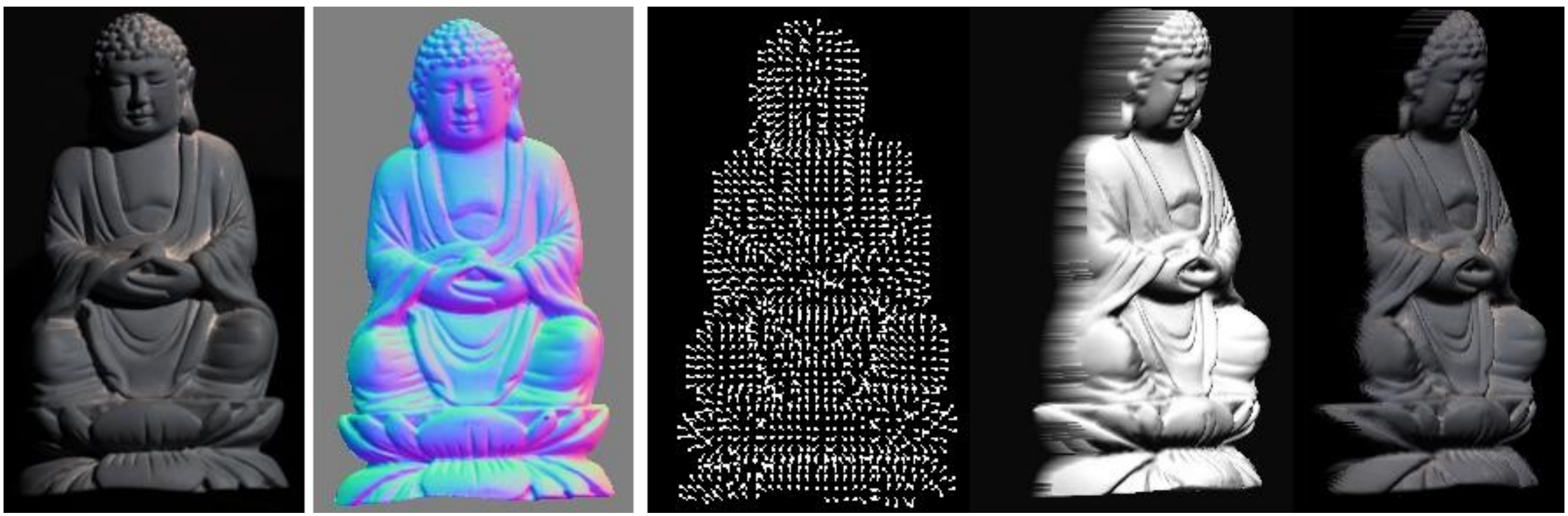
# CS4670/5760: Computer Vision

Kavita Bala

## Lecture 28: Photometric Stereo



Thanks to Scott Wehrwein

# Announcements

- PA 3 due at 1pm on Monday
- PA 4 out on Monday
- HW 2 out on weekend
- Next week: MVS, sFM

# Last Time: Two-View Stereo

# Last Time: Two-View Stereo

Key Idea: use feature motion to understand shape

# Today: Photometric Stereo

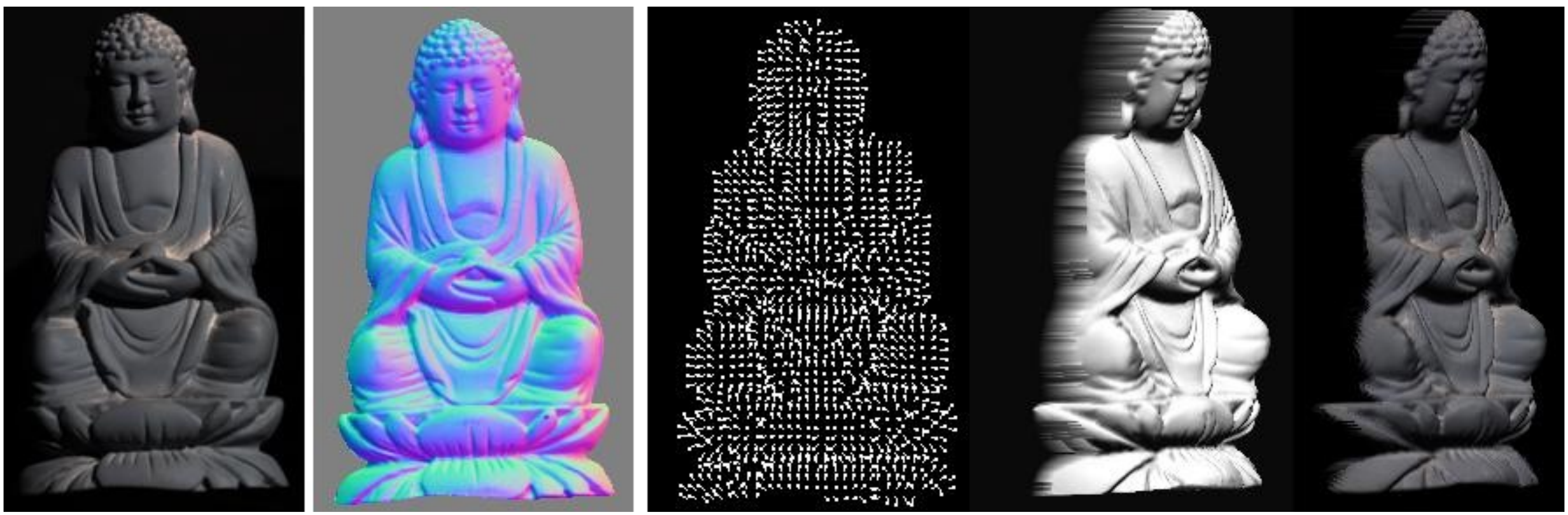
Key Idea: use pixel brightness to understand shape

