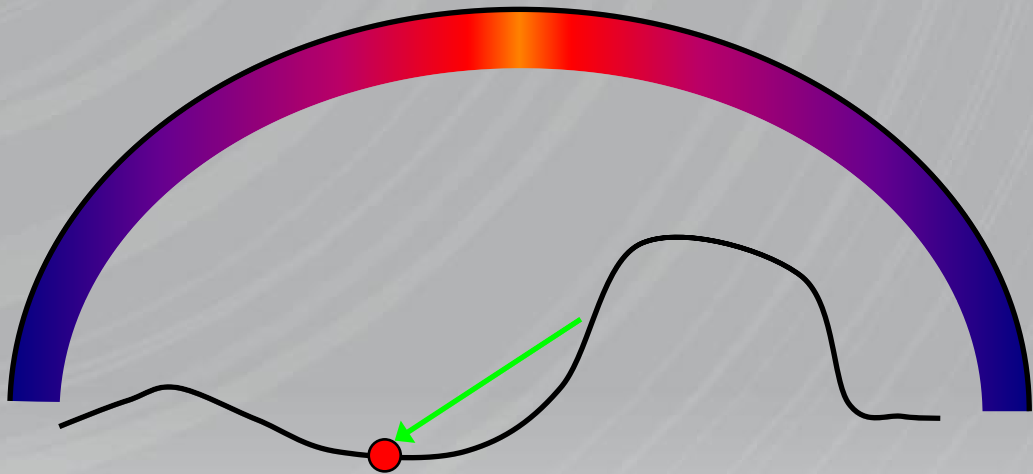
$$L_0\left(p\right) = \frac{\rho_d}{\pi} \int_{\Omega} \boxed{L_S\left(-\vec{s}\right)} V\left(p \to \vec{s}\right) H_{N_p}\left(-\vec{s}\right) ds$$



Diffuse PRT

$$L_0(p) = \frac{\rho_d}{\pi} \int_{\Omega} \boxed{L_S(-\vec{s})} V(p \rightarrow \vec{s}) H_{N_p}(-\vec{s}) ds$$

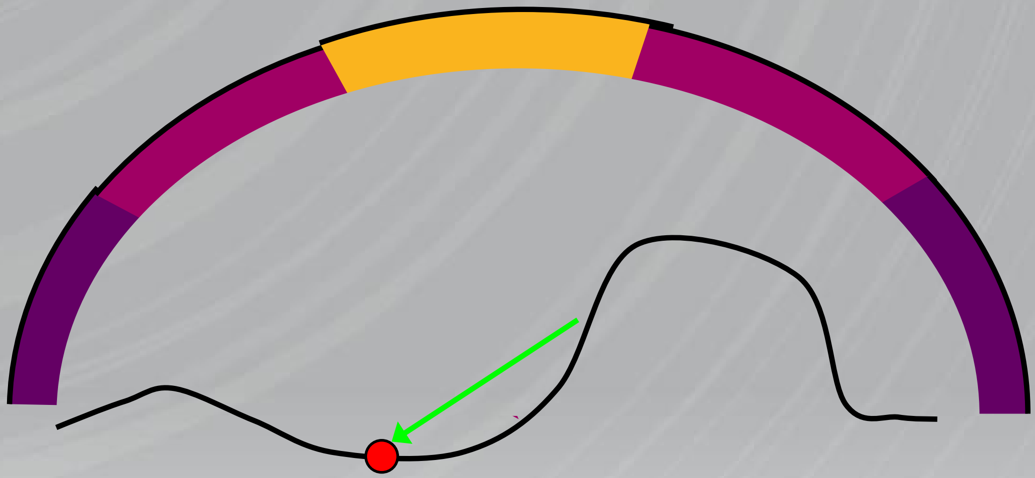
$$\boxed{L_S(-\vec{s}) \approx \sum_i l_i Y_i(-\vec{s})}$$



Diffuse PRT

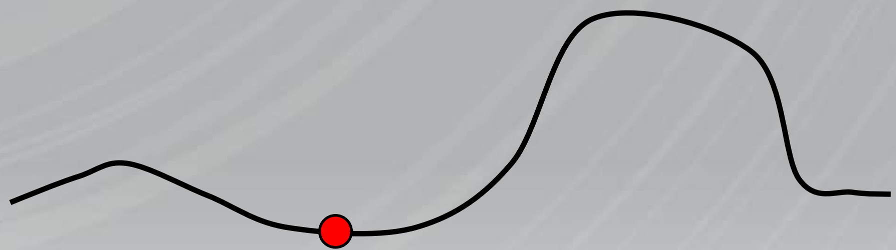
$$L_0(p) = \frac{\rho_d}{\pi} \int_{\Omega} L_S(-\vec{s}) V(p \to \vec{s}) H_{N_p}(-\vec{s}) \, ds$$

$$L_S(-\vec{s}) \approx \sum_i l_i Y_i(-\vec{s})$$



Diffuse PRT

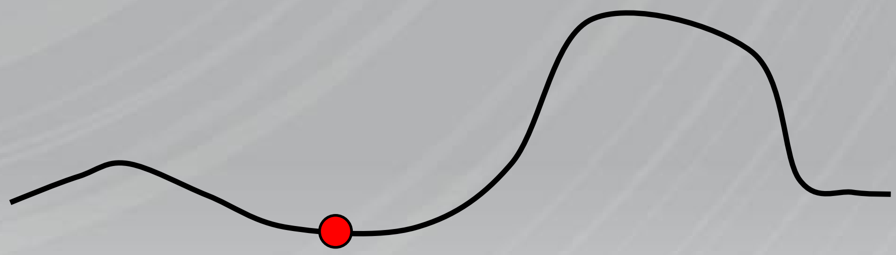
$$L_0(p) = \frac{\rho_d}{\pi} \int_{\Omega} \left(\sum_i l_i Y_i(-\vec{s}) \right) V(p \to \vec{s}) H_{N_p}(-\vec{s})\, ds$$



Diffuse PRT

$$L_0(p) = \frac{\rho_d}{\pi} \int_{\Omega} \left(\sum_i l_i Y_i(-\vec{s}) \right) V(p \to \vec{s}) H_{N_p}(-\vec{s}) ds$$

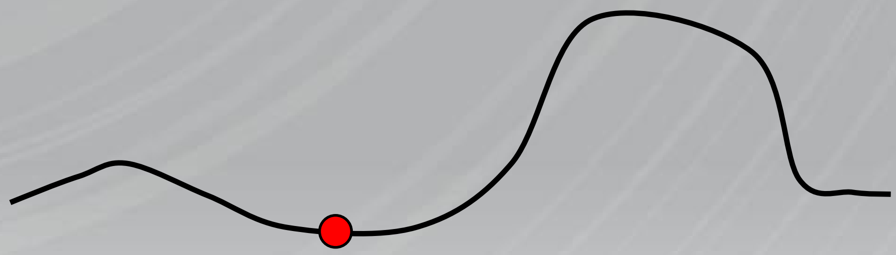
$$L_0(p) = \frac{\rho_d}{\pi} \sum_i l_i \int_{\Omega} Y_i(-\vec{s}) V(p \to \vec{s}) H_{N_p}(-\vec{s}) ds$$



Diffuse PRT

$$L_0(p) = \frac{\rho_d}{\pi} \int_\Omega \left(\sum_i l_i Y_i(-\vec{s}) \right) V(p \to \vec{s}) H_{N_p}(-\vec{s}) ds$$

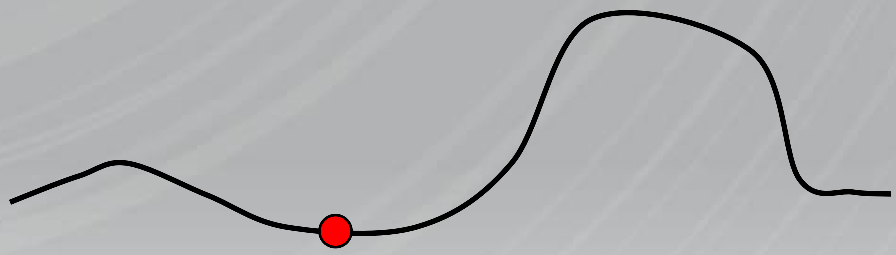
$$L_0(p) = \frac{\rho_d}{\pi} \sum_i l_i \int_\Omega Y_i(-\vec{s}) V(p \to \vec{s}) H_{N_p}(-\vec{s}) ds$$



Diffuse PRT

$$L_0(p) = \frac{\rho_d}{\pi} \sum_i l_i \int_{\Omega} Y_i(-\vec{s}) V(p \rightarrow \vec{s}) H_{N_p}(-\vec{s}) ds$$

$$L_0(p) = \frac{\rho_d}{\pi} \sum_i l_i t_{pi}^0$$

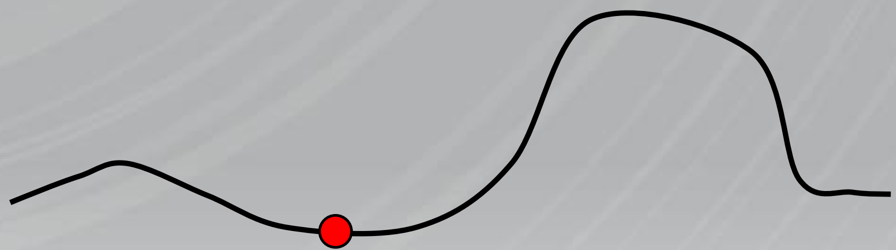


Diffuse PRT

$$L_0(p) = \frac{\rho_d}{\pi} \sum_i l_i \int_\Omega Y_i(-\vec{s}) V(p \to \vec{s}) H_{N_p}(-\vec{s}) ds$$

$$L_0(p) = \frac{\rho_d}{\pi} \sum_i l_i t_{pi}^0$$

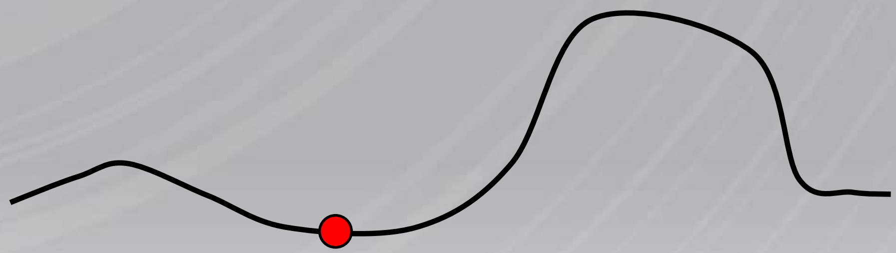
$$L_0(p) = \sum_i l_i t_{pi}^0$$



Diffuse PRT

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L(p) = \sum_i l_i \left(t_{pi}^0 + t_{pi}^1 + \cdots\right)$$

