Undergraduate Research in Photometric Stereo

By **William Nguyen**, and in Collaboration with:

- Isaac Tian (PhD Student)
- Brian Curless (PhD Advisor)
- Aditya Sankar (Lab Director)

Table of Contents

Table of Contents

Introduction

Berkeley Introduction to Shading

University of Washington: Shading Lecture

Introduction to Photometric Stereo

Direct and Global Illumination

Additional Photometric Stereo Content

Depths from Surface Normals

Cross Products

Initial Fragment Shader

Perspective Projection

Fragment Shader with Area Lighting

Possibilities for Modeling Area Lighting

Matrix Transformations

Microsoft Area Lighting Research

Fragment Shader for Area Lighting

Radiosity and Realistic Image Synthesis:

Area Lighting Measurement through Convolutions

Fourier Series

Fourier Transforms

Basics of Spherical Harmonics

Modeling Area Lighting through Spherical Harmonics

Spherical Harmonics for Computer Graphics

Spherical Integrations

Visualizing the Spherical Harmonics Model

Fast Transform for Spherical Harmonics

Python Spherical Harmonic Tools

Introduction

Reference Material:
https://www.scratchapixel.com/

Isaac had generally good results with the pinhole lights.

Need to Clarify whether good results were due to geometric distortion correction of camera or pinhole lighting

- 2 sets of photos taken (without geometric correction and without pinhole lights)

Need to check whether approximation of pinhole lights is accurate enough. Can be checked two ways:

1. Finding how much the pinhole light approximations deviate from the area light approximations
2. Modifying the pinhole light approximations to become accommodate area light approximations/parameters

Figure out how the 3D modelling works through the approximations of certain parameters

- Depth is partially being provided to help clarify whether it's brighter or deeper(?)
- Finding normals of surfaces based on lighting
- How lighting models are represented in code (lectures provided to help)

Model uses machine learning to develop the depths of model?

Hawaii is running similar simulations/project. Isaac found that their photos were distorted, which may have led to difficulties in creating the 3d models

May need to learn OpenGL

Berkeley Introduction to Shading

http://wla.berkeley.edu/~cs184/fa07/lectures/03-Shading.pdf

BRDF: Bidirectional Reflectance Distribution Function

- Given
  - Surface Material
  - Incoming light direction
  - Direction of viewer
  - Surface orientation
- Return
  - Fraction of light that reaches viewer
- p(v,l,n)
  - l = light vector
  - v = viewing direction
  - n = surface normal
  - p = p(θv, θL)
  - Better: p = p(θv, θL, λin, λout)

Approximate BRDF as a sum of:

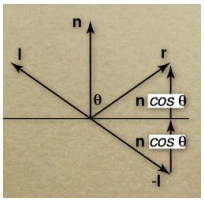
- Diffuse component
- Specular component
- Ambient term

Diffuse Component:

- Lambert's Law:
  - Intensity of reflected light (proportional) to cosine(angle between surface and incoming light)
  - Independent of viewing angle
- $k_d I(l * n)$
  - k_d =
  - I = intensity of light
- Able to plot light
  - Leaving in given direction
  - Leaving each point of surface

Specular Component

- Phong Illumination Model
- Depends on Viewing Angle (mirror-like reflection)
- Able to compute reflected direction
  - r = -l + 2(l * n)n
    - Reflected direction (halfway between l and -l)
  - h = (l + v)/ || l + v ||
    - Halfway between l and v



        - Used to approximate phong model

Ambient Term

- Cheap hack, accounting for ambient omnidirectional light

Summing Diffuse, Specular, and Ambient

$R = k_a I + k_d I \max(l * n, 0) + k_s I \max(r * v, 0)^p$

Direction VS. Point Lighting

- For Point light: light direction changes over surface
- For Distant light: direction is constant
    - Similar for orthographic/perspective viewer

Light Falloff:

- 1/r^2 is reality
- Looks bad so 1/r is often used

Surface Normals

- Normal vector at a point on a surface is perpendicular to all surface tangent vectors

Flat Shading

- Use constant normal for each triangle (polygon)
- Noticeably not smooth

Smooth Shading

- Computer average normal across vertices
- Interpolate across polygons
- Use a threshold for sharper edges
- Different normals for each face

Gouraud Shading:

- Compute Normals (shading) at each vertex
    - Fast and looks smooth
    - Bad for specular reflections

Phong Shading

- Compute Normals (Shading) at each pixel
    - Smooth, better speculars
    - Expensive to compute
- Interpolate Normals from vertices

Normal of a vertices is the average of the normals of the triangles around it
The normal of any point in the triangle is a baryocentric interpolation of the 3 vertices it is made of

- Basically based on proximity to each vertices

University of Washington: Shading Lecture

https://courses.cs.washington.edu/courses/cse457/17au/assets/lectures/shading-1pp.pdf

Approximating using Phong and Blinn-Phong Illumination Models
Given:

- Point **P** on a surface visible through pixel **p**
- The normal **N** at **P**
- The lighting direction, **L**, and intensity (color) I_L
- The viewing direction, **V**, at **P**
- The shading coefficients at **P**

Compute the color, *I*, of pixel **p**

1. $I = k_e$
   - I = resulting intensity
   - k_e = emissivity or intrinsic shade
2. $I = k_e + k_a I_{La}$
   - k_a is the ambient reflection coefficient
     - Actually the reflectance of ambient light
   - I_{La} is the ambient light intensity
   - k_e , k_a, I_{La}, are all functions over wavelengths λ

   For ambient shade:
   - $I(\lambda) = k_a(\lambda) I_{La}(\lambda)$
   - Traditionally, represented as rgb triples, compute above for each color channel

Diffuse/Lambertian Reflection

- Rough surface with microfacets or pigment particles that distribute light rays in all directions
- Reflected intensity independent of direction of viewer
- Incoming light depends on direction of source

3. $I = k_e + k_a I_{La} + k_d I_L B\cos\theta = k_e + k_a I_{La} + k_d I_L B(N * L)$

- K_d is diffuse reflection coefficient
- I_L is (color) intensity of light source
- N is normal to surface (unit vector)
- L is direction to light source (unit vector)
- B prevents contribution of light from below surface

$$B = \begin{cases} 1 & \text{if } N \cdot L > 0 \\ 0 & \text{if } N \cdot L \leq 0 \end{cases}$$

Specular Reflection

- Highlighting of objects, important for smooth, shiny surfaces
- Dependent on viewing direction **V**
- Color determined by color of light (if nonmetal)

$$I = \begin{cases} I_L & \text{if } \mathbf{V} = \mathbf{R} \\ 0 & \text{otherwise} \end{cases}$$

- Metal color can be altered
- Mirror Reflector, light reflected about N
    - With near perfect reflector, highlight will fall off quickly with increasing angle

Phong Specular Reflection:

- As n_s gets larger, highlight on curved surface gets smaller
- B is
- R is reflected light direction across normal
- V is viewing direction
- n_s is the specular exponent or shininess
- (x)+ =

$$I_{\text{specular}} \sim B(\mathbf{R} \cdot \mathbf{V})_+^{n_s}$$

$$\text{where } (x)_+ \equiv \max(0, x).$$

Blinn-Phong Specular Reflection

- Compute H = (L + V) / || L + V ||
- Computer specular contributions in terms of (N*H) raised to power n_s now:

Where $(x)_+$ = max (0,x)

$$I_{\text{specular}} \sim B(\mathbf{N} \cdot \mathbf{H})_+^{n_s}$$

- H is the unit vector halfway between L and V, where V is the viewing direction.

4. $I = k_e + k_a I_{La} + k_d I_L B(N * L) + k_s I_L B(N*H)^{ns}{}_+$

$$= k_e + k_a I_{La} + I_L B\left[k_d(\mathbf{N} \cdot \mathbf{L}) + k_s(\mathbf{N} \cdot \mathbf{H})_+^{n_s}\right]$$

applies to both spec & diffuse

Direction Lights:

- Single direction and intensity.
- Homogeneous coordinate for direction light is 0, it's a vector.

Point Lights:

- Direction determined by vector from light position to surface point
- L = (E - P)/ ||E - P||
- Intensity drops off inversely ($1/r^2$)
    - Common Alternative: $1/(a + br + cr^2)$
    - a,b,c user supplied constants
- Homogeneous coordinate for point light is 1, it's a point

Spotlights:

- Directional attenuation of a point light source, giving spotlight effect.

$$f_{\text{spot}} = \begin{cases} \dfrac{(\mathbf{L}\cdot\mathbf{S})^e}{a+br+cr^2} & \alpha \le \beta \\ 0 & \text{otherwise} \end{cases}$$

- **L** is the direction to the point lgiht
- **S** is the center direction of the spotlight
- α is the angle between **L** and **S**
- β is the cutoff angle for the spotlight
- e is the angular falloff coefficient
    - Similar to how n_s works.
- α <= β == $\cos^{-1}$(**L***S**) <= β == **L***S** >= cos(β)

5.

property of material

$$I = k_e + \sum_j k_a I_{La,j} + \frac{(\mathbf{L}_j\cdot\mathbf{S}_j)^{e_j}_{\beta_j}}{a_j + b_j r_j + c_j r_j^2} I_{L,j} B_j \left[k_d (\mathbf{N}\cdot\mathbf{L}_j) + k_s (\mathbf{N}\cdot\mathbf{H}_j)^{n_s}_{+} \right]$$

- Spatial Vectors: N, L, S, H
- RGB Triples: k, *I*
- Scalars: a, b, c, r, e, B, n_s

Ray casting: find out which surface point is visible at each pixel

- For each pixel center $\mathbf{P}_{ij}$ send ray from eye point (COP), **C**, through $\mathbf{P}_{ij}$ into scene
- Intersect each object with ray
- Select nearest intersection.

Alternative Approach

- Flip order of loops:
- For each triangle in scene
    - For each pixel, determine if triangle projects onto it,
    - Update pixel if this triangle is the closest one so far.

Warping Space: to determine which pixels a triangle projects onto

- Warp all of space so that all rays are parallel
- Drop z-coordinate to get pixel coordinates.

Graphics hardware follows “warping space” approach.

- Before being turned into pixels, piece of geometry goes through a number of transformations
    1. Model/Object Space
    2. World Space
    3. Eye/Camera Space
    4. Projection Space
    5. Image Space

Z-Buffer (Depth-Buffer):

- Used to determine which surface point is visible at each pixel

Pseudo for viewer looking down -z axis (bigger, -i.e. More positive -z's are closer)

```
for each pixel (i, j) do
    Z-buffer [i, j] ← FAR
    Framebuffer [i, j] ← <background c
end for
for each triangle A do
    for each pixel (i, j) in A do
        Compute depth z of A at (i, j)
        color ← shader(A, i, j )
        if z > Z-buffer [i, j ] then
            Z-buffer [i, j] ← z
            Framebuffer [i, j] ← color
        end if
    end for
end for
```

Far should be set to - infinity

Rasterization:

- Compute pixel coordinates of vertices of triangle (interior pixels through interpolation)
- Z-value computed incrementally (fast!)

Hardware Pipeline:

1. Vertex Processor
    - Vertex shader run for each vertex, outputs values interpolated across triangle
2. Primitive Assembler
    - Vertices grouped into triangles to be rasterized
3. Rasterizer
    - Iterate through scanlines, interpolating outputs from vertex shader at each pixel
4. Fragment Processor

- Fragment shader (pixel shader) called at each pixel, takes interpolated values, outputs final color.

Phong-interpolated normals pipeline:

- Vertex Shader:
  a. $V_i \leftarrow$ project v to image
  b. Out n_e
  c. Out v_e
  d. Out v_i
  e. $v_i^1, v_i^2, v_i^3 \rightarrow$ triangle
- Fragment Shader:
  a. L ← determine lighting direction (using v_e^p)
  b. V ← normalize ($-v_e^p$)
  c. N ← normalize (n_e^p)
  d. Color ← shade with L, V, N, k_d, k_s, n_s

Texture Mapping and Z-Buffer:

1. Supply per-vertex texture coordinates
2. Scan conversion is done in screen space, as usual
3. Texture coordinates are interpolated, as usual
4. Supply a **uniform** with the texture data
5. Each pixel is colored by looking up the texture at the interpolated coordinates
6. $$I = I_L B\left[k_d(\mathbf{N}\cdot\mathbf{L}) + k_s(\mathbf{N}\cdot\mathbf{L})\left(\mathbf{N}\cdot\frac{\mathbf{L}+\mathbf{V}}{\|\mathbf{L}+\mathbf{V}\|}\right)_+^{n_s}\right]$$
$$= I_L B(\mathbf{N}\cdot\mathbf{L})\left[k_d + k_s\left(\mathbf{N}\cdot\frac{\mathbf{L}+\mathbf{V}}{\|\mathbf{L}+\mathbf{V}\|}\right)_+^{n_s}\right]$$
$$= I_L B(\mathbf{N}\cdot\mathbf{L}) f_r(\mathbf{L},\mathbf{V})$$

Summary Questions:

Physical Meaning of each variable in Blinn-Phong Lighting Model in Iteration 5?

property of material

$$I = k_e + \sum_j k_a I_{La,j} + \frac{\left(\mathbf{L}_j \cdot \mathbf{S}_j\right)^{e_j}_{\beta_j}}{a_j + b_j r_j + c_j r_j^2} I_{L,j} B_j \left[k_d \left(\mathbf{N} \cdot \mathbf{L}_j\right) + k_s \left(\mathbf{N} \cdot \mathbf{H}_j\right)^{n_s}_{+} \right]$$

I = resulting intensity
K_e = intrinsic shade constant

$\Sigma k_a I_{La,j}$ = sum of ambient reflection coefficient (k_a) times ambient light intensity (I_{La}) for RGB

$(\mathbf{L_j} * \mathbf{S_j})_{\beta j}^{ej}$ = spotlight intensity where **L** is the direction to the point of light, **S** is the center direction of the spotlight, e is angular falloff coefficient, and β is the cutoff angle for the spotlight

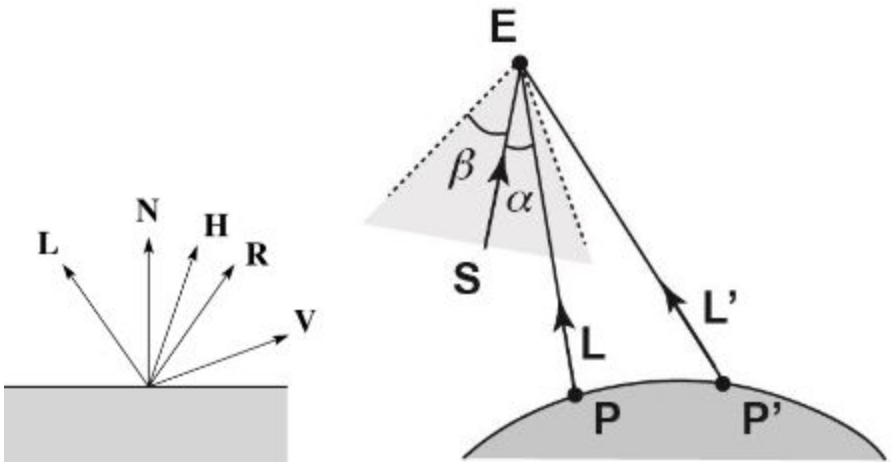
$a_j + b_j r_j + c_j r_j^2$ = user supplied constants combined with r^2 to model falloff

$I_{L,j} B_j$ = Color Intensity of the light source (I_L) * a constraint on contribution of light from below the surface (B)

k_d (**N * L**) = diffuse reflection coefficient (k_d) * dot product of surface normal and direction to light source (N*L)

k_s $(\mathbf{N} * \mathbf{H})_+^{ns}$ = specular reflection coefficient (k_s) * dot product of surface normal and unit vector halfway between L and V (**N * H**) to the power of the specular component (shininess n_s) and the specular reflection is either 0 or positive (can't be negative)

How terms are computed?
L = E - P / || E - P ||
H = L + V / ||L + V||
Effect Each Term Has On Image?



What does varying parameters do?
Understand Faceted, Gouraud, and Phong Interpolation.
Faceted: each face has a normal constant. Polygon
Gouraud: Compute normals at vertex, shade only vertex, interpolate resulting vertex colors
Phong: Computer normals at vertices (based on each triangle in proximity), interpolate normas and normalize shade using interpolated normals

Introduction to Photometric Stereo

http://pages.cs.wisc.edu/~lizhang/courses/cs766-2008f/syllabus/10-09-shading/shading.pdf
Photometric Stereo

Diffuse Reflection:

- $R_e = k_d \mathbf{N} * \mathbf{L} R_i$
- $I = k_d \mathbf{N} * \mathbf{L}$
    - Image intensity of P
- Simplifying assumptions
    - $I = R_e$: camera response function f is the identity function:
        - Solve for f and applying f^{-1} to each pixel in the image
    - $R_i = 1$: light source intensity is 1
        - Divide each pixel in this image by R_i

Shape from Shading:

- $K_d = 1$
    - $I = k_d\, N * L$
    - $= N * L$
    - $= \cos\Theta_i$
- You can measure angles between normal and light source
    - Need more info for surface shape
        - Assume some normals are known
        - Constraints on neighboring normals - “integrability”
        - Smoothness
    - Difficult in reality (inconsistent albedo)

Utilizing Linear Algebra:

- Write $I1 = k_d N*L_1$, $I2 = k_d N*L_2$, $I3 = k_d N*L_3$ into

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = k_d \begin{bmatrix} \mathbf{L_1}^T \\ \mathbf{L_2}^T \\ \mathbf{L_3}^T \end{bmatrix} \mathbf{N}$$

$$\underbrace{\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}}_{\mathbf{I}_{3\times 1}} = \underbrace{\begin{bmatrix} \mathbf{L_1}^T \\ \mathbf{L_2}^T \\ \mathbf{L_3}^T \end{bmatrix}}_{\mathbf{L}_{3\times 3}} \underbrace{k_d \mathbf{N}}_{\mathbf{G}_{3\times 1}}$$

$$\mathbf{G} = \mathbf{L}^{-1}\mathbf{I}$$

$$k_d = \|\mathbf{G}\|$$

More Lights Results In Better Results

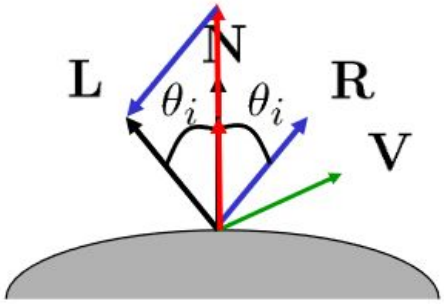
Least Squares Solution:

- $I = LG$
- $L^T I = L^T LG$
- $G = (L^T L)^{-1}(L^T I)$

Solve for N, k_d as before

Specular Reflection Rule:

- Perfect mirror, light is reflected about N



- Highlight when V = R
  - When viewing direction is the same as reflection
  - L becomes: L = 2(N * R) N - R

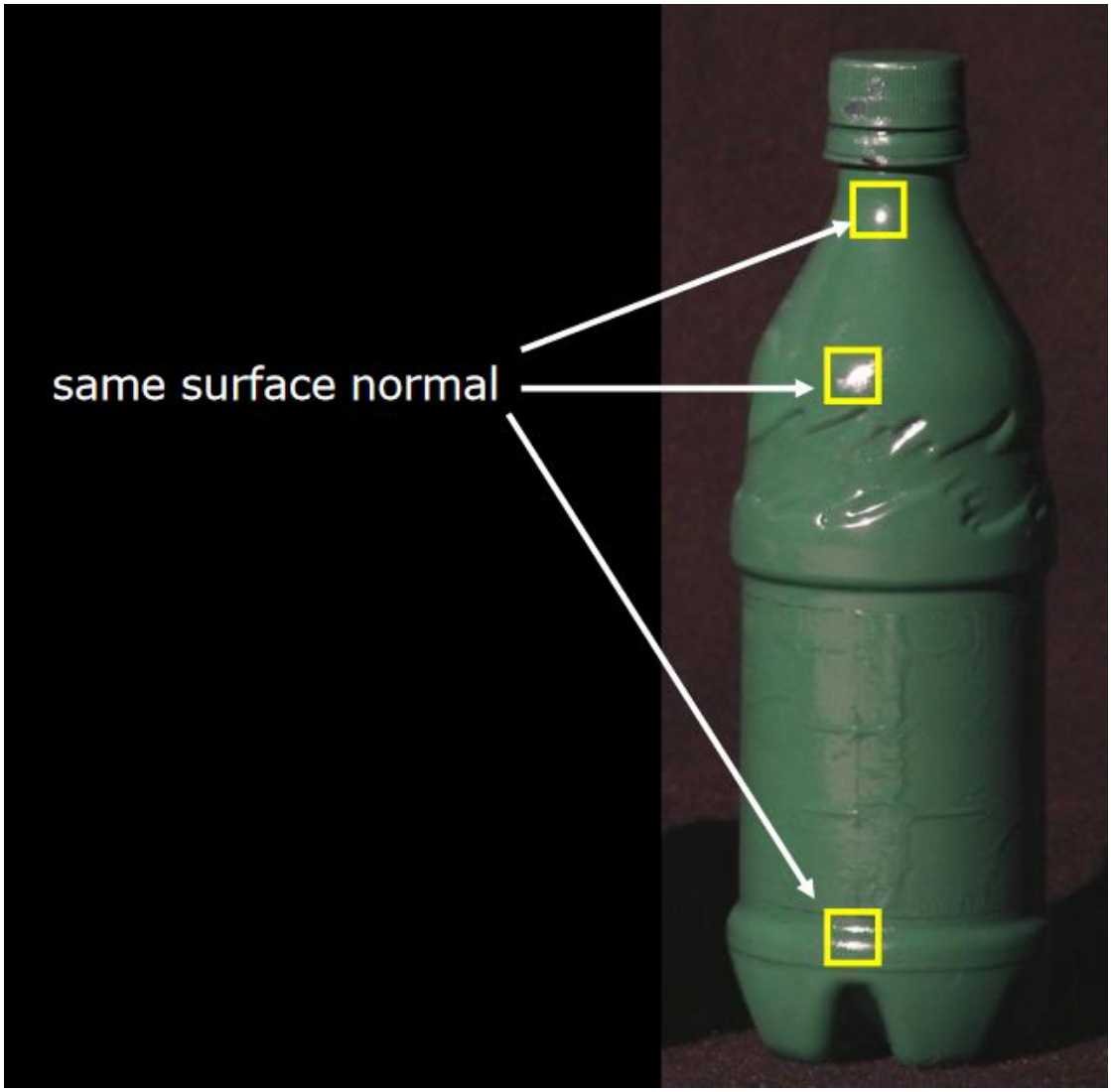
$$R_e = \begin{cases} R_i & \text{if } \mathbf{V} = \mathbf{R} \\ 0 & \text{otherwise} \end{cases}$$

Depth from Normals:

- $V1 = (x + 1, y, z_{x+1}, y) - (x, y, z_{xy}) = (1, 0, z_{x+1}, y - z_{xy})$
- $0 = N * V1 = (nx, ny, nz) * (1, 0, z_{x+1}, y - z_{xy}) = n_x + n_z(z_x + 1, y - z_{xy})$
- V2 will be similar
  - Each normal gives us two linear constraints on z
  - Compute z values by solving a matrix equation

Limitations of Photometric Stereo:

- Big Issues
  - Fails for shiny and semi-translucent things
  - Shadows, inter-reflections
- Smaller Problems
  - Camera and lights have to be distant
  - Calibration requirements
    - Measure light source directions, intensities
    - Camera response function
- Some of these are solved in modern works (1,2)

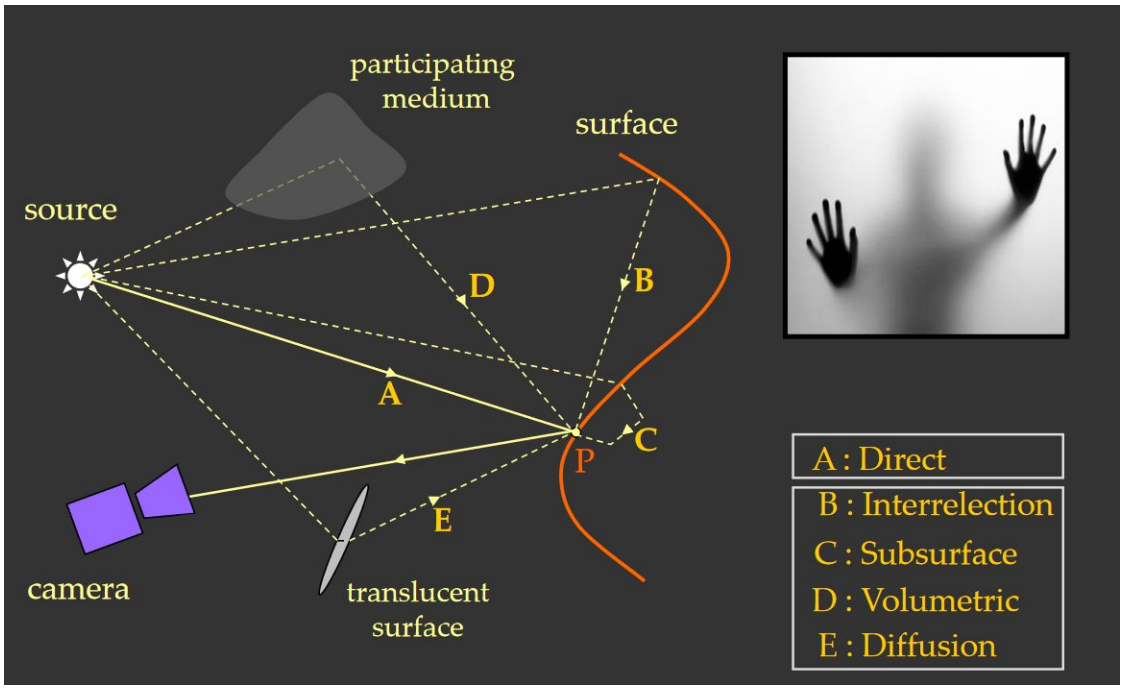


Photometric Stereo:

- Estimating 3D shape by varying illumination, fixed camera
    - Operating Conditions
    - Opaque material
    - Distant camera, lighting
    - Reference object available,
    - No shadows, interreflections, transparency

Direct and Global Illumination

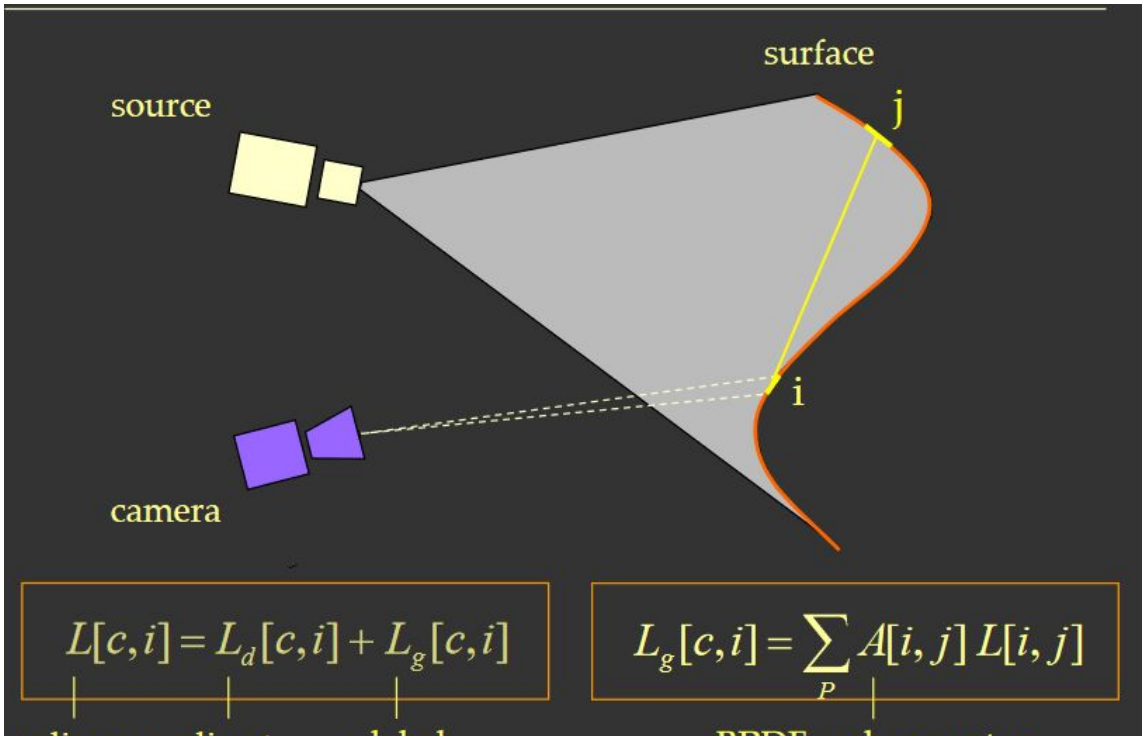
Direct and Global Illumination:



Fast Separation of Direct and Global Images

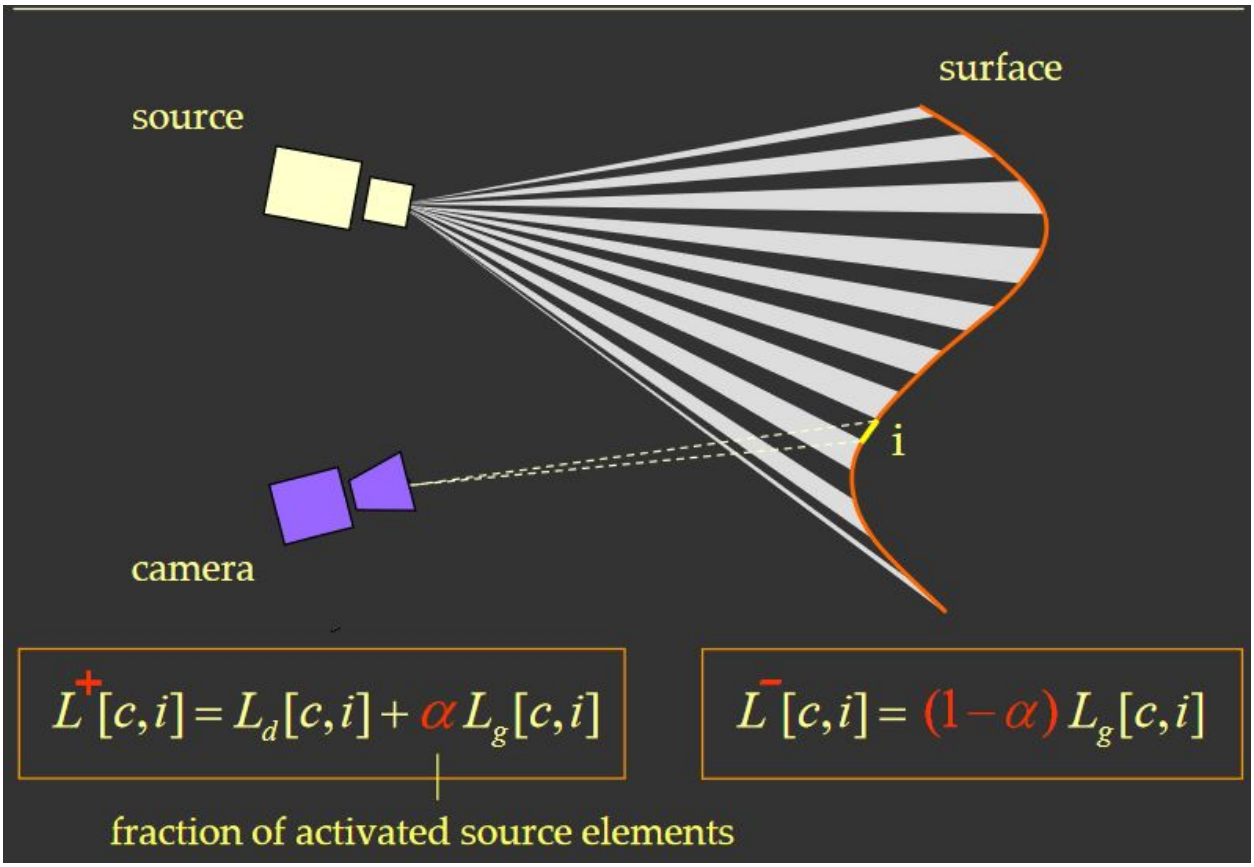
- Create images of scene
- Enhance brightness based vision methods
- New insights into material properties

Interreflections:



- Light may be reflected off another part of the surface and may be picked up where the camera/viewer is directed.