# Today: Photometric Stereo

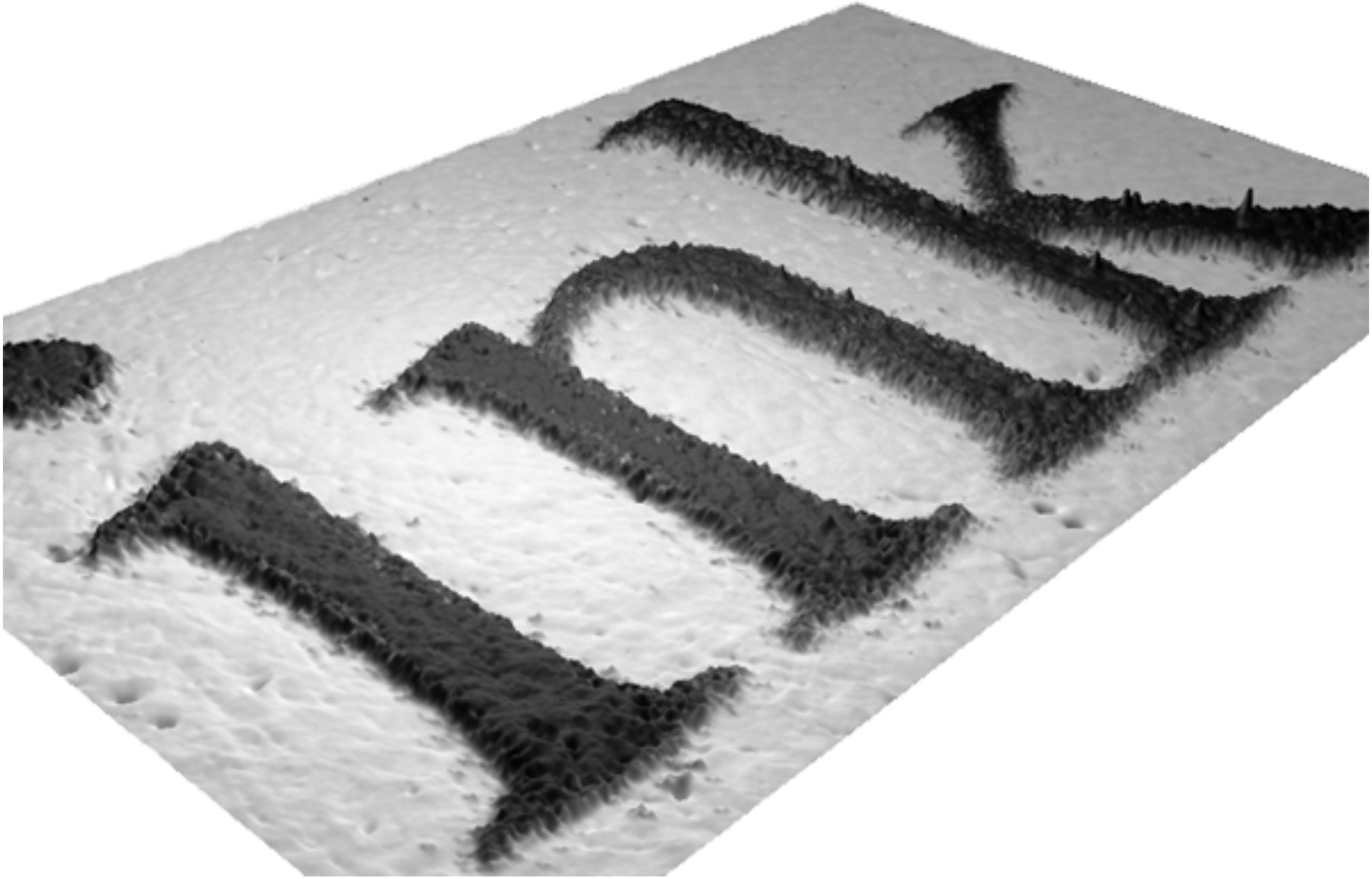
Key Idea: use pixel brightness to understand shape

# Photometric Stereo

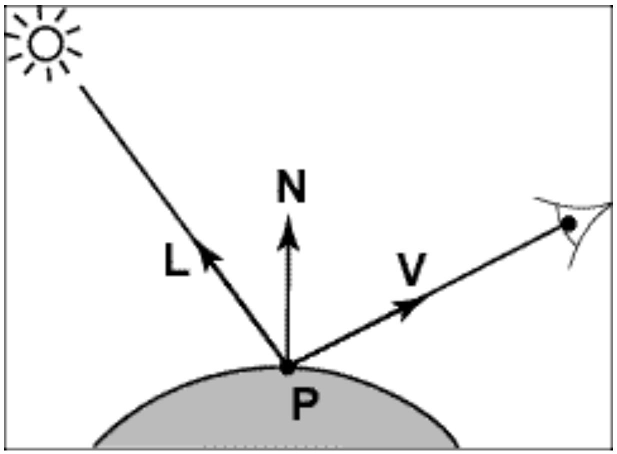
What results can you get?



Input (1 of 12) | Normals (RGB colormap) | Normals (vectors) | Shaded 3D rendering | Textured 3D rendering



# Modeling Image Formation

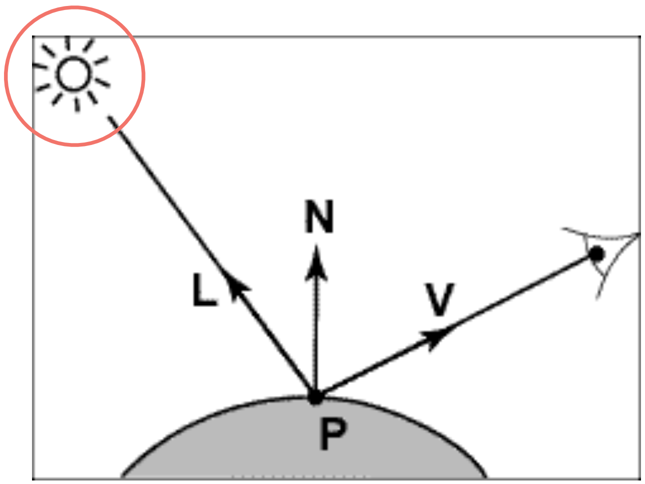



Now we need to reason about:

- How light interacts with the scene
- How a pixel value is related to light energy in the world

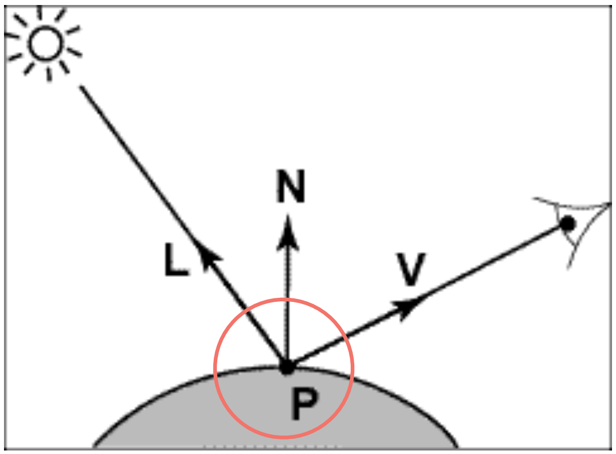
Track a "ray" of light all the way from light source to the sensor

