Light, Reflectance, and Global Illumination

TOPICS:

- Survey of Representations for Light and Visibility
- Color, Perception, and Light
- Reflectance
- Cost of Ray Tracing
- Global Illumination Overview
- Radiosity
- Yu's *Inverse Global Illumination* paper

15-869, Image-Based Modeling and Rendering

Paul Heckbert, 18 Oct. 1999

Representations for Light and Visibility

fixed viewpoint or planar scene	2-D	image (r,g,b)
diffuse surface	2.5-D/ 3-D	textured range data (z,r,g,b) *or pre-lit polygons*
specular surface in clear air, unoccluded	4-D	light field
specular surface in fog, or occluded. plenoptic function $I(x,y,z,\theta,\phi)$	5-D	5-D light field *or surface/volume model with general reflectance properties*

italics: not image-based technique

Physics of Light and Color

Light comes from electromagnetic radiation.

The amplitude of radiation is a function of wavelength λ.

Any quantity related to wavelength is called “spectral”.

Frequency $\nu = 2\pi / \lambda$

EM spectrum:

radio, infrared, **visible**, ultraviolet, x-ray, gamma-ray

← *long wavelength* *short wavelength* →

low frequency *high frequency*

- Don’t confuse EM ``spectrum’’ with ``spectrum’’ of signal in signal/image processing; they’re usually conceptually distinct.
- Likewise, don’t confuse spectral wavelength & frequency of EM spectrum with spatial wavelength & frequency in image processing.

Perception of Light and Color

Light is electromagnetic radiation visible to the human eye.

Humans have evolved to sense the dominant wavelengths emitted by the sun; space aliens have presumably evolved differently.

In low light, we perceive primarily with rods in the retina, which have broad spectral sensitivity ("at night, you can't see color").

In bright light, we perceive primarily with the three sets of cones in the retina, which have narrower spectral sensitivity roughly corresponding to red, yellow, and blue wavelengths.

We're most sensitive to greens & yellows; not very sens. to blue.

The eye has a huge dynamic range: 10 orders of magnitude.

In the brain, neurons combine cone signals into spot, edge, and line receptive fields (point spread functions). And each comes at a range of scales, orientations, and color channels.

What is Color?

Color is human perception of light.

Our perception is imperfect; we don't see spectral power $P(\lambda)$, instead we see three scalars:

at long wavelength (≈ red): $L = \int P(\lambda) s_L(\lambda) d\lambda$

at middle wavelength (≈ yellow): $M = \int P(\lambda) s_M(\lambda) d\lambda$

at short wavelength (≈ blue): $S = \int P(\lambda) s_S(\lambda) d\lambda$

where $s_L(\lambda)$, $s_M(\lambda)$, and $s_S(\lambda)$ are the sensitivity curves of our three sets of cones.

Color is three dimensional because any light we see is indistinguishable to our eyes from some mixture of three spectral (monochromatic, single-wavelength) **primaries**.

Color Spaces

Color can be described in various color spaces:

spectrum - allows non-visible radiation to be described, but usually impractical/unnecessary.

RGB - CRT-oriented color space

good for computer storage.

HSV - a more intuitive color space; good for user interfaces

H=hue -- the color wheel, the spectral colors

S=saturation (purity) -- how gray?

V=value (related to brightness, luminance) -- how bright?

a non-linear transform of RGB, since H is cyclic.

CIE XYZ - used by color scientists, a linear transform of RGB.

other color spaces, less commonly used...

For extremely realistic image synthesis, use of four or more samples of the spectrum may be necessary, but for most purposes, the three samples used by RGB color space is just fine.

Light is a Function of Many Variables

Light is a function of

- position x,y,z
- direction θ,ϕ
- wavelength λ
- time t
- polarization
- phase

In computer graphics, we typically ignore the last three by assuming static scenes, unpolarized, incoherent light, and assume that the speed of light is infinite.

But light is still a complicated function of many variables.

How do we measure light, what are the units?

Units of Light

quantity	*dimension*	*units*
solid angle	solid angle	[steradian]
power	energy/time	[watt]=[joule/sec]
radiance L a.k.a. intensity I	power/(area*solid angle)	[watt/(m^2*steradian)]

In vacuum or as an approximation for air, **radiance is constant along a ray**.

A picture is an array of incoming radiance values at imaginary projection plane; because of radiance-constancy, these are equal to outgoing radiances at intersection of ray with first surface hit

In general, light is absorbed and scattered along a ray.