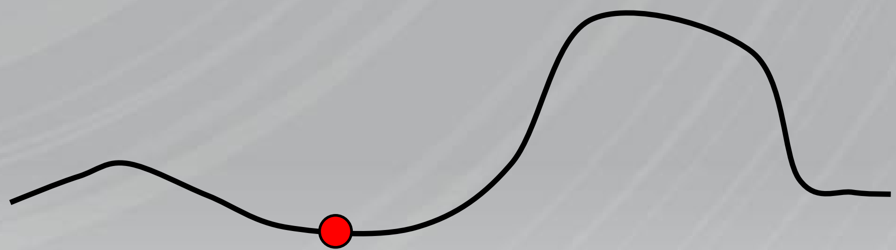


Diffuse PRT

$$L\left(p \to \vec{d}\right) = L_0\left(p \to \vec{d}\right) + L_1\left(p \to \vec{d}\right) + \cdots$$

$$L(p) = \sum_i l_i \left(t_{pi}^0 + t_{pi}^1 + \cdots\right)$$

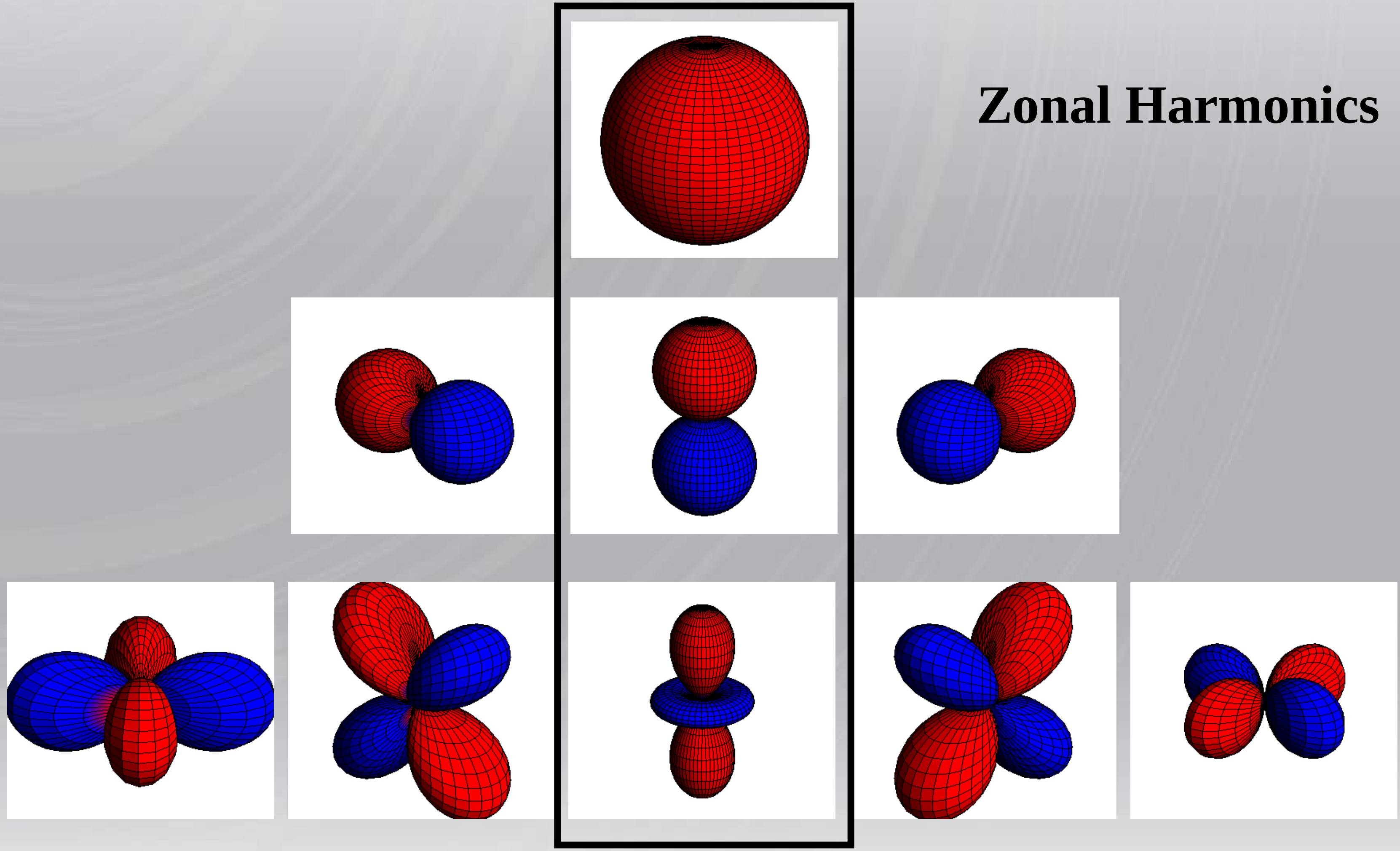
$$L(p) = \sum_i l_i t_{pi}$$



Spherical Harmonics

- Can be used to represent signals over the sphere
  - General form is for complex signals, real form used for graphics
- Analogous to the Fourier basis
- Can be expressed as trigonometric functions of theta/phi
  - Mostly just useful for derivations, computing analytic formulas
- Also represented as polynomials of coordinates of a point on the unit sphere