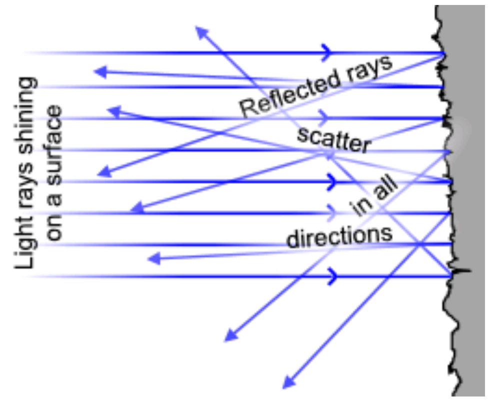
# Directional Lighting




- Key property: all rays are parallel
- Equivalent to an infinitely distant point source

# Lambertian Reflectance



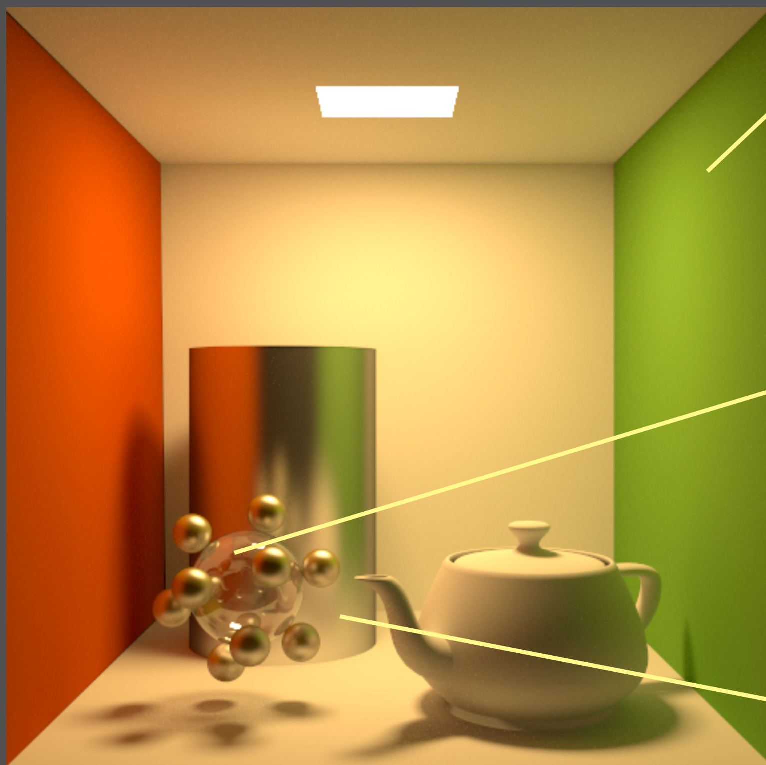



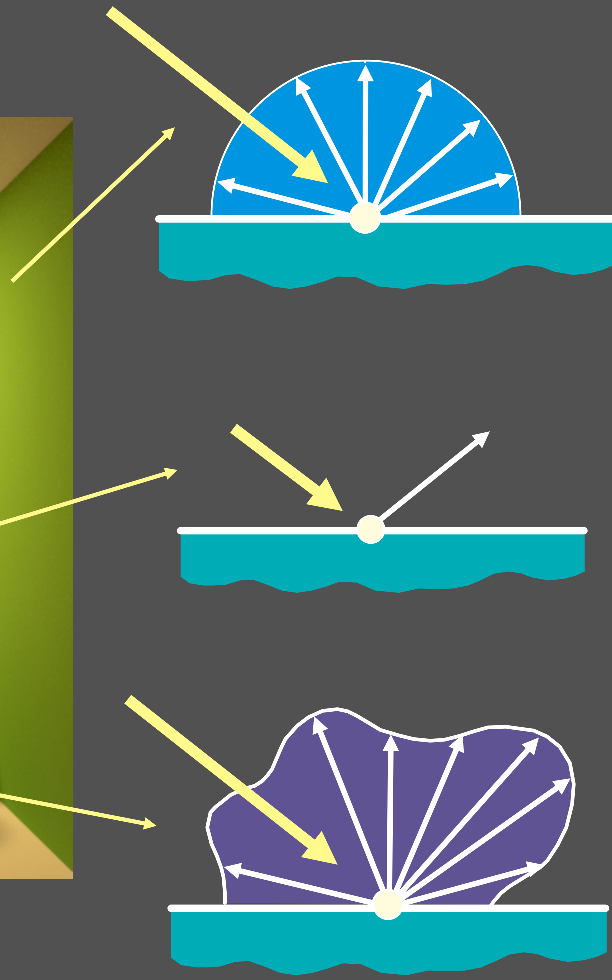

$$I = N \cdot L$$

Image intensity $=$ Surface normal $\cdot$ Light direction

Image intensity $\propto$ cos(angle between N and L)