High Frequency Illumination Pattern:



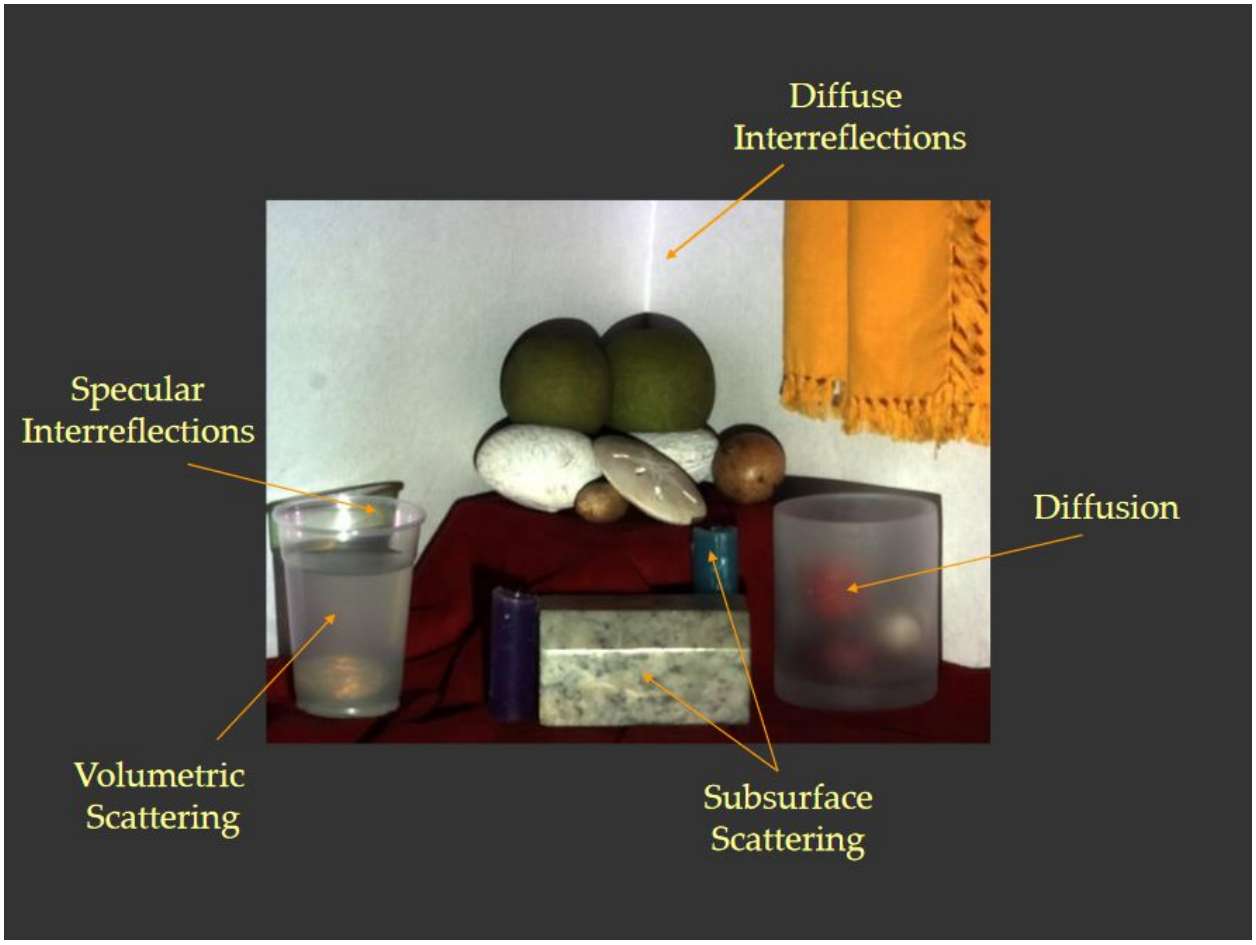
Separation from Two Images:
α = ½:
$L_d = L_{max} - L_{min}$, $L_g = 2L_{min}$

Subsurface scattering:

- Light going into a translucent surface will be recognized from the camera/viewer in different locations due to the subsurface scattering of the light.
- Subsurface scattering is what leads to a lot of color?

Volumetric Scattering: When there is a participating medium that the light source hits. The light hitting the participating medium may scatter into the the line of sight of the camera/viewer

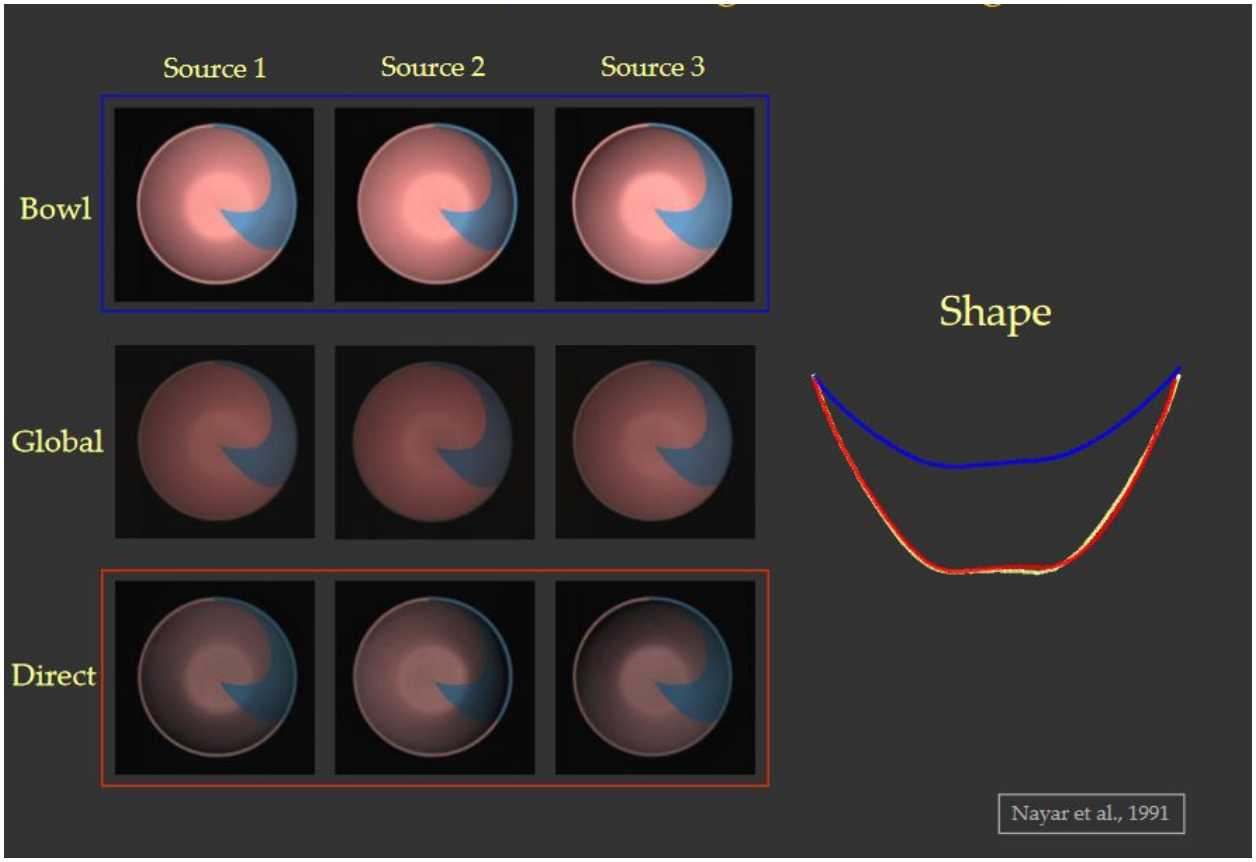
Real World Example:



Mirror Ball Fail Case:

- Mirror Ball reflects/absorbs light differently than other objects, and makes it difficult/impossible to successfully achieve a photometric stereo.

Photometric Stereo Using Direct Images:



- Direct Images recognize the ring of the bowl higher and center of the bowl as lower with the shadows and light on the sides showing change?

Variants of Separation Method:

- Coded Structured Light
- Shifted Sinusoids
- Shadow of Line Occluder
- Shadow of Mesh Occluders

Building Corners:

- $L_d = L_{max} - L_{min}$, $L_g = L_{min}$

Additional Photometric Stereo Content

https://www.sciencedirect.com/science/article/pii/S1076567008014018
https://homes.cs.washington.edu/~sagarwal/shadowcuts.pdf
http://www.macs.hw.ac.uk/texturelab/files/publications/phds_mscs/JW/chapter4.pdf

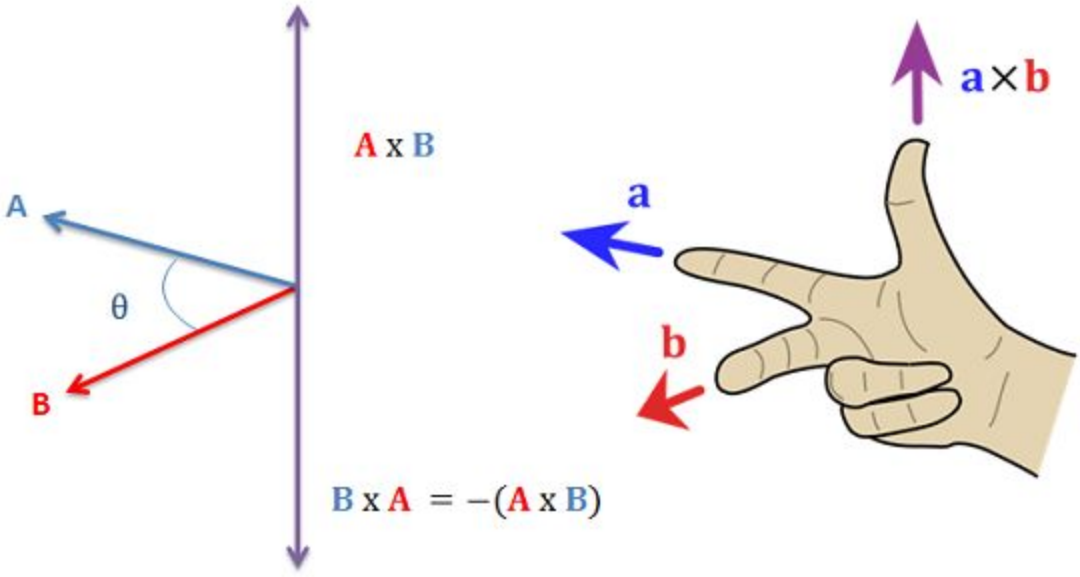
Depths from Surface Normals

- We have already transferred a 3D object to a 2D image
    - Depths are lost entirely
    - Object converted into pixels
    - Hjk
- 2D image is a bunch of pixels
    - We can convert image pixels/arrangement into triangles
    - For every vertex, we find the normal through taking a cross product of two vectors stemming from the vertex
    - From the vertex to another direction is +/-1 (x or y depending on direction, and usually + rather than -)
        - Vertex east of center vertex is $(x+1, y, z_e)$
        - Vertex south of center vertex is $(x,y+1, z_s)$
    - The normals are dependent on the z value, so we have to make function of N based on depth (z) → N(d)
    -

Cross Products

- Magnitude of cross product is area corresponding to the parallelogram of 2 vectors (repeating the 2 vectors to become a parallelogram)
- Resulting vector is orthogonal to both a and b
- ||a|| * ||b|| * sinΘ
- i,j,k == x,y,z
- $a = (a_1, a_2, a_3) = a_1i + a_2j + a_3k$
- $b = (b_1, b_2, b_3) = b_1i + b_2j + b_3k$
- $a \times b = (a_2b_3 - a_3b_2)i - (a_1b_3 - a_3b_1)j + (a_1b_2 - a_2b_1)k$
- Matrix Form

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} \mathbf{i} - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} \mathbf{j} + \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \mathbf{k}.$$



Scalar Projection (dot products):

$$a_b = \mathbf{a} \cdot \widehat{\mathbf{b}},$$

- where $\widehat{\mathbf{b}} = \mathbf{b} / \|\mathbf{b}\|$ is the unit vector in the direction of **b**.
- Scalar projection of vector a in direction of vector b
- == ||a||cosΘ

Initial Fragment Shader

```
#version 300 es
precision highp float;
precision highp int;
in vec3 vcolor;
in vec3 vnormal;
in vec3 wcoord;
in vec3 wvnormal;
in vec3 vdepth;
in vec3 vpos;
uniform float exposure;
uniform vec4 pointlight;
out vec4 color;

void main() {
	vec3 tmp = (normalize(vnormal));
	//uncomment this line for normal map
	//color = vec4(tmp,1);
	//uncomment here for depth map
	//color = vec4(vdepth, 1);
	//point light n*l here
	vec3 l = vec3(pointlight[0] - vpos[0], pointlight[1] - vpos[1], pointlight[2] - vpos[2]);
	vec3 colorrgb = dot(tmp , l)  * vec3(115.0/255.0, 161.0/255.0, 227.0/255.0); //vcolor;
	float r = length(l);
	colorrgb /= r*r*r;
	colorrgb *= pointlight[3];    //pointlight[3] is intensity
	colorrgb = clamp(colorrgb, 0.0, 1.0);

	//specular highlight (Phong model)
	float rscale = 2.0 * dot( l / r, tmp);
	vec3 reflect = vec3(0.0, 0.0, 0.0);
	if (rscale > 0.0) {    reflect = rscale * tmp - (l / r);}
	float rdotv = clamp(dot(reflect, - vpos / length(vpos)), 0.0, 1.0);
	float specular = 0.1 * clamp(pow(rdotv , 5.0) * pointlight[3], 0.0, 1.0);
	specular /= r*r;
	colorrgb += specular;

	//ambient component
	colorrgb += vec3(0.01, 0.01, 0.01);
```

```
            //compute S
            //multiply by

            colorrgb = clamp(colorrgb, 0.0, 1.0);
            color = vec4(colorrgb, 1);

    //uncomment here for mask
    //color = vec4(1, 1, 1, 1);
}
```

Summary:

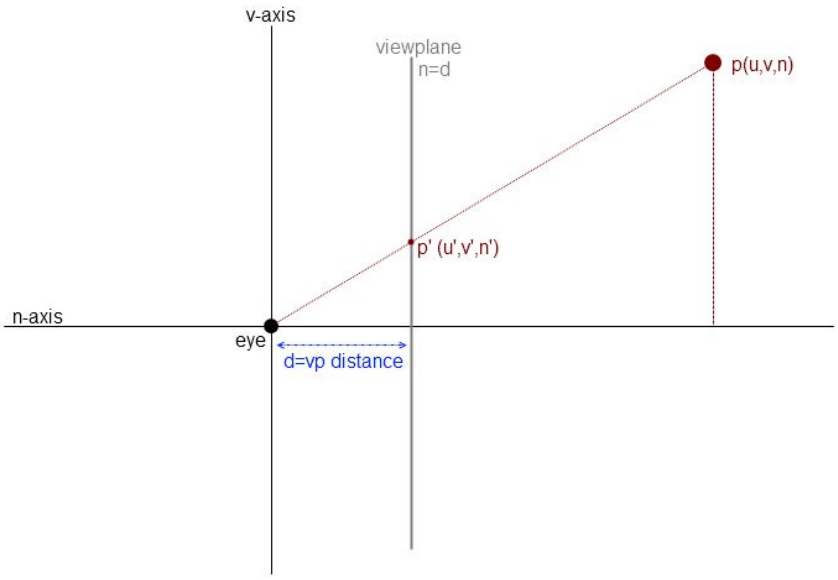
Tmp = unit vector of normal
Colorrgb

Perspective Projection

https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect5.pdf
http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/projection/perspective_projection.html
http://web.engr.illinois.edu/~slazebni/spring16/lec02_perspective-daf.pdf

Relative to position of eye and viewplane

- Eye lies on z-axis and viewplane is xy plane
- Project a point by finding intersection of line (from eye to point) with view plane
    - Calculated using similar triangles



Eye at origin, viewplane is d on n-axis, projected point p` found using similar triangles

$p`_v / d = p_v / p_n$
$v` = d * (p_u / -p_n)$
$p`_u = d * (p_u / -p_n)$
$P`_n = d$

Point (p_u,p_v,p_n) projected to $p`(d * (p_u / -p_n), d * (p_v / -p_n), d)$

Can replace d with n (for “near”). Done through multiplying the following homogeneous matrix.

$$\hat{\mathbf{M}} = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$\hat{\mathbf{M}}\vec{p} = \begin{pmatrix} np_u \\ np_v \\ np_n \\ -p_n \end{pmatrix}$$

To convert homogenous point to ordinary point, divide through by fourth component and discord

$$\rightarrow \left(n\frac{p_u}{-p_n}, n\frac{p_v}{-p_n}, n \right)$$

Pseudo Depth:

- Perspective projection discards info about vertex depth
- All projected vertices have depth d, unable to remove hidden surfaces without relative depth info
- Easily incorporated into projection matrix
- Points on near plane will be given depth of -1, points on far plane will be at +1
- Incorporated through

$$\hat{\mathbf{M}} = \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$p'_n = \frac{ap_n + b}{-p_n}$$

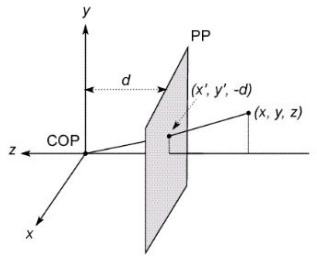
where

$$a = -\frac{f+n}{f-n}$$

$$b = \frac{-2fn}{f-n}$$

Modeling Projection:

- Compute intersection with image plane PP (projection plane) of ray from (x,y,z) to COP
- Derived using similar triangles
- Throw out last coordinate for projection coordinates on image



$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z},\ -d) \quad \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z})$$

Similar Triangles Definition:

If two angles of a triangle have measures equal to the measures of two angles of another triangle, then the triangles are similar. Corresponding sides of similar polygons are in proportion, and corresponding angles of similar polygons have the same measure.

Homogeneous Coordinates

- Extending N-d space to (N+1)-d space
    - A point in 2d image is treated as a ray in 3D projective space
    - Each point (x,y) on the image plane is represented by the ray (sx,sy,s)
        - All points on ray are equivalent: (x,y,1) = (sx, sy, s)
    - Go back to 2D by dividing with last coordinate: (sx, sy, s) /s → (x,y)

Modeling Projection:

- Matrix multiplication using homogeneous coordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate and throw it out to get image coords

- This is perspective projection
    - Matrix is projection matrix
- Scaling projection matrix leads to same result if scaled by any arbitrary constant c.
    - A larger object that is further away (scaled by x,y,z) can have the same size as a smaller object that is closer.

Fragment Shader with Area Lighting

```
precision highp float;
precision highp int;

in vec3 vpos;
in vec3 vnormal;

in vec3 vcolor;
in vec3 wcoord;
in vec3 wvnormal;
in vec3 vdepth;

uniform vec3 eone;
uniform float rone;
uniform float exposure;
uniform vec4 pointlight;

out vec4 color;

void main() {
   vec3 bright = vec3(0,0,0);
   float pi = 3.14159265358979323846264338327795;
   float incdegree = pi / 12;
   float incr = 0.1;
   vec3 z = vec3(0,0,1);

   //center point of area light based on pointlight vector
   vec3 l = vec3(pointlight[0] - vpos[0], pointlight[1] - vpos[1], pointlight[2]
- vpos[2]);



   //vcolor copied from pass.f.glsl
   vec3 colorrgb = vec3(115.0/255.0, 161.0/255.0, 227.0/255.0);
```

```
    for (float r = 0; r < rone; r+= incr){
        for(float theta = 0; theta < 2 * pi; theta += incdegree){

            //pp
            vec3 pp = vec3(r * cos(theta),r * sin(theta),0);

            //rotation plane for rodrigues rotation
            //crossing z x eone
            vec3 plane = cross( z / length(z), eone / length(eone));
            //normalizing rotation plane
            plane = normalize(plane);

            //rotation angle calculation
            float rot = acos( dot(eone/length(eone),z / length(z)) );

            //rotation vector of pp + transform (l)
            vec3 Wpp =
                pp * cos(rot)
                + normalize(cross(plane,
                pp))
                + plane * (dot(plane,
                pp)) * (1-cos(rot))
                + l;
            //scale by area divided by number of samples
            //Light intensity calculation//pointlight[3] = inherent light
intensity
            float intense = (pi * (pow(rone,2))) / ( (rone / incr)  *   ((2*pi) /
incdegree) );
            intense *= pointlight[3] * dot(-l / length(l)), eone / length(eone));

            vec3 change = vec3(Wpp[0] - vpos[0], Wpp[1] - vpos[1], Wpp[2] -
vpos[2]);

            //diffuse highlight
            vec3 tmp = (normalize(vnormal));
            colorrgb *= dot(tmp , change / length(change));
            colorrgb /= length(change) * length(change);
```

```
            colorrgb = clamp(colorrgb, 0.0, 1.0);


            //phong specular highlight
            float rscale = 2.0 * dot( change / length(change), tmp);
            vec3 reflect = vec3(0.0, 0.0, 0.0);
            if (rscale > 0.0) {
                reflect = rscale * tmp - (change / length(change));
            }
            float rdotv = clamp(dot(reflect, - vpos / length(vpos)), 0.0, 1.0);
            float specular = 0.1 * clamp(pow(rdotv , 5.0) * pointlight[3], 0.0,
1.0);
            specular /= length(change) * length(change);
            colorrgb += specular;

            //ambient component
            colorrgb += vec3(0.01, 0.01, 0.01);
            colorrgb = clamp(colorrgb, 0.0, 1.0);

            //multiply by the scaled intensity
            colorrgb *= intense

            //add shaded region to bright
            bright += colorrgb;
        }
    }

    //outputted color vector is rgb vector + 1.0 (100%) alpha value
    color = vec4(bright, 1);
}
```

Possibilities for Modeling Area Lighting

Converting Spot light to area light

- Spotlight is approximation of an area light
- Scale factor instead of L*S, is cross-section of light from vertex
    - Find projection through “deviation” from center
- Treated like point light and find cross-section for approximation
    - Similar to a spotlight
    - Further from center will be attenuated

- Spotlight simulation is 1-1

- Distance, radius,

- Area light is many to 1 model

1. Treat like point light, scale by cross-section
    a. Given radius
    b. Effectively like a spot light, e is based on area of light
    c. Still treats every light as 1-1, and simulating area light through attenuation from center of light
    d. Cross-section is just dot product
        i. Projection of spotlight onto surface
            1. Have distance, normal of light,
2. Discrete approximation of area light
    a. Position and orientation of area light
    b. For every area light, for loop to break it into discrete chunks
        i. Convert into polar coordinates and for loop over it (break into parts of r)
        ii. Concentric rings of point lights that are summed.
        iii. Replace attenuation factor withl oop in poalr.
    c. How to map polar in 2d
        i. Rtheta gives vector, can add to center of sphere
            1. Can arbitrarily pick 0,0 (where theta = 0)

Translation, Rotation on Computer Graphics slides (and scratch it)

- Start with berkeley for general, cross-reference with UW and scratch it.

Polar to Cartesian formula

Shading at Point V Based on Area Lighting:

- $\Sigma\ r = 0 \rightarrow$ radius $R\ \Sigma\ \Theta = 0 \rightarrow 2\ pi\ (L_{r,\Theta})$

Idea: Mapping polar coordinates on a 3D space

SCALE, THEN ROTATE, THEN TRANSLATE

1. Scale unit circle by radius of area light
2. Rotate unit circle on X,Y,Z → Direction S of the area light
    a. Find Z x S
    b. Find $\Theta = \cos^{-1}(z*s)$
    c. Rotate Z by Θ along the plane made by Z and S (utilizing the normal Z x S)
3. Translate unit circle on X,Y,Z → Position P_l of the area light
4. Multiply the matrix by to convert polar to cartesian

Matrix Transformations

https://people.eecs.berkeley.edu/~job/Classes/CS184/Fall-2014-Slides/05-3DTransformations.pdf

- Slide 1-23 for review

Translations for 2D: $\tilde{\mathbf{A}} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$

Translations for 3D: $\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Scales for 2D: $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Scales for 3D: $\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Rotations:

- Involve amount of rotation, **AND** axis of rotation (vector)
- Rotations are still orthonormal (orthogonal and normal)
- Determinant(R) = 1 =/= -1
    - Rotations are not commutative
- 2D rotations implicitly rotate about ⅓ out of plane axis

Axis Aligned 3D rotations

For VERTICAL Z Orientation:

$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$ $\mathbf{R}_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$ $\mathbf{R}_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Arbitrary Rotations:

- Can be built from axis-aligned rotations ($R = R_x * R_y * R_z$)

Exponential Map:

- Direct Representation of arbitrary rotations
- Rotate Θ degrees about some axis
- Encode Θ by length of vector
    - Θ = |r|

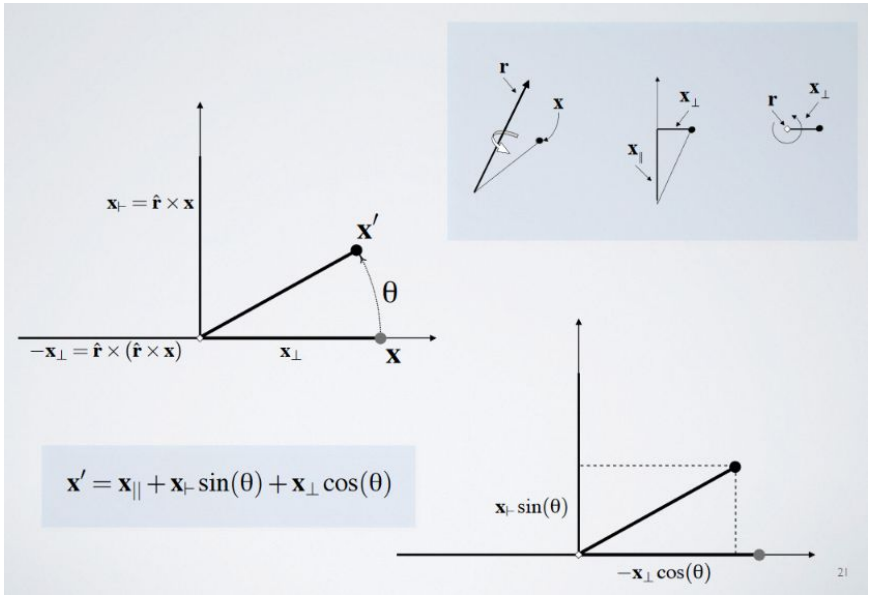
How to get matrix **R** from vector r

1. Rotate about x axis to put r into x-y plane

2. Rotate about z axis align r with x axis
3. Rotate Θ degrees about x axis
4. Undo #2 and then #1
5. Composite together

Vector expressing a point has two parts

- $X_{||}$ does not change
- $X_{\perp}$ rotates like a 2D point



Rodriguez Formula:

X` = r(r * x) + sin(Θ) (r_x x) - cos(Θ) (r_x (r_x x))

$$\mathbf{x}' = \left((\hat{\mathbf{r}}\hat{\mathbf{r}}^t) + \sin(\theta)(\hat{\mathbf{r}}\times) - \cos(\theta)(\hat{\mathbf{r}}\times)(\hat{\mathbf{r}}\times)\right)\mathbf{x}$$

$$(\hat{\mathbf{r}}\times) = \begin{bmatrix} 0 & -\hat{r}_z & \hat{r}_y \\ \hat{r}_z & 0 & -\hat{r}_x \\ -\hat{r}_y & \hat{r}_x & 0 \end{bmatrix}$$

http://www.cs.cmu.edu/afs/cs/academic/class/16741-s07/www/lectures/lecture7.pdf

Microsoft Area Lighting Research

https://www.microsoft.com/en-us/research/wp-content/uploads/1996/03/arealights.pdf

Benefits of Area Lighting vs point light source model:

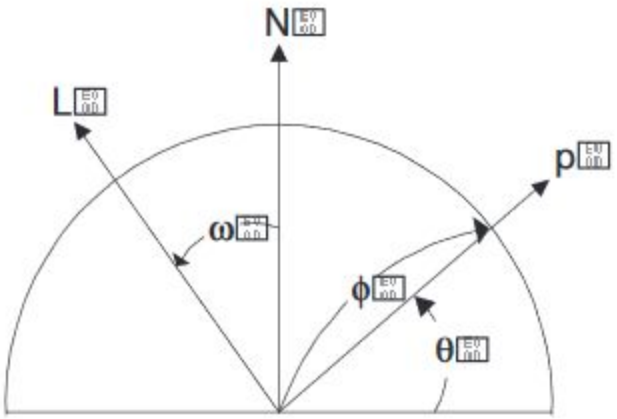
- Area light sources are softer, tending to reduce the variation of surface intensity
    - Real-time graphics sample the illumination at vertices of each polygon and linearly interpolates the shading function between the vertices
    - Nonlinear variation of surface shading thus requires finely tessellated polygons.
    - Using area light sources reduces this variation and allows larger polygons to be rendered with acceptable degree of fidelity
- Area light sources suffer from the problem of highlight burnout
    - Regions near the precise peak of highlights are clipped to maximum intensity, becoming regions of constant color

Lambertian Lighting:

- Simple light model in which light is scattered so as to appear equally bright from all directions. Given a point P on surface with unit-length normal N, Lambert's law states that the output intensity, *I*, is equal in any direction and given by
    - $$I \equiv \frac{1}{I_{\text{norm}}} \int_{H(N)} F(p)\,(p \cdot N)\,dA(p)$$
- H(N) is a unit-radius hemisphere around N, p is a point on this hemisphere representing the incident direction, F is the incident radiance as function of incident direction, and d*A* is the differential area of the surface element at p. I_{norm} is a factor that normalizes the BRDF
    - $$I_{\text{norm}} \equiv \int_{H(N)} (p \cdot N)\,dA(p) = \pi.$$

Infinite Hemispherical Light Sources:

- Light direction L which represents the "top" of the hemisphere. An arbitrary direction D receives light only if D * L >= 0.
    - Ignores location of point to be illuminated, only normal at point (N) matters
- Constant-weighted and cosine-weighted hemispherical light sources combined to produce a reasonable simulation of illumination from an overcast sky.
    - Parameterization assumes that coordinate system been arranged so that N is transformed to the z-axis, and perpendicular projection of L onto N is mapped to the negative x-axis. The hemisphere around N is given by

- $$p(\theta, \phi) \equiv \begin{pmatrix} \cos\theta \sin\phi \\ -\cos\phi \\ \sin\theta \sin\phi \end{pmatrix}$$
- 

- $\theta \in [0, \pi]$ and $\phi \in [0, \pi]$. The angle between N and L is denoted by ω, where $\omega \in [0, \pi]$.

dA = sinΦdΘdΦ ;Lambert's Law → p * N = sinΘsinΦ

- $$I_{\text{hemi}}(\omega) \equiv \frac{1}{\pi} \int_{\omega}^{\pi} d\theta \int_{0}^{\pi} d\phi \, F(\theta, \phi) \sin\theta \sin^2\phi.$$
- →

Constant-Weighted

- $$F(\theta, \phi) \equiv \begin{cases} 0, & \text{if } L \cdot p(\theta, \phi) < 0 \\ 1, & \text{otherwise.} \end{cases}$$
- Leads to $I_{\text{hemi-const}} = \frac{1}{2}(1 + \cos\omega).$ → $I_{\text{hemi-const}}(N, L) = \frac{1}{2}(1 + N \cdot L)$

Cosine-Weighted:

- $$F(\theta, \phi) \equiv L \cdot p(\theta, \phi)$$
- $$L \equiv \begin{pmatrix} \cos(\frac{\pi}{2} + \omega) \\ 0 \\ \sin(\frac{\pi}{2} + \omega) \end{pmatrix} = \begin{pmatrix} -\sin\omega \\ 0 \\ \cos\omega \end{pmatrix}.$$
- $$I_{\text{hemi-cos}}(\omega) = \frac{2}{3\pi} \left(\sin^3\omega + \cos\omega \left(\pi - \omega + \sin\omega \cos\omega\right)\right).$$
- In Global coordinates, $\omega = \cos^{-1}(N \cdot L)$

Combining Weights: The CIE Sky Model

- $$I_{\text{hemi-sum}}(\omega, \alpha, \beta) \equiv \alpha I_{\text{hemi-const}}(\omega) + \beta I_{\text{hemi-cos}}(\omega).$$
    - Can be used for overcast skies like so
    - $$I_{\text{hemi-sum}}(\omega, \frac{C}{3}, \frac{2C}{3})$$

Sub-Hemispherical Light Sources:

- Derivations of above can be used to construct shading function for a finite spherical light source that is perfectly diffuse emitter (or due to an infinite light source that subtends a fixed angle of less than pi)
- Sub-hemispherical is defined with direction L that points toward center of light source and an angle δ within 0 - pi/2.
    - Represents max angle an incident direction can make with L.
    - Direction D receives light only if D * L >= cos(δ)

$$I_{\text{hemi-sub}}(\omega, \sigma) \equiv \frac{1}{\pi} \begin{cases} \pi \cos\omega \sin^2\sigma, & \omega \in [0, \frac{\pi}{2} - \sigma] \\ \pi \cos\omega \sin^2\sigma + G(\omega, \sigma, \gamma) - H(\omega, \sigma, \gamma), & \omega \in [\frac{\pi}{2} - \sigma, \frac{\pi}{2}] \\ G(\omega, \sigma, \gamma) + H(\omega, \sigma, \gamma), & \omega \in [\frac{\pi}{2}, \frac{\pi}{2} + \sigma] \\ 0, & \omega \in [\frac{\pi}{2} + \sigma, \pi] \end{cases}$$

$$\gamma \equiv \sin^{-1}\left(\frac{\cos\sigma}{\sin\omega}\right),$$

$$G(\omega, \sigma, \gamma) \equiv -2\sin\omega\cos\sigma\cos\gamma + \frac{\pi}{2} - \gamma + \sin\gamma\cos\gamma,$$

$$H(\omega, \sigma, \gamma) \equiv \cos\omega\left[\cos\gamma\sqrt{\sin^2\sigma - \cos^2\gamma} + \sin^2\sigma\sin^{-1}\left(\frac{\cos\gamma}{\sin\sigma}\right)\right].$$

- $$I_{\text{hemi-const}}(\omega) = I_{\text{hemi-sub}}(\omega, \frac{\pi}{2}).$$
- Sub-hemispherical light source can model finite spherical light sources using the following
    - $$\sigma = \sin^{-1}\left(\frac{r}{\|P - P_L\|}\right)$$
    - R = radius
    - P_L = light source position
    - P = point to be illuminated

Approximating sub-hemispherical model:

- Can reduce computation by approximation “fillet” region of $I_{\text{hemi-sub}}$ as a cubic curve

- i.e., $\omega \in [\frac{\pi}{2} - \sigma, \frac{\pi}{2} + \sigma]$)
- One endpoint ω = pi/2 - δ
- Other endpoint ω = pi/2 + δ

$$I_{\text{hemi-sub}}(\frac{\pi}{2} - \sigma) = \cos(\frac{\pi}{2} - \sigma)\sin^2\sigma = \sin^3\sigma$$

$$\frac{d\,I_{\text{hemi-sub}}}{d\omega}(\frac{\pi}{2} - \sigma) = -\sin(\frac{\pi}{2} - \sigma)\sin^2\sigma = -\cos\sigma\sin^2\sigma.$$

$$I_{\text{hemi-sub}}(\frac{\pi}{2} + \sigma) = 0$$

- Hermite interpolation approximates a function by a cubic curve, given its value and the values of its first derivative at endpoints of the interval

$$f(x_0) = f_0,\ f(x_1) = f_1,\ f'(x_0) = f'_0,\ f'(x_1) = f'_1.$$

  - To increase accuracy, can break fillet region into two cubic curves instead of one by introducing another knot at ω = pi/2
  - Piecewise cubic curve defined using hermite interpolation at the 2 following intervals
    - $[\frac{\pi}{2} - \sigma, \frac{\pi}{2}]$ and $[\frac{\pi}{2}, \frac{\pi}{2} + \sigma]$.