General Reflection & Transmission

At a surface, reflectance is the fraction of incident (incoming) light that is reflected, **transmittance** is the fraction that is transmitted into the material. Opaque materials have zero transmittance.

In some books, reflectance is denoted ρ, and transmittance τ. Other books use k_{dr} and k_{sr} for diffuse and specular reflectance, and k_{dt} and k_{st} for transmittance.

A general reflectance function has the form of a **bidirectional reflectance distribution function (BRDF)**: $\rho(\theta_i,\phi_i,\theta_o,\phi_o)$, where the direction of incoming light is (θ_i,ϕ_i) and the direction of outgoing light is (θ_o,ϕ_o), θ is the polar angle measured from perpendicular, and ϕ is the azimuth.

There is a similar function for bidirectional transmittance, $\tau(\theta_i,\phi_i,\theta_o,\phi_o)$.

Light is absorbed and scattered by some media (e.g. fog).

Phong's Illumination model is an approximation to general reflectance.

Phong Illumination Model

A point light source with radiance I_l, illuminating an opaque surface, reflects light of the following radiance:

If surface is perfectly **diffuse** (Lambertian). It is **independent of viewing direction**!

$I = I_l \, k_{dr} \max\{N \cdot L, 0\} / r^2$

where k_{dr} = coefficient of diffuse reflection [1/steradian]

N = unit normal vector

L = unit direction vector to light, r is distance to light

If surface is perfectly **specular**. It is **not independent of viewing direction**.

$I = I_l \, k_{sr} \max\{N \cdot H, 0\}^e / r^2$

where k_{sr} = coefficient of specular reflection [1/steradian]

H = unit "halfway vector" = $(V+L)/\|V+L\|$

V = unit direction vector to viewer

e = exponent, controlling apparent roughness: small=rough, big=smooth

- *There are more realistic reflection models than Phong's.*
- *Don't confuse Phong Illumination with Phong Shading.*

Cost of Basic Rendering Algorithms

s = #surfaces (e.g. polygons) t_s = time per surface (transforming, ...)

p = #pixels t_p = time per pixel (writing, incrementing, ...)

ℓ = #lights t_ℓ = time to light surface point w.r.t. one light

a = Σ screen areas of surfs t_i = time for one ray/surface intersection test

Painter's or Z-buffer algorithm, with flat shading

(assuming no sorting in painter's algorithm)

worst case cost = $s(t_s + \ell t_\ell) + at_p$ $\approx at_p$ if polygons big

Painter's or Z-buffer algorithm, with per pixel shading (e.g. Phong)

worst case cost = $st_s + a(\ell t_\ell + t_p)$ $\approx a\ell t_\ell$ if polygons big

Ray casting with no shadows, no spatial data structures

worst case cost = $p(st_i + \ell t_\ell + t_p)$ $\approx pst_i$ if many surfaces

Ray tracing to max depth d with shadows, refl&tran, no spat. DS, no supersampling

2^d–1 intersections/pixel, for each of which there are ℓ shadow rays

worst case cost = $p(2^d–1)[(\ell +1)st_i + \ell t_\ell]$ $\approx 2^d p\ell st_i$ if many surfaces

Note: time constants vary, e.g. t_p is larger for z-buffer than for painter's.

Interreflection

We typically simulate just **direct illumination**: light traveling on a straight, unoccluded line from light source to surface, reflected there, then traveling in a straight, unoccluded line into eye.

Light travels by a variety of paths:

light source → eye *(0 bounces: looking at light source)*

light source → surface1 → eye *(1 bounce: direct illumination)*

light source → surf1 → surf2 → eye *(2 bounces)*

light source → surf1 → surf2 → surf3 → eye *(3 bounces)*

...

Illumination via a path of 2 or more bounces is called **indirect illumination** or **interreflection**. It also happens with transmission.

Global Illumination

Observation: light comes from other surfaces, not just designated light sources.

Goal: simulate interreflection of light in 3-D scenes.

***Difficulty*: you can no longer shade surfaces one at a time, since they're now interrelated!**

Two general classes of algorithms:

1. **ray tracing methods**: simulate motion of photons one by one, tracing photon paths either backwards ("eye ray tracing") or forwards ("light ray tracing") -- good for specular scenes

2. **radiosity methods**: set up a system of linear equations whose solution is the light distribution -- good for diffuse scenes

The Unit of Radiosity

Radiance (a.k.a. **intensity**) is power from/to an area in a given direction.

units: power / (area $\times$ solid angle)

Radiosity is outgoing power per unit area due to emission or reflection over a hemisphere of directions.

units: power / area

radiosity = radiance $\times$ integral of [cos(polar angle) $\times$ d(solid angle)] over a hemisphere = $\pi \times$ radiance

So radiosity and radiance are linearly interrelated.