# Materials - Three Forms

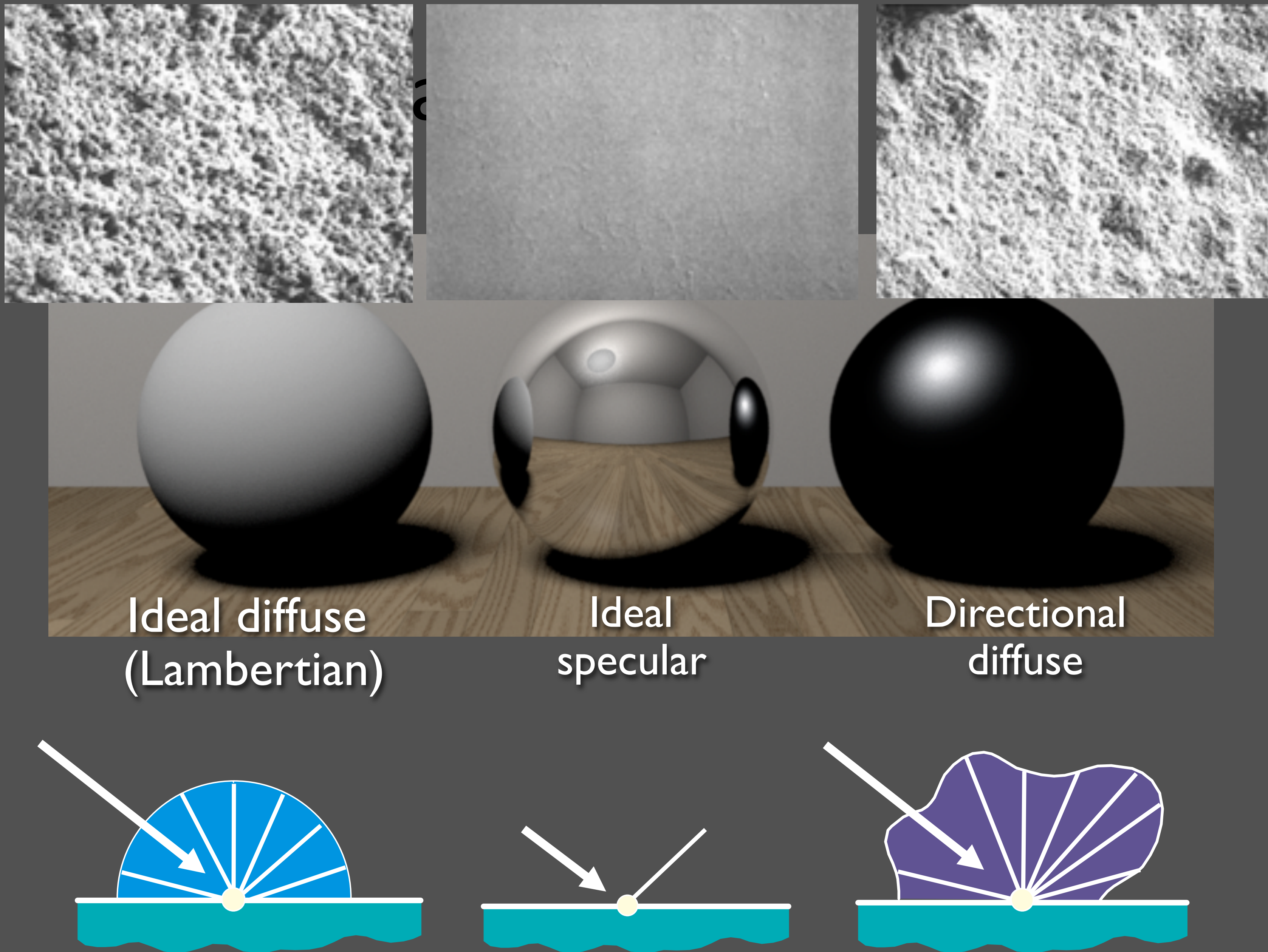




Ideal diffuse (Lambertian)

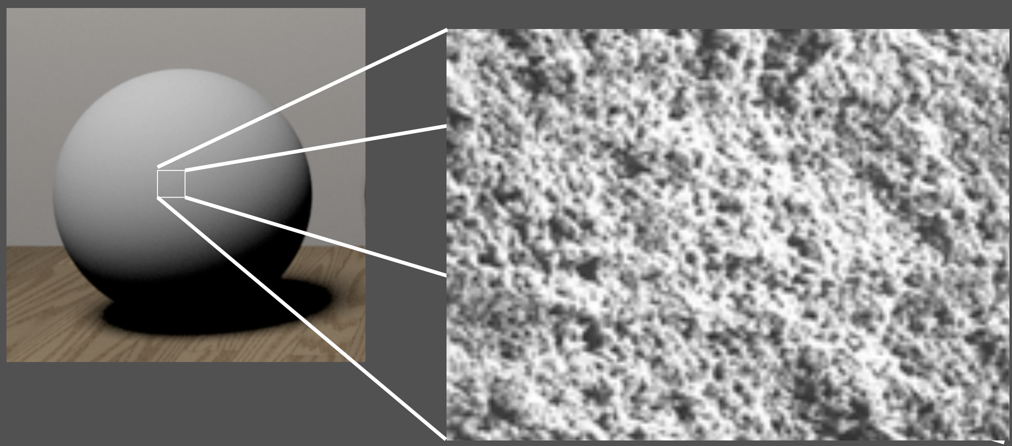
Ideal specular

Directional diffuse


Ideal diffuse
(Lambertian)
Ideal
specular
Directional
diffuse

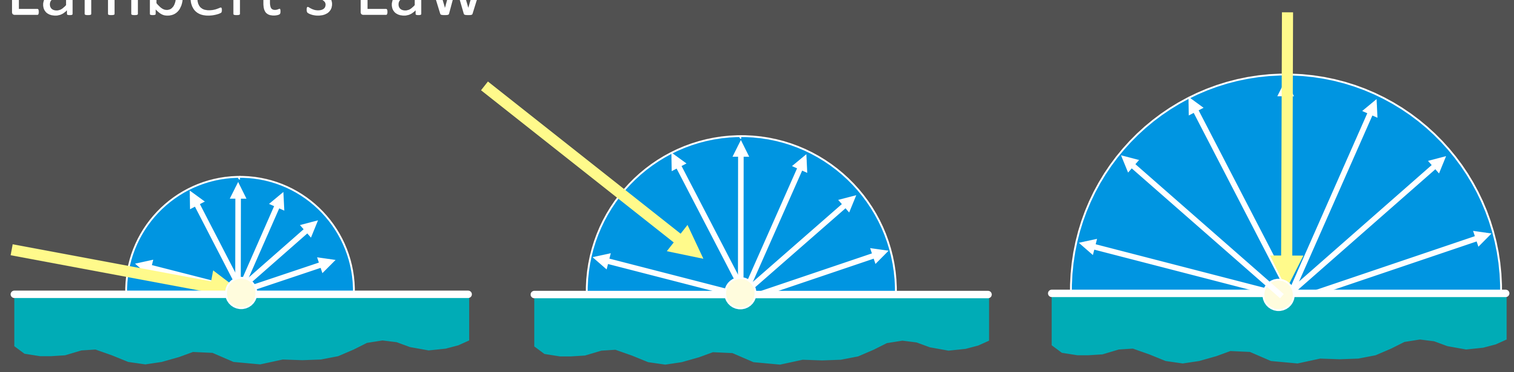
# Ideal Diffuse Reflection

- Characteristic of multiple scattering materials
- An idealization but reasonable for matte surfaces
- Basis of most radiosity methods