Polygonal Light Sources:

- Illumination from uniformly bright polygonal light sources can be analytically integrated;
  - Collection of m vertices $V_1, V_2, \ldots, V_m$.

$$I_{\text{poly}} \equiv \frac{1}{2\pi}\sum_{i=1}^{m}\Theta_i(P)\,(\Gamma_i(P)\cdot N)$$

$$\Theta_i(P) \equiv \cos^{-1}\left(\frac{V_i - P}{\|V_i - P\|}\cdot\frac{V_{i+1} - P}{\|V_{i+1} - P\|}\right)$$

$$\Gamma_i(P) \equiv \frac{(V_i - P)\times(V_{i+1} - P)}{\|(V_i - P)\times(V_{i+1} - P)\|}$$

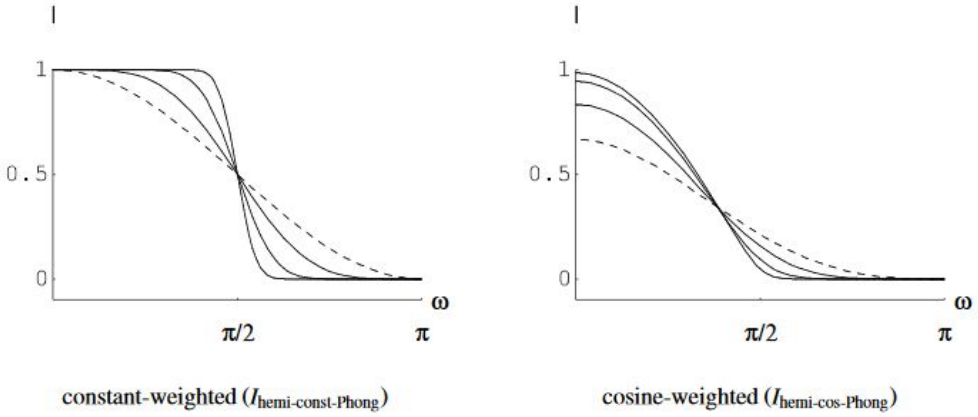
- Assuming light source is entirely within hemisphere around N at P. light source must be restricted.
  - Can be easily accomplished by clipping light source polygon to plane containing P perpendicular to N (Q - P) * N = 0

Non-Lambertian Lighting:

- Phong power-law model, output intensity in viewing direction V is given by

- $$I \equiv \frac{1}{I_{\text{norm-Phong}}} \int_{H(R)} F(p)\,(p \cdot R)^n \, dA(p).$$
- *R is reflection direction (reflecting viewing direction through normal N)*
- Integral exponent *n* controls shininess (larger exponent → shinier surface).
- H(R) is unit-radius hemisphere (around R rather than N)
- P is point on hemisphere representing incident direction
- F is incident radiance as function of incident direction
- d*A* is differential area of surface element at p. $I_{\text{norm-phong}}$ is normalization factor
  - $$I_{\text{norm-Phong}} \equiv \int_{H(R)} (p \cdot R)^n \, dA(p) = \frac{2\pi}{n+1}$$
- Different models than OpenGL and Arvo95
  - Specular intensity falls off as angle of plane formed by V and p becomes perpendicular to *N*

Phong Illumination from Infinite Hemispherical Light Sources: (Equations found on PDF)



constant-weighted ($I_{\text{hemi-const-Phong}}$) cosine-weighted ($I_{\text{hemi-cos-Phong}}$)

Figure 4: Infinite Hemisphere Phong Illumination: The left side plots constant-weighted model, the right plots the cosine-weighted model, for $n = 1, 4, 16, 64$. The dashed curve represents $n = 1$ on both sides.

Figure 4 compares the infinite hemisphere Phong illumination model (both constant and cosine weighted) for various values of n. Note that $I_{\text{hemi-const-Phong}}$ and $I_{\text{hemi-cos-Phong}}$ are generalizations of the functions $I_{\text{hemi-const}}$ and $I_{\text{hemi-cos}}$, respectively, from Sections 2.1.1 and 2.1.2, where

$$I_{\text{hemi-const}}(\omega) = I_{\text{hemi-const-Phong}}(\omega, 1)$$
$$I_{\text{hemi-cos}}(\omega) = I_{\text{hemi-cos-Phong}}(\omega, 1).$$

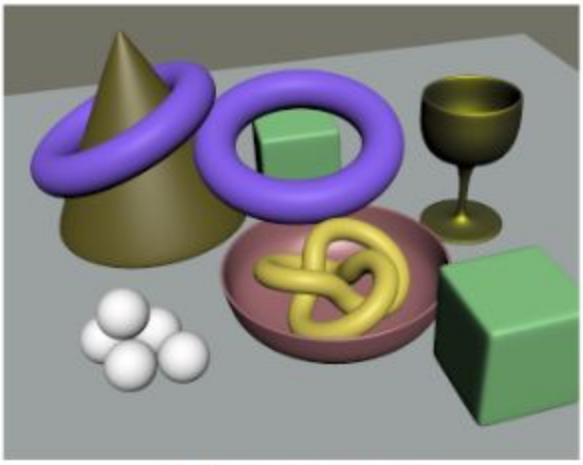
Phong Illumination from Sub-Hemispherical Light Sources:

- Actual code for evaluation of the sub-hemispherical Phong model can be found in Appendix F. As in the case of Lambertian lighting, a finite spherical light source can be modeled by computing σ via Equation 7. An infinite light source of fixed angular
- dimension is modeled using a fixed σ, for which $\cos\sigma$ and $\sin\sigma$ can be pre-computed.

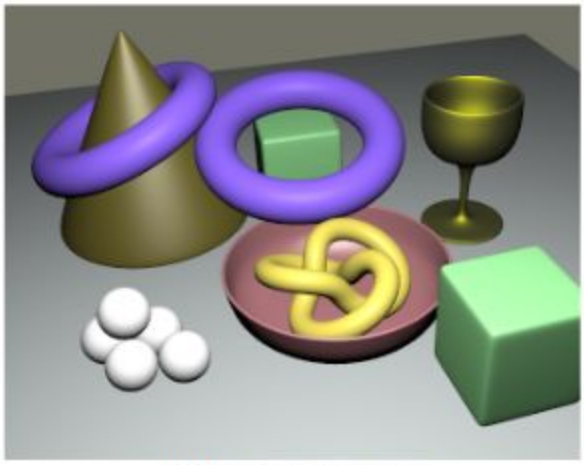
Phong Illumination from Polygonal Light Sources:

Lambertian and Phong Model Summary:

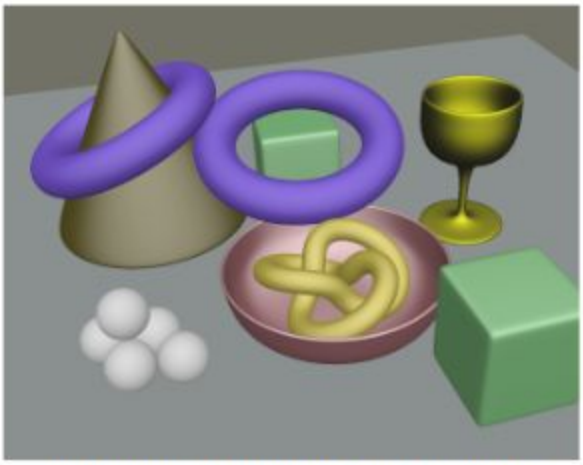
Lighting Model	Light Source	Name	Equation Number
Lambertian	constant-weighted hemisphere	$I_{\text{hemi-const}}$	(3)
Lambertian	cosine-weighted hemisphere	$I_{\text{hemi-cos}}$	(5)
Lambertian	constant-weighted sub-hemisphere	$I_{\text{hemi-sub}}$	(6)
Lambertian	constant-weighted polygon	I_{poly}	(8)
Phong	constant-weighted hemisphere	$I_{\text{hemi-const-Phong}}$	(14)
Phong	cosine-weighted hemisphere	$I_{\text{hemi-cos-Phong}}$	(15)
Phong	constant-weighted sub-hemisphere	$I_{\text{hemi-sub-Phong}}$	(24)
Phong	constant-weighted polygon	$I_{\text{poly-Phong}}$	(25)



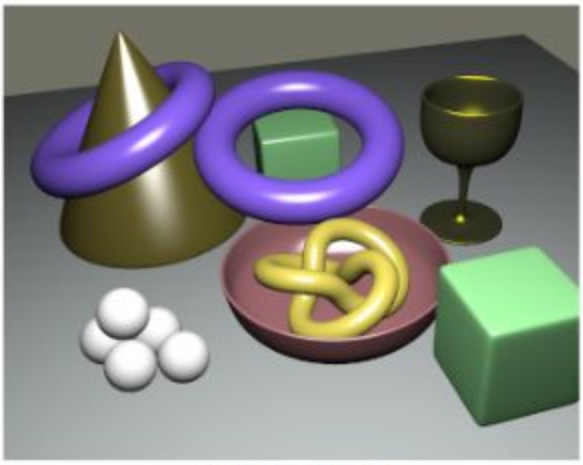
(a) Non-Area Directional



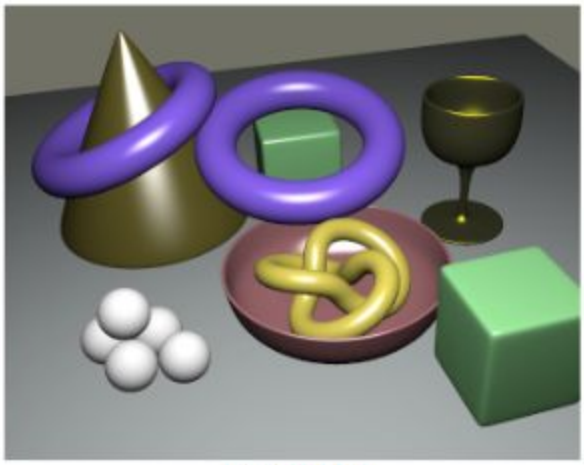
(b) Non-Area Positional



(c) Overcast Sky Hemispherical Light Source



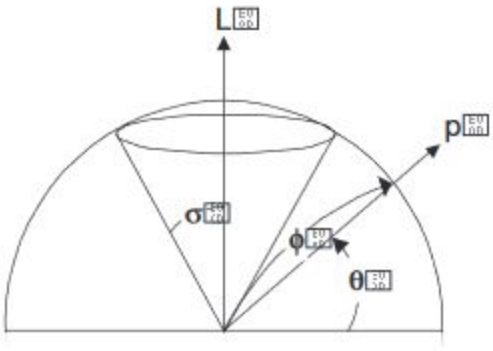
(d) Finite Spherical



(e) Polygonal

Figure 7: Comparison of Area Light Source Results.

Derivation of Lambertian Luminance from Sub-Hemispherical Light Sources:

- Derives equation 6:
    - Region of integration has 3 forms
    1. Sub-hemispherical light is entirely within hemisphere around normal
        - Perform integration analytically by parameterizing the subsphere about L
        - $$p(\theta, \phi) \equiv \begin{pmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{pmatrix}$$
        - Coordinate system is transformed so L maps to z-axis
        - $$N = \begin{pmatrix} \sin\omega \\ 0 \\ \cos\omega \end{pmatrix}.$$
    2. Light source is partially within normal's hemisphere
        - Transform coordinates and parameterize light source via
        - $$p(\theta, \phi) \equiv \begin{pmatrix} \cos\theta \sin\phi \\ -\cos\phi \\ \sin\theta \sin\phi \end{pmatrix}$$
        - 

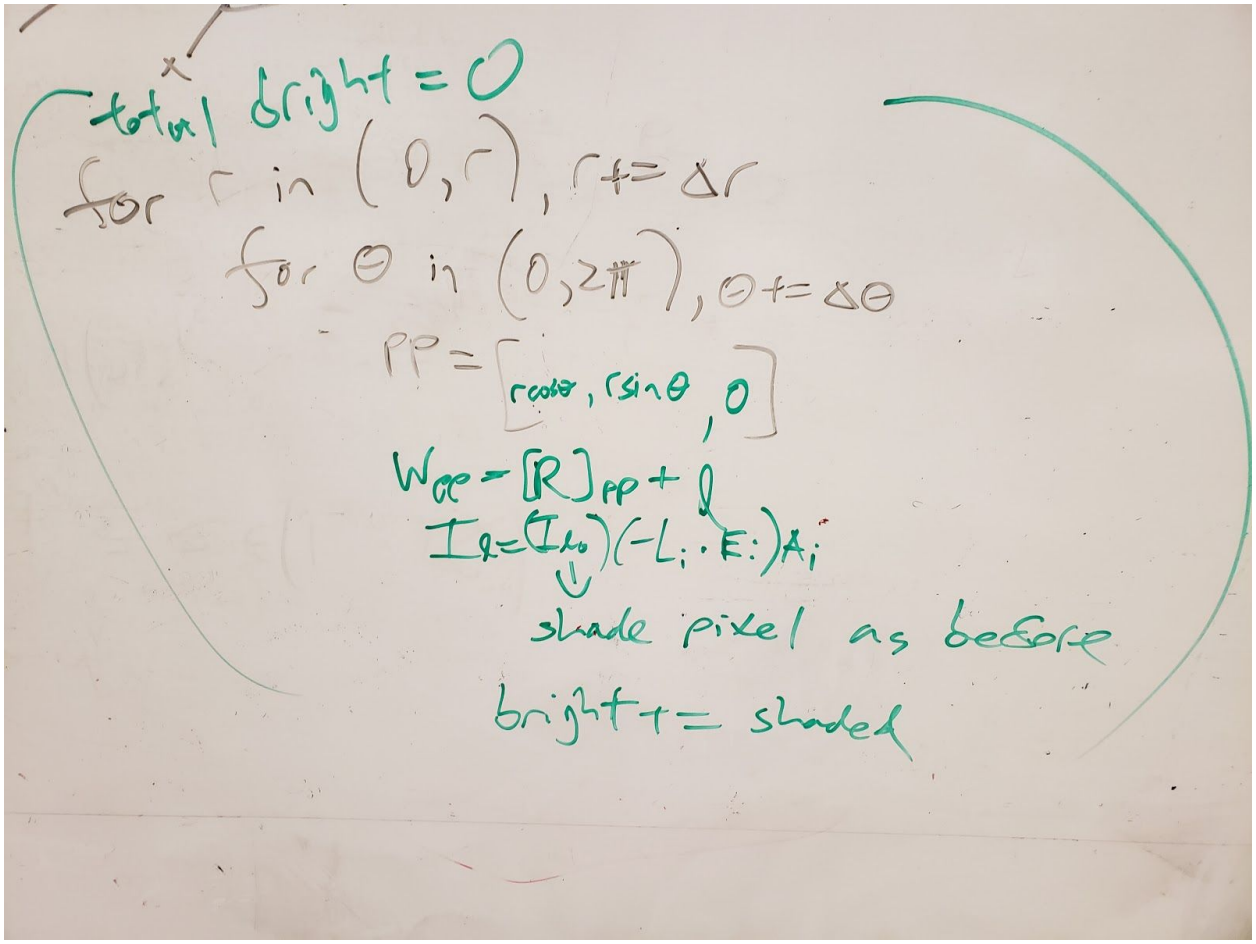
        - L transformed to map onto z-axis, and plane formed by N and L become xz plane
        - Resulting Intensity
        - $$I = \pi \cos\omega \sin^2\sigma - \cos\omega \left[\cos\gamma \sqrt{\sin^2\sigma - \cos^2\gamma} + \sin^2\sigma \sin^{-1}\left(\frac{\cos\gamma}{\sin\sigma}\right)\right] - 2\sin\omega \cos\sigma \cos\gamma + \frac{\pi}{2} - \gamma + \sin\gamma$$
    3. Light source is entirely outside normal's hemisphere
        - $$\begin{aligned} \omega \in [0, \tfrac{\pi}{2} - \sigma] &\rightarrow \text{entirely inside} \\ \omega \in [\tfrac{\pi}{2} - \sigma, \tfrac{\pi}{2} + \sigma] &\rightarrow \text{partially inside} \\ \omega \in [\tfrac{\pi}{2} + \sigma, \pi] &\rightarrow \text{entirely outside} \end{aligned}$$

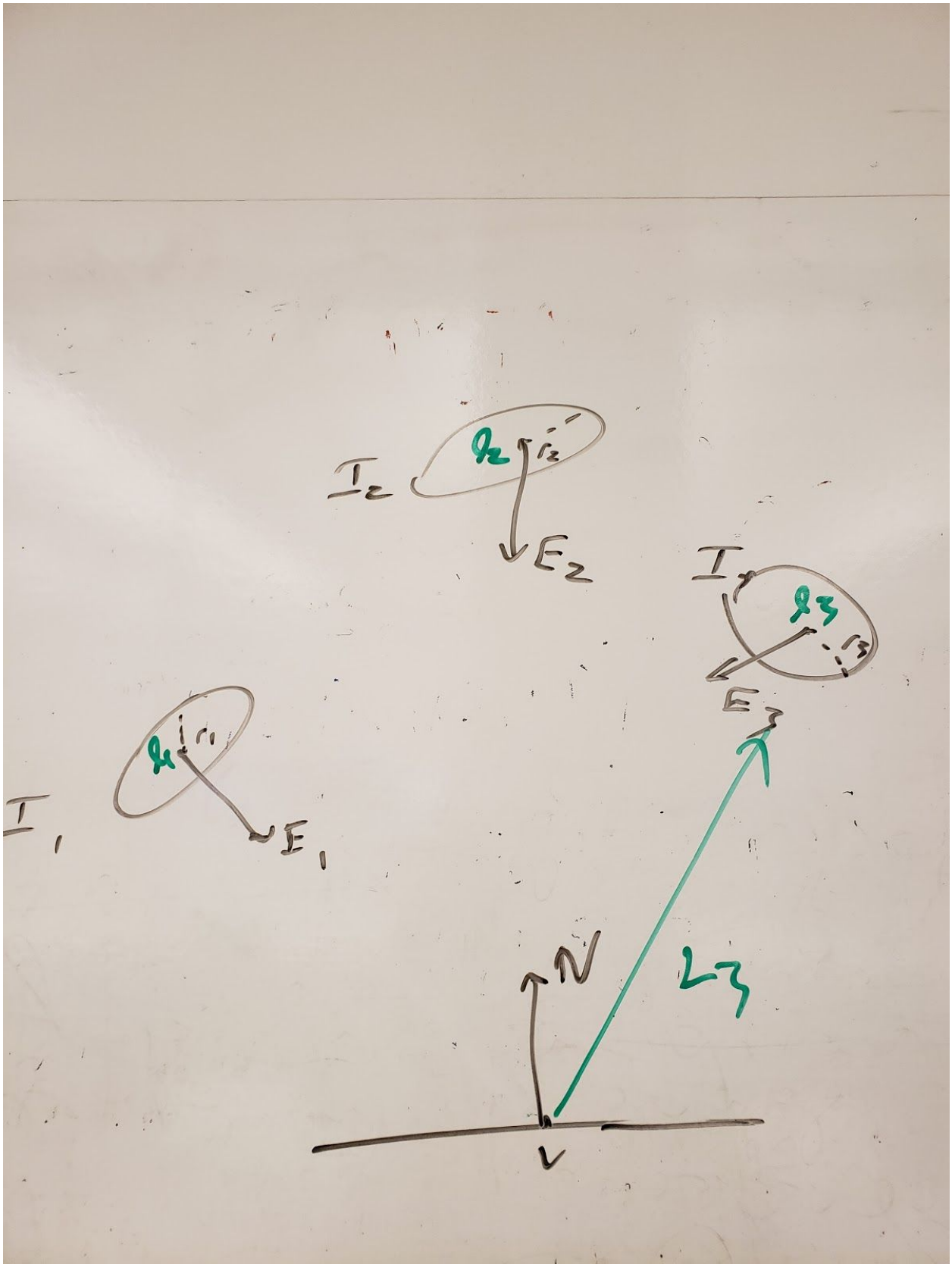
Code Available:

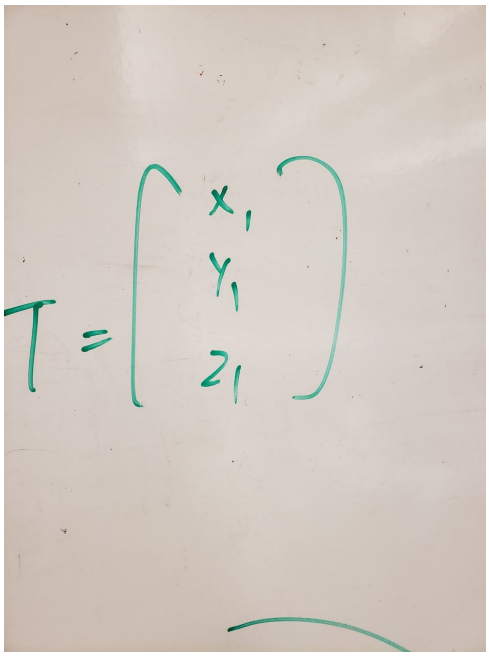
- Sine Power Integral Evaluation
- Stoke's Theorem Derivation of Boundary Integral Series for Specular Integral
- Recurrence Relations for Integrals of the form $(a + b\cos\Phi)^n$
- Evaluation of Finite Series of Integrals of Form $(a + b\cos\Phi)^n$

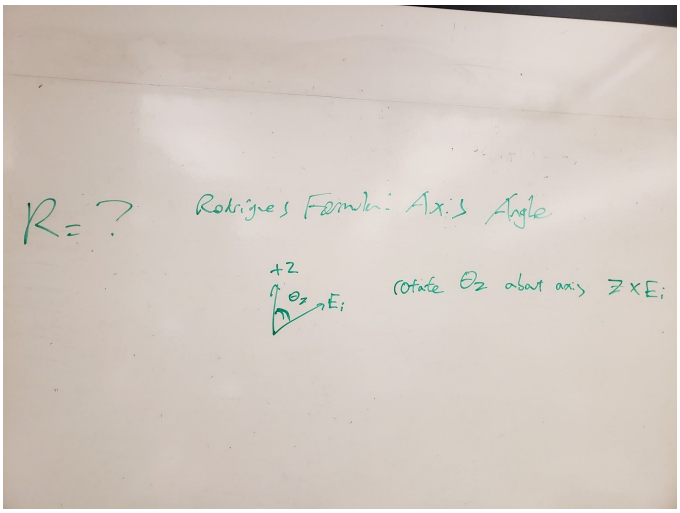
- Evaluation of Sub-hemispherical Phong Illumination
- Evaluation of Finite Series of Integrals of Form $(a + \cos\Theta + b \sin\Theta)^n$
- Evaluation of Polygonal Phong Illumination
- Side Note: We need to scale the models by the cross-sectional area to effectively convey the reduction/attentuation(?) of the lighting

Fragment Shader for Area Lighting




I_2
q_2
r_2
E_2
I_3
q_3
r_3
E_3
q_1
r_1
I_1
E_1
N
L_3
V

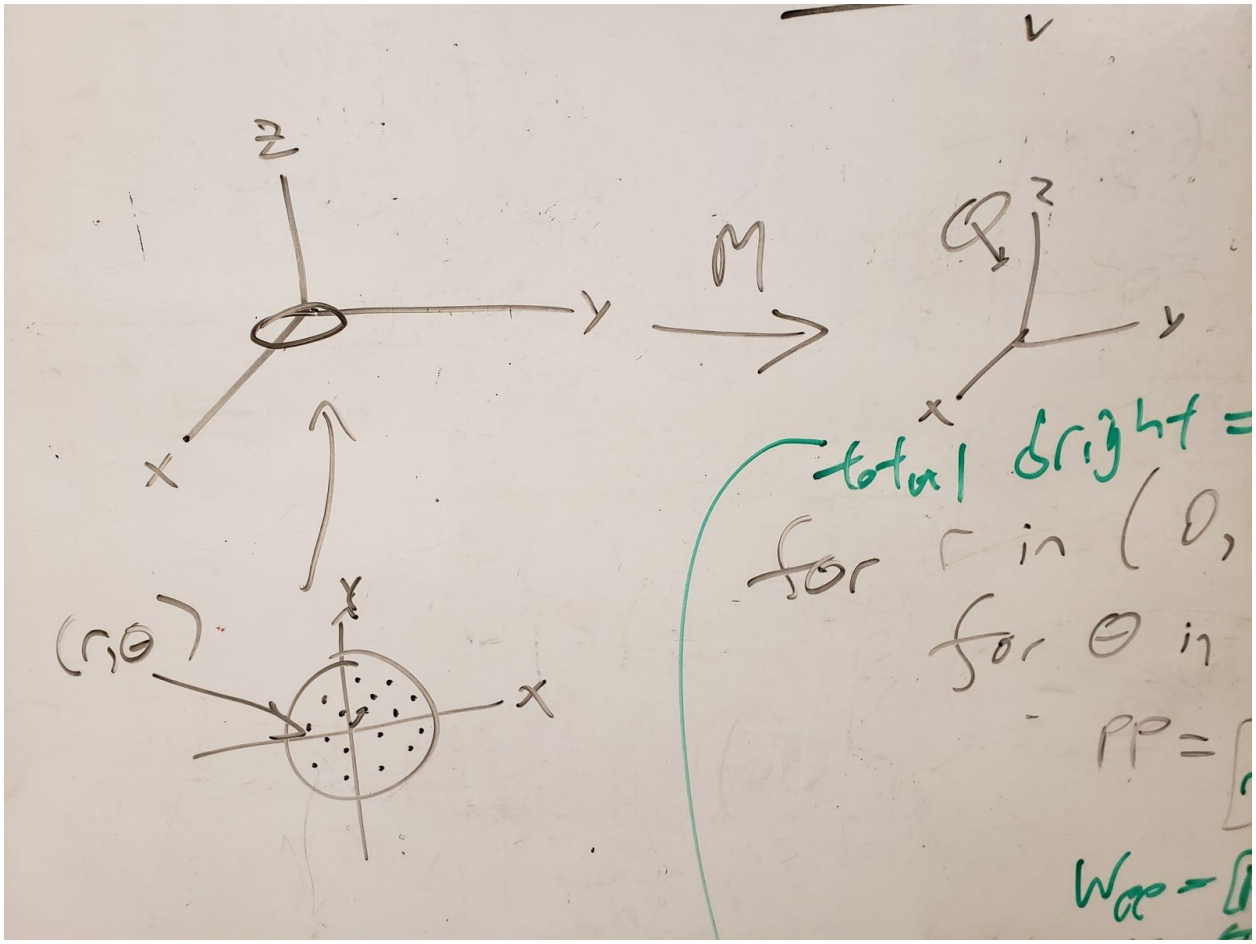




$$\mathbf{v}_{\text{rot}} = \mathbf{v}\cos\theta + (\mathbf{k}\times\mathbf{v})\sin\theta + \mathbf{k}\,(\mathbf{k}\cdot\mathbf{v})(1-\cos\theta)\,.$$

$$\mathbf{k} = \frac{\mathbf{a}\times\mathbf{b}}{|\mathbf{a}\times\mathbf{b}|} = \frac{\mathbf{a}\times\mathbf{b}}{|\mathbf{a}||\mathbf{b}|\sin\alpha}\,.$$

- vectors **a** and **b** which define the plane of rotation, and the sense of the angle θ is measured away from **a** and towards **b**. Letting α denote the angle between these vectors


z
y
x
M
z
y
x
(r,θ)
y
x
total bright =
for r in (0,
for θ in
PP =
W_ap =

Radiosity and Realistic Image Synthesis:

- Radiosity: measure of radiant energy, in particular, the energy leaving a surface per unit area per unit time.
    - Utilized/Developed into a set of computational techniques for computing global illumination
- Radiosity used in global illumination.
    - Ray tracing evaluates illumination equation for directions and locations determined by view and pixels of image
    - Radiosity solves illumination equation at locations distributed over the surfaces of environment
        - View-independent technique
    - Computation complete before viewing parameters, leading to interactive walkthroughs of simulated environments
    - Includes reflection (glossy/ideal) and media (smoke/haze)
- Finite Element Methods:
    1. Solves difficult integral equations by approximation unknown functions by subdividing the domain of a function into smaller pieces/elements
    2. approximated using relatively simple functions (polynomials).
    3. Projected into finit function space, characterized by finite number of unknowns
    4. System solved numerically

Optics:

- Light is a form of electromagnetic radiation
    - Sinusoidal wave formed by coupled electric and magnetic fields.
    - Electric and magnetic are perpendicular to direction of propagation
    - Frequency → wavelength size, and can exist at any wavelength
    - Pure light source (Laser) is single frequency
    - Coherent: source is tuned so that wave stays in phase as it propagates
    - Incoherent: wave doesn't stay in phase. Natural light.
    - Polarization: preferential orientation of electric and magnetic field vectors relative to direction of propagation. (directed vs random orientation)
    - Geometric (ray) optics, Physical (wave) optics, and quantum (photon) optics

Radiometry and Photometry

- Radiometry: science of physical measurement of electromagnetic energy
- Photometry: psychophysical measurement of visual sensation produced by electromagnetic spectrum
- Spectral Luminous Relative Efficiency: a plot of relative sensitivity of eye across visible spectrum (between 400nm and 700nm)

$R = \int V(\lambda)S(\lambda)d\lambda$ from $\lambda = 380$ to 770

- V = relative efficiency
- S = spectral energy

Integral of time/power = exposure

Global: total power in system must equal power being absorbed by surfaces.
Local: energy flowing into a region of space/surface element must equal energy flowing out

- Absorb, reflect, transmitted light never greater than incident light.

Light Field

- Transport Theory:
    - Study of how mass, charge, energy, or light flows
    - Easiest to represent as particles
    - P(x) =Total number of particles
        - absolute/extrinsic quantity
    - p(x) = particle density
        - intrinsic/differential quantity
    - dV = differential volume
        - A
- Count total number of particles flowing across small differential surface dA in slice of time dt
    - Computed considering tube formed by sweeping dA a distance vdt in direction -v
    - All particles across dA (between t and t + dt) must have initially been inside tube.
    - Can compute number of particles crossing surface element by multiplying the particle volume density times volume of tube
        - Volume of tube = base (dA) times height ($v\cos(\Theta)dt$)
        - Θ = angle between v and normal to dA
    - $P(x) = p(x)(vdt\cos\Theta)dA$
- Flow across a surface depends on the cosine of the angle of incidence between the surface normal and the direction of the flow.
    - Max flow through a surface occurs when particles flow perpendicular to direction of flow. (vice versa for flow parallel to surface
    - # particle flow proportional both to differential area of the surface element and to the interval of time used
    - Divide through by dt and surface area (dA) and take limit as these quantities goes to zero. This quantity is called flux.
        - Particle density is now a function of position x and direction w.
        - $P(x, w) = p(x,w)\cos\Theta dwdA$

Radiance and Luminance

- Light Field theory can be applied to light transport
    - Rendering system disregard quantum mechanics of light

    - Light transport of flux (stuff that flows) is radiant energy per unit time, or radiant power Φ, rather than particles.
- Radiance = Energy per unit volume → photon volume density * energy of a single photon (h c/ λ)
    - H is Planck's constant and c is the speed of light.
    - Power per unit projected area perpendicular to the ray
    - Characterizes distribution of light in a scene
    - 5 variables
        - 3 for position, 2 for direction
        - Differential flux in small beam with cross-sectional area dA and differential solid angle dw is $d\Phi = L(x,w)\ \cos\Theta dw dA$
1. Radiance in direction of light ray remains constant as it propagates along the ray
    - Total flux leaving first surface must equal flux arriving on the second surface
    - Throughput: large the throughput, larger the beam.
        - $T = dw_1 dA_1 = dw_2 dA_2 = (dA_1 dA_2)/ r^2$
2. Response of sensor is proportional to the radiance of the surface visible to the sensor.
    - Constant of proportionality is the throughput, which is only a function of geometry of sensor
    - If a surface is far, the sensor sees more of it, the sensor also subtends a smaller angle with respect to the surface.
    - Increase of energy resulting from integrating over a larger surface area is counterbalanced by the decrease in percentage of light that makes it to the sensor.

Angles and Solid Angles:

- Positions on sphere represented by number of degrees from North Pole (zenith Θ) and number of degrees about equator (azimuth Φ). Directions (unit vectors) and spherical coordinates can be used interchangeably
- $r\sin\Theta$, $dA = r^2 \sin\Theta d\Theta d\Phi$
    - $rd\Theta$ is the length of longitudinal arc generated as Θ goes to $\Theta + d\Theta$
    - $r\sin\Theta d\Phi$ is length of latitudinal arc generated as Φ goes to $\Phi + d\Phi$
    - Product of both is differntial area
- Solid angle subtended by sperical area a is equal to a/r^2
    - measure of angle in steradians, denoted sr
- Differential solid angle = $dw = dA/r^2 = \sin\Theta d\Theta d\Phi$

Irradiance and Illuminance

- Total energy per unit area incident onto a surface with a fixed orientation
    - Computed integrating incident, or incoming radiance, L_i, over a hemisphere Ω
    - $d\Phi = [\int_\Omega L\cos(\Theta)\ dw\ ]\ dA$
- Irradiance, E: radiant energy per unit area falling on a surface

    - $E = d\Phi/dA$
    - $E = \int_\Omega L\cos(\Theta)\, dw$
- $\cos\Theta\, dw$ = projected solid angle
- $E = E_0\cos\Theta$

Radiosity and Luminosity

- $B = \int_\Omega L_0 \cos\Theta\, dw$
- L_0 is outgoing radiance
- Photometric equivalent → luminosity

Radiant and Luminous Intensity

- Radiant intensity of point light source with units of power per unit solid angle.
    - Equivalent photometric quantity is the luminous intensity
    - $d\Phi = I(w)dw$
- Intensity in a given direction is equal to irradiance at a point on unit sphere centered at the source.
    - Intensity defined to be power per unit area (rather than per unit solid angle).
- For an isotropic point light source,
    - $I = \Phi/ 4pi$
- Point source acts like a spotlight and radiates different amounts of light in different directions. Total energy emitted is
    - $\Phi = \int_\Omega I(w)dw$
- Irradiance on differential surface due to a single point light source can be computed by calculating solid angle subtended by the surface element from the point of view of the light source
    - $E = I(dw/dA) = \Phi/4pi * \cos\Theta/|x - x_s|^2$
    - $|x - x_s|$ distance from point to surface element
        - $1/r^2$ is from falloff

Summary of Radiometric and Photometric Quantities

- 6 radiometric quanitites
    - Radiant energy
    - Radiant power
    - Radiance
    - Irradiance
    - Radiosity
    - Radiant Intensity

Reflection Functions

- Process by which light incident on a surface leaves that surface from the same side.