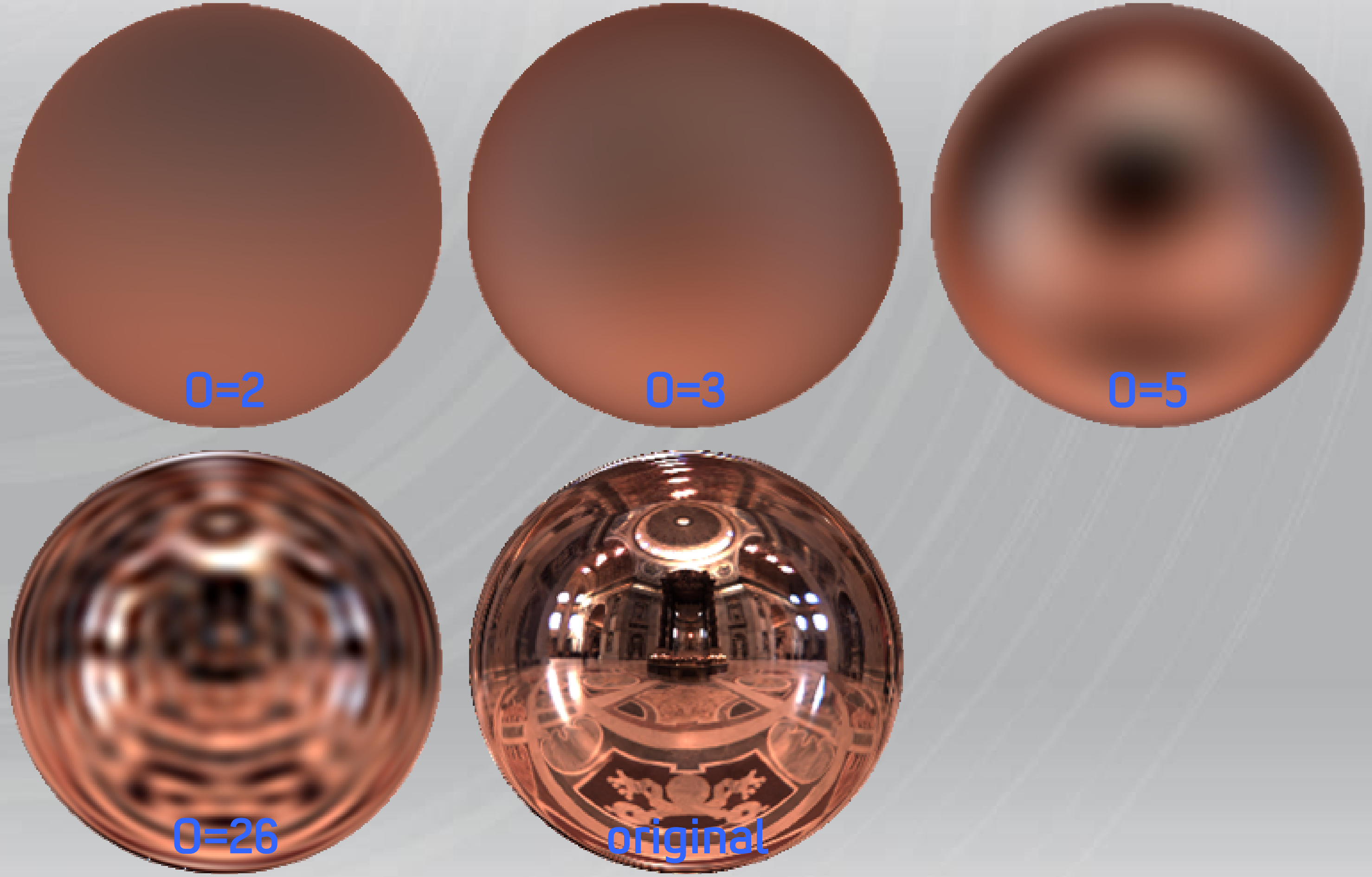
Spherical Harmonics



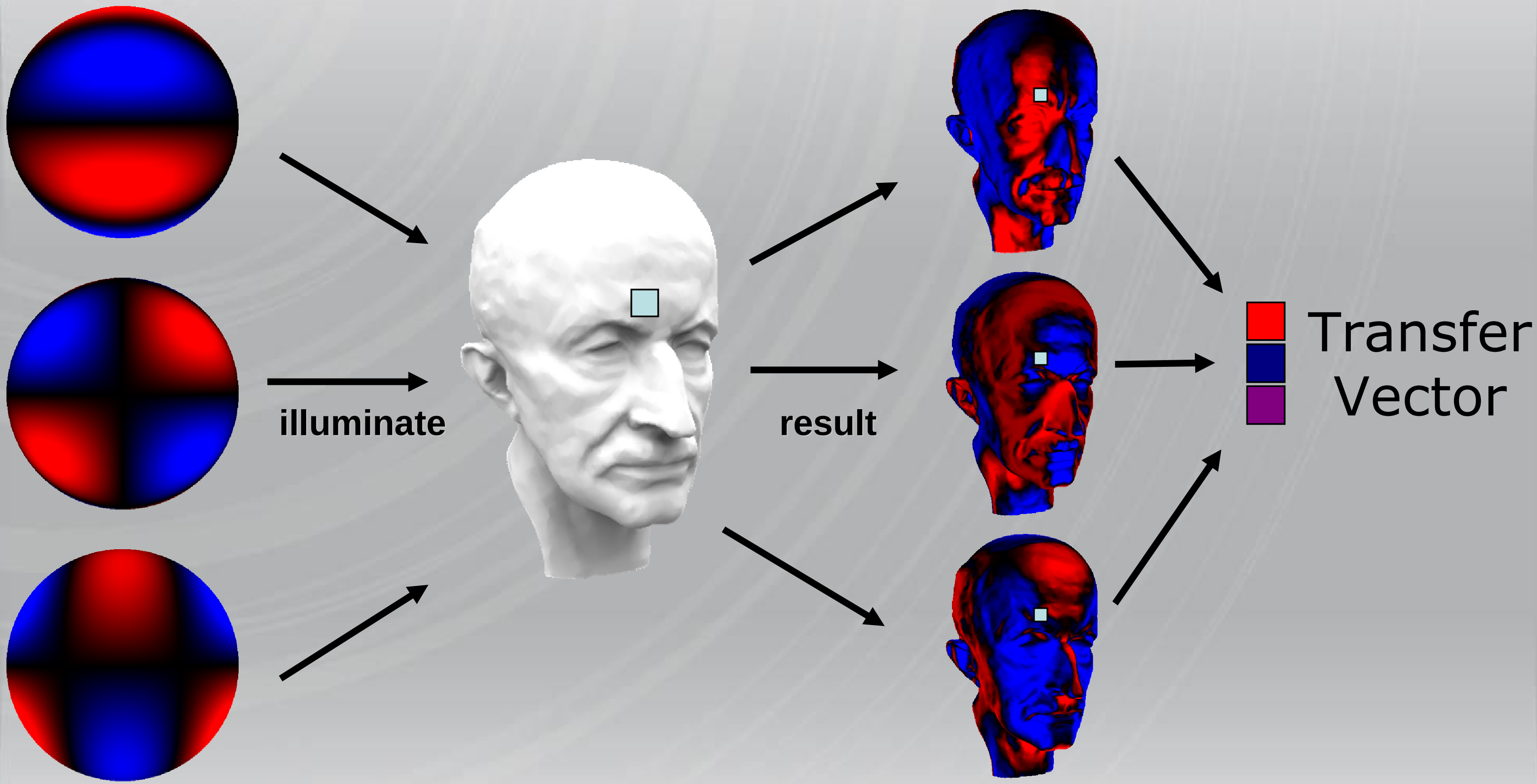
SH Properties

- Basis functions are orthogonal
  - Integrate against themselves = 1
  - Integrate against other SH = 0
  - Makes projecting functions/signals simple
- Rotation invariance
  - Project(Rot(f)) = Rot(Project(f))
  - No aliasing as an object rotates, or lights rotate around an ojbect (ie: lights don't "wobble")
- Band limiting
  - Capture finer and finer frequencies as you add more terms (also helps with aliasing)

Spherical Harmonics



Precomputed Radiance Transfer (PRT)



Precomputed Radiance Transfer (PRT)

- Compute objects response to a given number “light basis functions” off-line
  - Include arbitrarily complex light transport
- At run time rotate lighting into frame of the object
- Dot product of transfer vector and lighting vector generates exit radiance
- Limitations
  - Assumes rigid objects (see LDPRT later)
  - Lots of data (compression)
  - Distant lighting (see gradients, PRV later)

Demo

PRT

General Scene Response

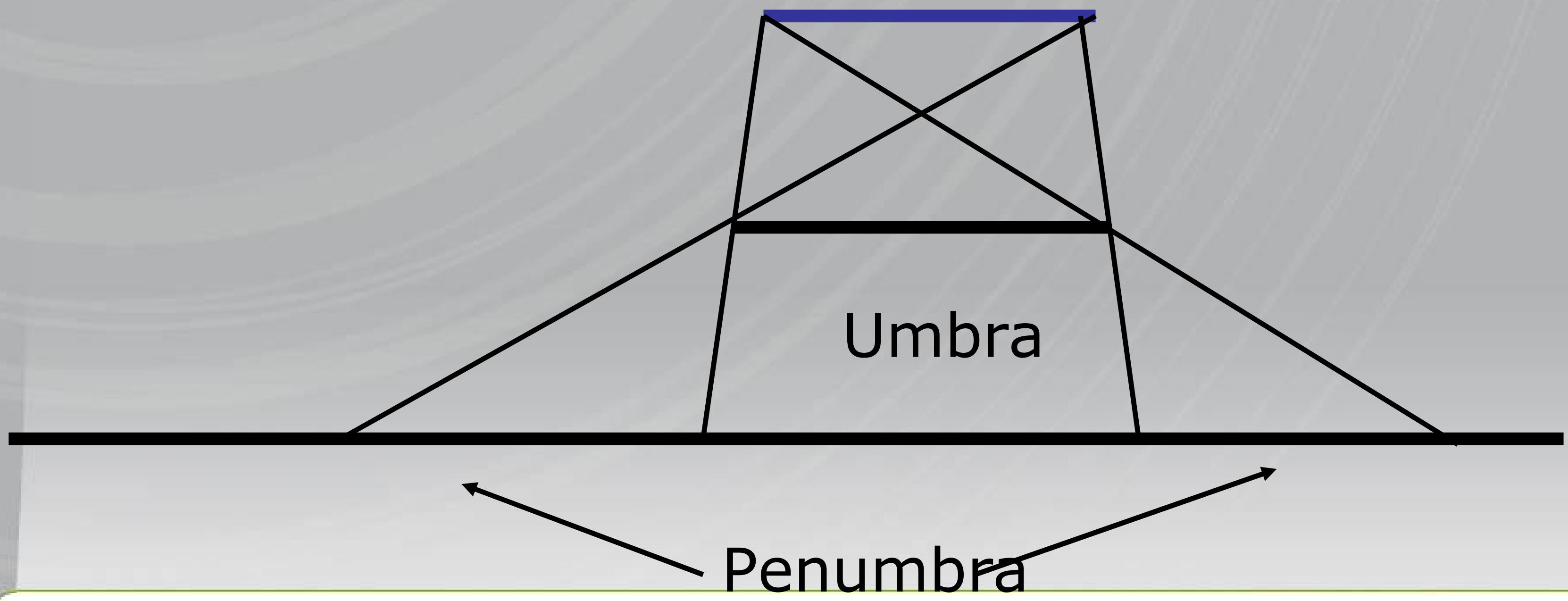
- Rendering is scaling scene response by intensity of each "light" and summing
- Dot product generates an image
  - Factored form of an integral equation
- Relies on the spatial relationship between objects in the scene to be consistent (static scene)

Precomputed Basis

- Polynomial Texture Maps
  - [Malzbender2002]
  - Bi-quadratic polynomial (2 DOF)
- Steerable Illumination Textures
  - [Ashikhmin2002]
  - Steerable basis to model path of small area light (49 DOF)
- Directional Basis
  - [Hao2003]
  - Subsurface Scattering from directional lights
- Wavelets
  - [Ng03]
  - Models “all-frequencies”
  - Extended to glossy [Liu04,Ng04,Wang04]