# Ideal Diffuse

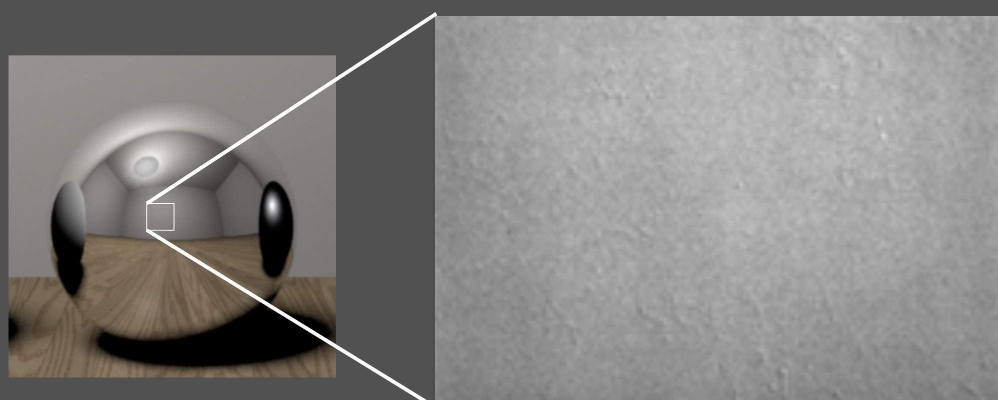
- Lambert's Law



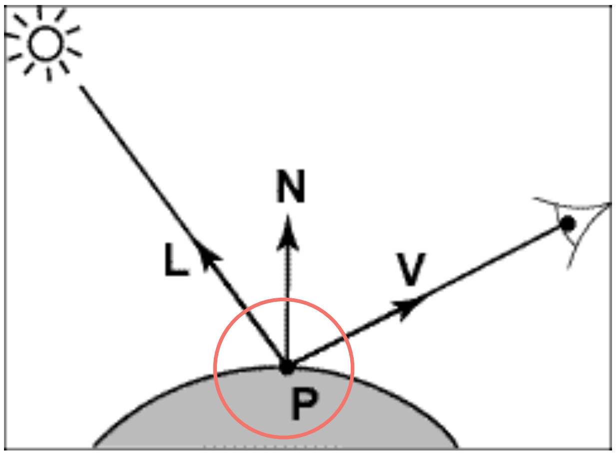
$$I_{diffuse} = I_{light} k_d cos(\theta)$$
$$I_{diffuse} = I_{light} k_d N.L$$

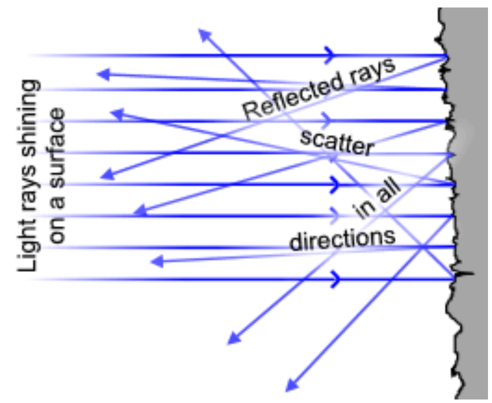
# Ideal Specular Reflection

- Calculated from Fresnel's equations
- Exact for polished surfaces
- Basis of early ray-tracing methods



# Lambertian Reflectance




1. Reflected energy is proportional to cosine of angle between L and N **(incoming)**

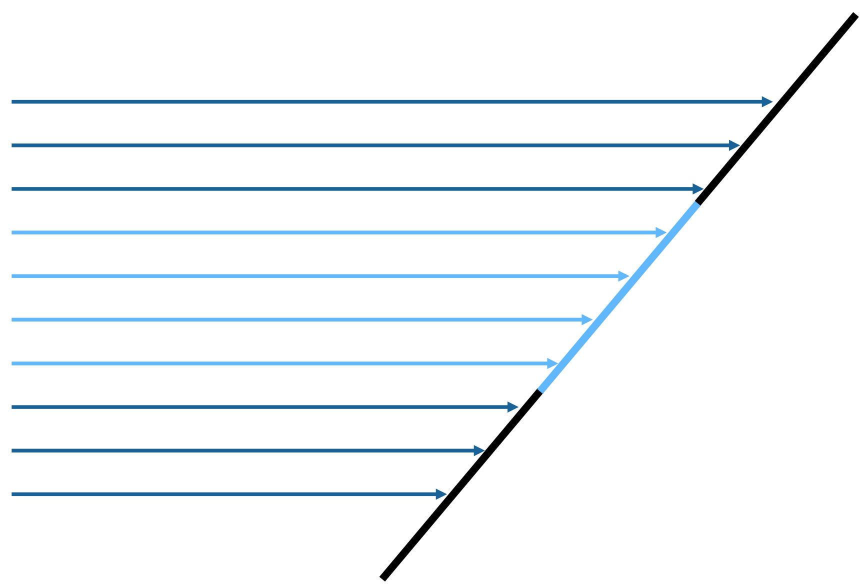
2. Measured intensity is viewpoint-independent **(outgoing)**