Bidirectional Reflection Distribution Function

- Increase in incident light energy per unit area results in corresponding increase in reflected energy
- $dL_r(w_r) \in dE(w_i)$
- Increase irradiance through increasing solid angle or energy density

BRDF

- $f_r(w_i \rightarrow w_r) = (dL_r(w_r))/ (dL_i(w_i)cos\Theta_i dw_i$
- Ratio of reflected radiance in the direction w_r to the differential irradiance from the incident direction w_i that produces it
- Bidirectional because it depends on two directions
    - Strictly positive function (0 → infinity)

1. If BRDF based on physical laws, it will remain unchanged if the incident and reflected directions are interchanged
    a. $w_r \rightarrow w_i == w_i \rightarrow w_r$
2. BRDF is anisotropic
    a. If incident and reflected are fixed, underlying surface rotated about normal, percentage of light may change
    b. Smooth surface (most materials) don't depend on surface orientation, thus reflection is unchanged if rotated. (implies 3 degrees of freedom instead of four)
    c. Reflection behaves linearly, total amount of light reflected by a surface in a specific direction is given by hemispherical integral over all possible incident directions
        i. Reflected radiance in a direction is due to radiance arriving from all directions weighted by the BRDF relating the incoming and reflected directions and by the projected solid angle.

Mirror Reflection:

- Angle of reflectance is equal to angle of incidence
    - Reflected vector is in the plane determined by incident ray and surface normal vector
- Reflected radiance is exactly equal to incident radiance
    - Can be expressed using delta functions (properties below)
        1. $\delta(x) = 0$ if $x \neq 0$
        2. $\int\delta(x)dx = 1$ from 0 to infinity
        3. $\int\delta(x - y) f(x)dx = f(y)$ from 0 to infinity

Reflectance:

- Delta function interpreted as infinitesimally thin, infinitely high spike with unit area
    - Implies BRDF may be infinite
- Biconical reflectance → easier to work with quantity between 0 and 1
- Reflectance depends on the distribution of incoming light, L_i.

    - Assumed L_i is uniform and isotropic, then L_i can be taken out from integral in both numerator and denominator.
- Reflectance uses double integral over incident and reflected directions for which limits of integration haven't been set
- Directional, conical, hemispherical
- Amount of light scattered into entire hmeisphere from a single incident direction
    - Amount of light scattered into

Lambertian Diffuse Reflection

- Diffuse reflectance modeled by assuming light equally scattered in any direction regardless of incident direction

1. Value of reflected radiance is proportional to incident irradiance
2. Reflected radiance is a constant and hence the same in all directions

- BRDF is a constant, and the reflectance is also a constant
- Can parameterize BRDF in terms of the reflectance: $f_{r,d} = p_d/pi$.
    - Intuitive to describe materials using reflectance
    - P used in text, can be assumed between 0 and 1

Glossy Reflection

- Treat general BRDF as sum of mirror specular reflection, Lambertian diffuse reflection, and glossy reflection
    - Real environment not smooth, BRDF contains component where light reflected into a complex distribution of outgoing directions

- Glossy: scattering of light from rough surfaces
    - Glossy reflection increases at glancing angles of incidences and reflection.
    - Mirror specular comes from perfectly smooth
    - Lambertian diffuse from multiple surface reflections from rough surfaces and subsurface scattering
- Microfacet Theory: surface is made of little reflective facets, each behaving as a small mirror; each reflecting perfectly.
    - Amount of light from source toward eye == relative number of microfacets oriented halfway between eye and light source

Rendering Equation

- specify/compute incident light distribution
    - Illumination model
- Local and Indirect Illumination

Local/Direct Illumination

- Direct lighting from point light sources
    - Depends on individual properties of light sources and surface being shaded.
- If there are n light sources, hemispherical integral collapses to a sum over n sources

- Extending to light sources with arbitrary directional distributions
    - radiant intensity in direction of surface
    - Linear and Area light sources can be used, but involves integrating the reflectance function over the range of possible direction incident from the light source.

Global or Indirect Illumination

- Light may come from any surface in environment, very important in shadowing
- Relate illumination to reflected light distribution from another surface.
- Spatial dependence of radiance needs to be explicit
- Radiance is invariant along a ray
- Switch hemispherical integral over all incident directions to an area integral over all other surfaces.

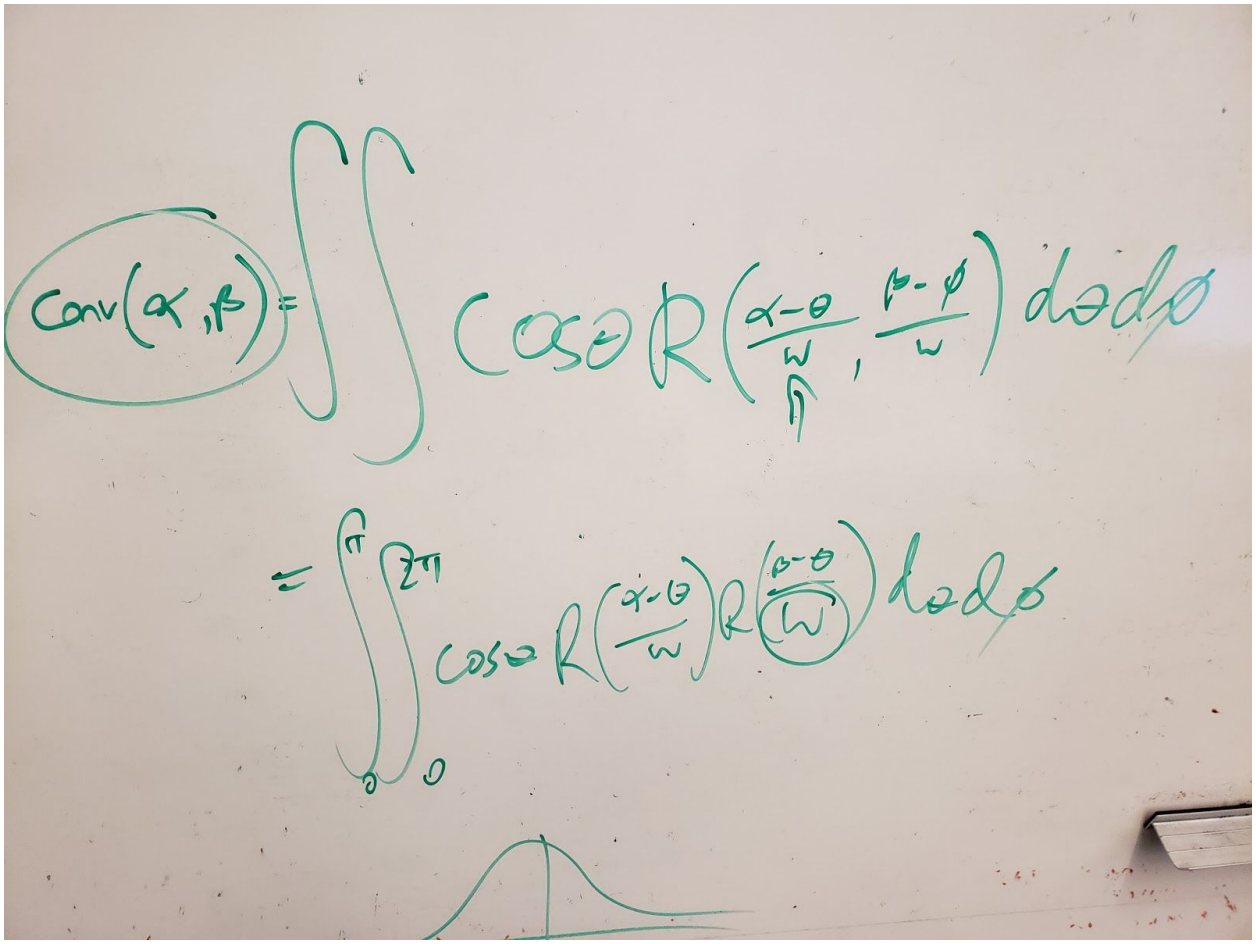
Radiosity Equation

- BRDF independent of incoming and outgoing directions and can be taken out from under integral.
    - Outgoing radiance from Lambertian surface is same in all directions and equals radiosity B divided by pi
- Rendering equation expresses conservation of light energy at all

Area Lighting Measurement through Convolutions

Python/Matlab:
Solve double integral
conv(α, β) = $\int_0^{pi}\int_0^{2pi}$ cosθ R((α - θ)/ w) R((β-θ)/ w)dθdΦ
Finding sum of light from f * g(t)
Trying to figure out impact of w
Plot the results

Links:
https://docs.anaconda.com/anaconda/user-guide/getting-started
http://www.scipy-lectures.org/index.html
https://code.visualstudio.com/docs/python/python-tutorial#_prerequisites
https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.dblquad.html
https://docs.scipy.org/doc/numpy/reference/generated/numpy.convolve.html
https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html#general-multiple-integration-dblquad-tplquad-nquad
https://matplotlib.org/users/pyplot_tutorial.html
https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot



Parameters: **func : *callable***

A Python function or method of at least two variables: y must be the first argument and x the second argument.

a, b : *float*

The limits of integration in x: $a < b$

gfun : *callable or float*

The lower boundary curve in y which is a function taking a single floating point argument (x) and returning a floating point result or a float indicating a constant boundary curve.

hfun : *callable or float*

The upper boundary curve in y (same requirements as *gfun*).

args : *sequence, optional*

Extra arguments to pass to *func*.

epsabs : *float, optional*

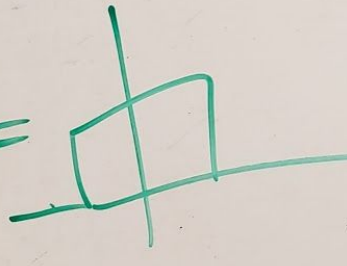
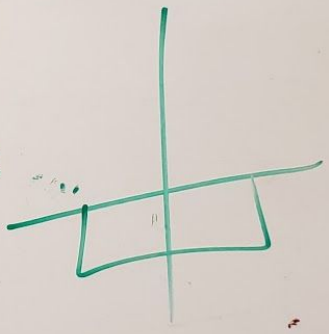
Absolute tolerance passed directly to the inner 1-D quadrature integration. Default is 1.49e-8.

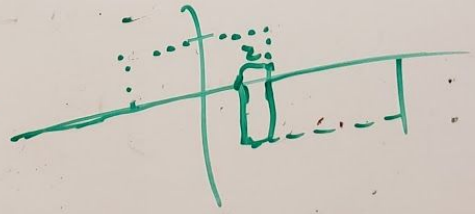
epsrel : *float, optional*

Relative tolerance of the inner 1-D integrals. Default is 1.49e-8.

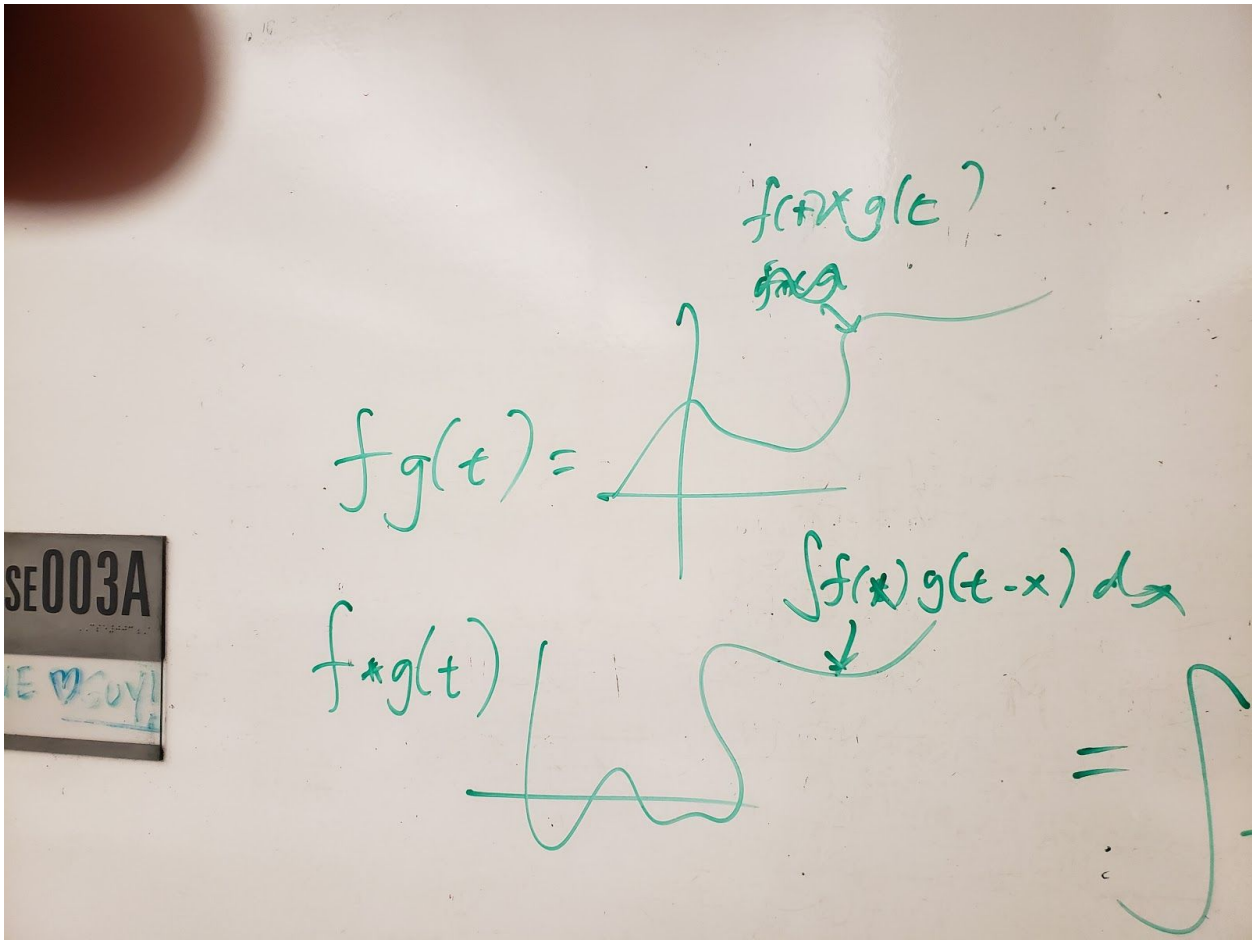
$) dx \quad f * g\,(t)$

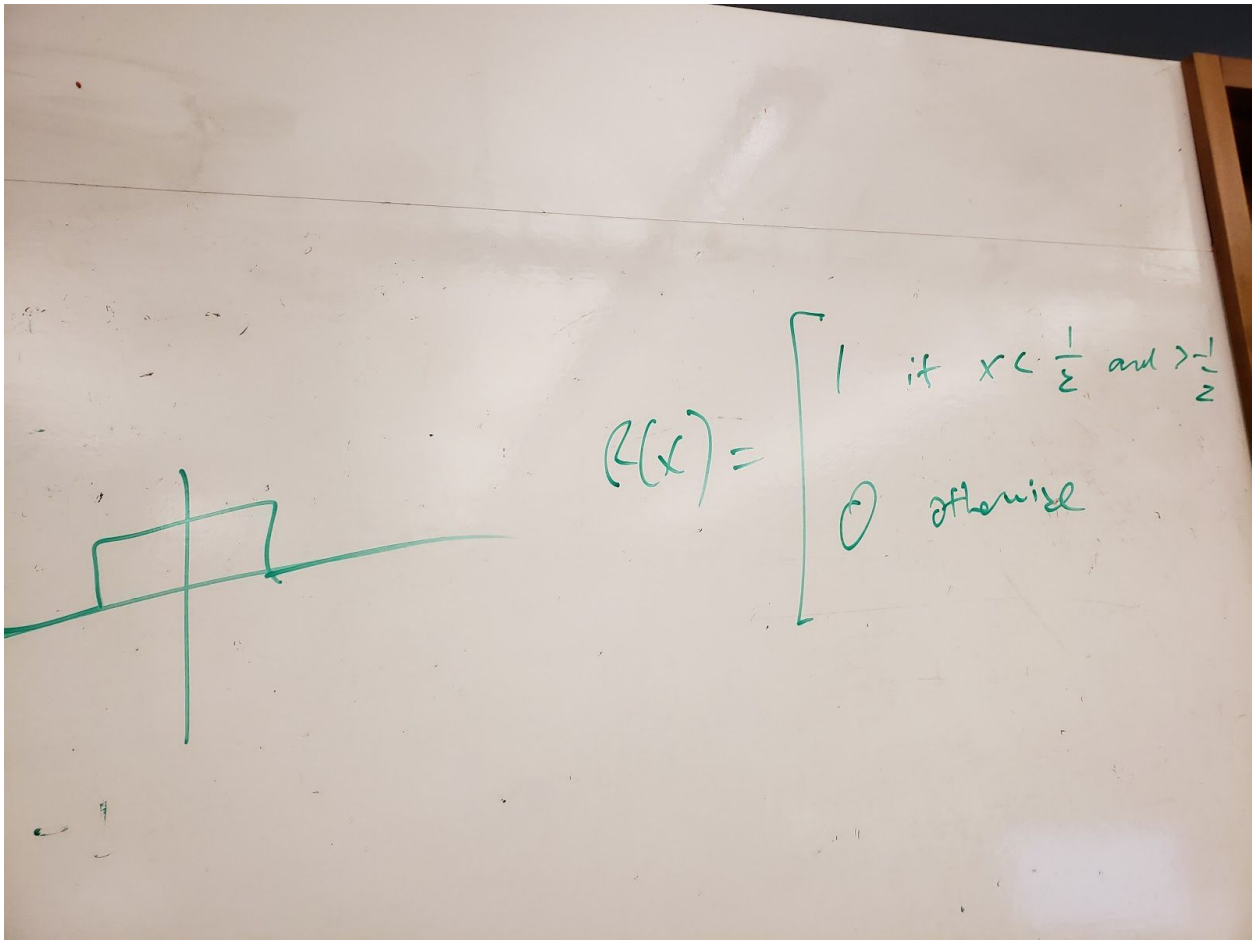
$$= \int f(x)\, g(t-x)\, dx$$

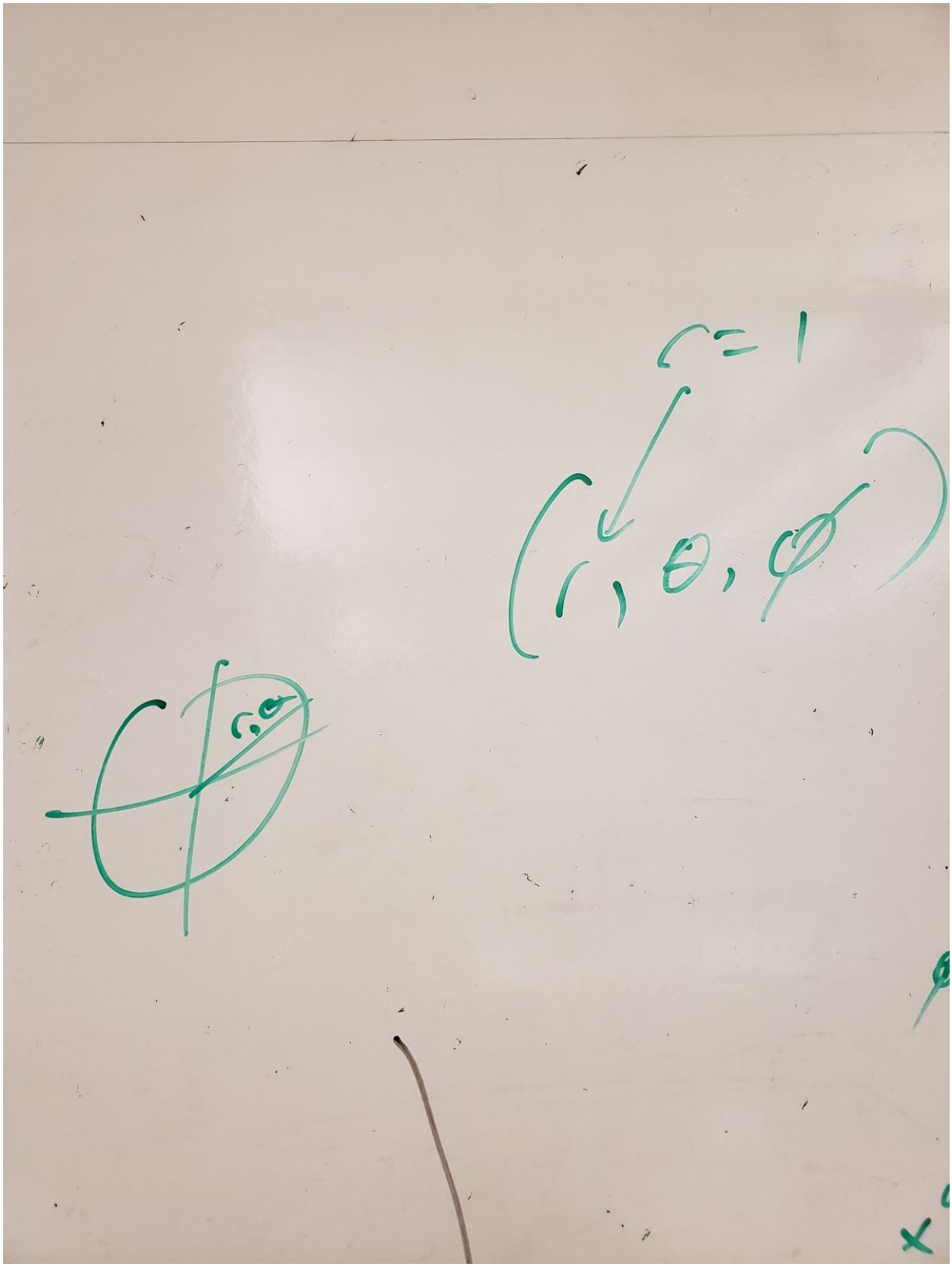
$f =$  , $g =$ 

$f * g(3) =$ 

$= -2$


f(t)*g(t)
f*g
f*g(t)
∫f(x) g(t-x) dx
=


$R(x) = \begin{cases} 1 & \text{if } x < \frac{1}{2} \text{ and } > -\frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$

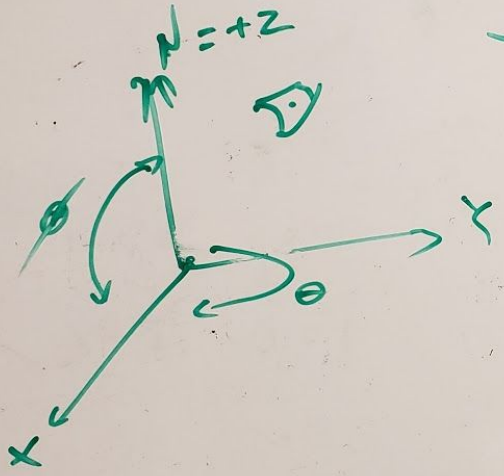

r = 1
(r, θ, φ)

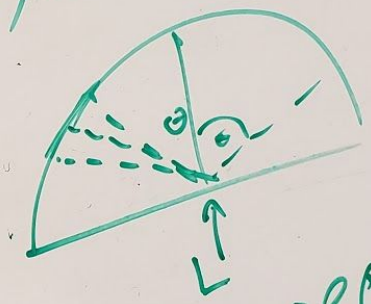
$L \cdot N = |L||N| \cos\theta$

$\cos\theta$

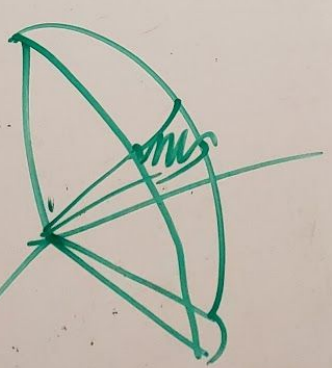
L?

$L = \int (L \cdot N) I$ R





$= \sum \cos\theta \cdot R\left(\frac{\theta}{\omega}\right)$



```
#unsure how to plug in function for alpha beta
#https://stackoverflow.com/questions/6289646/python-function-as-a-function-argume
nt
#https://docs.python.org/3/reference/expressions.html#lambda
```

Returning values from python script:
https://stackoverflow.com/questions/18231415/best-way-to-return-a-value-from-a-python-script

https://opencv.org/

Spherical Harmonics

- Robby papers in 2000s

Fourier Series

- Any periodic functions is a sum of sines and cosines
- Various frequencies

Integral here is multiplication in fourier

Any function in terms of spherical harmonics

- Can rotate it back at any point

Can computer spherical harmonic expansion for disk being convolved
Multiply spherical harmonic coefficients (in harmonics domain) and convert back
9 coefficients is fine (don't need infinite)

Spherical harmonics on rectangles and cosine

Sensitivity analysis of photometric stereo with area lights.

- Other research

Fourier Series

Integrals from 0 → 2pi

- Integral sin(mt)
  - = 0 for any integer
- Integral cos(mt)
  - = 0 for any non-zero integer
- Integral sin(mt) * cos
  - Break down with trig functions
  - Leads to 2 cases above, and thus = 0 for any integers
- Integral sin(mt) * sin(mt)
  - = 0 for integers m =/= n or m =/= -n
  - Leads to cases above through trig
- Integral $\sin^2(mt)$
  - = pi when m is non-zero int
- Integral cos(mt) * cos(mt)
  - = 0 when int m =/= n or m =/= 0
- Integral $\cos^2(mt)$
  - = pi when m is non-zero int

Fourier Series Expression

f(t) =

a_0	
+ $a_1\cos(t)$	+ $b_1\sin(t)$
+ $a_2\cos(2t)$	+ $b_2\sin(2t)$
...	...
+ $a_n\cos(nt)$	+ $b_n\sin(nt)$

First Term Approximation

1. Take integrals of all terms from 0 → 2pi
2. Take coefficients outside
3. All integrals of non-zero integer is = 0 except term a_0
4. $a_0 \int_0^{2pi} dt = a_0(t)|_0^{2pi}$
5. $a_0(2pi - 0) = a_0 2pi$

6. $\int_0^{2pi} f(t)dt = a_0 2pi$
7. $a_0 = 1/2pi \int_0^{2pi} f(t)\, dt$
    - Average value of f from $0 \rightarrow 2pi$

Fourier coefficients for cosine terms

Find a_n where n =/= 0

f(t)cos(nt) =

$a_0\cos(nt)$	
+ $a_1\cos(t)\cos(nt)$	+ $b_1\sin(t)\cos(nt)$
+ $a_2\cos(2t)\cos(nt)$	+ $b_2\sin(2t)\cos(nt)$
...	...
+ $a_n\cos(nt)\cos(nt)$	+ $b_n\sin(nt)\cos(nt)$

1. Multiply f(t) by cos(nt), and every term by cos(nt)
2. Take integral from $0 \rightarrow 2pi$ of every term and f(t)
3. All terms cancel out to be 0 except for $a_n \int_0^{2pi} \cos^2(nt)dt$
4. $a_n * pi = \int_0^{2pi} f(t)\cos(nt)dt$
5. $a_n = 1/pi \int_0^{2pi} f(t)\cos(nt)\, dt$

Fourier Coefficients for Sine Terms

f(t) sin(nt) = $a_0\sin(nt)$

$a_0\sin(nt)$	
+ $a_1\cos(t)\sin(nt)$	+ $b_1\sin(t)\sin(nt)$
+ $a_2\cos(2t)\sin(nt)$	+ $b_2\sin(2t)\sin(nt)$
...	...
+ $a_n\sin(nt)\sin(nt)$	+ $b_n\sin(nt)\sin(nt)$

1. Multiply f(t) by sin(nt), and every term by sin(nt)
2. Integral from $0 \rightarrow 2pi$ of every term and f(t)
3. All terms cancel out to be 0 except for $b_n \int_0^{2pi} \sin^2(nt)dt$

4. $b_n pi = \int_0^{2pi} f(t) \sin(nt)dt$

$B_n = 1/pi \int_0^{2pi} f(t)\sin(nt)dt$

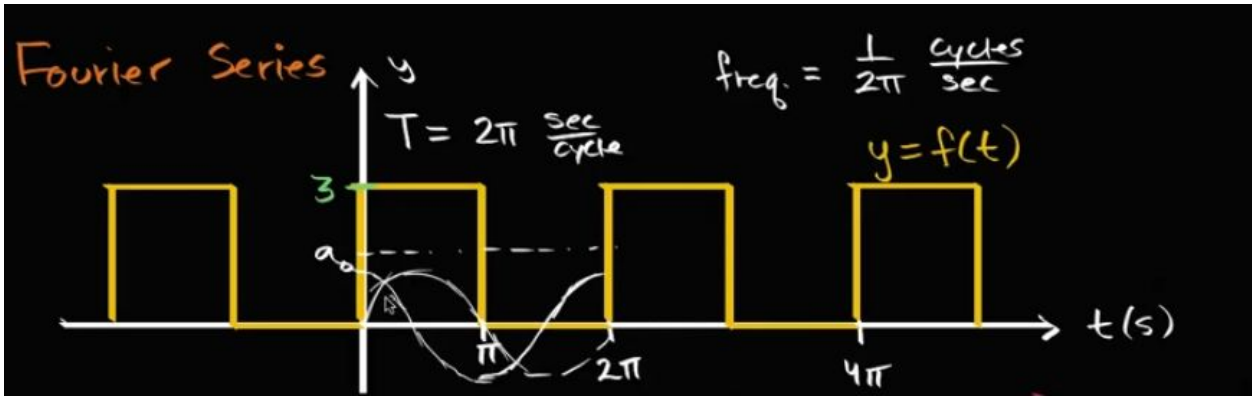
Fourier Coefficients for Square Wave

$a_0 = 1 / 2pi \int_0^{2pi} f(t)\, dt$; $a_n = 1/pi \int_0^{2pi} f(t)\cos(nt)dt$; $b_n = 1/pi \int_0^{2pi} f(t)\sin(nt)dt$

$a_0 = 1/2pi (\int_0^{pi} 3dt + \int_{pi}^{2pi} 0\, dt) = 1/2pi(3t)_0^{pi} = 3pi/2pi = 3/2$
$a_n = 1/pi \int_0^{pi} 3\cos(nt)dt = 1/n * 3/pi \int_0^{pi} n * \cos(nt) = 3/(n * pi)\, (\sin(nt))_0^{pi}$
$= 3/(n * pi)\, (0 - 0) = 0$
$b_n = 1/pi \int_0^{pi} 3 \sin(nt)dt = -1/n* 3/pi \int_0^{pi} \sin(nt)dt * -n =$
$-3/(n * pi)(\cos(nt))|_0^{pi} = -3/(n*pi)(\cos(n * pi) -1)$

- N is even
    - $b_n = 0$
- N is odd
    - $-6/(n * pi)$

$f(t) = a_0 + a_n + b_n$
$f(t) = 3/2 + 0 + 6/pi \sin(t) + 6/(3pi) * \sin(3t) + 6/(5 * pi) * \sin(5t) + \ldots$



Fourier Transforms

https://www.youtube.com/watch?v=1JnayXHhjlg

$$F(v) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i v t} dt$$

- Function of time becomes a function of frequency

Inverse Fourier transform

$$f(t) = \int_{-\infty}^{\infty} F(v) e^{2\pi i v t} dv$$

- Function of frequency becomes a function of time

Transformations

- Mapping between domains
- Fourier focuses on the transformations between time and Frequency
- Frequency F(v) = w/2pi = Hertz
- Still the same information, but different descriptions/phrasing
    - Can situationally be more effective

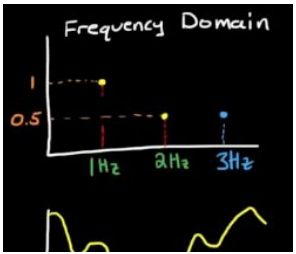
Any time domain can be represented by the sum several sinusoids

- Sinusoids described by amplitude, frequency, and phase
- All information necessary to describe signal
- Plot amplitude and phase at every frequency across entire spectrum
- Signals are comprised of multiple sinusoids of different amplitudes and phases across the entire spectrum
- Sinusoids only waveform doesn't change shape in a Linear Time-Invariant System

Transform signal (Frequency function) into something easier to work with

- Can think in terms of change in amplitude (or gain) and change in phase through a system and any given frequency
- Time Domain of signals represents real-life use
- Frequency Domain of signals useful in analyzing control systems
- Fourier transform helps move from one to other

Fourier extends into Laplace Transform



After 3 points in frequency domain, reduce into summations adding all frequencies between negative infinity and positive infinity

$f(t) = \cos(2\pi t) + 0.5 \cdot \cos(4\pi t) + 0.5 \cdot \cos(6\pi t)$ = $f(t) = \sum_{-\infty}^{\infty} A(v) \cos(2\pi v t)$

A(v) = all amplitudes at each frequency

- Multiplied by cos(each specific frequency)

Can replace with continuous spectrum of frequencies

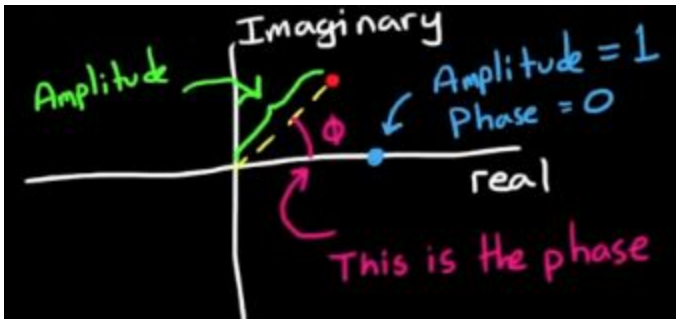
- Replace summation with integral
- Similar format to Inverse Fourier Transform

$$f(t) = \int_{-\infty}^{\infty} A(v) \cos(2\pi v t) dv$$

Taking parameters (amplitude, etc) and multiplying them by sinusoids and adding them all up to get the signal in the time domain