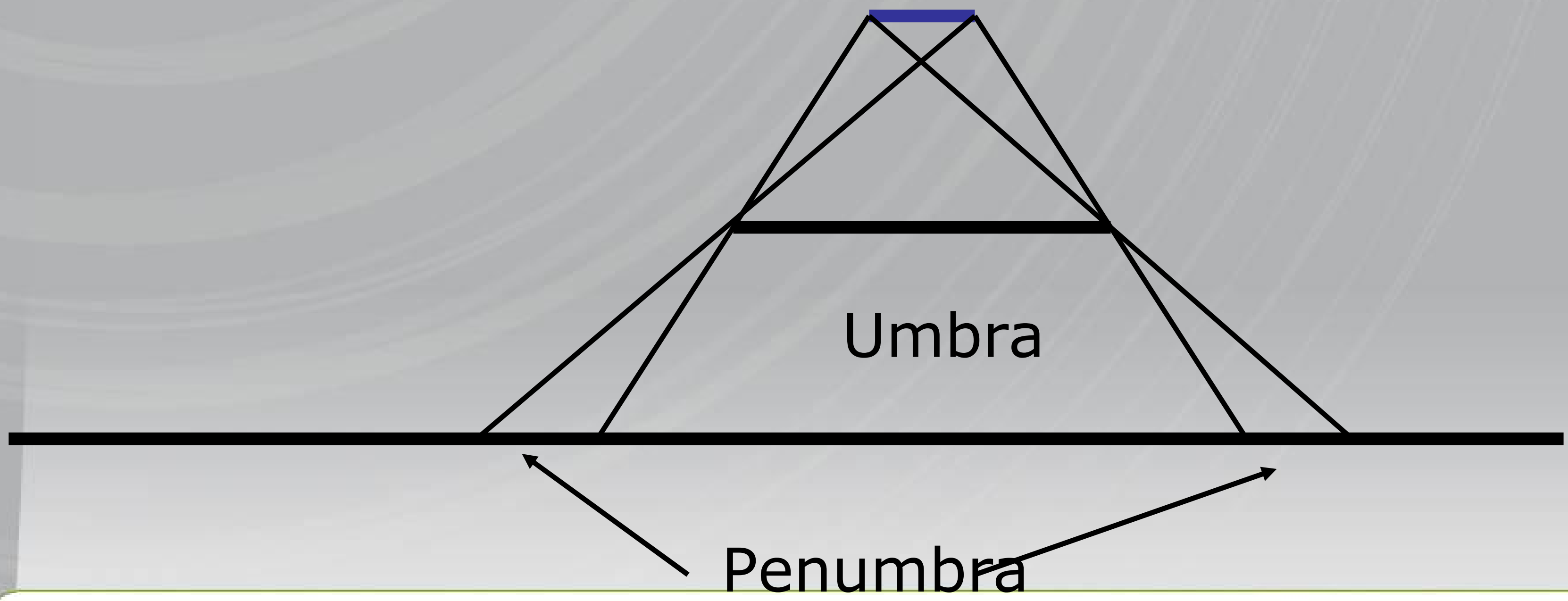
Spatial Sampling Issues

- Relationship between spatial sampling densities over objects and light frequencies



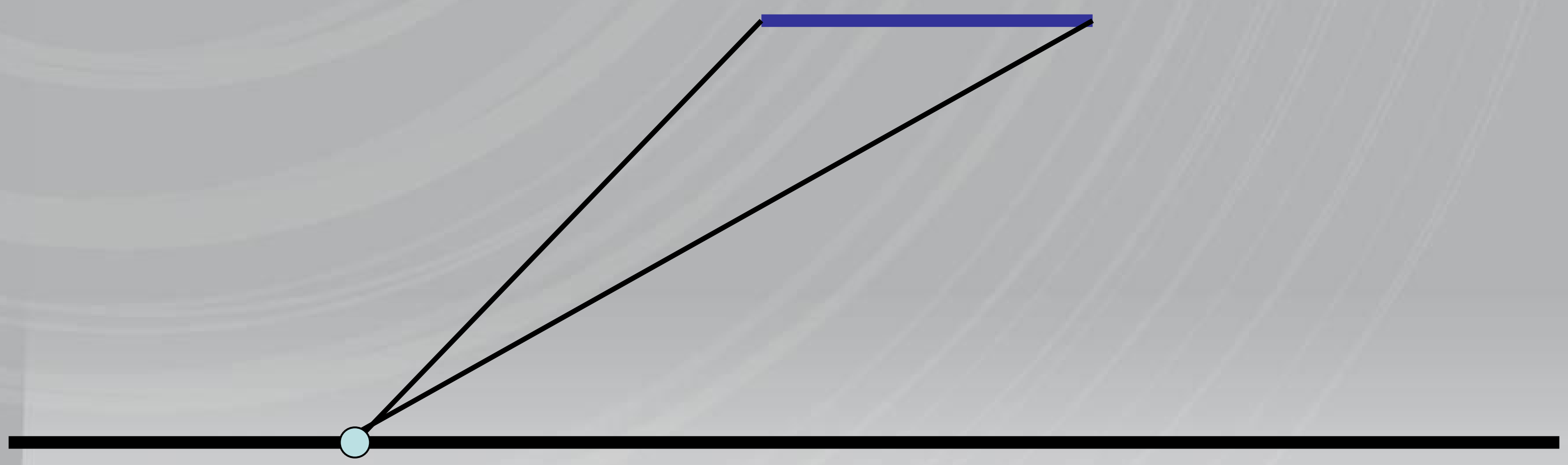
Spatial Sampling Issues

- Light shrinks -> penumbra tightens
- Higher sampling density to “move” light over object/scene



Angular Sampling Issues

- Large lights need to be sampled a lot



Angular Sampling Issues

- Small lights clearly less

Sampling Issues

- Large (low frequency) lights
  - Coarse spatial sampling
    - Not a lot of storage
  - Large solid angles
    - Run time integration would be expensive
- Small (high frequency) lights
  - Fine spatial sampling
    - High storage
  - Small solid angles
    - Run time integration isn’t that bad

Sampling Issues: Transport

- Bounced light is pretty much always low frequency
  - An illuminated wall is an area light
  - Even from a point (high frequency) light source
- Do you need to use high frequency basis to model high frequency inter-reflections???
  - Similar to duality discussed in [Ramamoorthi2001], inter-reflections are implicitly large area lights

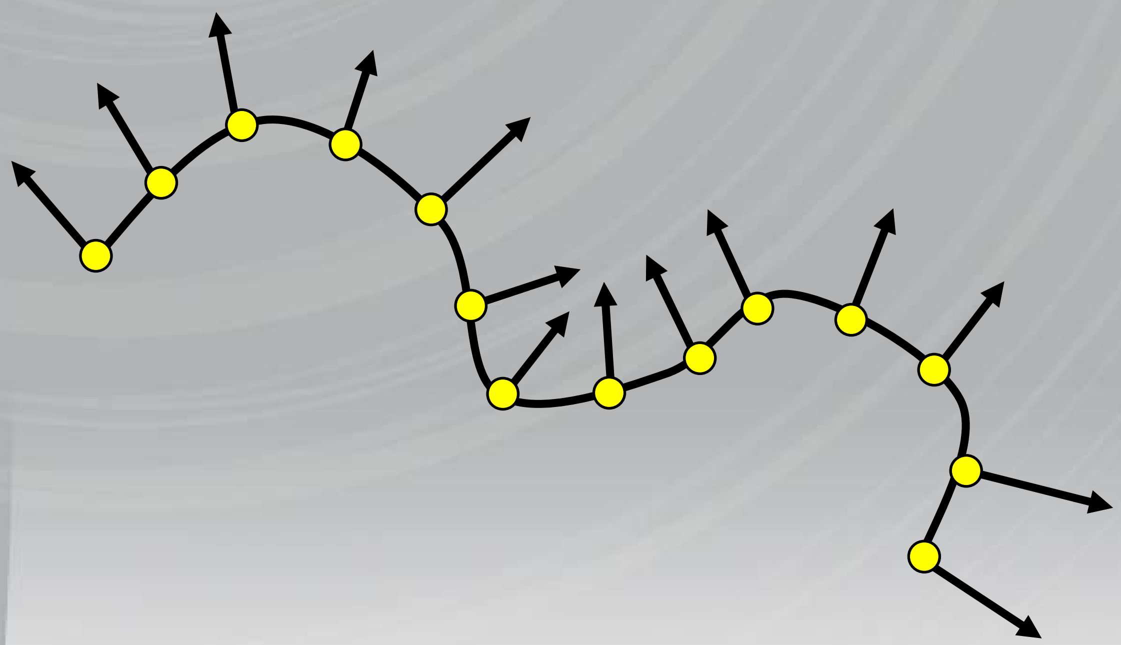
Compression Goals

- Decode efficiently
  - As much on the GPU as possible
  - Render compressed representation directly
- Increase rendering performance
  - Big dot products can be expensive
- Reduce memory consumption
  - Not just on disk