# Lambertian Reflectance: Incoming

1. Reflected energy is proportional to cosine of angle between L and N

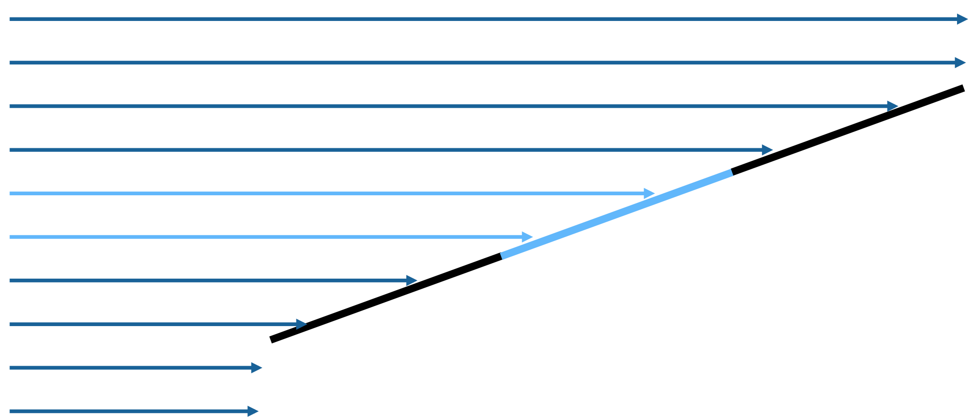
# Lambertian Reflectance: Incoming

1. Reflected energy is proportional to cosine of angle between L and N



# Lambertian Reflectance: Incoming

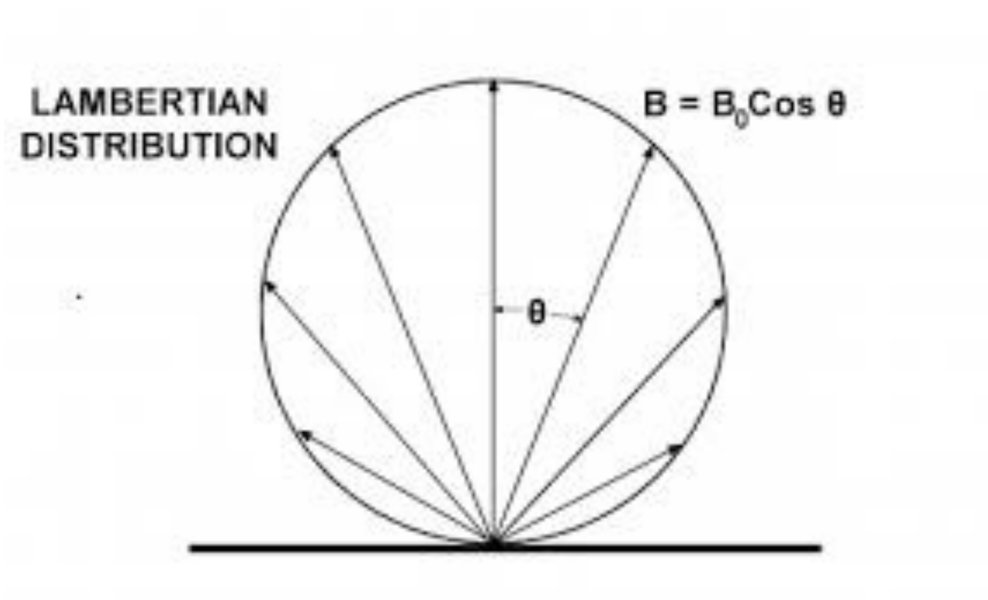
1. Reflected energy is proportional to cosine of angle between L and N



Light hitting surface is proportional to the cosine

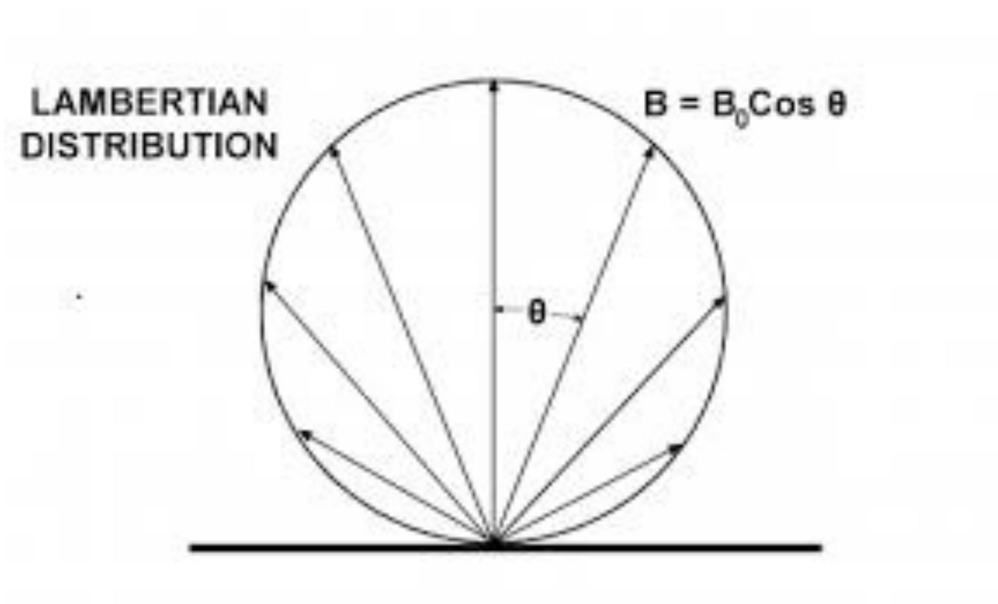
# Lambertian Reflectance: Outgoing

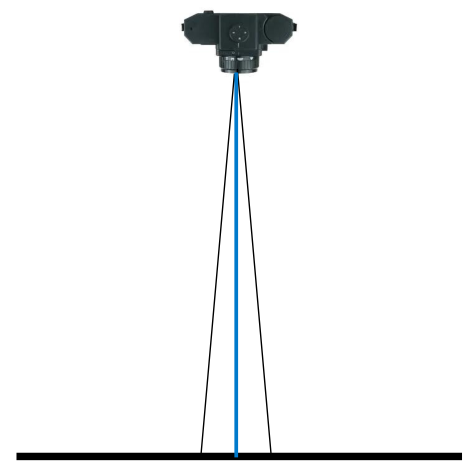
1. Radiance (what we see) is viewpoint-independent



# Lambertian Reflectance: Outgoing

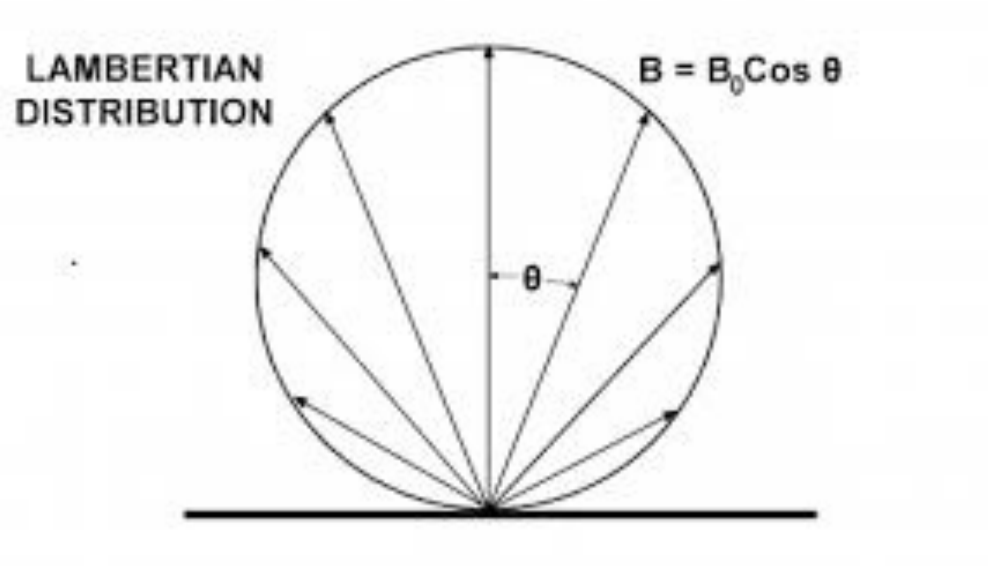
1. Radiance (what the eye sees) is viewpoint-independent