- Currently no phase shift information
- Chose cosine waves that had no phase shift degrees

Phase and amplitude can be described by a complex number



- Amplitude - sqrt($real^2$ + $imag^2$)
- tan(θ) = imag/real

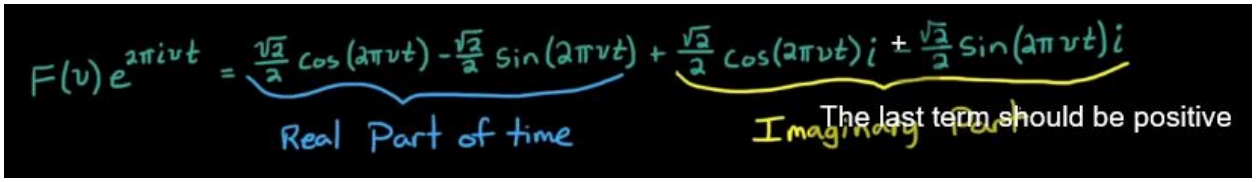
Results in complex number information for amplitude that includes phase information

→ F(v) = sqrt(2)/2 + sqrt(2)*i* / 2

Sines and cosines in complex plane using

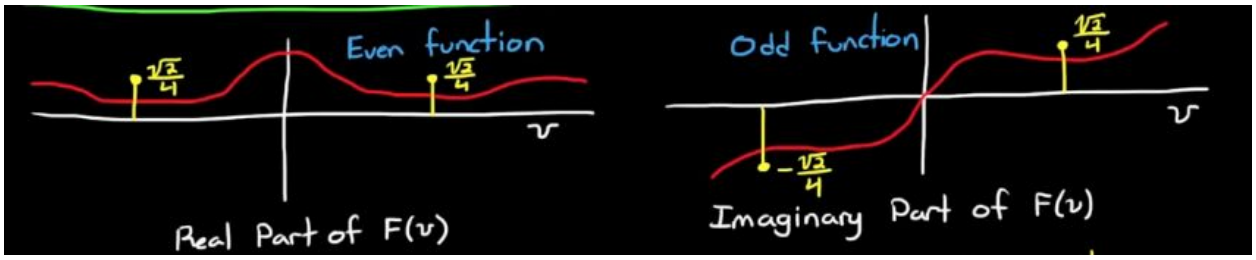
Euler's formula → $e^{it} = \cos t + i \sin t$

THEREFORE:
With phase A(v) cos(2 pi vt) becomes complex function $F(v)e^{2(pi)vt}$

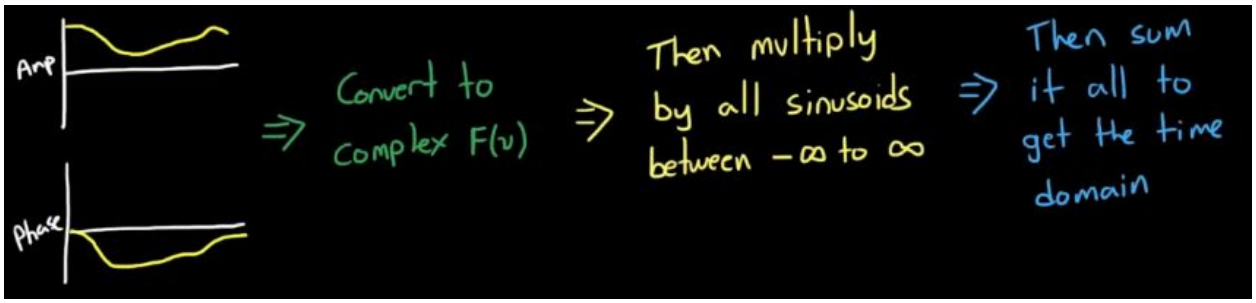


^--Positive Frequency

Negative Frequency cancels out imaginary part

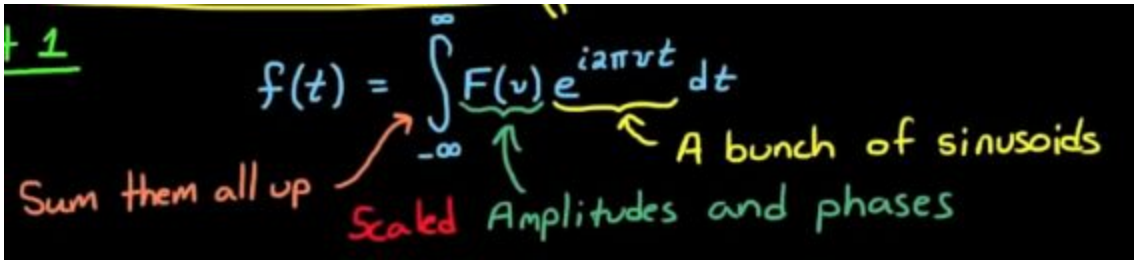


REVIEW:



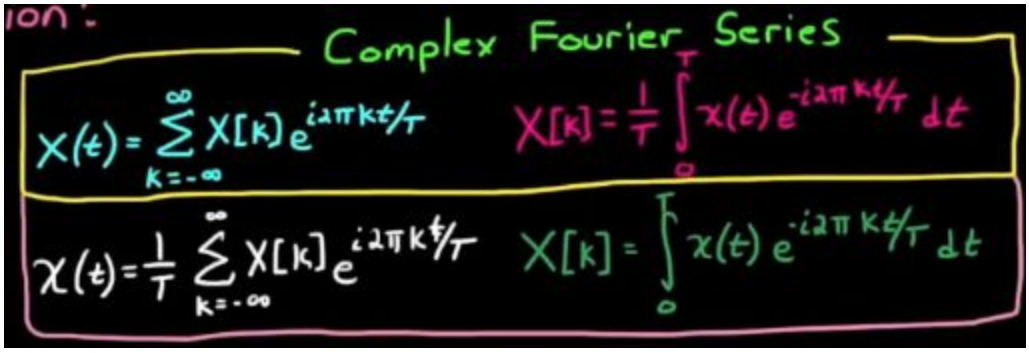
https://www.youtube.com/watch?v=kKu6JDqNma8
Scaling term from periodic time function to an a-periodic time function

- Part 1 was to simplify Inverse Fourier Transform into multiplication and summation steps which are analogous to

- 

Complex Fourier Series

- Used for periodic time functions
- Converts continuous-time signal into a discrete frequency signal
- Pair is forward and inverse functions together
    - Will undo each other
- Multiple variations of Complex Fourier Series of these equations
    1. Using frequency instead of angular velocity (ω), ω = 2pi * frequency
    2. $\int_0^T$ or $\int_{-T/2}^{T/2}$ or $\int_{-T}^{0}$
    3. Replacing T with f_o, $f_o = 1/T$
    4. Moving the scaling term to the other equation
        - Can write scaling factor (1/T) in the inverse equation.
        - Still retaining pairs, but redefining output signals

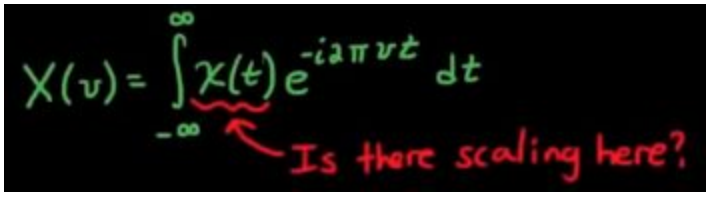


- Top are unscaled. Bottom are scaled by period. Both are correct equations, but scaling needs to be undone on the way out.
    - Make sure to use the transform pairs.

When period increases, frequency gets smaller

- Each harmonic of that fundamental frequency also gets smaller
- Limit of T as it approaches infinity produces an aperiodic time signal
    - Period of infinite time

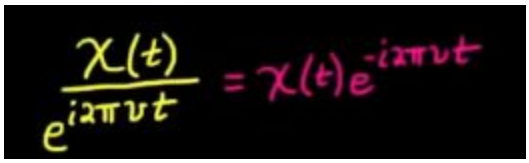
Fourier Transform Conceptualization



How many $5 are in $15?

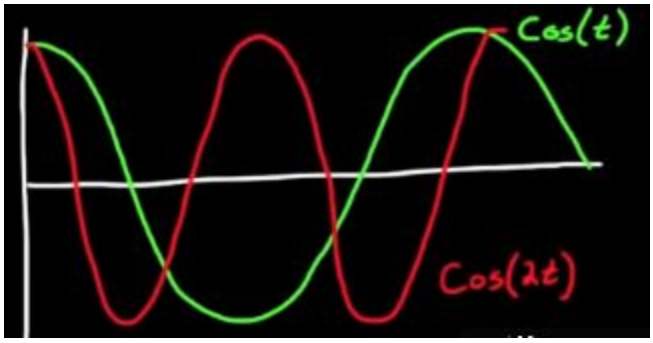
- 15 / 5 = 3

How much 15Hz signal is in X(t)



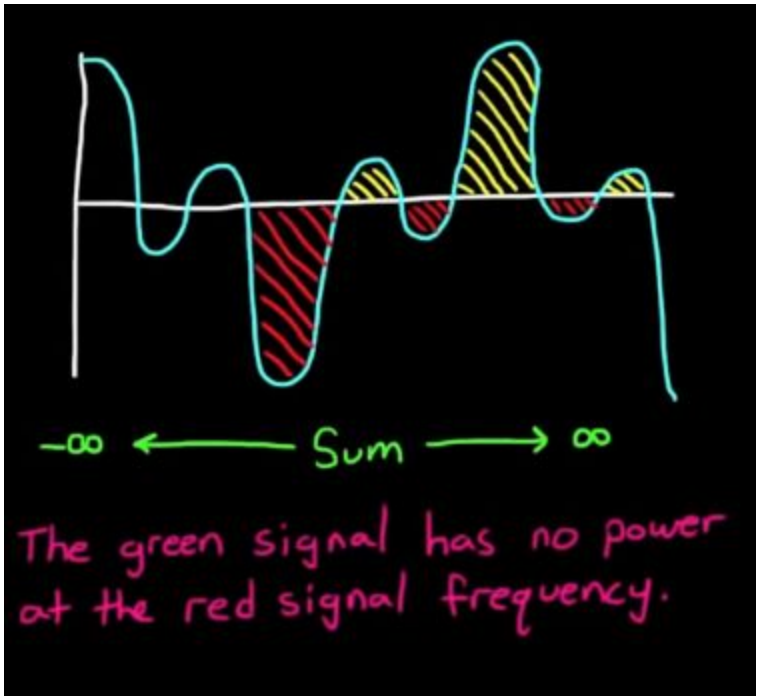
- Integrate across all time
- Integrate for all frequencies across the spectrum
    - We'll get the frequency domain representation

Graphical Example:



- Sum the result across all time will = 0
    - Multiplication since we moved e^{i2pivt} from denominator to numerator with negative exponent
    - Euler's Formula
    - 

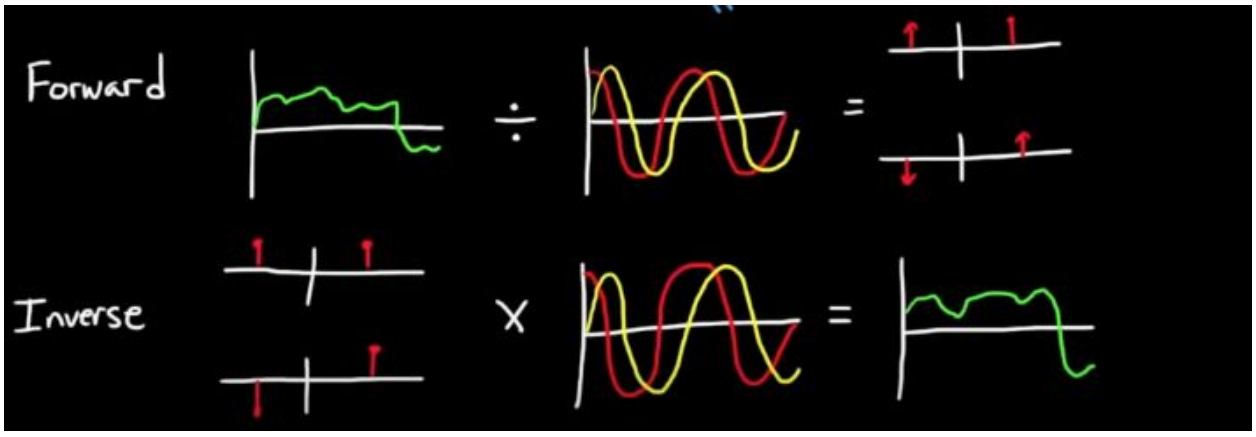
Resulting cos(t) * cos(2t)



On Imag/Real Graph, line on real axis, with no imag. So no phase shift → cosine wave. Amplitude is infinity, but no information from infinity. Need to scale by infinity, which still loses information

Direct Delta solves for these infinite losses of information
δ(n) =

- Infinitely tall and infinitesimally thing.
    - Area nder curve is 1.
    - Multiply amplitude by dirac delta
    - Take integral for all time leaves area under curve == amplitude.



Basics of Spherical Harmonics

https://dickyjim.wordpress.com/2013/09/04/spherical-harmonics-for-beginners/

- Values based on direction (i.e. light at a specific position)
- Sample it in every possible direction, store it using spherical harmonics
    - Stored as approximation, diffuses (blurry)
    - Not using them as ray-traced reflections
- Spherical Harmonics are infinite series
    - Usually cut at certain bands of detail.
    - Bands 0 indexed, each band B adds 2B +1 values to the series
    - Order 1 = 1 value, Order 2 = 4 value, Order 3 = 9 value (stops here usually)
- Meaning behind bands
    - Single value band 0 could be used as ambient occlusion term
    - Three values for band 1 could be like a bend normal
- Once you have coefficients, you an add, scale, and rotate them.
    - Adding to accumulate effects of multiple lights
    - Scaling to lerp between different values (different points)
    - Rotation means that you can easily move your SH into the space of your model rather than transforming per vertex or per pixel normals to the same space

http://www.paulsprojects.net/opengl/sh/technical.html

http://www.ohiouniversityfaculty.com/mohlenka/research/uguide.pdf

L^2 inner product of two functions → $\langle f, g \rangle = \int f(s)\bar{g}(s)ds$

- G denotes complex conjugation and the integral is over a space of interest
  - Sphere → $\int_0^{2\pi} \int_0^{\pi} \sin\phi d\phi d\theta$
- L^2 norm by ||f|| = sqrt(<f,f>)
  - Space L^2 consists of all functions such that ||f|| < infinity
- Basis for L^2 is set of functions $\{\varphi_i\}$ with properties
  - Orthogonal: $\langle \varphi_i, \varphi_j \rangle = 0$ for i =/= j
  - Normalized: $||\varphi_i|| = 1$
  - Span: Linear combination of φ_i`s; $f = \sum_i \alpha_i \psi_i$, a_i is a complex number

R^2: Fourier Series

- Polynomials in R^2 {(x,y): x,y ∈ R}
- Specifically Harmonic Polynomials
  - $\Delta_2 p(x, y) = 0,$
  - $\Delta_2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}.$
- Specifically Homogeneous Harmonic Polynomials:
  - Homogeneous of degree n if $p(tx, ty) = t^n p(x,y)$ for positive t
  - Homogeneous

f()
Function = weighted function $Y_{l,m}$

If functione nds up being only a function of theta.
f(theta,phy) will go away

Only a theta dependece for one of the functions
True for both
Apply convolution theorem

- Some f(theta,phi) convolved with g(theta)
- Can just multiply coefficients together to
- $A_{l,m}$for f(theata, phi), and $b_{l,0}$ for g(theta)
- Multiplying pairs of ^
- Look up convolution theorem

$H_{l,m} = a_{l,m}\ b_{l,0}$

Plug it back into original function

If m nonzero, wiped out due to euler's formula

$H_{l,0} = a_{l,0}\ \ b_{l,0}$

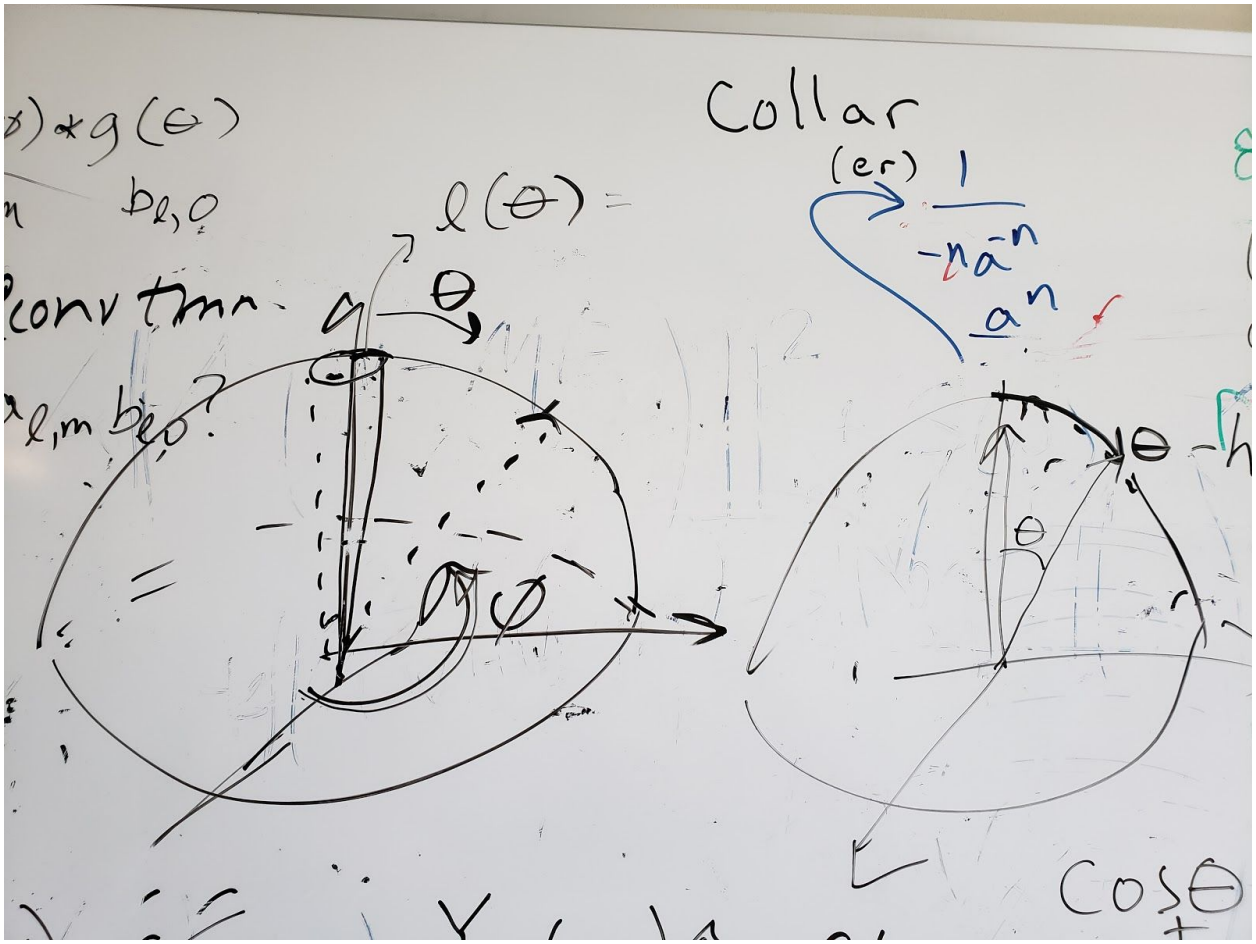
Ends up with a function purely of theta

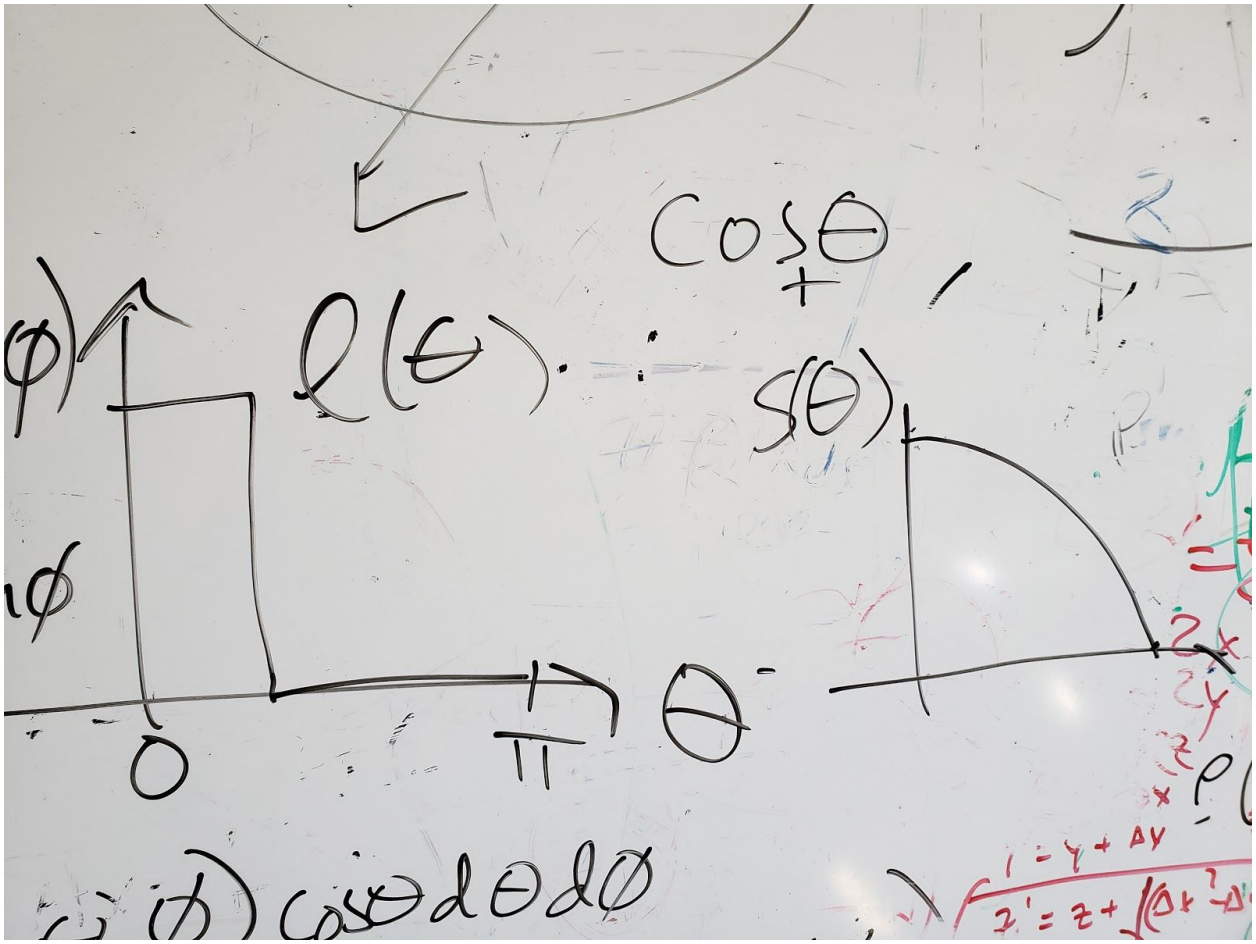
- No dependence on phi

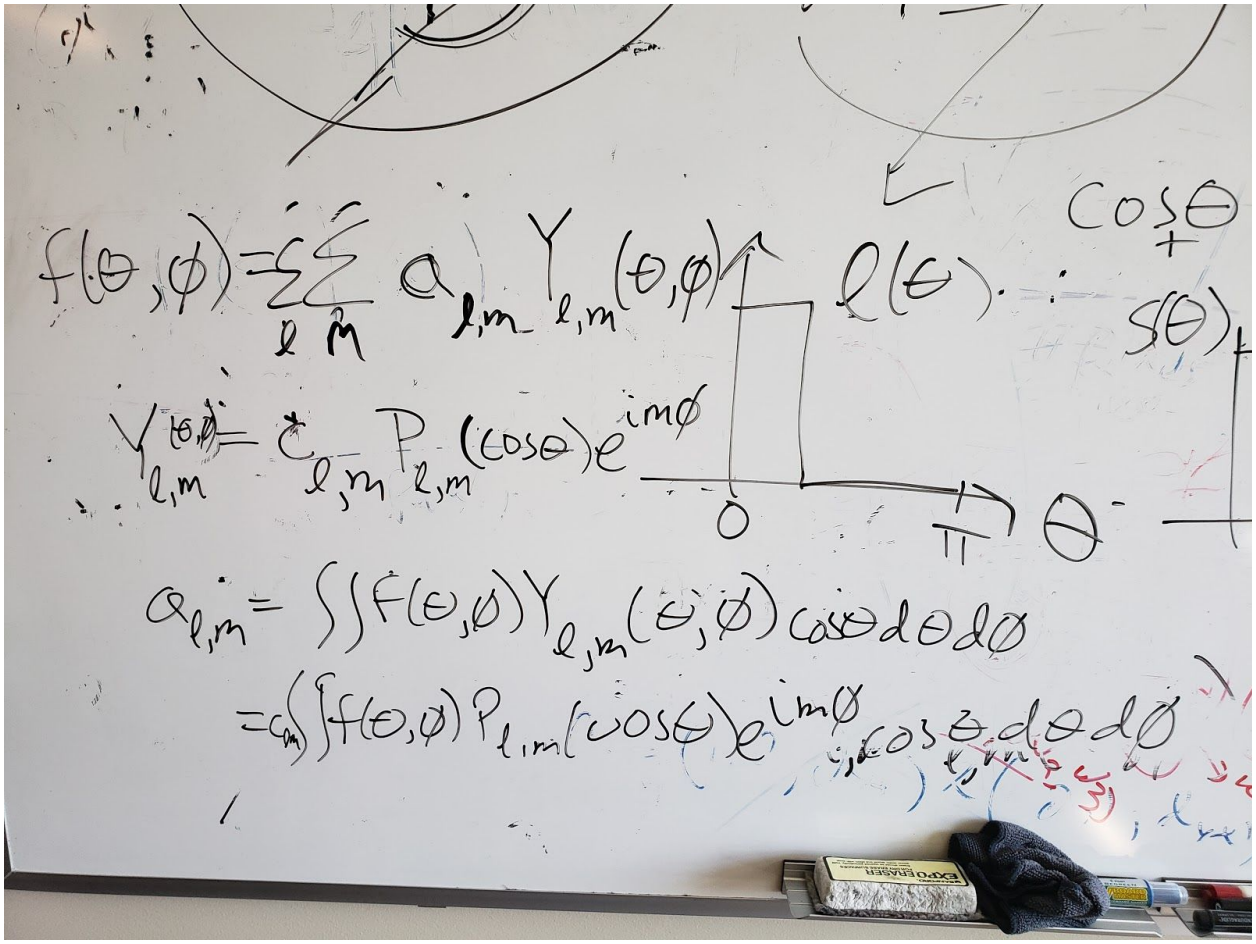
https://math.stackexchange.com/questions/2033332/converting-a-function-to-spherical-hermonic

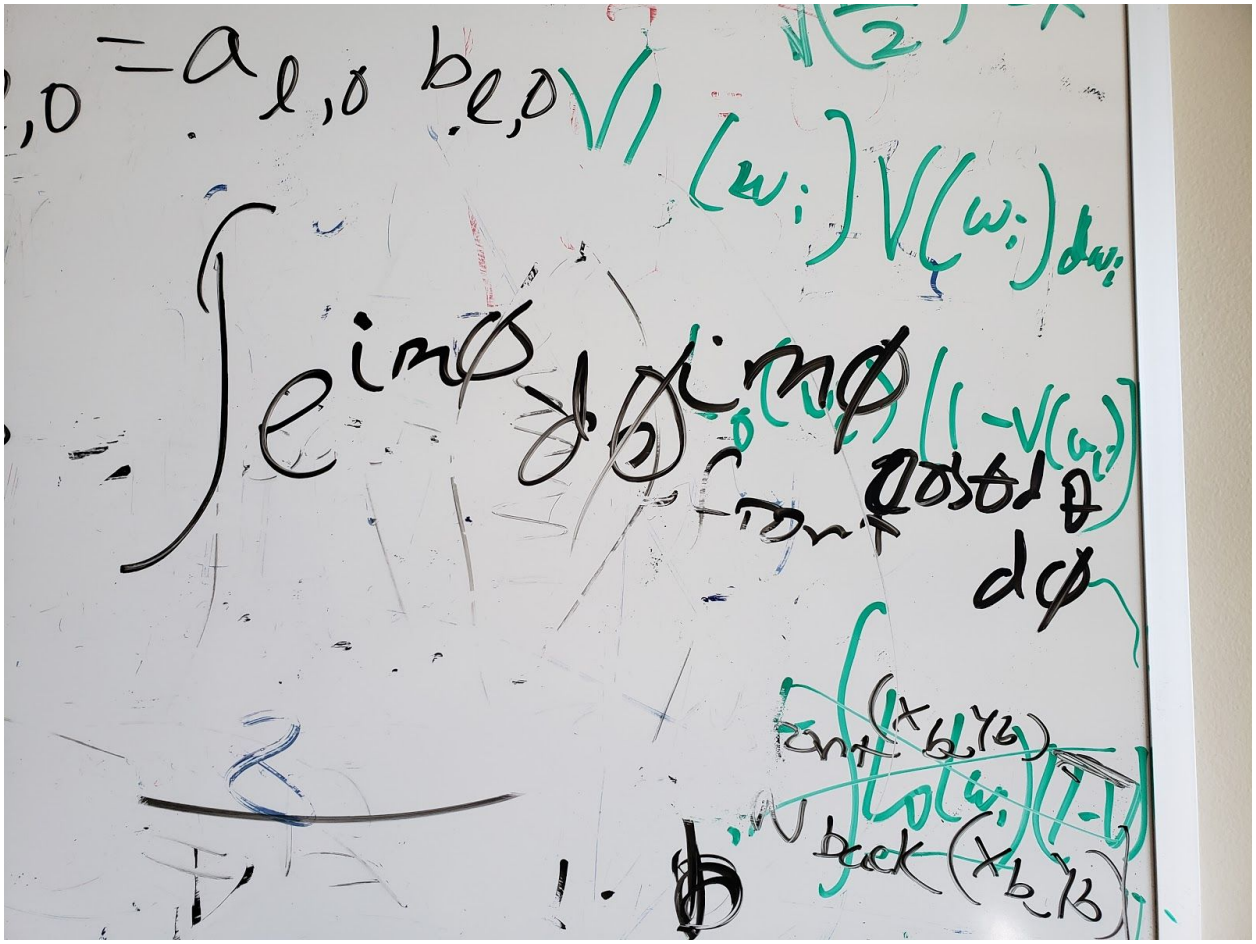
http://mathworld.wolfram.com/SphericalHarmonic.html

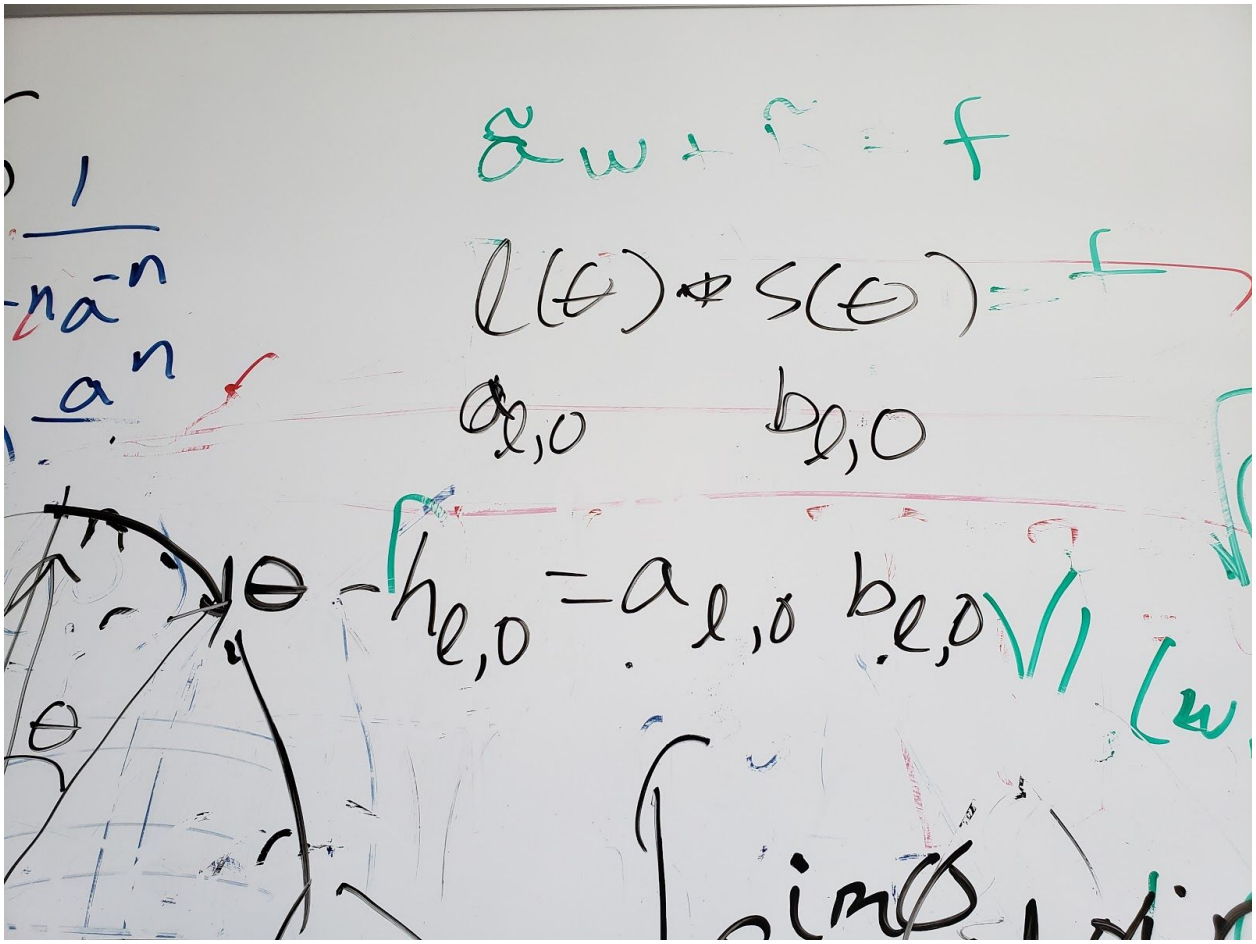
Modeling Area Lighting through Spherical Harmonics