- Not just for PRT, useful for any type of signal

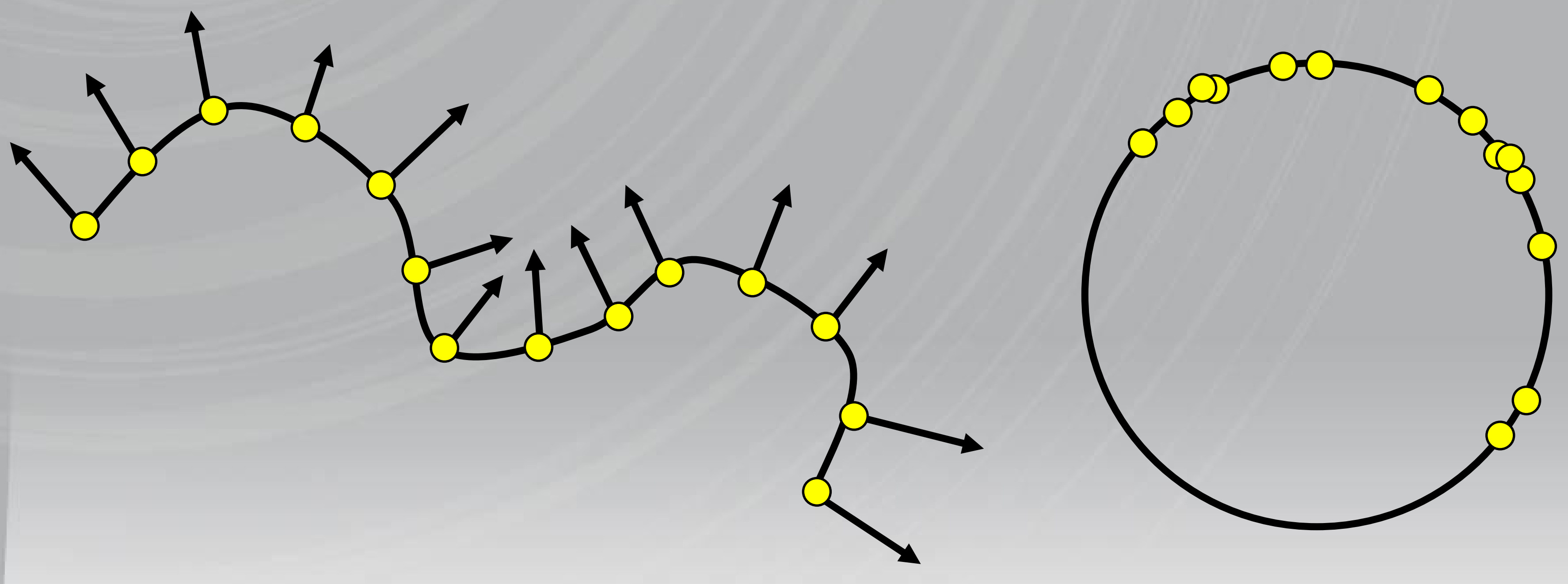
Compression Example

Surface is curve, signal is normal



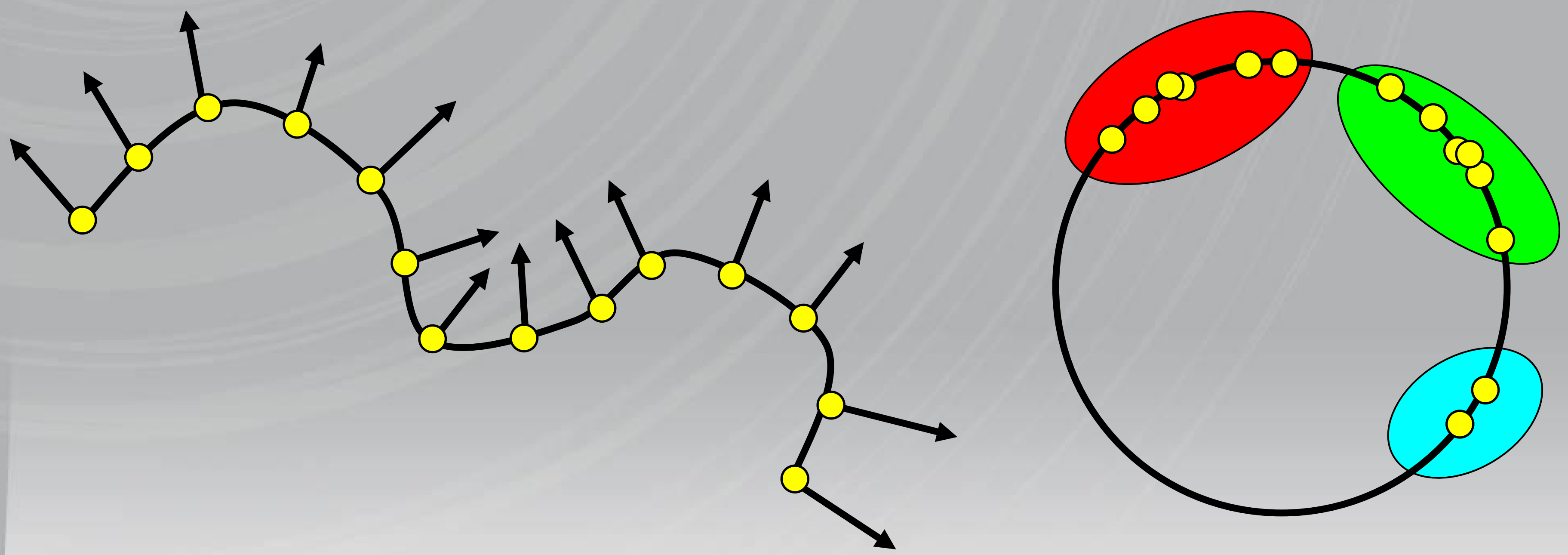
Compression Example

Signal Space



VQ

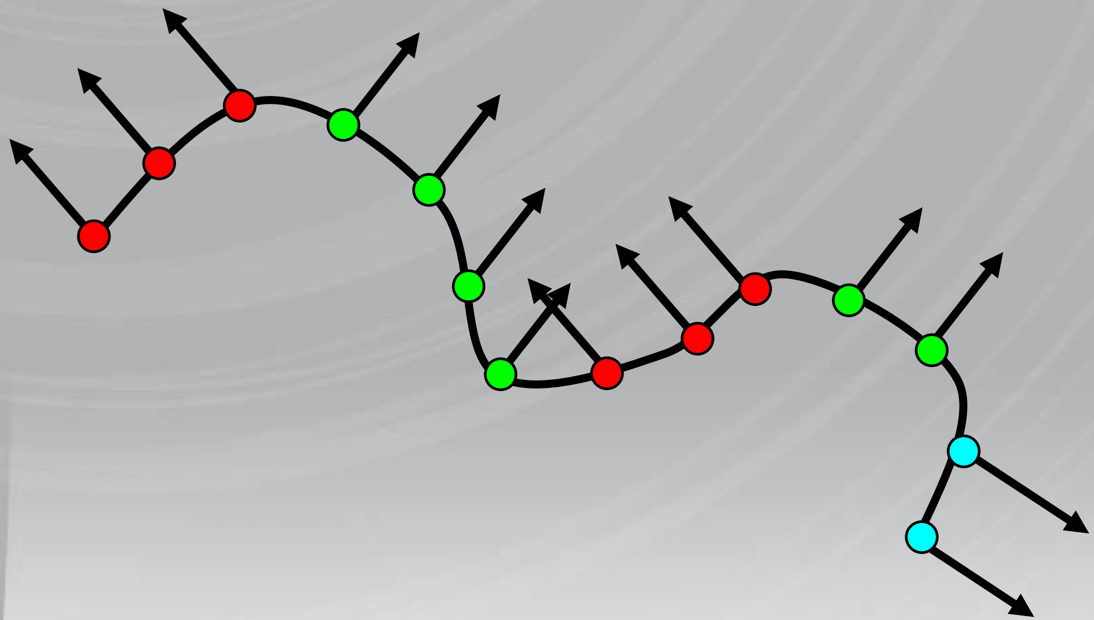
Cluster normals

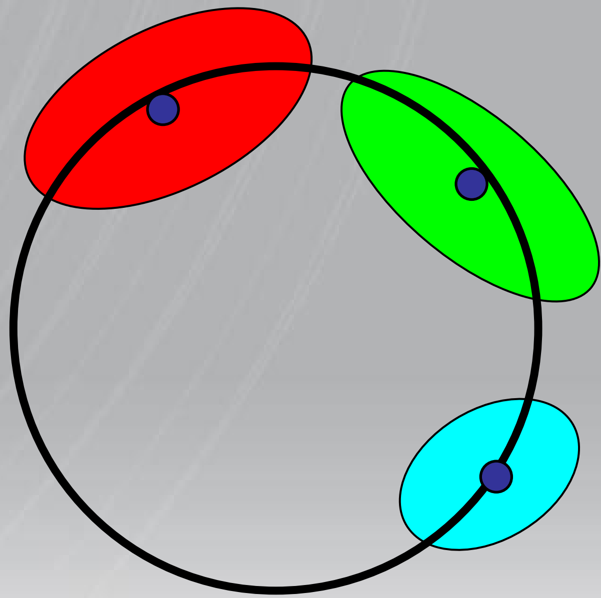


VQ

Replace samples with cluster mean

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

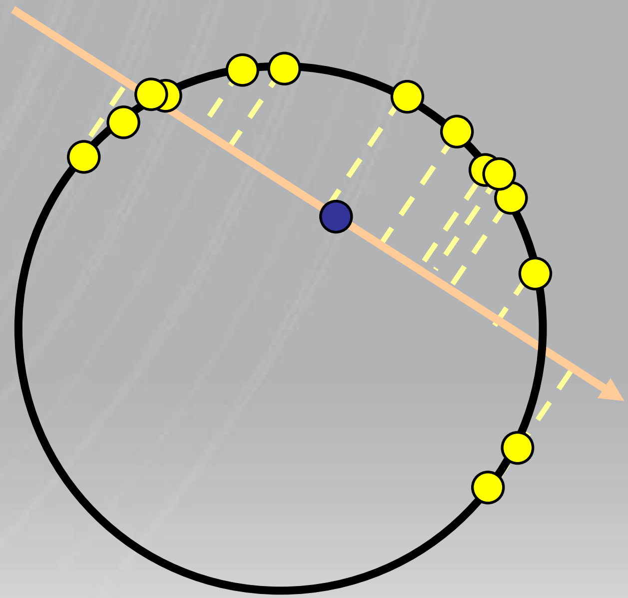




PCA

Replace samples with mean + linear combination

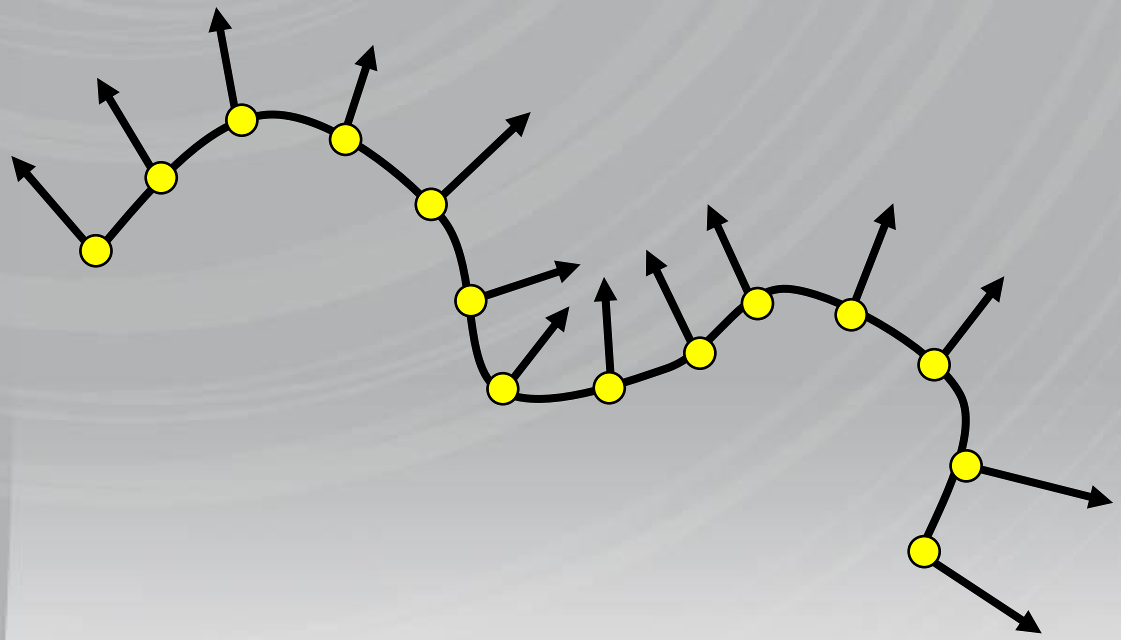
$$\mathbf{M}_p \approx \tilde{\mathbf{M}}_p = \mathbf{M}^0 + \sum_{i=1}^{N} w_p^i \mathbf{M}^i$$