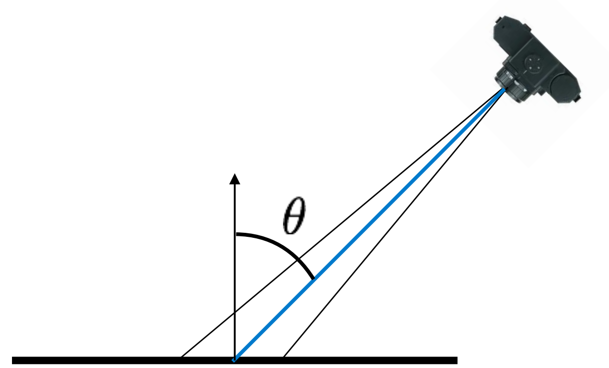


# Lambertian Reflectance: Outgoing

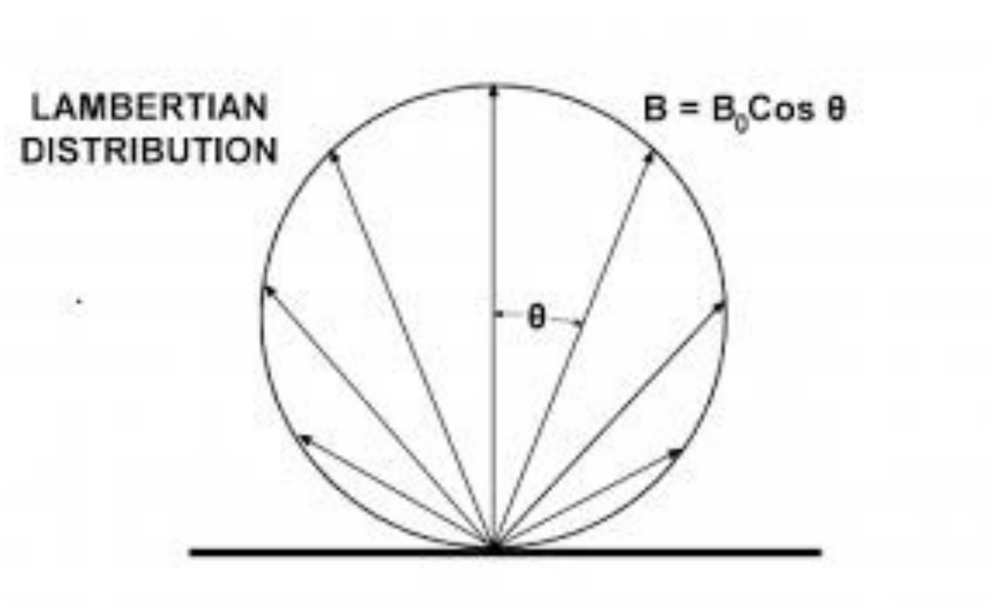
1. Measured intensity is viewpoint-independent

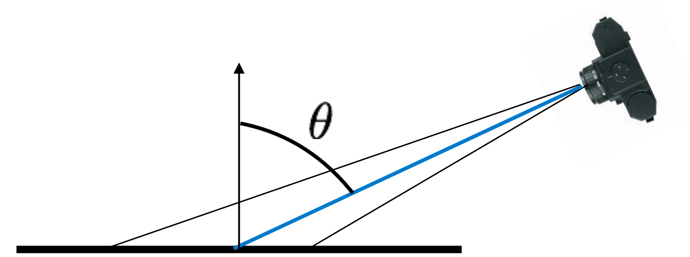


# Lambertian Reflectance: Outgoing

1. Measured intensity is viewpoint-independent



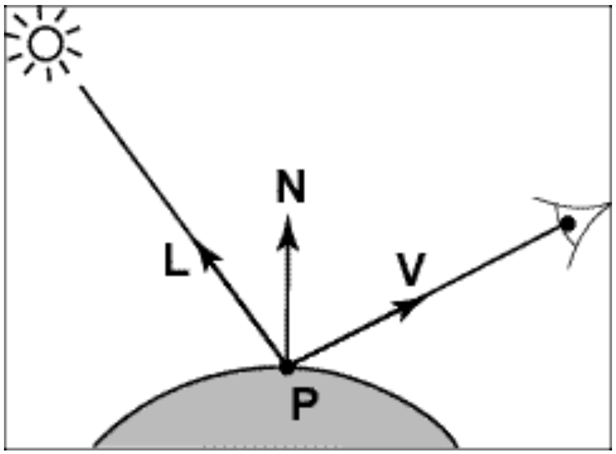




A cos (θ)

Radiance
(what eye sees)

$$\propto B_0 \cos(\theta) \frac{1}{\cos(\theta)}$$

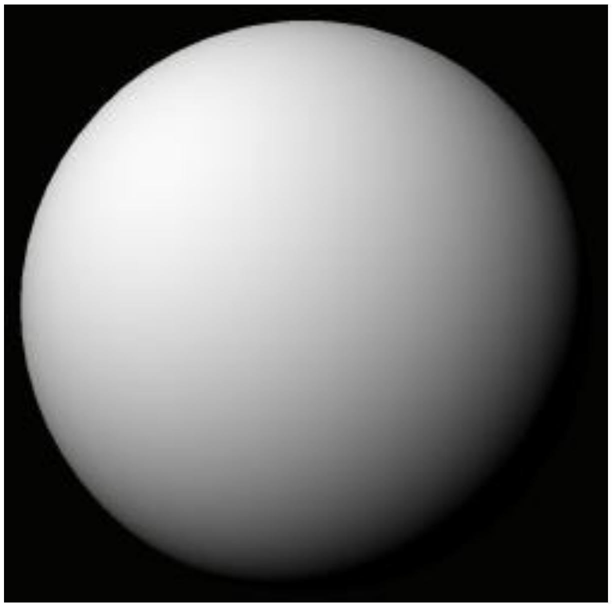
# Image Formation Model: Final




$$I = k_d \mathbf{N} \cdot \mathbf{L}$$

1. Diffuse albedo: what fraction of incoming light is reflected?
   - Introduce scale factor $k_d$
2. Light intensity: how much light is arriving?
   - Compensate with camera exposure (global scale factor)
3. Camera response function
   - Assume pixel value is linearly proportional to incoming energy (perform radiometric calibration if not)

# A Single Image: Shape from Shading



$$I = k_d \mathbf{N} \cdot \mathbf{L}$$

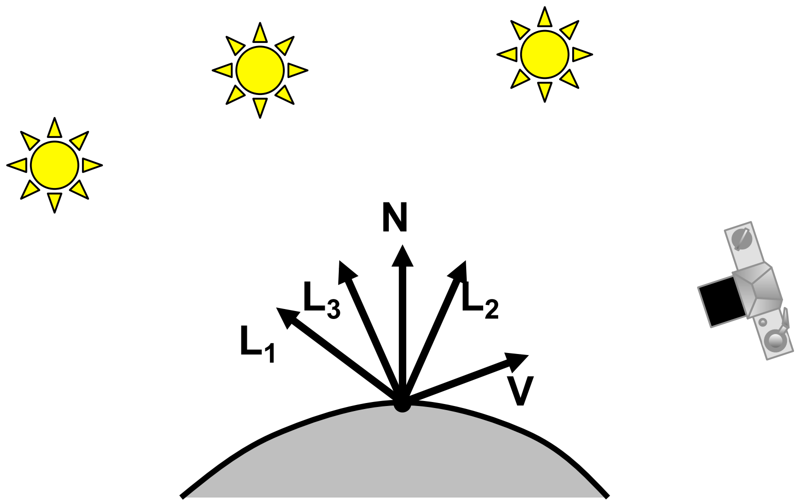
Assume $k_d$ is 1 for now.