Thus, “radiosity” is both a unit of light and an algorithm.

Radiant emitted flux density is the unit for light emission.

units: power / area

Radiosity as an Integral Equation

$$\text{radiosity}(x) = \text{emitted}(x) + \text{reflectance}(x) \times \int_{\text{other surface points } t} \text{formfactor}(x,t) \times \text{radiosity}(t)\, dt$$

where x and t are surface points

This is called an **integral equation** because the unknown function "radiosity(x)" appears inside an integral.

Can be solved by radiosity methods or randomized "Monte Carlo" techniques also, by simulating millions of photon paths.

Radiosity methods are a discrete way to think about and simulate global illumination.

Classical Radiosity Method

Definitions:

surfaces are divided into **elements**

radiosity = integral of emitted radiance plus reflected radiance over a hemisphere. units: [power/area]

Assumptions:

- no participating media (no fog) → *shade surfaces only, not vols.*
- opaque surfaces (no transmission)
- **reflection and emission are diffuse** → ***radiance is direction-indep.***, *radiance is a function of 2-D surface parameters and* λ
- reflection and emission are independent of λ within each of several wavelength bands; typically use 3 bands: R,G,B → *solve 3 linear systems of equations*
- radiosity is constant across each element → *one RGB radiosity per element*

Typically (but not exclusively):

- surfaces are polygons, elements are quadrilaterals or triangles

Deriving Radiosity Equations, 1

$$\begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } i\end{pmatrix} = \begin{pmatrix}\text{power}\\ \text{emitted}\\ \text{by elem } i\end{pmatrix} + \begin{pmatrix}\text{power}\\ \text{reflected}\\ \text{by elem } i\end{pmatrix}$$

$$\begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } i\end{pmatrix} = \begin{pmatrix}\text{power}\\ \text{emitted}\\ \text{by elem } i\end{pmatrix} + \begin{pmatrix}\text{reflectance}\\ \text{of elem } i\end{pmatrix} \times \sum_{\text{elem } j} \begin{pmatrix}\text{fraction of power}\\ \text{leaving elem } j \text{ that}\\ \text{arrives at elem } i\end{pmatrix} \times \begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } j\end{pmatrix}$$

Let

A_i = area of element i (computable)

e_i = radiant emitted flux density of element i (given)

ρ_i = reflectance of element i (given)

b_i = radiosity of element i (unknown)

F_{ij} = form factor from i to j = fraction of power leaving i that arrives at j (computable)

So the equation above can be rewritten :

$$A_i b_i = A_i e_i + \rho_i \sum_{j=1}^{n} F_{ji} A_j b_j$$

Form Factors

Define the **form factor** F_{ij} to be the fraction of light leaving element i that arrives at element j

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} v_{ij} dA_j dA_i$$

Where

- v_{ij} is a boolean **visibility** function: 0 if point on i is occluded with respect to point on j, 1 if unoccluded.
- This is a double area integral. Difficult! We end up approximating it.
- dA_i and dA_j are infinitesimal areas on elements i and j, respectively
- θ_i and θ_j are polar angles: the angles between ray and normals on elements i and j, respectively
- Projected area of dA_i from j is $\cos\theta_i \, dA_i$, hence the cosines
- r is distance from point on i to point on j

Reciprocity law: $A_i F_{ij} = A_j F_{ji}$.

Deriving Radiosity Equations, 2

Earlier, we had : $A_i b_i = A_i e_i + \rho_i \sum_{j=1}^{n} F_{ji} A_j b_j$

Dividing by A_i : $b_i = e_i + \rho_i \sum_{j=1}^{n} F_{ji} \frac{A_j}{A_i} b_j$

By the reciprocity law, $F_{ji}(A_j / A_i) = F_{ij}$, so $b_i = e_i + \rho_i \sum_{j=1}^{n} F_{ij} b_j$ for all elems i

Or, in matrix/vector notation :

b = e + Kb

where **b** is the vector of unknown radiosities, **e** is the vector of known emissions, and **K** is a square matrix of reflectance times form factor : $K_{ij} = \rho_i F_{ij}$

Subtracting **Kb** from both sides, we get **b - Kb = e**, or

(I - K)b = e

where **I** is an identity matrix.