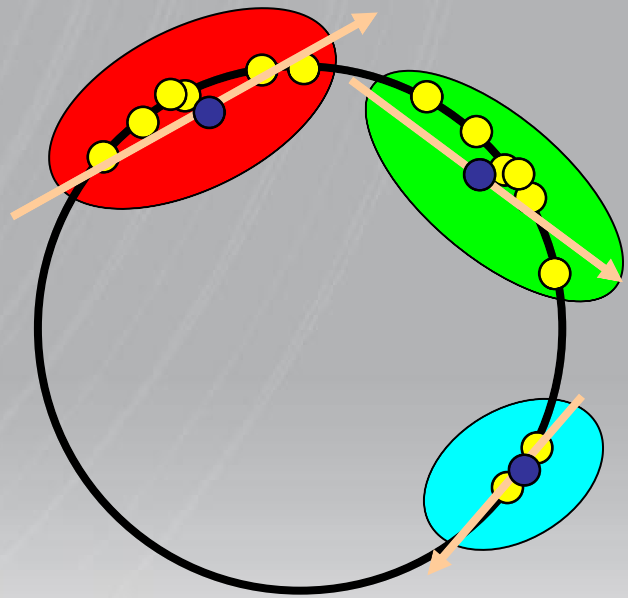


CPCA

Compute a linear subspace in each cluster

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

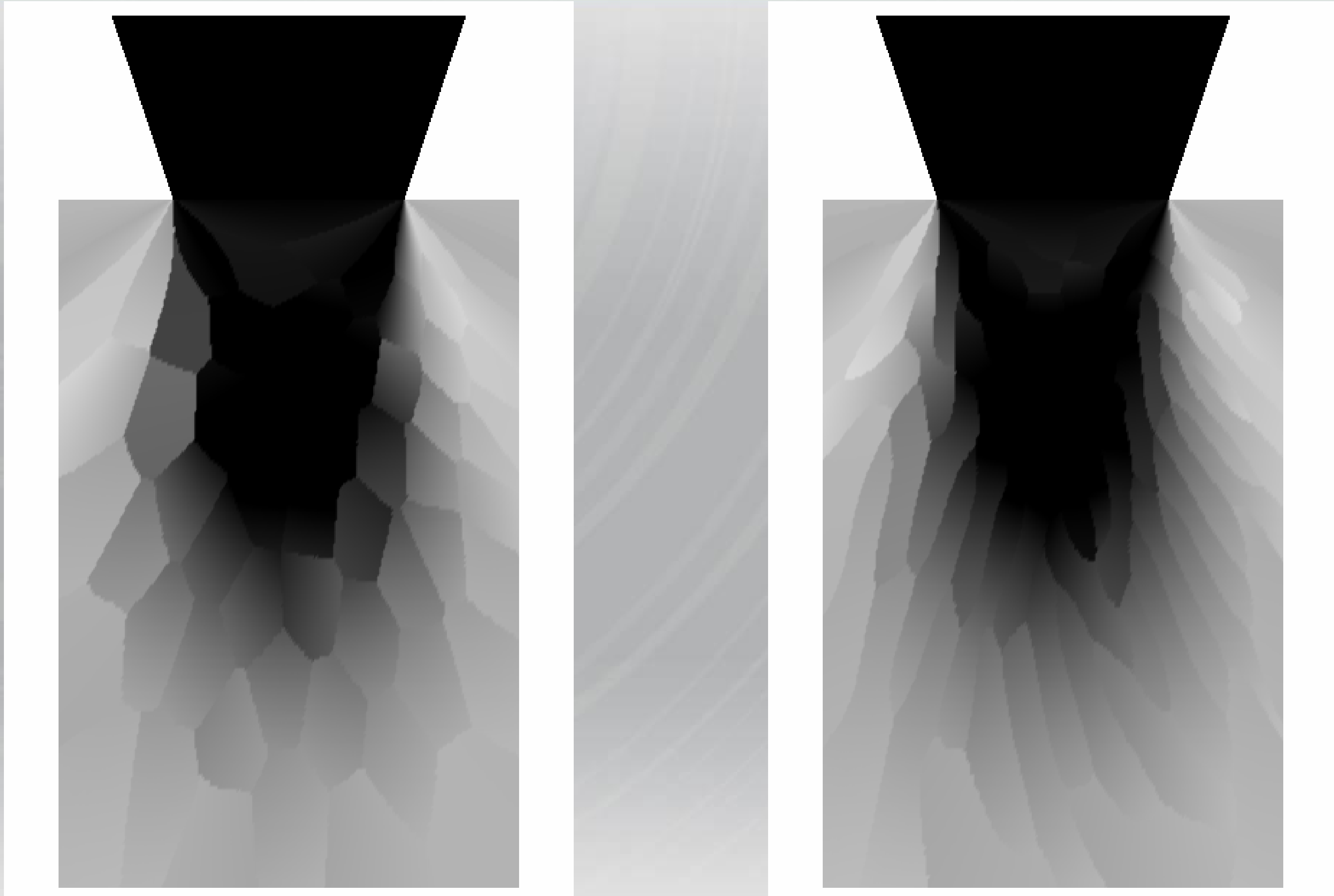




CPCA

- Clusters with low dimensional affine models
- How should clustering be done?
- 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

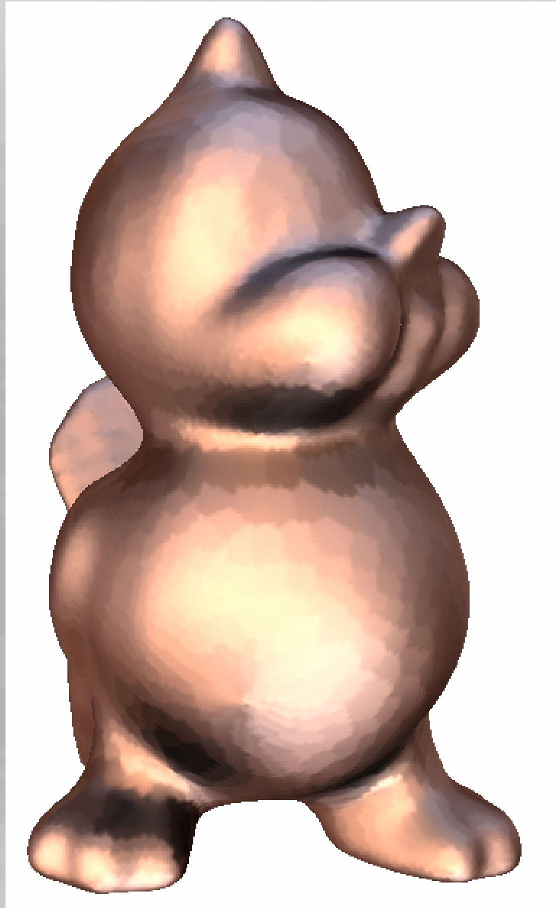
Static vs. Iterative



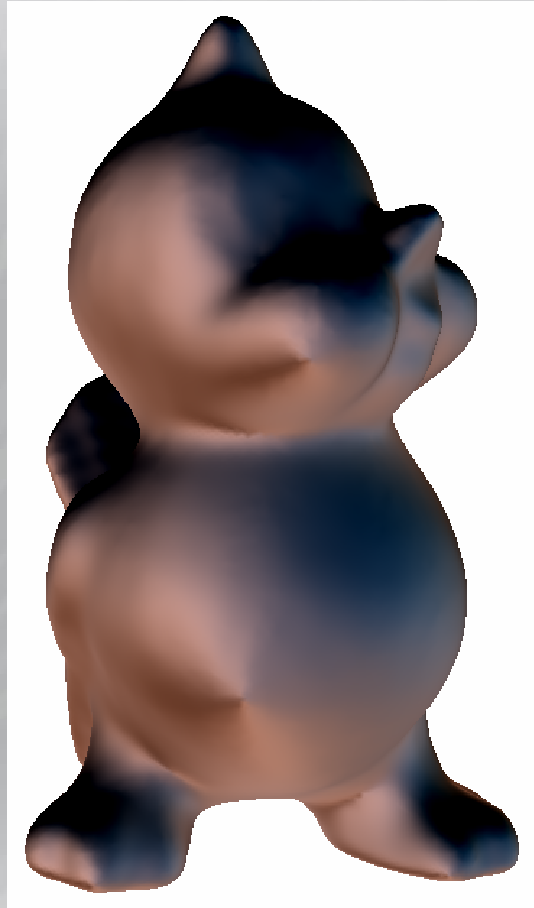
Related Work

- VQ+PCA [Kambhatla94] (static)
- VQPCA [Khambhatla97] (iterative)
- Mixture PC [Dony95] (iterative)
- Independently used with BTF's [Mueller03]
- More sophisticated models exist
  - [Brand03], [Roweis02]
  - Mapping to GPUs is challenging
    - Variable storage per vertex
    - Partitioning is more difficult (or requires more passes)
    - Worth investigating again on current GPU's – like the Xbox360