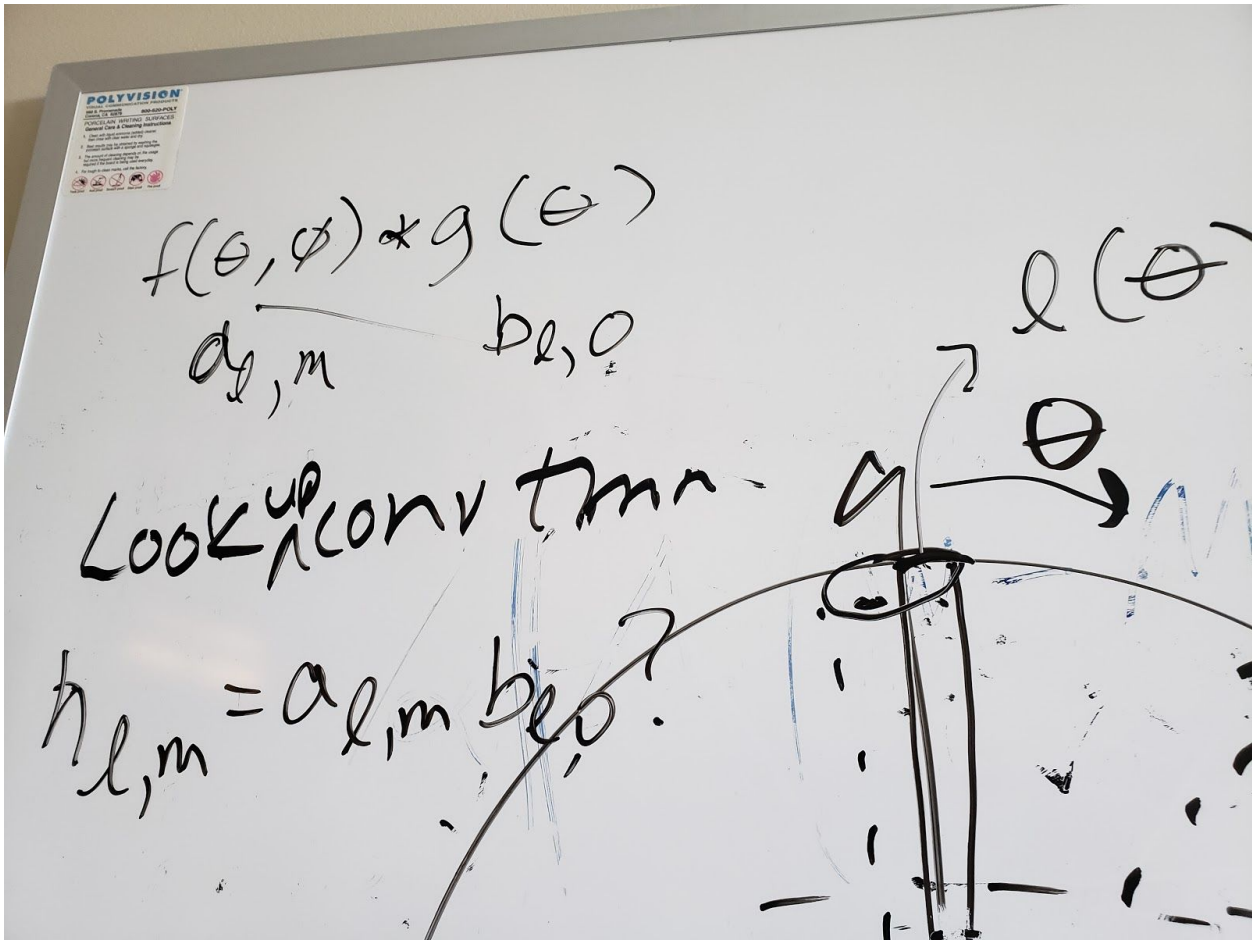


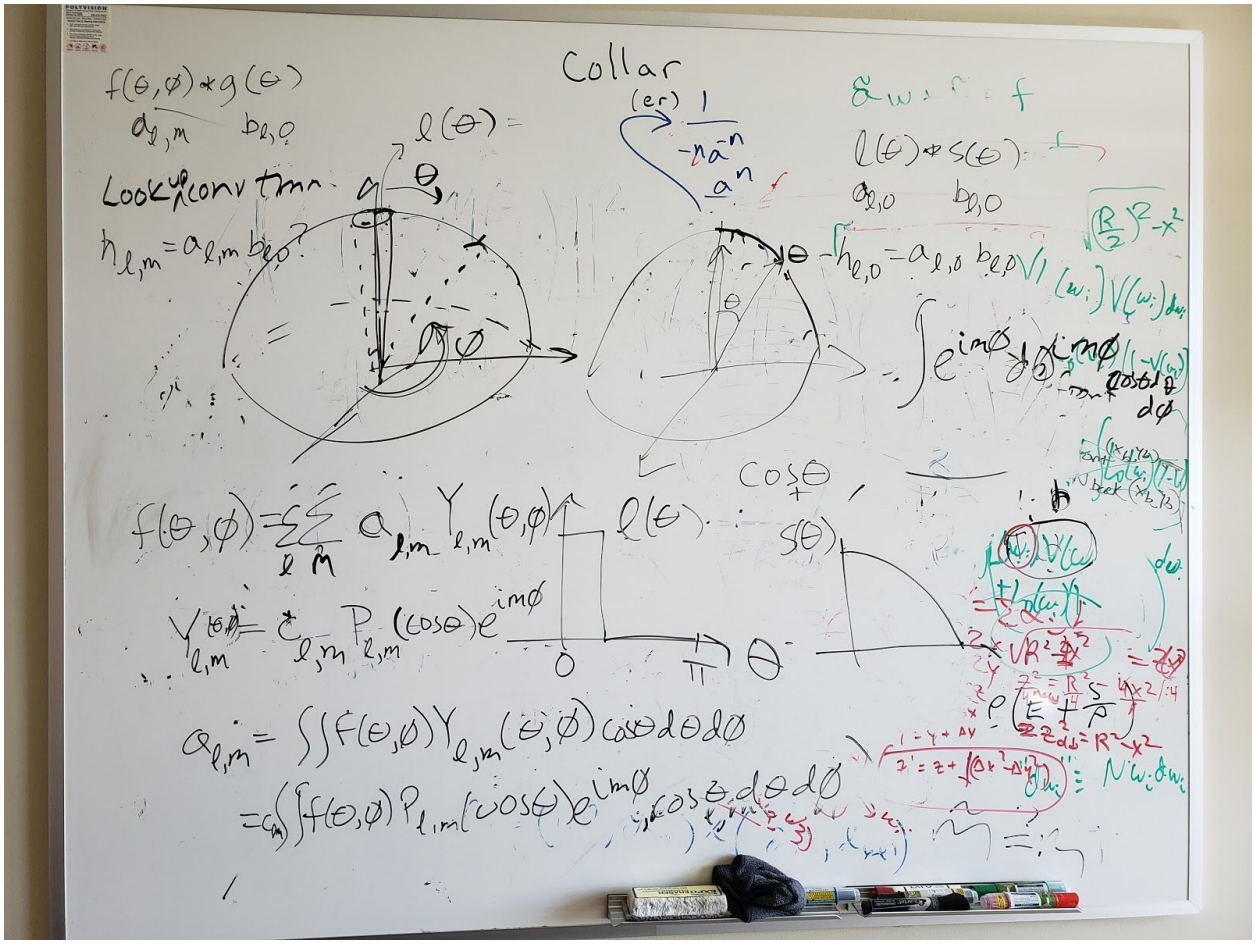

Cos θ
ℓ(θ)
S(θ)
φ
0
π
θ


$f(\theta,\phi) = \sum_l \sum_m a_{l,m} Y_{l,m}(\theta,\phi)$
$Y_{l,m} = C_{l,m} P_{l,m}(\cos\theta) e^{im\phi}$
$a_{l,m} = \int\int f(\theta,\phi) Y_{l,m}(\theta,\phi) \cos\theta \, d\theta \, d\phi$
$= c_m \int\int f(\theta,\phi) P_{l,m}(\cos\theta) e^{im\phi} \cos\theta \, d\theta \, d\phi$
$\cos\theta$
$l(\theta)$
$s(\theta)$
0
π
θ




$f(\theta) * g(\theta) = f$
$a_{\ell,0}$ $b_{\ell,0}$
$h_{\ell,0} = a_{\ell,0} b_{\ell,0}$
θ


POLYVISION
PORCELAIN WRITING SURFACES
General Care & Cleaning Instructions
$f(\theta,\phi) * g(\theta)$
$a_{l,m}$
$b_{l,0}$
Look up convention
$h_{l,m} = a_{l,m} b_{l,0}$?
$l(\theta)$
θ



We are trying to find the convolution between the 2 spheres above

- Convolution: a function derived from two given functions by integration that expresses how the shape of one is modified by the other.
- Sphere (theta from 0 - pi,phi from 0 - 2pi) and n length from center to sphere edge
    - Area lighting based on disk at the top (constrained within theta
- Second Sphere**???** Resulting in some cos(theta)**???**
    - **Actually the normal vector. Normal deviates from vertical by theta**
- Superimposing one on other to account for the effect of all instances of light on a certain position

Function (Θ, Φ) = sum of l (light function???), sum of m (**???**) of $\mathbf{a_{l,m}}$ **(a weighted function) *** $\mathbf{Y_{l,m}(\Theta,\Phi)}$**???**

- $\mathbf{Y_{l,m}(\Theta,\Phi) = C_{l,m}P_{l,m}(\cos(\Theta))e^{im(\Phi)}}$
- **Isolate** $\mathbf{a_{l,m}}$**???**
- $\mathbf{a_{l,m} = \iint f(\Theta,\Phi) * Y_{l,m}(\Theta,\Phi)\cos(\Theta)d(\Theta)d(\Phi)}$
    - $\mathbf{= C_{l,m} \iint f(\Theta,\Phi)\, P_{l,m}(\cos(\Theta))e^{im(\Phi)}\cos(\Theta)d(\Theta)d(\Phi)}$

- If m is nonzero, the sum of $e^{im(\Phi)} = 0$ resulting in nonzero m = 0

    - Integral of euler's function, so net area of sin and cos functions which would be zero
    - This makes Φ phi irrelevant in the equation

Convolution Theorem

- Fourier transform of a convolution is the pointwise product of Fourier transforms
- convolution in one domain (e.g., time domain) equals point-wise multiplication in the other domain (e.g., frequency domain)

If $\mathcal{F}$ denotes the Fourier transform operator, then $\mathcal{F}\{f\}$ and $\mathcal{F}\{g\}$ are the Fourier transforms of f and g, respectively. Then

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

where $\cdot$ denotes point-wise multiplication. It also works the other way around:

$$\mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\}$$

By applying the inverse Fourier transform $\mathcal{F}^{-1}$, we can write:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

and:

$$f \cdot g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} * \mathcal{F}\{g\}\}$$

???

- f(theta,phi) * g(theta)
    - Convolution of f and g
    - f(theta,phi) → $a_{l,m}$
    - g(theta) → $b_{l,0}$
- $H_{l,m} = a_{l,m}\, b_{l,m}$
    - $H_{l,0} = a_{l,0}\, b_{l,0}$

So rather than doing

Find u * v in terms of spherical coordinates

- 1 vector is (theta,phi)
- 1 vector is (alpha,beta)

Spherical Harmonics for Computer Graphics

https://cs.dartmouth.edu/~wjarosz/publications/dissertation/appendixB.pdf

Definition:
A harmonic is a function that satisfies Laplace's equation:

$$\nabla^2 f = 0.$$

Spherical harmonics are an infinite set of harmonic functions defined on the sphere

- Computer graphics focuses on real-valued basis

Representing a direction vector w using standard spherical parameterization

$$\vec{\omega} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta),$$

Real Spherical Harmonic Basis Functions are defined as

$$y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi)P_l^m(\cos\theta) & \text{if } m > 0, \\ K_l^0 P_l^0(\cos\theta) & \text{if } m = 0, \\ \sqrt{2}K_l^m \sin(-m\phi)P_l^{-m}(\cos\theta) & \text{if } m < 0. \end{cases}$$

-

K_l^m are normalization constants

$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}},$$

-

Recurrence Relations of Legendre Polynomials:

$$P_0^0(z) = 1,$$

$$P_m^m(z) = (2m-1)!!(1-z^2)^{m/2},$$

$$P_{m+1}^m(z) = z(2m+1)P_m^m(z),$$

$$P_l^m(z) = \frac{z(2l-1)}{l-m}P_{l-1}^m(z) - \frac{(l+m-1)}{l-m}P_{l-2}^m(z).$$

Basis Function Indexes:

- Order (l)
- Degree (m^2)
- l ∈ N and -l <= m <= l

- 2l+1 basis functions of order l
- Order l determines frequency of basis functions over sphere

Example Spherical Harmonic Expansions:

		Spherical	Cartesian
$l=0$	$y_0^0(\theta,\phi) =$	$\sqrt{\frac{1}{4\pi}}$	$\sqrt{\frac{1}{4\pi}},$
$l=1$	$y_1^{-1}(\theta,\phi) =$	$\sqrt{\frac{3}{4\pi}}\sin\phi\sin\theta$	$\sqrt{\frac{3}{4\pi}}x,$
	$y_1^0(\theta,\phi) =$	$\sqrt{\frac{3}{4\pi}}\cos\theta$	$\sqrt{\frac{3}{4\pi}}z,$
	$y_1^1(\theta,\phi) =$	$\sqrt{\frac{3}{4\pi}}\cos\phi\sin\theta$	$\sqrt{\frac{3}{4\pi}}y,$
$l=2$	$y_2^{-2}(\theta,\phi) =$	$\sqrt{\frac{15}{4\pi}}\sin\phi\cos\phi\sin^2\theta$	$\sqrt{\frac{15}{4\pi}}xy,$
	$y_2^{-1}(\theta,\phi) =$	$\sqrt{\frac{15}{4\pi}}\sin\phi\sin\theta\cos\theta$	$\sqrt{\frac{15}{4\pi}}yz,$
	$y_2^0(\theta,\phi) =$	$\sqrt{\frac{5}{16\pi}}(3\cos^2\theta-1)$	$\sqrt{\frac{5}{16\pi}}(3z^2-1),$
	$y_2^1(\theta,\phi) =$	$\sqrt{\frac{15}{4\pi}}\cos\phi\sin\theta\cos\theta$	$\sqrt{\frac{15}{8\pi}}xz,$
	$y_2^2(\theta,\phi) =$	$\sqrt{\frac{15}{16\pi}}(\cos^2\phi-\sin^2\phi)\sin^2\theta$	$\sqrt{\frac{15}{32\pi}}(x^2-y^2).$

Projection and Expansion

Spherical harmonics define a complete basis over the sphere. Any real-valued spherical function f may be expanded as a linear combination of the basis functions

$$f(\vec{\omega}) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} y_l^m(\vec{\omega})\, f_l^m,$$

f_l^m coefficients are computed projecting f onto each basis function y_l^m

$$f_l^m = \int_{\Omega_{4\pi}} y_l^m(\vec{\omega})\, f(\vec{\omega})\, d\vec{\omega}.$$

- Exact as long as l → infinity
    - Leads to infinite number of coefficients
- Need to limit number of bands to l = n -1
    - Frequencies up to some threshhold

- N^{th} order band-limited approximations of f

$$\tilde{f}(\vec{\omega}) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} y_l^m(\vec{\omega}) f_l^m.$$

- The higher the frequency, more coefficients necessary.

Reformat indexing to use single parameter

- l = l(l+1) + m
- N^{th} order approximation can be reconstructed using n^2 coefficients

Properties

1. Convolution
   - Spherical Harmonics is a Fourier Domain Basis defined over a sphere. Similar Space Convolution Property. If h(z) is a circularly symmetric kernel, then the convolution h * f is equivalent to weighted multiplication in the SH domain
   - $$(h \star f)_l^m = \sqrt{\frac{4\pi}{2l+1}} h_l^0 f_l^m.$$
   - Computation of prefiltered environment maps and irradiance environment maps
2. Orthonormality
   - Spherical Harmonics are orthogonal for different l and different m
   - Inner product of 2 distinct basis functions is 0
   - Inner product of basis function with itself is 1
   - $$\int_{\Omega_{4\pi}} y_i(\vec{\omega}) y_j(\vec{\omega}) d\vec{\omega} = \delta_{ij},$$
     - = Kronecker Delta Function
3. Double Product Integral
   - Integral product of two SH functions can be expanded as
   - $$\int_{\Omega_{4\pi}} \tilde{a}(\vec{\omega}) \tilde{b}(\vec{\omega}) d\vec{\omega} = \int_{\Omega_{4\pi}} \left(\sum_i a_i y_i(\vec{\omega}) \right) \left(\sum_j b_j y_j(\vec{\omega}) \right) d\vec{\omega},$$
     $$= \sum_i \sum_j a_i b_j \underbrace{\int_{\Omega_{4\pi}} y_i(\vec{\omega}) y_j(\vec{\omega}) d\vec{\omega}}_{C_{ij}},$$
   - $C_{i,j}$ are called coupling coefficients, $\rightarrow C_{i,j} = \delta_{i,j}$ which leads to

$$\int_{\Omega_{4\pi}} \bar{a}(\vec{\omega})\,\bar{b}(\vec{\omega})\,d\vec{\omega} = \sum_i \sum_j a_i\,b_j\,C_{ij},$$
$$= \sum_i \sum_j a_i\,b_j\,\delta_{ij},$$
$$= \sum_i a_i\,b_i.$$

-
  - Integrated product of two SH functions is the dot product of their coefficient vectors

4. Triple Product Integral

$$\int_{\Omega_{4\pi}} \tilde{a}(\vec{\omega})\,\tilde{b}(\vec{\omega})\,\tilde{c}(\vec{\omega})\,d\vec{\omega} = \int_{\Omega_{4\pi}} \left(\sum_i a_i\,y_i(\vec{\omega})\right)\left(\sum_j b_j\,y_j(\vec{\omega})\right)\left(\sum_k c_k\,y_k(\vec{\omega})\right) d\vec{\omega},$$
$$= \sum_i \sum_j \sum_k a_i\,b_j\,c_k \int_{\Omega_{4\pi}} y_i(\vec{\omega})\,y_j(\vec{\omega})\,y_k(\vec{\omega})\,d\vec{\omega},$$
$$= \sum_i \sum_j \sum_k a_i\,b_j\,c_k\,C_{ijk},$$

-
  - Tripling coefficients $C_{i,j,k}$

5. Double Product Projection

$$c_i = \int_{\Omega_{4\pi}} y_i(\vec{\omega})\,c(\vec{\omega})\,d\vec{\omega},$$
$$= \int_{\Omega_{4\pi}} y_i(\vec{\omega})\,a(\vec{\omega})\,b(\vec{\omega})\,d\vec{\omega},$$
$$= \int_{\Omega_{4\pi}} y_i(\vec{\omega}) \left(\sum_j a_j\,y_j(\vec{\omega})\right)\left(\sum_k b_k\,y_k(\vec{\omega})\right) d\vec{\omega},$$
$$= \sum_j \sum_k a_j\,b_k \int_{\Omega_{4\pi}} y_i(\vec{\omega})\,y_j(\vec{\omega})\,y_k(\vec{\omega})\,d\vec{\omega},$$
$$= \sum_j \sum_k a_j\,b_k\,C_{ijk}.$$

-
  - i^{th} coefficient of c is a linear combination of, up to, j x k from a and b
    - Weight of terms determined by tripling coefficients (which are independent of a and b
  - Only need to compute tripling coefficients once to compute product projection of many pairs

6. Rotational Invariance

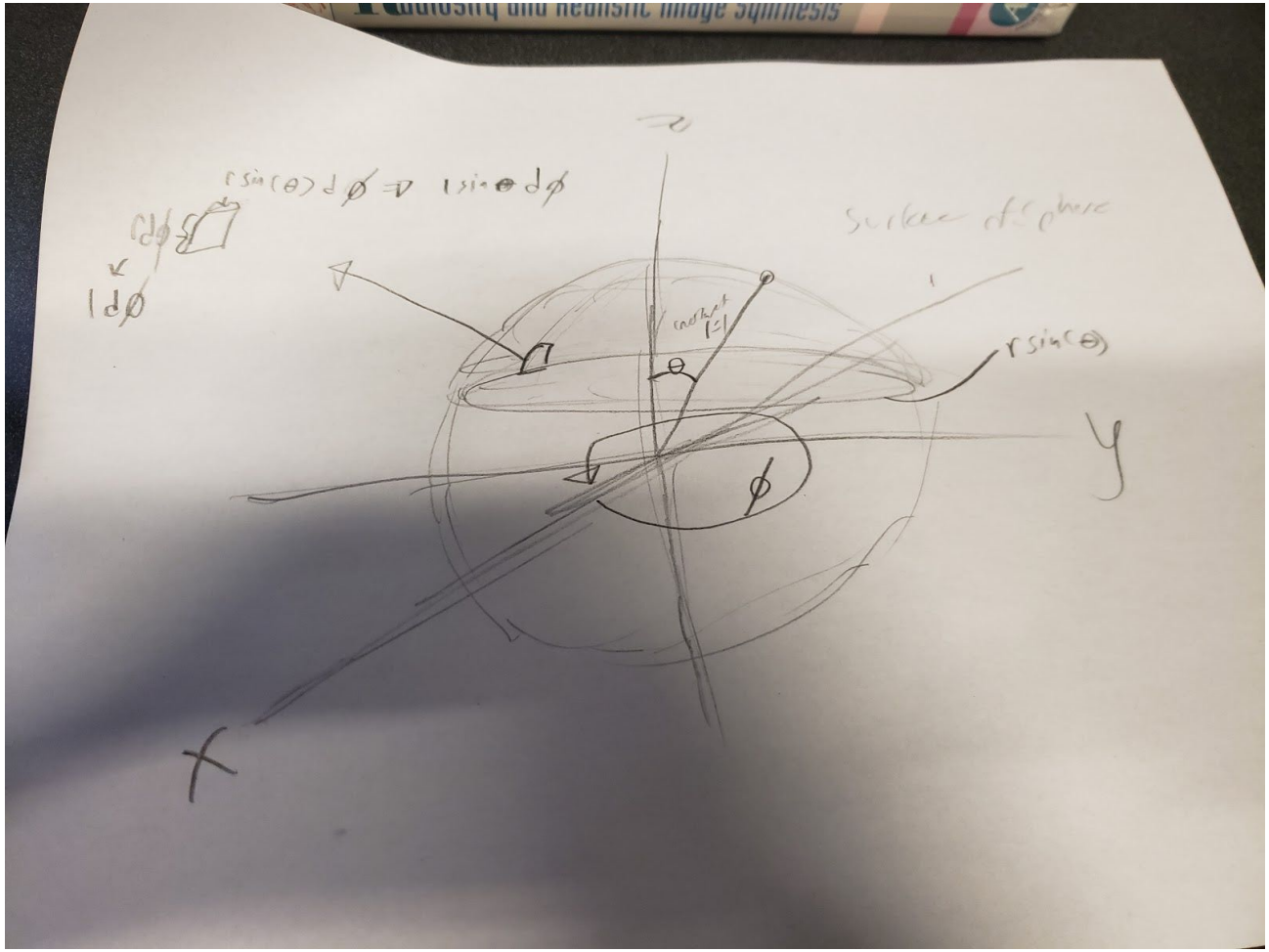
- $g(\vec{\omega}) = f(\mathbf{R}\vec{\omega}),$
- $\tilde{g}(\vec{\omega}) = \tilde{f}(\mathbf{R}\vec{\omega}).$
- Rotate the lookup into unrotated approximation f, or lookup directly into the rotated approximation g
- Spherical harmonic projections produce no aliasing

7. Rotation
    - If we know f, we can computer SH coefficients of rotated function g exactly by applying linear transformation to projection coefficients of f.
    - Ith coefficient of rotated function g is
    - $$g_i = \sum_j f_j \tilde{\mathbf{R}}_{ij}.$$
    - Coefficients in one band of f only influence same band of coefficients in rotated representation
    - $$\tilde{\mathbf{R}} = \left[\begin{array}{c|ccc|ccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \hline 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \cdots \\ \hline 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right].$$
    -
8.

ADJUSTING MODEL FOR NEW RECTANGLE FUNCTION AND NEW COS(OMEGA)
Look up integrals in spherical Coordinates
Double integrals in polar coordinates

Spherical Integrations

- Find proof that equation doesn't need an additional cos or sin
- Our case is specifically for r =1 (the surface of the sphere)
    - Requires only 2 integrals instead of 3


z
surface of sphere
r sin(θ)
θ
φ
y
x

Visualizing the Spherical Harmonics Model

https://stackoverflow.com/questions/39249594/numerical-computation-of-the-spherical-harmonics-expansion
https://en.wikipedia.org/wiki/Table_of_spherical_harmonics
https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.fft.rfftn.html#numpy.fft.rfftn
https://shtools.oca.eu/shtools/tag_python.html
http://reference.wolfram.com/language/ref/SphericalHarmonicY.html

Since

$$Y_1^{-1} = \frac{1}{2}\sqrt{\frac{3}{2\pi}}e^{-i\phi}\sin\theta$$
$$Y_1^{1} = -\frac{1}{2}\sqrt{\frac{3}{2\pi}}e^{i\phi}\sin\theta$$
$$Y_1^{0} = \frac{1}{2}\sqrt{\frac{3}{\pi}}\cos\theta$$

You can write

$$e^{-i\phi}\sin\theta = 2\sqrt{\frac{2\pi}{3}}Y_1^{-1}$$
$$e^{i\phi}\sin\theta = -2\sqrt{\frac{2\pi}{3}}Y_1^{1}$$
$$\cos\theta = 2\sqrt{\frac{\pi}{3}}Y_1^{0}$$

Fast Transform for Spherical Harmonics

http://www.ohiouniversityfaculty.com/mohlenka/research/MOHLEN1999P.pdf

Spherical Harmonics on sphere S^2 in the same way that Fourier Exponential functions are on circle

- Grid $O(N^2)$ on sphere, a direct calculation is $O(N^4)$. Separation of variables reduces to $O(N^3)$
- This research presents $O(N^{5/2}\log(N))$ and $O(N^2(\log N)^2)$
    - Fast application of matrices dense and oscillatory, locally can be represented efficiently in trigonometric series.

Spherical Harmonics can be represented as $P_n^m(\cos(\text{phi})e^{im(\text{phi})}$

- $n >= |m|$.
- n & m are integers
- P_n^m are associated legendre functions
- After normalization, forms a basis for sphere with sin(phi)d(phi)d(theta)

Statement of Problem:

$$\left\{\tilde{P}_n^m(\cos\phi)e^{im\theta}/\sqrt{2\pi}\right\}_{(m,n)\in\mathcal{I}_N}$$

- $(\phi \in (0,\pi),\ \theta \in [0,2\pi))$
- P_n^m associated legendre function of degree n and order m
- normalized so that our set of functions is orthonormal on the sphere.
- $$\mathcal{I}_N = \{(m,n) \in \mathbf{Z}\times\mathbf{Z} : 0 \le n < N, -n \le m \le n\}.$$
- $$f(\phi,\theta) = \sum_{\mathcal{I}_N} \alpha_n^m \tilde{P}_n^m(\cos\phi)e^{im\theta}/\sqrt{2\pi}.$$
- Evaluated on grid phi x theta with @N equispaced points in theta and N gaussian nodes in cos(theta)
- $$\left\{(\phi_j,\theta_k) = \left(\cos^{-1}(g_j^N), 2\pi\frac{k}{2N}\right) : j,k \in \mathbf{Z}; 0 \le j < N, 0 \le k < 2N\right\}.$$
- Evaluation: Need to compute $O(N^2)$ f(phi, theta) output values from $O(N^2)$ input values a_m^n
- Expansion: given $O(N^2)$ sample values of $f(\phi_j,\theta_k)$ and wish to compute $O(N^2)$ output values a_n^m

Result Summary:
Fast Transform reduces to a set of 2N problems

$$M_{n,j}^m = \left(\sqrt{\sin\phi_j}\tilde{P}_n^m(\cos\phi_j)\right)_{n,j}.$$

- Finding a fast application for the matrices
- These matrices are dense and oscillatory

Modeled using quasi-classical (WKB) frequency estimates.

- $$M_{n,j}^m \approx \left(\exp\left(\int^{\phi_j} i\sqrt{(n+1/2)^2 - \frac{m^2-1/4}{\sin^2 t}}dt\right)\right)_{n,j} = (\exp(in\phi_j\Phi^m(n,\phi_j)))_{n,j}. \quad (6)$$

One Dimensional Algorithm:

- Fixing m, if for each n $\Phi^m(n,\phi)$ is a constant of phi, matrix application becomes evaluation of Fourier series achieved through FFT
    - Can partition phi-space into small intervals where $\Phi^m(n,\phi)$ is constant. Find suitable partition for each, yet total of intervals used is manageable, can represent entire matrix in terms of $O(N^{3/2}\log(N))$ exponentials, apply it in this same $O(N^{3/2}\log(N))$ time.

Two Dimensional Algorithm:

- If for each phi, $\Phi^m(n,\phi)$ is a constant function of n, this matrix application becomes an expansion into a fourier series, which can be done using FFT. we can partition in both coordinates at the same time, representing each rectangle by the interactions of small number of exponentials in n with a small number of exponentials in phi

What are Spherical Harmonics

- Spherical harmonics are from generalizing Fourier Series to the next dimension
    - Built from associated legendre functions
    - Eigenfunctions of the Spherical Laplacian
    -

Python Spherical Harmonic Tools

https://shtools.oca.eu/shtools/

- Python spherical harmonic tool
- https://github.com/SHTOOLS/SHTOOLS/issues/62
- https://github.com/SHTOOLS/SHTOOLS/issues/147
- https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyspharm
- https://shtools.oca.eu/shtools/pages/mydoc/notebooks/Introduction-1.html
-

https://www.cs.dartmouth.edu/~geelong/sphere/

- C spherical harmonic tool
    - Can use c in python if compiled correctly

Estimating Human Shape and Pose from a Single Image

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5459300

- Computing shape and pose parameters of 3D human body model directly from monocular image cues
    a. Given user-supplied estimate of subject's height and a few clicked point on body, can estimate initial 3D articulated body pose and shape
    b. Using initial guess, can generate a tri-map of inside/outside/on boundary of human for graph cuts.
    c. Low-dimensional linear model of human shape in which variations due to height are concentrated along single dimension, enabling height-constrained estimation of body shape
    d. Formulate problem of parametric human shape from shading.

Introduction:

- Single image silhouette is generally insufficient to constrain 3D body shape
    - Propose use of additional monocular cues including smooth shading
    - Given initial guess of body pose, we optimize pose, shape, and reflectance properties of 3D body model
- With monocular image, no background can be assumed.
    - Outline of body provides strong constraint on body shape
    - Initial pose, obtained by manual clicking on a few image locations of major joints, can create initial foreground region to derive tri-map for GrabCut segmentation
- Parametric body representation is based on SCAPE model.
    - Body pose is ambiguous, body limb lengths unknown.

    - Constrain height of subject during optimization
    - Previous methods use linear shape deformation bases with principal component analysis
- Body shape from shading
    - Estimating body shape parameters, pose of body, its reflectance properties and lighting best explains shading and shadows observed in image
    - Assume single point light source.
    - Skin has significant specular component, approximate with Blinn-Phong.
    - Exploiting shading requires accurate surface normals
    - Body shape, pose, light direction, skin reflectance result in synthesized image of person with observed image

Related Work

- 3D body pose from monocular images
    - Automated methods extracting 2D human pose from monocular image use to initialize our method
    - Recover set of body poses consistent with clicked 2D points corresponding to joins
        - Assumes orthographic camera and only relative depths.
- Body shape from images
    - Body shape → pose independent representation that characterizes the fixed skeletal structure and distribution of soft tissue (muscle and fat)
    1. Non-parametric models (visual hulls, point clouds, and voxel representations)
    2. Part-based models using generic shape primitives (cylinders or cones), superquadrics, or “metaballs”
    3. Humanoid models with set of pre-specified parameters (limb length, etc)
    4. Data-Driven models where human body shape variation is learned from a training set of 3D body shapes
- Machine vision algorithms
    - Utilize structured light, photometric stereo, or multiple calibrated camera views in controlled setting
    - Image evidence decreases, needed human-specific models increases
    -
    - Estimate limited aspects of body shape (scaling parameter or join locations) but fails to capture range of natural body shapes
- Data-driven methods encoding statistics of human body shape
    - Learned deformable body model estimating shape from multiple photos (controlled environment, predefined pose)
    - For estimating arbitrary pose, desirable for a body model that factors in shape due to pose and identity
    - SCAPE model derived from laser scans and realistic articulated and nonrigid pose-dependent deformations, as well as shape variations between individuals.
- Challenge of Single Monocular Image

    - Sigal et al. train a mixture of experts model to predict 3D body pose and shape directly from various 2D shape features computed from image silhouette
    - From internet. Requires manual segmentation, and doesn't accurate estimate pose.
    - While silhouettes constrain surface normals, non-rigid deformation articulation and self occlusion make silhouette boundary
- Body Shape From Shading
    - Recovering shape of unknown surfaces
    - Computer surface normals at each point on the boyd mesh.
    - Formulate & optimize robust shape from shading
    - Go beyond to deal with a learned shape deformation model and articulation
- Humaning body and shading usually calibrated lab environments
    - Theobalt recovers human body and detailed reflectance properties, but needs several cameras with calibrated environment/lighting
    - Balan using multiple known poses to recover albedo, but doesn't estimate shape
    - La Gorce used accurate hand-shape model and shading for monocular tracking.