This is a linear system of n equations in n unknowns (the b_i).

There are three such systems of equations, one for the red channel, one for green, and one for blue. The variables ρ_i, e_i, and b_i are RGB vectors.

Computing Visibility for Form Factors

Computing visibility in the form factor integral is like solving a hidden surface problem from the point of view of each surface in the scene.

Two methods:

ray tracing: easy to implement, but can be slow without spatial subdivision methods (grids, octrees, hierarchical bounding volumes) to speed up ray-surface intersection testing

hemicube: exploit speed of z-buffer algorithm, compute visibility between one element and all other elements. Good when you have z-buffer hardware, but some tricky issues regarding hemicube resolution

You end up approximating the double area integral with a double summation, just like numerical methods for approximating integrals.

When two elements are known to be inter-visible (no occluders), you can use analytic form factor formulas and skip all this.

Radiosity Systems Issues

All algorithms require the following operations:

1. Input scene (geometry, emissions, reflectances).
2. Choose mesh (important!), subdividing polygons into elements.
3. Compute form factors using ray tracing or hemicube for visibility (expensive).
4. Solve system of equations (indirectly, if progressive radiosity).
5. Display picture.

If mesh is chosen too coarse, approximation is poor, you get blocky shadows.

If mesh is chosen too fine, algorithm is slow. A good mesh is critical!

In radiosity simulations, because scene is assumed diffuse, surfaces' radiance will be view-independent.

Changes in viewpoint require only visibility computations, not shading. Do with z-buffer hardware for speed. This is commonly used for "architectural walkthroughs" and virtual reality.

Changes in scene geometry or reflectance require a new radiosity simulation.

Summary of Radiosity Algorithms

Radiosity algorithms allow indirect lighting to be simulated.

Classical radiosity algorithms:

- Generality: limited to diffuse, polygonal scenes.
- Realism: acceptable for simple scenes; blocky shadows on complex scenes. Trial and error is used to find the right mesh.
- Speed: good for simple scenes. If all form factors are computed, $O(n^2)$, but if progressive radiosity or newer "hierarchical radiosity" algorithms are used, sometimes $O(n)$.

Generalizations:

- **curved surfaces**: easy - radiosity samples are like a surface texture.
- **non-diffuse (specular or general) reflectance**: much harder; radiosity is now a function of not just 2-D position on surface, but 2-D position and 2-D direction. Lots of memory required, but it can be done.

Inverse Global Illumination: Recovering Reflectance Models of Real Scenes from Photographs

Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins

SIGGRAPH ‘99

15-869, Image-Based Modeling and Rendering
Paul Heckbert, 20 Oct. 1999

Yu: Motivation

Most IBMR methods permit novel viewpoints but not novel lighting; they assume surfaces are diffuse.

Want to permit changes in lighting.

Would like to to recover (non-diffuse) reflectance maps:

specularity and roughness at each surface point

Yu: Simplifying Assumptions

- static scene of opaque surfaces
- known shape
- known light source positions
- calibrated cameras, known camera positions
- high dynamic range photographs
- specular reflection parameters (specular reflection coefficient and roughness) constant over large surface regions
- each surface point captured in at least one image
- each light source captured in at least one image
- image of a highlight in each specular surface region in at least one photograph

Yu Algorithm Overview

1. **Solve for shape** (use FAÇADE or similar system)
2. **Solve for coarse diffuse reflectances**
   - coarsely mesh the surfaces into triangles
   - inverse radiosity problem: known form factors and radiosities (from image radiances), solve for diffuse reflectances (linear system)
3. **Solve for coarse specular reflectance**
   - find highlights for each surface from known surface geometry & lights
     - check that highlight unoccluded
   - sample image at points around the highlight center
   - repeat until convergence
     - solve for diffuse and specular reflectances (linear system) for each region
     - solve for roughness of each region (nonlinear)
     - update estimates of interreflection between surfaces
4. **Solve for detailed diffuse reflectance** (albedo maps)
   - look at all images of a given surface point, subtract specular component
   - throw out outlier (highlight-tainted values), average the rest

Yu: Acquisition Details

- spherical, frosted light bulbs
- 180 volt DC power to avoid 60Hz light flicker
- black strips of tape on walls for digitization
- shot 150 images with digital camera, merged to create 40 high dynamic range images (represent with floating point pixel values)
- block light source to camera in some of the images
- correct for radial distortion and vignetting