Equal Rendering Cost



VQ



PCA

CPCA

How Compression Improves Rendering

- Original expression

$$e_p = \mathbf{T}_p \bullet \mathbf{L}$$

- Approximated with compression

$$e_p \approx \left(\mathbf{T}_{C_p}^0 + \sum_{i=1}^{N} w_p^i \mathbf{T}_{C_p}^i \right) \bullet \mathbf{L}$$

- Looks more expensive
- “cluster” work is constant across vertices/texels

How Compression Improves Rendering

- Factor, do highlighted portion on the CPU

$$e_p \approx \boxed{\left(\mathbf{T}_{C_p}^{0} \bullet \mathbf{L}\right)} + \sum_{i=1}^{N} w_p^i \boxed{\left(\mathbf{T}_{C_p}^{i} \bullet \mathbf{L}\right)}$$

- Rendering only depends on number of PCA vectors in a cluster
- Anything that is “linear” can be optimized this way
  - Combination of light maps, significantly reduce compute and storage, change intensities on the fly

PRT HLSL shader

```
float4 vAccumR = 0, vAccumG = 0, vAccumB = 0;
for (int i=0; i < (NUM_PCA/4); i++)
{
  vAccumR += vPCAWeights[i] * aConsts[nOffset+1+(NUM_PCA/4)*0+i];
  vAccumG += vPCAWeights[i] * aConsts[nOffset+1+(NUM_PCA/4)*1+i];
  vAccumB += vPCAWeights[i] * aConsts[nOffset+1+(NUM_PCA/4)*2+i];
}

float4 vDiffuse = aConsts[nOffset];
vDiffuse.r += dot(vAccumR,1);
vDiffuse.g += dot(vAccumG,1);
vDiffuse.b += dot(vAccumB,1);
```

- Very lossy (4 PCA)
  - 11 instructions
  - NUM_CLUSTERS * 4 consts
  - 1 short4 + 1 byte per vertex
- Less lossy (12 PCA)
  - 17 instructions
  - NUM_CLUSTERS * 10 consts
  - 3 short4 + 1 byte per vertex
- DX SDK sample for details

What is a Transfer Vector?

- Each coefficient models how light expressed in the corresponding basis function contributes to exit radiance at a point
- Spherical function that integrates against distant lighting to compute exit radiance
  - Dot product generates exit radiance

Why Animating PRT is Hard

- Lighting and Transfer Vector have to be expressed in same coordinate system
  - For rigid objects just rotate the light once
  - For skinned characters you could compute rotated lighting at each bone and blend Rotating SH is expensive
- Trade off accuracy in transfer vector with efficient rotation…

Local Deformable PRT

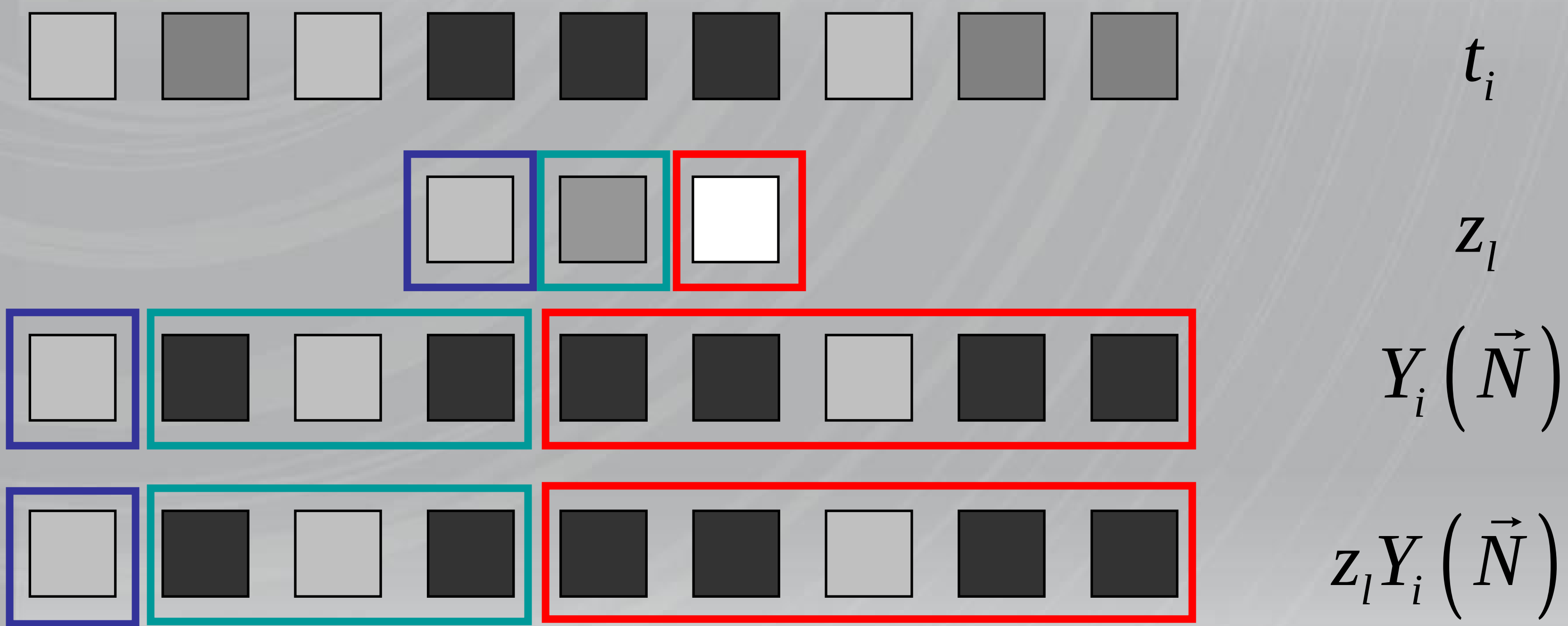
- Irradiance Environment Maps are easy to rotate, but they don’t model GI effects
- PRT Models complex GI effects, but hard to rotate
- We want something in between – easy to rotate + handles some amount of GI effects

Local Deformable PRT

- Functions with circular symmetry around the Z axis project only into the Zonal Harmonic basis functions
  - These are the only class of functions that the SH convolution theorem can be applied to
- Evaluating basis functions in a direction (projection of a rotated delta function) generates a rotated form of the Zonal Harmonic basis functions (due to rotational invariance of SH)
- Given any circular symmetric function (in Z), rotating is just evaluating basis functions and scaling by ZH coefficients for given band

Local Deformable PRT

- Shading normal + coefficient per band

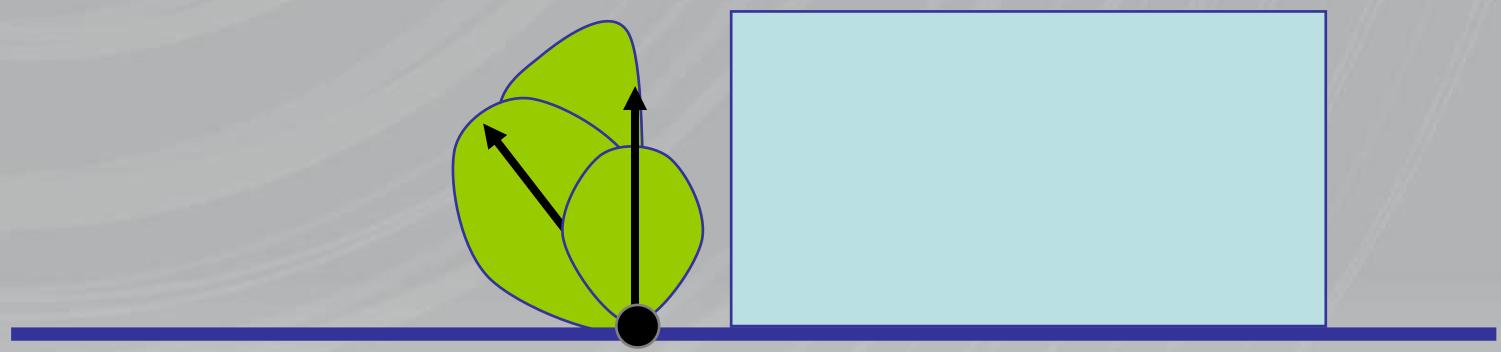


How Does LDPRT Work?

- Approximates transfer vector as a spherical function that has circular symmetry (around the shading normal) – zonal harmonic
- Just application of SH convolution theorem
- In between irradiance environment maps and PRT
- Shading normal from linear terms (direction of "maximal visibility")
- For a given shading normal, ZH coefficients that minimize squared error can be computed in closed form

Shading Normals

- Often used with ambient occlusion (sometimes called "bent normals")



How to use LDPRT?