Body Model and Fitting

- SCAPE
    - Deformable, triangulated mesh model of human body
        - accounts different body shapes, different poses, and non-rigid deformations due to articulation
        - Mesh m = 12,500 vertices
- Pose is is parametrized by a set of rigid body part rotation (θ)
    - Changes in body shape are captured by shape deformation gradients between a reference mesh and a new mesh
    - Low-dim, statistical model of body shape is learned using principal component analysis (PCA)
    - 2 gender-specific models with >1000 men/women
- Goal is to estimate shape parameters β (vector of linear coefficients that characterizes given shape) and pose parameters θ that explain image
    - Model parameters used to produce 3D mesh Y(β,θ) projected onto image plane to obtain silhouettes, edges, or shaded images
        - Body parameters $\boldsymbol{\theta}_B$ = [β,θ]
        - Standard distance function for silhouettes and edges
        - New shading term
        - Objective function minimized using gradient-free direct search simplex method

3.1 Pose Initialization

- 3D pose initialization
    - Camera coordinate using clicked 2D points corresponding to joins
    - Perspective method requires estimate of focal length extracted from EXIF metadata
    - An approximate focal length produces better than orthographic assumption

- Perspective projection requirements
    - Position the root join in 3D
    - First, limb most parallel to image plane identified as one that minimizes ratio between image length and actual length
        - If limb parallel to image plane,Depth is uniquely determined using ratio of similar triangles,
        - If not, use foreshortening factor similar to scale parameter.
- Do not explicitly require limb length as input
    - Can predict from a database
    - Use database to build a height constrained shape spaced
    - Deforms mesh to match mean person of specified gender and height
    - Extract limb lengths from linear combination of vertices
- Extend previous methods to initialize head pose
    - Solving for neck rotation that minimizes distance between user-clicked 2D face point sand 3D vertex position on mesh projected into image

Region-Based Segmentation

- Given initial mesh, render silhouette into image
- Initial guess for foreground segmenetation
    - Tri-map defining each pixel as foreground, background, or uncertain
    - Eroding and dilating initial region by 5%. Then use grabcut

Internal Edges:

- Silhouettes do not provide pose constraints in regions where one body part occludes another
    - Combining edge information with silhouettes
    - SCAPE mode, edges provide better fit to image evidence than previous models
    - Detector image edges using standard edge detector
        - Thresholded distance transform to define edge cost map normalized to [0,1].
        - Model edges correspond to visible vertex edges for which there is sign change in dot product between triangle normals and ray from camera to midpoint of edge
        - Trapezoid rule to evaluate line integral of all model edges over edge cost image

Attribute Preserving Shape Spaces:

- 3d pose and shape from a single images means that we must constrain the search space as much as possible
    - Wrong body shape can be compensated for by a change in pose (equal explanation value)
    - Additional information (e.g. height) reduces ambiguity

    - SCAPE eigen-shape representation does not provide direct control parameters corresponding to intuitive attributes (gender, height, weight, etc)
        - If derived as functions of linear coefficients, can be included as constraints during body shape estimation.
    - Direct approach of constructing a rotation of original eigen-shape space that height variation is remove from all but one bases.
        - Allows optimization over body shapes without varying height
- Previously, computed direction in shape coefficient space such that any movement along this axis manipulates certain attribute the most and other minimally
    - Axis not optimized for an fails to preserve attribute value along orthogonal directions
- Linear mapping from fixed set of attributes to shape parameters, could optimize body shape using parameters instead of PCA coefficients
    - Preserving attribute through keeping it fixed.
- Explicitly searching for attribute-preserving directions in the eigen-space and reorients the bases along these directions
    - Focusing on height, can be applied to other attributes (volume, geodesic distances, etc)
    - Body height H(β) measured by reconstructing mesh in predefined standing pose
    - We seek a new orthonormal basis such that none of its bases account for height except one, which becomes the height axis
    - G should also preserve the representational power of the original bases:
        - Subspace spanned by first h bases is same after change of bases (absent height axis)

1. Selecting candidate basis e_k for height ais as one that maximizes absolute correlations between height and shape coefficients of training examples
2. Iterate over remaining bases, rotating current plane to make e_j height preserving.
3. Rotation matrix is used to update, at iteration j, the orthonormal basis G_j

$$\text{basis } \mathbf{G}_j = \mathbf{G}_{j-1}\mathbf{R}_{jk}\left(\arg\min_{\varphi}\left(H(\vec{0}_n) - H(\mathbf{G}_{j-1}\mathbf{R}_{jk}(\varphi)\vec{e}_j)\right)^2\right),$$

where $\mathbf{R}_{jk}(\varphi)$ is a $n \times n$ rotation in the $(\vec{e}_j, \vec{e}_k)$ plane:

$$\mathbf{R}_{jk}(\varphi) = \begin{array}{c} \\ j \\ k \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} j & & k & \end{array} \\ \begin{pmatrix} \mathbf{I} & & & 0 \\ & \cos(\varphi) & -\sin(\varphi) & \\ & \mathbf{I} & & \\ & \sin(\varphi) & \cos(\varphi) & \\ 0 & & & \mathbf{I} \end{pmatrix} \end{array}.$$

- Body shape in new shape space expressed as
- $d = (UG_n)\beta + \mu$, where $\beta = (G_n)^{-1}\beta$

Body Shape from Shading

- Approximate body's reflectance using Blinn-Phong model
  - Also diffuse and specular components
  - Single light source and ambient illumination
- Let $X(\Theta_B) = [x_1, x_2, ..., x_m]$ be positions of the m vertices of a body mesh, $N(\Theta_B) = [n_1, n_2, \ldots, n_m]$ be associated unit length normals.
  - X and N are functions of pose and shape parameters
  - Formulate a parametric shape from shading problem
- Let $a = [a_1, a_2, \ldots, a_m]$ be **albedo** of each vertex and $s = [s_1, s_2, \ldots, s_m]$ **specularity** of each vertex
  - Shading value of each surface point i is:

$$\hat{r}_i = b + a_i(\vec{\ell}_i \cdot \vec{n}_i)l + s_i(\vec{h}_i \cdot \vec{n}_i)^{\alpha} l \qquad (1)$$

    - l_i is the direction from vertex x_i toward light source
    - h_i is the halfway vector between l_i from vertex i toward camera
    - b is ambient illumination
    - l is light intensity,
    - **α** is specular exponent

- For distant directional light source (outdoor scene)
  - L is constant for every vertex
- For point light source (indoor scene)
  - Use quadratic attenuation function for light intensity with distance from source

- Optimization
  - Body is placed at origin of spherical coordinate system and light position is parametrized as Θ_L = **[γ, φ, z]** w.r.t body center
    - **Γ** is azimuth
    - **φ** is elevation and respectively
    - **z** is distance between light source and body
  - Parameter l_i, h_i, and l in Eq. 1 depend on Θ_L
- Θ_R = Reflectance Parameters = [a, s, b, **α**]

- r_i is linearly interpolated intensity in input image where vertex i is projected, our goal is to minimize energy function $E_{sh}(\Theta_B, \Theta_R, \Theta_L) \propto$

$$\sum_{i \in visible} \Big\{ \rho_{\eta_1}(\hat{r}_i(\Theta_B, \Theta_R, \Theta_L) - r_i) \qquad (2)$$

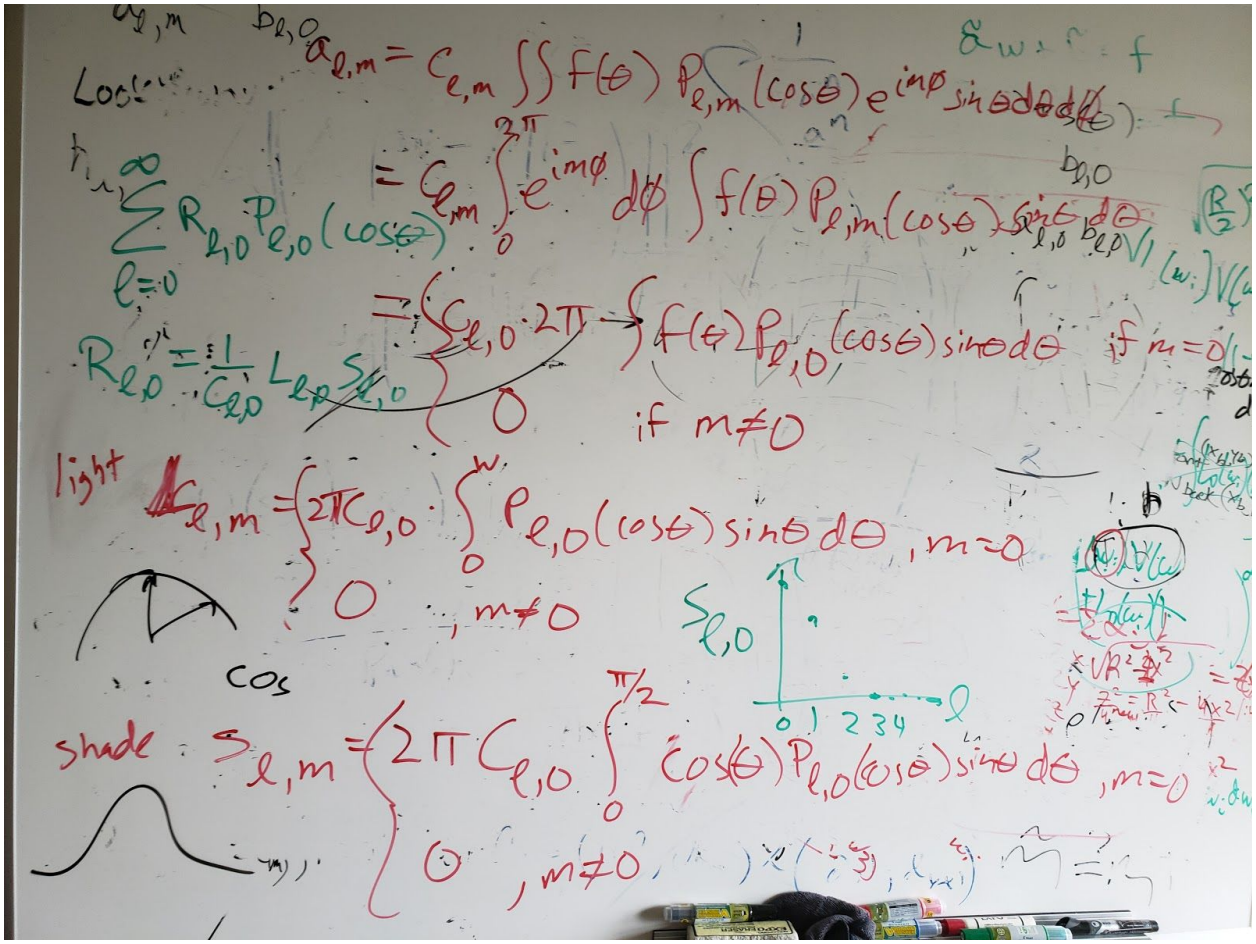
$$+\lambda_1 \sum_{j \in \mathcal{N}(i)} \frac{\rho_{\eta_2}(a_j - a_i)}{d_{j,i}} + \lambda_2 \sum_{j \in \mathcal{N}(i)} \frac{\rho_{\eta_3}(s_j - s_i)}{d_{j,i}} \Big\}$$

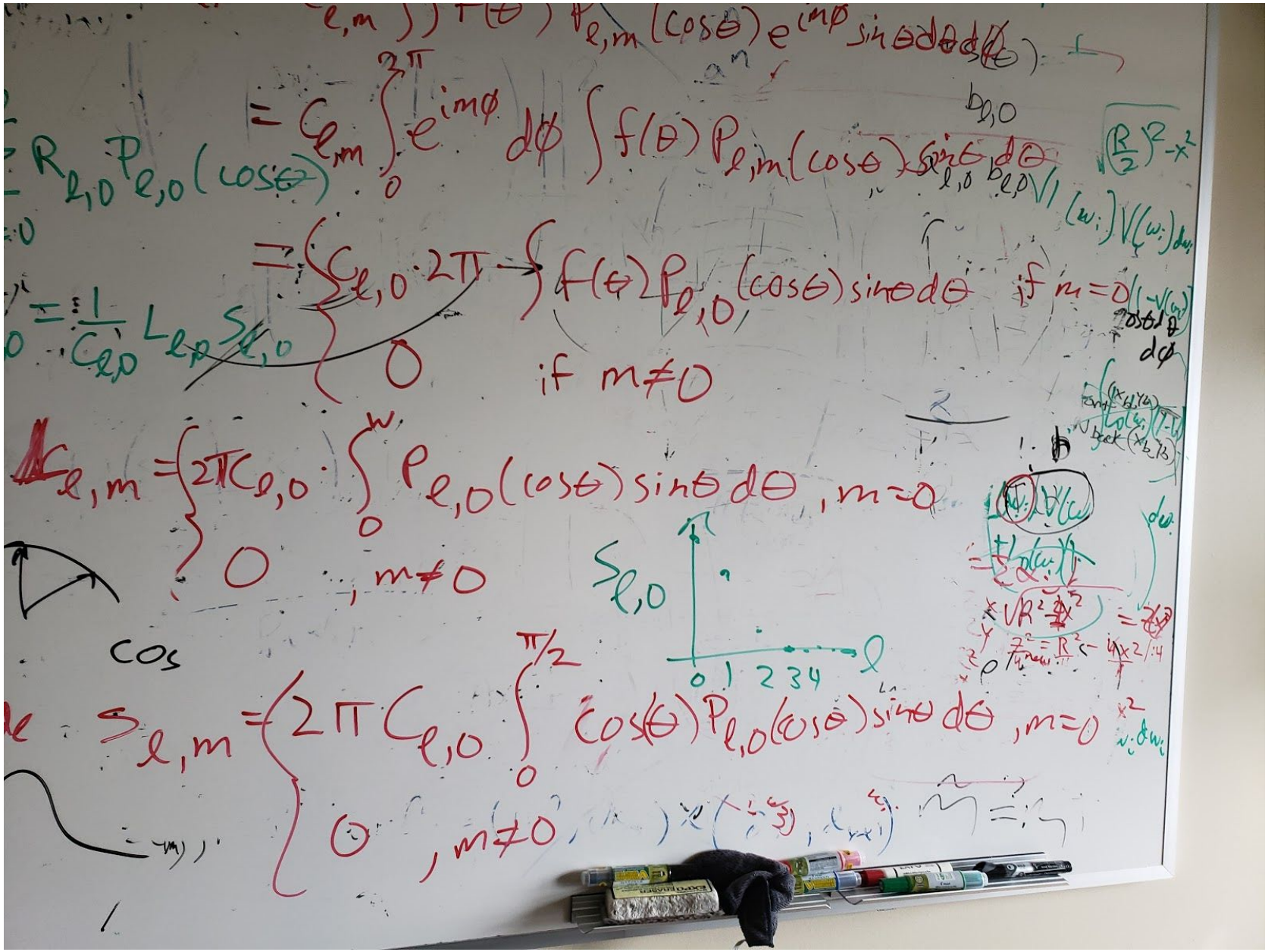
- N(i) = vertices connected to vertex i, $d_{j,i}$ is $|x_j - x_i|$
- $p_n(x) = x^2 / (n^2 + x^2)$ is robust error function for outliers
- First term penalize differences between measured intensity in observed image, r_i, and the predicted brightness of corresponding model vertices, $r_i(.)$
- Second term ensures neighboring vertices have similar albedo
- Robust formulation provides piecewise smooth prior on albedo that allows spatial variation (for clothing, hair, variation in skin color, etc)
- Third term provides a piecewise smooth prior over specularity
- $\boldsymbol{\lambda_1}$ and $\boldsymbol{\lambda_2}$ weight relative faithfulness to observed data and spatial smoothness assumptions

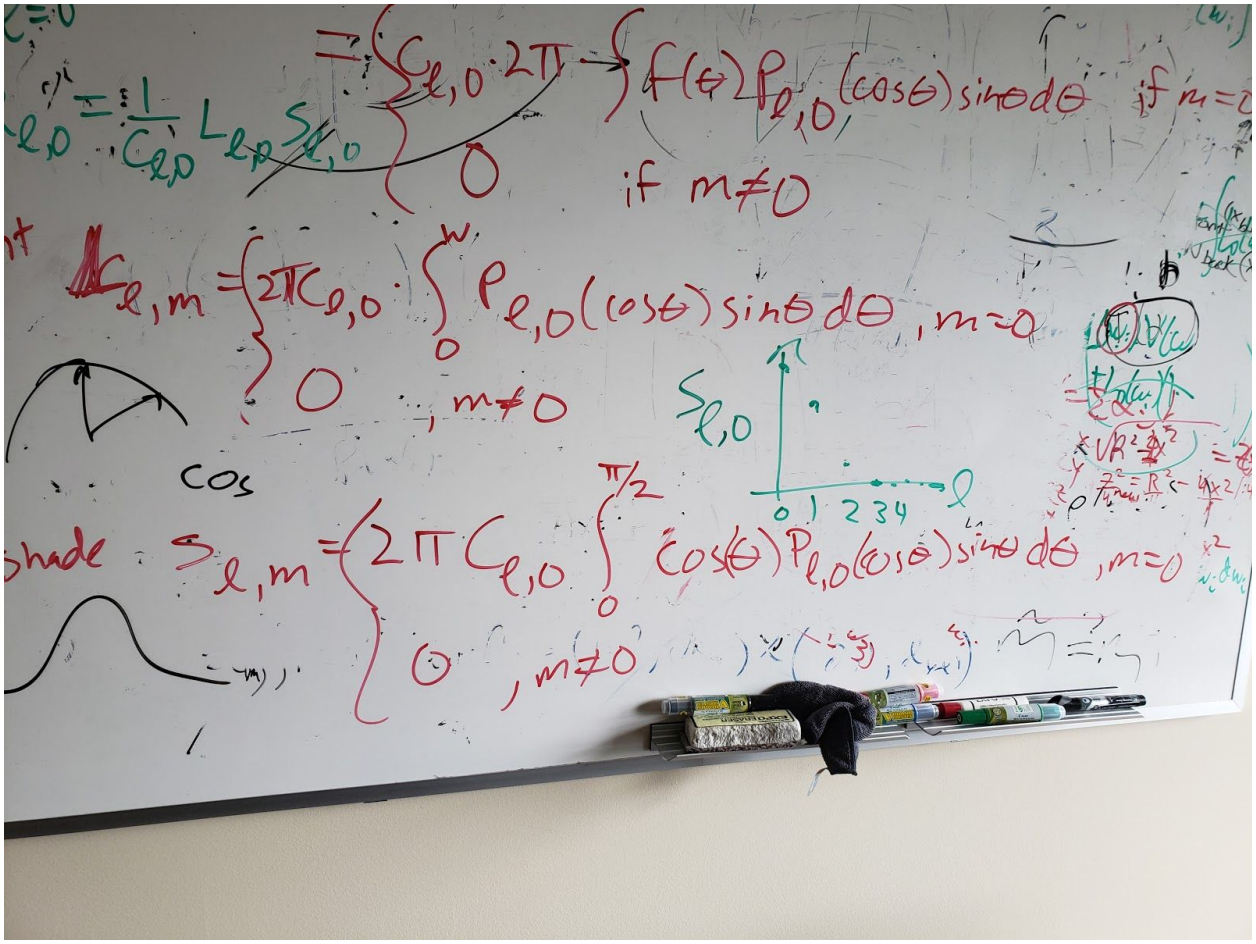
- User coarsely initializes Θ_L and then energy function is minimized in alternating fashion.
    1. Θ_L is optimized given fixed Θ_B and Θ_R
        a. In first iteration, Θ_B is initial guess of pose and shape
        b. Albedo and specularity in Θ_R considered uniform
    2. Optimize Θ_R with fixed Θ_L and Θ_B
        a. No closed form solution possible so we minimize using gradient descent.
    3. Fix Θ_L, Θ_R and optimize Θ_B but here optimization is more difficult
        a. Changing Θ_B affects brightness
        b. Gradient-free simplex method is employed to solve step 3.
        c. Alternate between three steps until convergence criterion is met.
    - Vary $\boldsymbol{\lambda}$ during optimization (larger → smaller)
        - shape is forced to change in order to make predicted brightness match image observation
        - Initial pose needs to be accurate, but illumination direction is insensitive to initialization

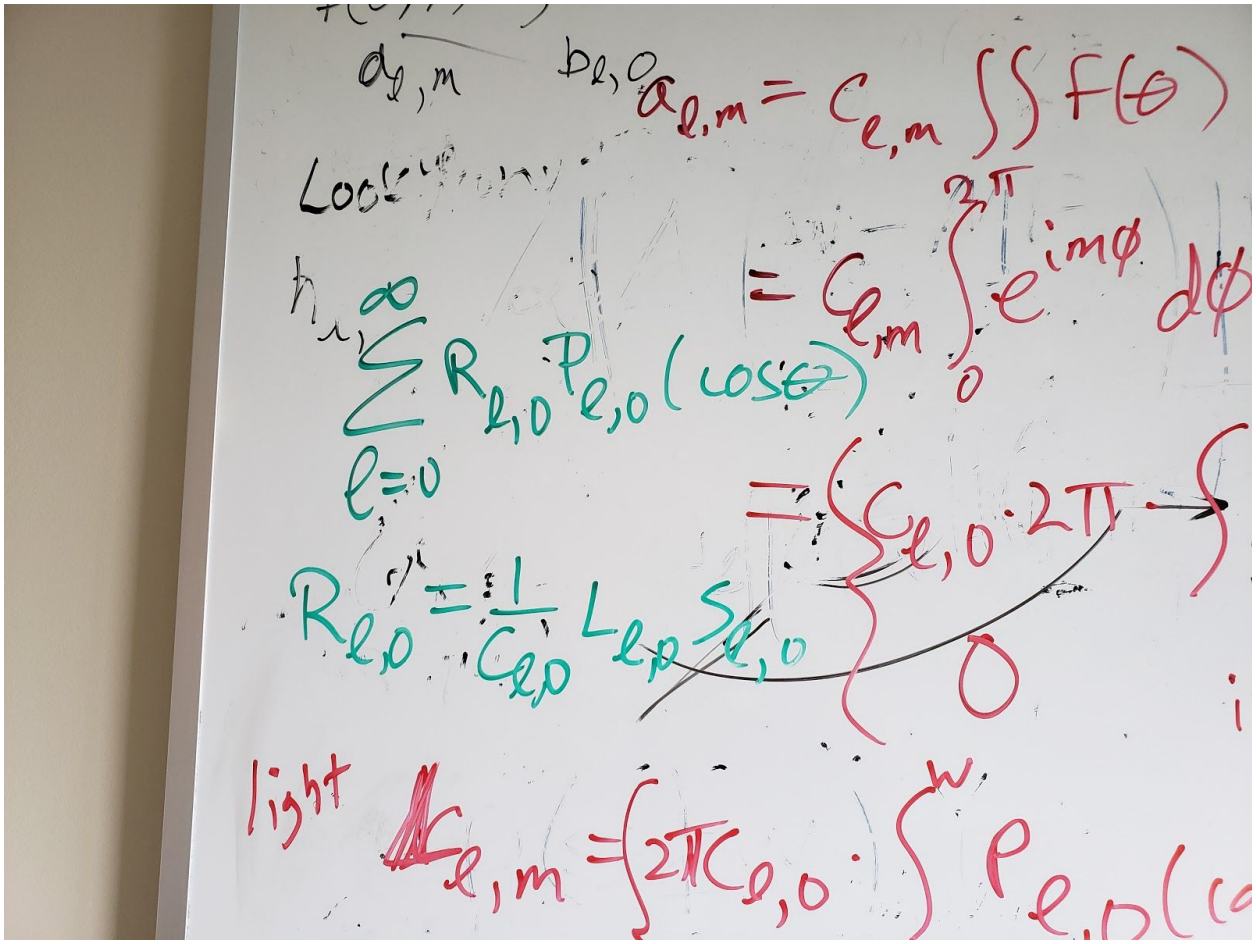
Summarizing Body Shape from Shading

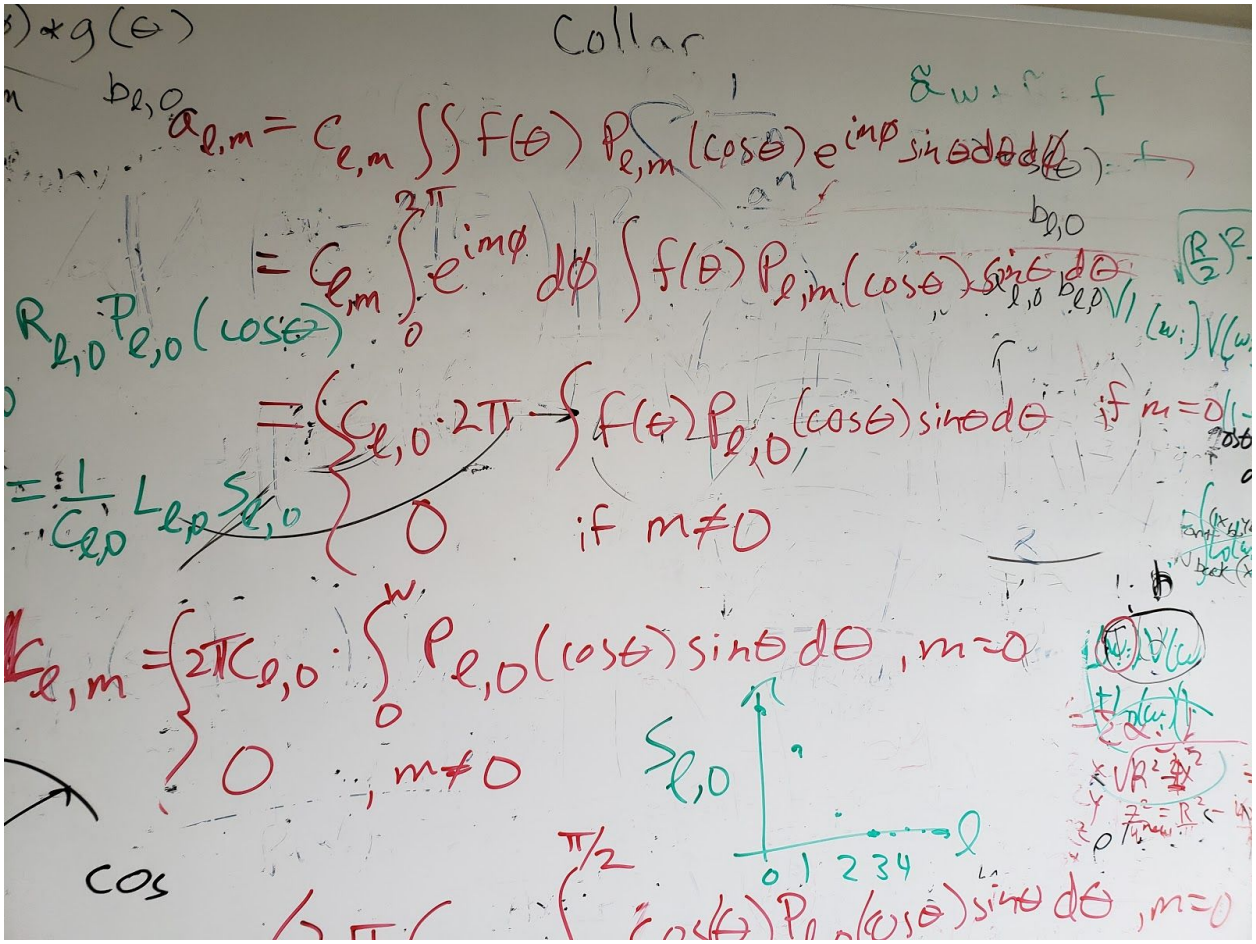
Result

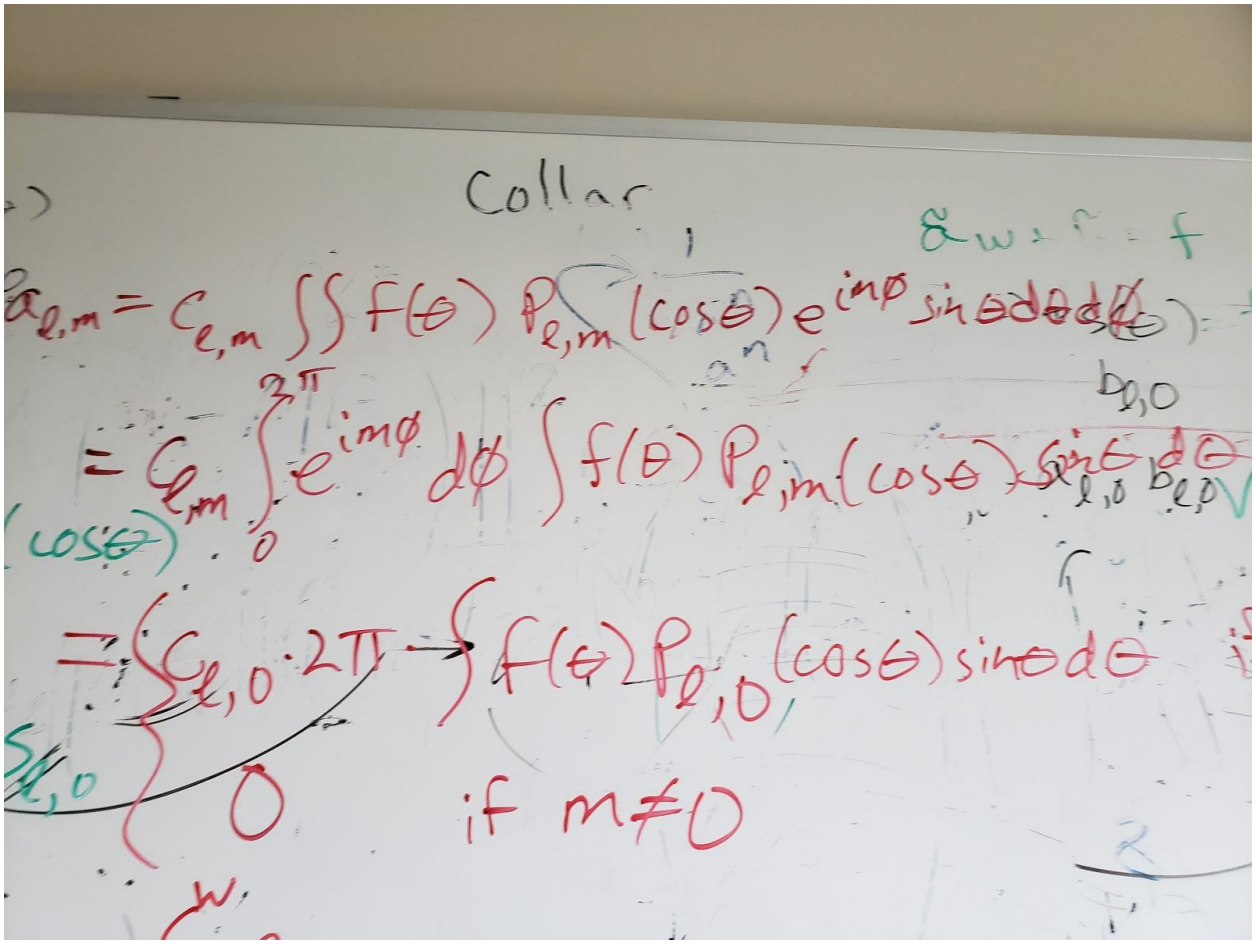

$a_{\ell,m} = c_{\ell,m} \iint f(\theta) P_{\ell,m}(\cos\theta) e^{im\phi} \sin\theta \, d\theta \, d\phi$
$= c_{\ell,m} \int_0^{2\pi} e^{im\phi} \, d\phi \int f(\theta) P_{\ell,m}(\cos\theta) \sin\theta \, d\theta$
$\sum_{\ell=0}^{\infty} R_{\ell,0} P_{\ell,0}(\cos\theta)$
$R_{\ell,0} = \frac{1}{C_{\ell,0}} L_{\ell,0} S_{\ell,0}$
$= \begin{cases} c_{\ell,0} \cdot 2\pi \int f(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta & \text{if } m=0 \\ 0 & \text{if } m \neq 0 \end{cases}$
light
$c_{\ell,m} = \begin{cases} 2\pi c_{\ell,0} \int_0^{\pi} P_{\ell,0}(\cos\theta) \sin\theta \, d\theta & , m=0 \\ 0 & , m \neq 0 \end{cases}$
cos
$S_{\ell,0}$
0 1 2 3 4 ℓ
shade
$S_{\ell,m} = \begin{cases} 2\pi C_{\ell,0} \int_0^{\pi/2} \cos(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta & , m=0 \\ 0 & , m \neq 0 \end{cases}$


$= C_{\ell,m} \int_0^{2\pi} e^{im\phi} d\phi \int f(\theta) P_{\ell,m}(\cos\theta) \sin\theta \, d\theta$
$= \begin{cases} C_{\ell,0} \cdot 2\pi \int f(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases}$
$C_{\ell,m} = \begin{cases} 2\pi C_{\ell,0} \int_0^{\pi} P_{\ell,0}(\cos\theta) \sin\theta \, d\theta, & m = 0 \\ 0, & m \neq 0 \end{cases}$
$S_{\ell,0}$
0 1 2 3 4 ℓ
cos
$S_{\ell,m} = \begin{cases} 2\pi C_{\ell,0} \int_0^{\pi/2} \cos(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta, & m = 0 \\ 0, & m \neq 0 \end{cases}$


$C_{\ell,0} = \frac{1}{L_{\ell,0}} S_{\ell,0}$
$= \begin{cases} C_{\ell,0} \cdot 2\pi \int_0^{\pi} f(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases}$
$C_{\ell,m} = \begin{cases} 2\pi C_{\ell,0} \int_0^{\pi} P_{\ell,0}(\cos\theta) \sin\theta \, d\theta & , m = 0 \\ 0 & , m \neq 0 \end{cases}$
$S_{\ell,0}$
0 1 2 3 4 ℓ
cos
shade
$S_{\ell,m} = \begin{cases} 2\pi C_{\ell,0} \int_0^{\pi/2} \cos(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta & , m = 0 \\ 0 & , m \neq 0 \end{cases}$


$a_{\ell,m} = c_{\ell,m} \iint f(\theta)$
$= c_{\ell,m} \int_0^{2\pi} e^{im\phi} d\phi$
$= c_{\ell,0} \cdot 2\pi$
$\sum_{\ell=0}^{\infty} R_{\ell,0} P_{\ell,0}(\cos\theta)$
$R_{\ell,0} = \frac{1}{C_{\ell,0}} L_{\ell,0} S_{\ell,0}$
0
light
$c_{\ell,m} = 2\pi c_{\ell,0} \cdot \int$


Collar
$a_{\ell,m} = C_{\ell,m} \int\int f(\theta) P_{\ell,m}(\cos\theta) e^{im\phi} \sin\theta \, d\theta \, d\phi$
$= C_{\ell,m} \int_0^{2\pi} e^{im\phi} \, d\phi \int f(\theta) P_{\ell,m}(\cos\theta) \sin\theta \, d\theta$
$R_{\ell,0} P_{\ell,0}(\cos\theta)$
$= \begin{cases} C_{\ell,0} \cdot 2\pi \int f(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases}$
$= \frac{1}{C_{\ell,0}} L_{\ell,0} S_{\ell,0}$
$C_{\ell,m} = \begin{cases} 2\pi C_{\ell,0} \cdot \int_0^{\pi} P_{\ell,0}(\cos\theta) \sin\theta \, d\theta, & m = 0 \\ 0, & m \neq 0 \end{cases}$
$S_{\ell,0}$
0 1 2 3 4
cos
$\pi/2$
$\int f(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta, \; m = 0$


Collar
$a_{\ell,m} = c_{\ell,m} \iint f(\theta) P_{\ell,m}(\cos\theta) e^{im\phi} \sin\theta \, d\theta \, d\phi$
$= c_{\ell,m} \int_0^{2\pi} e^{im\phi} \, d\phi \int f(\theta) P_{\ell,m}(\cos\theta) \sin\theta \, d\theta$
$= \begin{cases} c_{\ell,0} \cdot 2\pi \int f(\theta) P_{\ell,0}(\cos\theta) \sin\theta \, d\theta \\ 0 \end{cases}$
if $m \neq 0$

https://cseweb.ucsd.edu/~ravir/p1004-ramamoorthi.pdf

A Signal-Processing Framework for Reflection

Signal-processing framework for analyzing reflected light field from homogeneous convex curved surface under distant illumination
Practical and theoretical application in:

- Determining lighting distributions
- Determining bidirectional reflectance distribution functions (BRDFs)
- Rendering Environment Maps
- Image-Based Rendering

Reflection operator behaves qualitatively like a convolution

- Reflected light field can be thought of by convolving light and BRDF
    - Filtering incident illumination using BRDF
- Able to express frequency-space coefficients of reflected light field as a product of the spherical harmonic coefficients of the illumination and BRDF
- Inverse Rendering
    - Estimation of BRDF and lighting parameters from real photographs
    - Able to derive analytic formulae for spherical harmonic coefficients of many common BRDF and lighting models.
    - Able to determine precise conditions under which estimation of BRDFs and lighting distributions are well posed and well-conditioned

- Forward Rendering
    - Efficient rendering of objects under complex lighting conditions specified by environment maps.

Introduction:

- Want to perform inverse rendering
    - Estimation of material and lighting properties from real photographs
- Increasing importance in graphics for obtaining input models for (forward) rendering
- Computer vision mainly on estimating shape from images, little work on estimatin material properties or lighting
    - Computer graphics, global illumination fairly well-developed.
    - Little work on simpler reflected equation (direct illumination incident on a surface)
- Significant knowledge of properties of reflection operator
    - Reflection from Lambertian surface blurs illumination
    - Usually represent diffuse reflection map at low resolutions (for environment map)
    - Use mirror surfaces like gazing spheres.
- Goal of paper is to formalize observations and present a mathematical theory of reflection for general complex lighting environment and arbitrary BRDFs

    - Describe a signal-processing framework for analyzing reflected light field from a homogeneous convex curved surface under distant illumination
    - Able to derive formula for reflected light field in terms of spherical harmonic coefficients of BRDF and Lighting
- Signal Processing Framework for Reflection as Convolution
    - Reflection operator behaves like a convolution
    - Reflected light field obtained by convolving lighting and BRDF, or filtering incident illumination using BRDF
    - Able to express frequency-space coefficients of reflected light field as product of spherical harmonic coefficient of illumination and BRDF
        - Forward Rendering can be viewed as Convolution
        - Inverse Rendering can be viewed as Deconvolution
- Well-Posedness and Conditioning of Forward and Inverse Problems
    - Inverse problems can have several solutions and are often numerically ill-conditioned (infeasible to solve with practical algorithms)
    - Able to analyze well-posedness and conditioning of some inverse problems
    - Additional assumptions likely required to address ill-conditioned or ill-posed inverse problems
    - Ill-conditioned inverse problem corresponds to forward problem where final results are not sensitive to certain components of initial conditions
        - Allows us to approximate initial conditions - such as incident illumination -
- Basri and Jacobs [2003] have applied ideas to lighting-insensitive recognition and photometric stereo

Reflection Equation

1. Assumptions
    - Curved Surfaces: reflection of distant illumination field by curved surfaces. Specifically, variation of reflected light field as a function of surface orientation and exitant direction
        - Different orientations of curved surface corresponding to different orientations of upper hemisphere and BRDF.
        - Each orientation of surface corresponds to different integral over lighting
    - Distant Illumination:
        - Illumination field assumed to be generated by distant sources, allowing usame lighting function anywhere on object surface
        - represented by a single environment map indexed by incident angle
    - Convex Objects
        - Assumption of convexity ensures there is no shadowing or interreflection
        - Incident illumination is only because of distant illumination field
        - Convexity allows parameterizing object simply by surface orientation
        - Isotropic surfaces, surface orientation is specified uniquely by normal vector

    - Anisotropic surfaces, specify direction of anisotropy: orientation of local tangent frame
- Homogeneous Surfaces: Assume untextured surfaces with same BRDF everywhere
- Discussion
    - Assumptions are very similar to those made in most interactive graphics application
    - Also similar to those in computer vision and inverse rendering
    - Only significant additional assumption is that of homogeneous surfaces.
        - Spatially varying BRDFs are often approximated in practical graphics/vision using spatially varying texture that modulates one or more components of BRDF
        - A

2. Flatland 2D Case
    - All surface normals, measurements, and illumination restricted to single plane

$$B(\mathbf{x}, \theta_o') = \int_{-\pi/2}^{\pi/2} L(\mathbf{x}, \theta_i')\rho(\theta_i', \theta_o') \cos\theta_i' \, d\theta_i'.$$

- Reflection equation
- B is reflected radiance
- L is incident radiance (illumination)
- p is BRDF (function of local incident and outgoing angles (θ'_i,θ'_o) in 2D
- Limits of integration correspond to visible half-circle

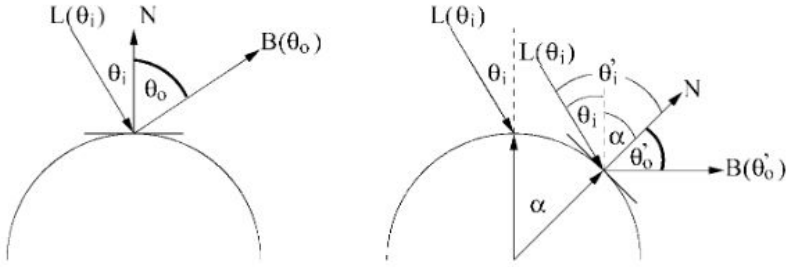


Fig. 2. Schematic of reflection in 2D. On the left, we show the situation with respect to one point on the surface (the north pole or 0° location, where global and local coordinates are the same). The right figure shows the effect of the surface orientation α. Different orientations of the surface correspond to rotations of the upper hemisphere and BRDF, with the global incident direction θ_i corresponding to a rotation by α of the local incident direction θ_i'. Note that we also keep the local outgoing angle (between N and B) fixed between the two figures.

- Assumption of convex surface → no shadowing or interreflection
    - Assumed in Equation 1
    - Depends only on distant illumination field L and surface BRDF p.
- Assumption of distant illumination
    - Implies reflected light field depends directly only on surface orientation, not directly on position x

    - Reparameterize surface by its angular coordinates
- Assumption of distant sources
    - represent incident illumination by single environment map for all surface positions using function L regardless of surface position.
- Transfer function p = pcos(θ_i) to absorb cosine term in integrand

$$B(\alpha, \theta_o') = \int_{-\pi/2}^{\pi/2} L(\theta_i)\hat{\rho}(\theta_i', \theta_o')\, d\theta_i'.$$

- Mixed local (primed) and global (unprimed coordinates
- Lighting is a global function, expressed in global coordinates frame as a function of global angles
- BRDF expressed in local incident and reflected angles
    - When expressed in local coordinate frame, same everywhere for homogeneous surface
    - Lighting same everywhere in global coordinate frame.
- Integration conveniently done over local or global coordinates, but upper hemisphere easier to keep track of in local coordinates

Rotations - Converting between Local and Global coordinates:

- Convert between these by applying rotation corresponding to local surface normal α.
    - Up-vector in local coordinates 0', is surface normal
    - R_α operator that rotates θ_i in global coordinates, given in 2D simply by θ

$$\theta_i = R_\alpha(\theta_i') = \alpha + \theta_i'$$

    - To convert global to local, apply inverse

$$\theta_i' = R_\alpha^{-1}(\theta_i) = -\alpha + \theta_i.$$

$$\begin{aligned} B(\alpha, \theta_o') &= \int_{-\pi/2}^{\pi/2} L(R_\alpha(\theta_i'))\hat{\rho}(\theta_i', \theta_o')\, d\theta_i' \\ &= \int_{-\pi/2+\alpha}^{\pi/2+\alpha} L(\theta_i)\, \hat{\rho}\left(R_\alpha^{-1}(\theta_i), \theta_o'\right) d\theta_i. \end{aligned}$$

$$\begin{aligned} B(\alpha, \theta_o') &= \int_{-\pi/2}^{\pi/2} L(\alpha + \theta_i')\hat{\rho}(\theta_i', \theta_o')\, d\theta_i' \\ &= \int_{-\pi/2+\alpha}^{\pi/2+\alpha} L(\theta_i)\, \hat{\rho}\,(-\alpha + \theta_i, \theta_o')\, d\theta_i. \end{aligned}$$

Interpretation as Convolution:

- Reflected light field at a given surface orientation corresponds to rotating the BRDF to that orientation, then integrating over the upper half-circle

- BRDF can be thought as a filter, while lighting is input signal. Reflected light field is obtained by filtering input signal using filter derived from BRDF.
- Symmetrically, reflected light field at given surface orientation may be computed by rotating lighting into local coordinate system of BRDF, then integrating over half circle

3. Generalization to 3D

$$B(\mathbf{x}, \theta_o', \phi_o') = \int_{\Omega'} L(\mathbf{x}, \theta_i', \phi_i') \rho(\theta_i', \phi_i', \theta_o', \phi_o') \cos\theta_i' \, d\omega_i'.$$

- Reflection Equation converted to 3D

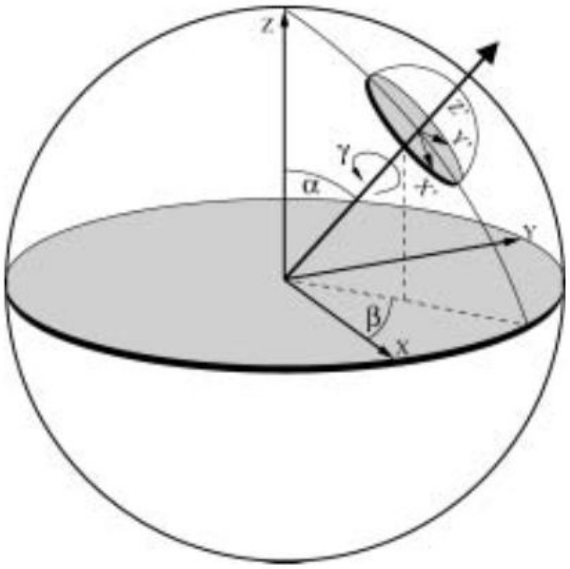


Fig. 3. Diagram showing how the rotation corresponding to (α, β, γ) transforms between local (primed) and global (unprimed) coordinates. The net rotation is composed of three independent rotations about Z, Y, and Z, with the angles α, β, and γ corresponding directly to the Euler angles.

- 3d upper hemisphere rather than 2D half-circle
  - Consider (local) azimuthal angles φ'_i and φ'_o
- Same substitutions as we did in 2D
  - Reparameterize surface position x by angular coordinate (α, β, γ)
  - Surface normal given by N = $[\sin\alpha\cos\beta, \sin\alpha\sin\beta, \cos\alpha]$.
  - Angular parameter γ important for anisotropic surface (but not isotropic surfaces)
    - Controls rotation of local tangent-frame about the surface normal

$$B(\alpha, \beta, \gamma, \theta_o', \phi_o') = \int_{\Omega_i'} L(\theta_i, \phi_i) \hat{\rho}(\theta_i', \phi_i', \theta_o', \phi_o') \, d\omega_i'.$$

Rotations - Converting between Local and Global Coordinates:

- Apply rotation to convert between local and global coordinates

- North pole (0', 0') or +Z axis in local coordinates is the surface normal, and the corresponding global coordinates are (α, β)
- Rotation of form $R_z(\beta)\, R_y(\alpha)$ correctly performs this transformation
  - Subscript z denotes rotation about Z axis
  - Subscript y denotes rotation about Y axis
- Generally, rotation between local and global coordinates also specify transformation of local tangent frame
- General rotation operator given by $R_{\alpha,\beta,\gamma} = R_z(\beta) R_y(\alpha) R_z(\gamma).$

- Leads to 3D equivalent of rotation

$$(\theta_i, \phi_i) = R_{\alpha,\beta,\gamma}(\theta_i', \phi_i') = R_z(\beta) R_y(\alpha) R_z(\gamma) \{\theta_i', \phi_i'\}$$
$$(\theta_i', \phi_i') = R^{-1}_{\alpha,\beta,\gamma}(\theta_i, \phi_i) = R_z(-\gamma) R_y(-\alpha) R_z(-\beta) \{\theta_i, \phi_i\}.$$

- Substitution yields

$$\begin{aligned} B(\alpha, \beta, \gamma, \theta_o', \phi_o') &= \int_{\Omega_i'} L(R_{\alpha,\beta,\gamma}(\theta_i', \phi_i')) \hat{\rho}(\theta_i', \phi_i', \theta_o', \phi_o')\, d\omega_i' \\ &= \int_{\Omega_i} L(\theta_i, \phi_i) \hat{\rho}\left(R^{-1}_{\alpha,\beta,\gamma}(\theta_i, \phi_i), \theta_o', \phi_o'\right) d\omega_i. \end{aligned}$$

- Utilizes spherical coordinates
- Could be formatted as 3x3 rotation matrix, result may be rewritten as

$$\begin{aligned} B(R, \omega_o') &= \int_{\Omega_i'} L(R\omega_i') \hat{\rho}(\omega_i', \omega_o')\, d\omega_i' \\ &= \int_{\Omega_i} L(\omega_i) \hat{\rho}(R^{-1}\omega_i, \omega_o')\, d\omega_i, \end{aligned}$$

Interpretation as Convolution:

- In spatial domain, convolution is result generated when filter is translated over input signal. Can generalize to transformations T_a where T_a is function of a, and write

$$(f \otimes g)(a) = \int_t f(T_a(t)) g(t)\, dt.$$

  - Standard expression for spatial convolution. When T_a is rotation by angle a, formula defines convolution in angular domain

Frequency Space Analysis:

- Natural to analyze it in frequency-space.
  - 2D reflection, then 3D generalization

1. Fourier Analysis in 2D

- Fourier basis functions F_k by

$$F_k(\theta) = \frac{1}{\sqrt{2\pi}} e^{Ik\theta}.$$

2. Spherical Harmonic Analysis in 3D

- Spherical harmonics are appropriate signal-processing tool for sphere
    - Equivalent for that domain to fourier series in 2D (on a circle)
- Spherical Harmonic Y_{lm} is given by

$$N_{lm} = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}$$
$$Y_{lm}(\theta,\phi) = N_{lm} P_{lm}(\cos\theta) e^{Im\phi},$$

    - N_{lm} is a normalization factor.
    - Azimuthal dependence expanded in terms of Fourier basis functions
    - Θ dependence is expanded in terms of associated Legendre functions P_{lm}
    - l >= 0
    - -l <= m <= l
    - 2l+1 basis functions for given order l
    - Can be written as either trigonometric functions of sphericals coordinates (θ,φ) or as polynomials of cartesian components (x,y,z) with x^2, y^2, z^2 = 1
- Rotation formula for spherical harmonics is

$$Y_{lm}(R_{\alpha,\beta,\gamma}(\theta,\phi)) = \sum_{m'=-l}^{l} D^l_{mm'}(\alpha,\beta,\gamma) Y_{lm'}(\theta,\phi).$$

- M indices are mixed
    - Spherical harmonic after rotation must be expressed as a combination of other spherical harmonics with different m indices.
- L indices are mixed
    - Rotations of spherical harmonics with order l composed of spherical harmonics with order l
- D^l matrix that tells us how spherical harmonic transforms under rotation
    - rewrite rotated spherical harmonic as linear combination of all spherical harmonics of same order.
    - D^l is (2l + 1) dimensional representation of rotation group SO(3)

$$D^l_{mm'}(\alpha,\beta,\gamma) = \int_{\phi=0}^{2\pi}\int_{\theta=0}^{\pi} Y_{lm}(R_{\alpha,\beta,\gamma}(\theta,\phi)) Y^*_{lm'}(\theta,\phi)\sin\theta\, d\theta\, d\phi.$$

- Can be simplified to be without z rotations (β = γ = 0)

$$d^l_{m\,m'}(\alpha) = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi} Y_{lm}(R_y(\alpha)(\theta,\phi)) Y^*_{lm'}(\theta,\phi) \sin\theta \, d\theta \, d\phi.$$

First 3 Representations of $d^l_{mm'}$ with l = 0,1, 2

$$d^0(\alpha) = 1$$

$$d^1(\alpha) = \begin{pmatrix} \cos^2\frac{\alpha}{2} & \frac{\sin\alpha}{\sqrt{2}} & \sin^2\frac{\alpha}{2} \\ -\frac{\sin\alpha}{\sqrt{2}} & \cos\alpha & \frac{\sin\alpha}{\sqrt{2}} \\ \sin^2\frac{\alpha}{2} & -\frac{\sin\alpha}{\sqrt{2}} & \cos^2\frac{\alpha}{2} \end{pmatrix}$$

$$d^2(\alpha) = \begin{pmatrix} \cos^4\frac{\alpha}{2} & 2\cos^3\frac{\alpha}{2}\sin\frac{\alpha}{2} & \frac{1}{2}\sqrt{\frac{3}{2}}\sin^2\alpha & 2\cos\frac{\alpha}{2}\sin^3\frac{\alpha}{2} & \sin^4\frac{\alpha}{2} \\ -2\cos^3\frac{\alpha}{2}\sin\frac{\alpha}{2} & \cos^2\frac{\alpha}{2}(-1+2\cos\alpha) & \sqrt{\frac{3}{2}}\cos\alpha\sin\alpha & (1+2\cos\alpha)\sin^2\frac{\alpha}{2} & 2\cos\frac{\alpha}{2}\sin^3\frac{\alpha}{2} \\ \frac{1}{2}\sqrt{\frac{3}{2}}\sin^2\alpha & -\sqrt{\frac{3}{2}}\cos\alpha\sin\alpha & \frac{1}{2}(3\cos^2\alpha-1) & \sqrt{\frac{3}{2}}\cos\alpha\sin\alpha & \frac{1}{2}\sqrt{\frac{3}{2}}\sin^2\alpha \\ -2\cos\frac{\alpha}{2}\sin^3\frac{\alpha}{2} & (1+2\cos\alpha)\sin^2\frac{\alpha}{2} & -\sqrt{\frac{3}{2}}\cos\alpha\sin\alpha & \cos^2\frac{\alpha}{2}(-1+2\cos\alpha) & 2\cos^3\frac{\alpha}{2}\sin\frac{\alpha}{2} \\ \sin^4\frac{\alpha}{2} & -2\cos\frac{\alpha}{2}\sin^3\frac{\alpha}{2} & \frac{1}{2}\sqrt{\frac{3}{2}}\sin^2\alpha & -2\cos^3\frac{\alpha}{2}\sin\frac{\alpha}{2} & \cos^4\frac{\alpha}{2} \end{pmatrix}$$

Two representations of matrices D^l:

$$D^l_{0\,m'}(\alpha,\beta,0) = d^l_{0m'}(\alpha) = \sqrt{\frac{4\pi}{2l+1}} Y^*_{lm'}(\alpha,\pi)$$

$$D^l_{m0}(\alpha,\beta,\gamma) = d^l_{m0}(\alpha)\, e^{Im\beta} = \sqrt{\frac{4\pi}{2l+1}} Y_{lm}(\alpha,\beta).$$

Decomposition into Spherical harmonics

- Expanding lighting in global coordinates

$$L(\theta_i,\phi_i) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} L_{lm} Y_{lm}(\theta_i,\phi_i).$$

- To obtain lighting in local coordinates, utilize same rotation in 2D

$$L(\theta_i,\phi_i) = L(R_{\alpha,\beta,\gamma}(\theta'_i,\phi'_i)) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{m'=-l}^{l} L_{lm} D^l_{m\,m'}(\alpha,\beta,\gamma) Y_{lm'}(\theta'_i,\phi'_i).$$

- Now convert transfer function p = pcosθ`$_i$
  - p is nonzero only over upper hemisphere, where cosθ`$_i$ > 0 $_{and}$ cosθ`$_0$ > 0

$$\hat{\rho}(\theta_i', \phi_i', \theta_o', \phi_o') = \sum_{l=0}^{\infty} \sum_{n=-l}^{l} \sum_{p=0}^{\infty} \sum_{q=-p}^{p} \hat{\rho}_{ln,pq} Y_{ln}^*(\theta_i', \phi_i') Y_{pq}(\theta_o', \phi_o')$$

Spherical Harmonic Reflection Equation:

$$B(\alpha, \beta, \gamma, \theta_o', \phi_o') = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{m'=-l}^{l} \sum_{l'=0}^{\infty} \sum_{n=-l'}^{l'} \sum_{p=0}^{\infty} \sum_{q=-p}^{p} L_{lm} \hat{\rho}_{l'n,pq} D_{mm'}^{l}(\alpha, \beta, \gamma) Y_{pq}(\theta_o', \phi_o') T_{lm'l'n}$$

$$\begin{aligned} T_{lm'l'n} &= \int_{\phi_i'=0}^{2\pi} \int_{\theta_i'=0}^{\pi} Y_{lm'}(\theta_i', \phi_i') Y_{l'n}^*(\theta_i', \phi_i') \sin\theta_i' \, d\theta_i' d\phi_i' \\ &= \delta_{ll'} \delta_{m'n}. \end{aligned}$$

- l` = l
- n = m`
- Results in:

$$B(\alpha, \beta, \gamma, \theta_o', \phi_o') = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{n=-l}^{l} \sum_{p=0}^{\infty} \sum_{q=-p}^{p} L_{lm} \hat{\rho}_{ln,pq} \left(D_{mn}^{l}(\alpha, \beta, \gamma) Y_{pq}(\theta_o', \phi_o') \right).$$

- Should expand reflected light field B in terms of new basis
  - $C_{lmnpq} = D_{mn}^{l}(\alpha, \beta, \gamma) Y_{pq}(\theta_o', \phi_o').$
  - D^l in basis comes directly from rotation formula for spherical harmonics
  - Basis functions are a product of matrices D^l and spherical harmonics Y_{pq}
  - Reflected direction is a unit vector described by (θ'_o, φ'_o)
  - Surface parameterization is rotation described by (α, β, γ)
  - Orthogonality relation for D^l is

$$\int_{\gamma=0}^{2\pi} \int_{\beta=0}^{2\pi} \int_{\alpha=0}^{\pi} \left(D_{mn}^{l}(\alpha, \beta, \gamma) \right)^* \left(D_{m'n'}^{l'}(\alpha, \beta, \gamma) \right) \sin\alpha \, d\alpha \, d\beta \, d\gamma = \frac{8\pi^2}{2l+1} \delta^{ll'} \delta_{mm'} \delta_{nn'}.$$

- Group-invariant dμ(g) of rotation group g = SO(3) is sin(α) d(α) d(β) d(γ)
- Integral of μ(g) = $8pi^2$
  - To obtain orthonormal basis, must normalize

$$C_{lmnpq} = \sqrt{\frac{2l+1}{8\pi^2}} D^l_{mn}(\alpha, \beta, \gamma) Y_{pq}(\theta'_o, \phi'_o)$$

$$B = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{n=-l}^{l} \sum_{p=0}^{\infty} \sum_{q=-p}^{p} B_{lmnpq} C_{lmnpq}(\alpha, \beta, \gamma, \theta'_o, \phi'_o)$$

$$B_{lmnpq} = \int_{\phi'_o=0}^{2\pi} \int_{\theta'_o=0}^{\pi} \int_{\gamma=0}^{2\pi} \int_{\beta=0}^{2\pi} \int_{\alpha=0}^{\pi} B(\alpha, \beta, \gamma, \theta'_o, \phi'_o) C^*_{lmnpq}(\alpha, \beta, \gamma, \theta'_o, \phi'_o) \sin\alpha \sin\theta'_o \, d\alpha \, d\beta \, d\gamma \, d\theta'_o \, d\phi'_o.$$

- Expansion of reflected light field in terms of orthonormal basis functions
- Assuming anisotropic surfaces for full generality, reflected light field is a function of five variables.
- Impractical to have full range of values for anisotropic parameter, tangent frame rotation, y for every surface orientation,
    - Y is often function of surface orientation (α, β)
- In frequency-space, reflected light field obtained simply by multiplying together coefficients of lighting and BRDF
    - Convolving incident illumination with BRDF

$$B_{lmnpq} = \sqrt{\frac{8\pi^2}{2l+1}} L_{lm} \hat{\rho}_{ln,pq}.$$

Alternative result without expanding output dependence

$$B_{lmn}(\theta'_o, \phi'_o) = \sqrt{\frac{8\pi^2}{2l+1}} L_{lm} \hat{\rho}_{ln}(\theta'_o, \phi'_o).$$

- Fixed local outgoing angle does NOT correspond to single image
    - Corresponds to general slice of reflected light field
- Local viewing angle is different for different points in image, depending on relative orientation between surface normal and viewing direction
    - Single image corresponds to single global viewing direction, and hence a single global outgoing angle.

3. Alternative Forms

Isotropic BRDFs

- Rotating local tangent frame makes no difference: functions of only 3 variables
    - $\hat{\rho}(\theta'_i, \phi'_i, \theta'_o, \phi'_o) = \hat{\rho}(\theta'_i, \theta'_o, | \phi'_o - \phi'_i |).$
    - With respect to reflected light field parameter γ
- Orientation of local tangent frame has no physical significance
- Terms that satisfy isotropy, are invariant to adding angle $\Delta\phi$ to incident and outgoing azimuthal angles are nonzero
    - Requires n = q

- BRDFs only depend on $|\phi'_o - \phi'_i|$,
    - Able to negate both incident and outgoing azimuthal angles without changing results

$$\hat{\rho}_{lpq} = \hat{\rho}_{lq,pq} = \hat{\rho}_{l(-q),p(-q)}.$$

- Isotropy reduces dimensionality of BRDF from 4D to 3D
    - Only 3 independent indices
    - Half degrees of freedom constrained since we can negate azimuthal angle
- Remove dependence of reflected light field on γ by arbitrarily setting γ = 0.
    - γ mathematically controls origin or 0-angle for φ'_o
    - Can be set arbitrarily
- Rotation operator given by
    - $$R_{\alpha,\beta} = R_{\alpha,\beta,0} = R_z(\beta) R_y(\alpha).$$
- Leading to new orthonormality relation

$$\int_{\beta=0}^{2\pi} \int_{\alpha=0}^{\pi} \left(D^l_{mn}(\alpha,\beta)\right)^* \left(D^{l'}_{m'n}(\alpha,\beta)\right) \sin\alpha \, d\alpha \, d\beta = \frac{4\pi}{2l+1} \delta^{ll'} \delta_{mm'}.$$

    - Orthogonality relation for index n no longer holds.
    - Absence of integral over γ leads to slight weakening of orthogonality relation
- Define normalization constants by

$$\Lambda_l = \sqrt{\frac{4\pi}{2l+1}}.$$

- Expanding lighting into global coordinates

$$L(\theta_i, \phi_i) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} L_{lm} Y_{lm}(\theta_i, \phi_i)$$

$$L(\theta_i, \phi_i) = L(R_{\alpha,\beta}(\theta'_i, \phi'_i)) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{m'=-l}^{l} L_{lm} D^l_{mm'}(\alpha,\beta) Y_{lm'}(\theta'_i, \phi'_i).$$

- Expansion of isotropic BRDF

$$\hat{\rho}(\theta'_i, \theta'_o, |\, \phi'_o - \phi'_i \,|) = \sum_{l=0}^{\infty} \sum_{p=0}^{\infty} \sum_{q=-\min(l,p)}^{\min(l,p)} \hat{\rho}_{lpq} Y^*_{lq}(\theta'_i, \phi'_i) Y_{pq}(\theta'_o, \phi'_o).$$

- Reflected light field can be expanded using product of representation matrices and spherical harmonics

$$C_{lmpq}(\alpha, \beta, \theta_o', \phi_o') = \Lambda_l^{-1} D_{mq}^l(\alpha, \beta) Y_{pq}(\theta_o', \phi_o')$$

$$B(\alpha, \beta, \theta_o', \phi_o') = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{p=0}^{\infty} \sum_{q=-\min(l,p)}^{\min(l,p)} B_{lmpq} C_{lmpq}(\alpha, \beta, \theta_o', \phi_o')$$

$$B_{lmpq} = \int_{\phi_o'=0}^{2\pi} \int_{\theta_o'=0}^{\pi} \int_{\beta=0}^{2\pi} \int_{\alpha=0}^{\pi} B(\alpha, \beta, \theta_o', \phi_o') C_{lmpq}^*(\alpha, \beta, \theta_o', \phi_o') \sin\alpha \sin\theta_o' \, d\alpha \, d\beta \, d\theta_o' \, d\phi_o'.$$

- Basis functions C_{lmpq} orthonormal in spite of weakened orthogonality of functions D^l_{mq}
- Although representation matrices D^l
- Derive an analytic expression (convolution formula) for reflection equation in terms of coefficients

$$\boxed{B_{lmpq} = \Lambda_l L_{lm} \hat{\rho}_{lpq}}$$

- Different normalization and removal of y and corresponding index n
  - Still essentially similar to $B_{lmnpq} = \sqrt{\frac{8\pi^2}{2l+1}} L_{lm} \hat{\rho}_{ln,pq}.$
- Alternative form fixing outgoing elevation angle
  - Isotropic BRDF depends on $|\varphi'_o - \varphi'_i|$, do not hold φ'_0 fixed
  - Analogous to equation 40

$$\hat{\rho}(\theta_i', \theta_o', |\phi_o' - \phi_i'|) = \sum_{l=0}^{\infty} \sum_{q=-l}^{l} \hat{\rho}_{lq}(\theta_o') \left(\frac{1}{\sqrt{2\pi}} Y_{lq}^*(\theta_i', \phi_i') \exp(Iq\phi_o') \right)$$

$$B(\alpha, \beta, \theta_o', \phi_o') = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{q=-l}^{l} B_{lmq}(\theta_o') \left(\frac{1}{\sqrt{2\pi}} \Lambda_l^{-1} D_{mq}^l(\alpha, \beta) \exp(Iq\phi_o') \right).$$

$$B_{lmq}(\theta_o') = \Lambda_l L_{lm} \hat{\rho}_{lq}(\theta_o')$$

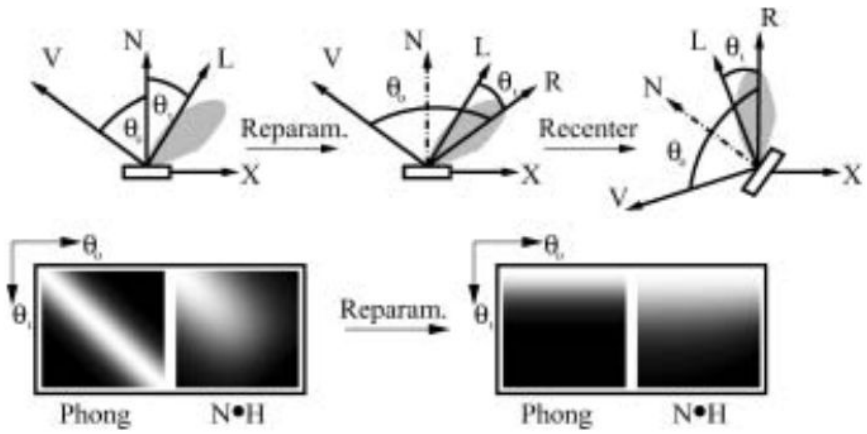


Fig. 5. Reparameterization involves recentering about the reflection vector. BRDFs become more compact, and in special cases (Phong) become 1D functions.

Reciprocity Preserving:

- Physical property of BRDFs is reciprocity
  - Symmetric w.r.t interchange of incident and outgoing angles
  - Transfer function $\hat{\rho} = \rho \cos \theta_i'$ does NOT preserver reciprocity
  - Multiply transfer function and reflected light by cos θ'$_0$,

$$\tilde{\rho} = \hat{\rho} \cos \theta_o' = \rho \cos \theta_i' \cos \theta_o'$$
$$\tilde{B} = B \cos \theta_o'$$

  - Will use frequency-space reflection formula whenever explicitly maintaining reciprocity of BRDF

$$\tilde{B}_{lmpq} = \Lambda_l L_{lm} \tilde{\rho}_{lpq}$$

Reparameterization by Central BRDF Direction

- First special case of radially symmetric 1D BRDFs
  - BRDF single symmetric lobe of fixed shape, orientation on well-defined central direction
  - BRDF 1D function u → $\hat{\rho} = u(\vec{C} \cdot \vec{L})$.
  -

4. asdasdasd

http://silviojemma.com/public/papers/lighting/spherical-harmonic-lighting.pdf