- Can be used as a replacement for normal maps
  - No unique parameterization required
  - Works better for “local” surface properties
- Can be synthesized from scratch
  - Build leaf/skin textures for example
- Can be used instead of PRT
  - Less accurate transfer approximation

Demo

LDPRT Single Lobe

LDPRT Multiple Lobes

- One lobe only can represent circularly symmetric functions
  - Works ok for some textures, not as well for others
- Fit multiple lobes to approximate transfer vector
  - More data/reconstruction costs
  - Siggraph2005 paper
- Non-linear optimization problem
  - Used analytic gradients of objective function and BFGS
  - Fairly well behaved, but standard techniques are useful
    - Iteratively approximate a single lobe, compute residual, repeat
    - Multiple starts to avoid local minima
  - Any 3 lobe directions that aren't on a great circle can represent linears
    - >= 3 lobes, just explicitly store (easy to rotate linear)
    - 1 specific lobe (used in single lobe case) can represent as well

LDPRT Rendering

- Demo does full evaluation from previous slide (evaluates all basis functions in shading normal direction, scales each band by zonal harmonic coefficient…)
- Simple Optimization:

$$\sum_{l}\sum_{i\in l} z_l Y_i\left(\vec{N}\right)L_i \qquad \sum_{l} z_l \boxed{\sum_{i\in l} Y_i\left(\vec{N}\right)L_i}$$

LDPRT

- Light specialized rendering
  - Complexity O(n) instead of O(n^2)
  - Fill a texture that contains dot(Y,L) for each band in each direction dynamically
    - 8x8 or 16x16 are probably doable with border textures
    - Without border textures another parameterization (Lat/Long in demo) works at 64x32
    - Use SIMD instructions to fill dynamic textures, do 4 texels at a time

Parameterized Models

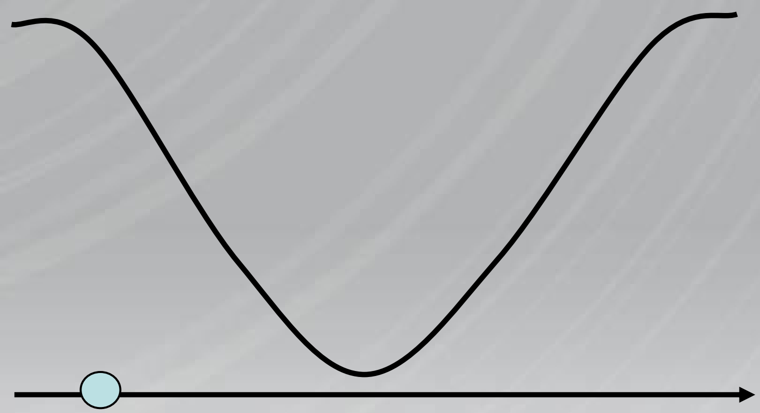
- Can just fit to results of PRT simulation
- Can build an ad-hoc model that has intuitive parameters (build transfer vector on the fly)
- Can fit an intuitive model to simulation data

Simple Translucency

- Single degree of freedom (DOF)
  - "Optical Thickness", how much light bleeds through in the negative normal direction
  - Could be based on subsurface scattering simulation, demo is ad-hoc

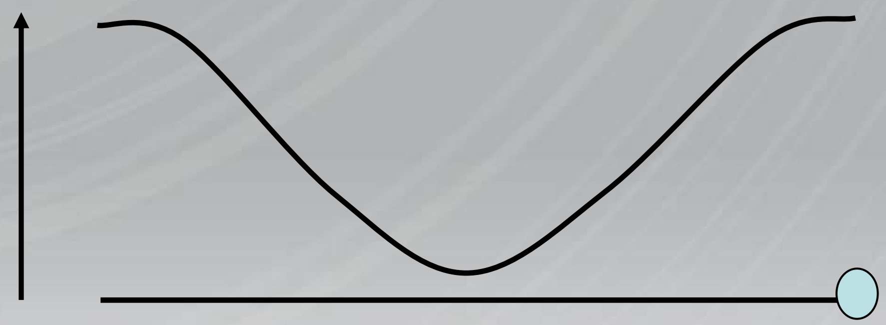
Wrinkle Model

- Two DOF
  - Phase, position along canonical wrinkle



Wrinkle Model

- Two DOF
  - Phase, position along canonical wrinkle
  - Amplitude, max magnitude of wrinkle



Fit

- Compute several simulations
  - 64 discrete amplitudes
  - 255 unique points in phase
- Fit 32x32 textures
  - DC,Linear (4 numbers) using linear least squares
  - 3 lobes fit using non-linear least squares, all DOF fit at once (18k DOF – about 5-10 min)
    - Quasi-Newton techniques don't work, approximate inverse Hessian is to large
    - Used non-linear conjugate gradients instead

Demo

LDPRT Multiple Lobes

Light flowing through space

- Lightmaps/PRT strictly deals with lighting response on surfaces
- What about in a volume?
- Needed to light objects moving through a scene that is modeled with transfer vectors

Representations for Lighting

- Plenoptic function [Adelson91]
- $F(x,y,z,\theta,\phi,\lambda,t)$
  - x,y,z : Position in space
  - θ,ϕ : Direction
  - λ : Wavelength
  - t : Time
- Common to use 3 wavelengths (RGB)
- Time is a bit simplistic as the only means to model changes in illumination

Irradiance Volumes

- Approximation of Plenoptic Function convolved with normalized cosine kernel
- Irradiance Volumes [Greger98]
  - Original paper used “diffuse cube maps”
  - SH used in later papers
  - Static lighting, diffuse objects (no transport) scene not effected by objects

SH Gradients

- Model "mid range" illumination by using a Taylor expansion (in space) of projection into SH [Annen2004]
- This is a really good idea – fairly inexpensive way to handle "local" lights
- Using N-gradient directions requires N times more work
  - 1 or 2 directional derivatives might make sense
- Also see Chris Oat's (ATI) GDC talk this year
  - But use compression!

DEMO

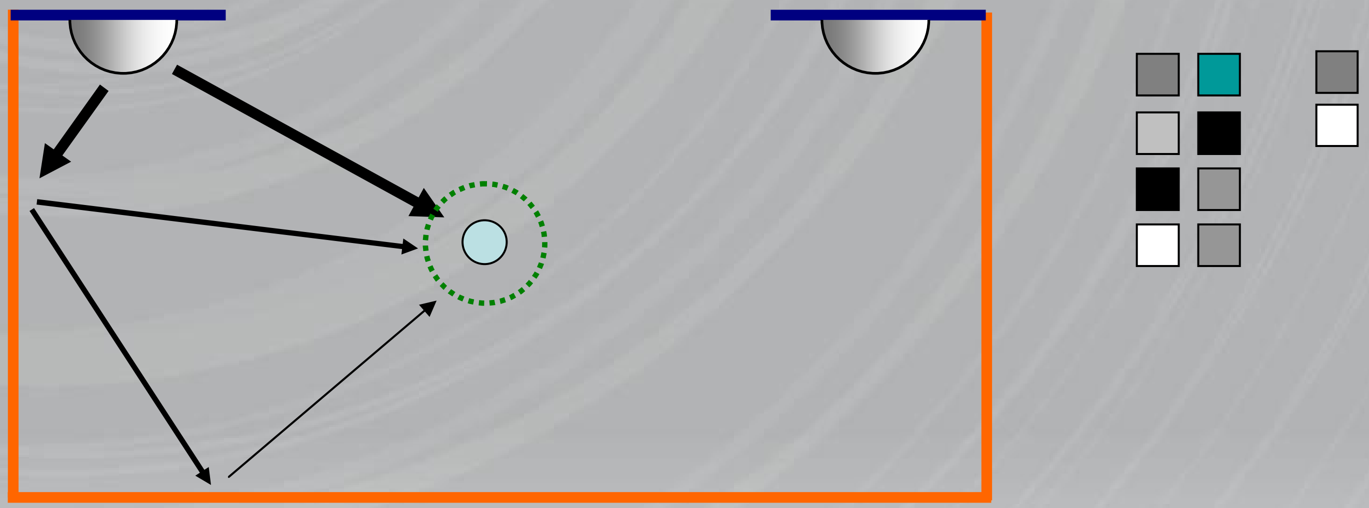
Irradiance Volumes + Gradients

Parameterized Radiance Volumes

- Extend Irradiance Volumes to handle dynamic lighting, render objects with PRT, hacks for how object effects lighting in scene
- Challenges
  - How to mix with SH Gradients
  - Compression

Parameterized Radiance Volumes

SH Light Probes from set of lights is a transfer matrix



Response in space is a SH Light Probe

PRV Representation

- Use smooth (differentiable) BF so you can easily generate gradients
- Multi-level uniform quadratic b-splines
  - Not very expensive at run time
  - C1 continuous (gradients behave better)
  - Multi-level enables more aggressive compression
- K-nearest neighbor and radial basis functions also worth investigating
  - Cost/continuity a concern
  - Oct-trees used by Chris Oat as well

DEMO

Parameterized Radiance Volumes

Generating a PRV

- Compute transfer matrices at a moderate density in space
- Fit coarse b-spline volume to transfer matrices
  - Linear least squares
- Compute residual, threshold
- Fit finer samples to residuals

PRV Optimizations

- Only encode matrices in a single "slice"
  - Chest height in a game
  - Encode derivative out of slice explicitly
- For finer scales, only encode luminance
  - Kind of like image/video compression
  - Lower angular frequency for chroma?
- PCA transfer matrices
  - Trade off PCA decoding with less interpolation

Final Thoughts

- PRT
  - Enables effects that are difficult with traditional techniques
    - Soft shadows from large area lights
    - Inter-reflections
    - Subsurface scattering
  - Easy to mix with traditional techniques
    - Split techniques based on light frequency (PRT for low, shadow maps for high)
    - Split based on transport path (PRT for indirect lighting, something else for direct)
  - Outdoor game
    - PRT for direct+indirect lighting from “skylight” (minus sun)
    - PRT for indirect lighting from sun
    - Conventional techniques for direct lighting from sun

Final Thoughts

- LDPRT
  - Can be used for surface details
  - Trivial to skin/deform (but shadows in tangent space or rest configuration)
- PRV
  - Tying objects into the lighting used for the scene is a good idea, and done already in games
  - Parameterizing lighting makes sense going forward
    - Independent light maps
    - Outdoor lighting (sky light model)
- Compression
  - Always use with PRT
  - Worth using for other scenarios (multiple light maps in particular)

Acknowledgments

- Coauthors
  - John Snyder, Jan Kautz, Ben Luna, John Hart, Jesse Hall
- Samples/Artwork/Slides
  - Jason Sandlin, John Steed, Shanon Drone
- Light Probes
  - Paul Debevec