What can we measure from one image?

- $\cos^{-1}(I)$ is the angle between N and L
- Add assumptions:
  - A few known normals (e.g. silhouettes)
  - Smoothness of normals

In practice, SFS doesn't work very well:
assumptions are too restrictive,
too much ambiguity in nontrivial scenes.

# Multiple Images: Photometric Stereo




$$I_1 = k_d \mathbf{N} \cdot \mathbf{L_1}$$
$$I_2 = k_d \mathbf{N} \cdot \mathbf{L_2}$$
$$I_3 = k_d \mathbf{N} \cdot \mathbf{L_3}$$

Write this as a matrix equation:

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

# Solving the Equations

$$\underbrace{\left[\begin{array}{ccc} I_1 & I_2 & I_3 \end{array}\right]}_{\substack{\mathbf{I} \\ 1 \times 3}} = \underbrace{k_d \mathbf{N}^T}_{\substack{\mathbf{G} \\ 1 \times 3}} \underbrace{\left[\begin{array}{ccc} \mathbf{L_1} & \mathbf{L_2} & \mathbf{L_3} \end{array}\right]}_{\substack{\mathcal{L} \\ 3 \times 3}}$$

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

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

$$\mathbf{N} = \frac{1}{k_d}\mathbf{G}$$

# Solving the Equations

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

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

- When is L nonsingular (invertible)?
  - >= 3 light directions are linearly independent, or:
  - All light direction vectors cannot lie in a plane.
- What if we have more than one pixel?
  - Stack them all into one big system.

# More than Three Lights

$$\left[\begin{array}{ccc} I_1 & \dots & I_n \end{array}\right] = k_d \mathbf{N}^T \left[\begin{array}{ccc} \mathbf{L_1} & \dots & \mathbf{L_n} \end{array}\right]$$

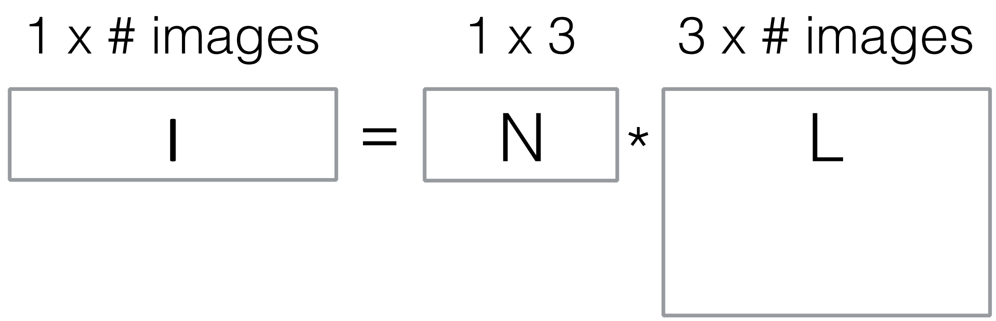
- Solve using least squares (normal equations):

$$\begin{aligned} \mathbf{I} &= \mathbf{GL} \\ \mathbf{IL^T} &= \mathbf{GLL^T} \\ \mathbf{G} &= \mathbf{(IL^T)(LL^T)^{-1}} \end{aligned}$$

- Equivalently use SVD
- Given G, solve for N and $k_d$ as before.

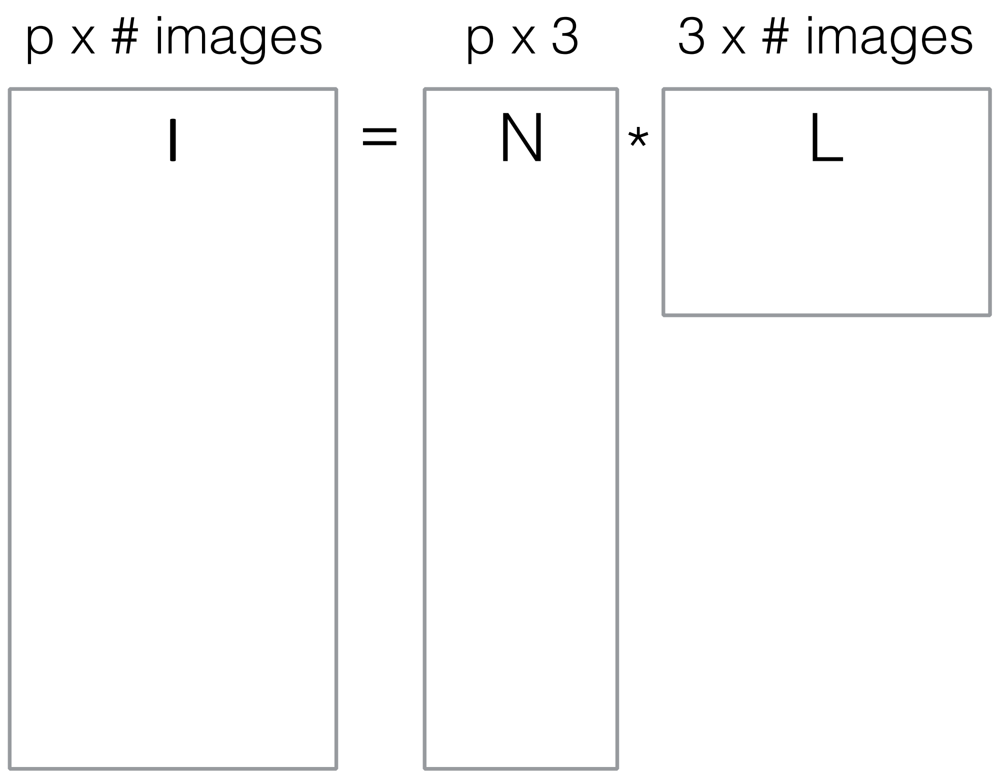
# More than one pixel

Previously:



# More than one pixel

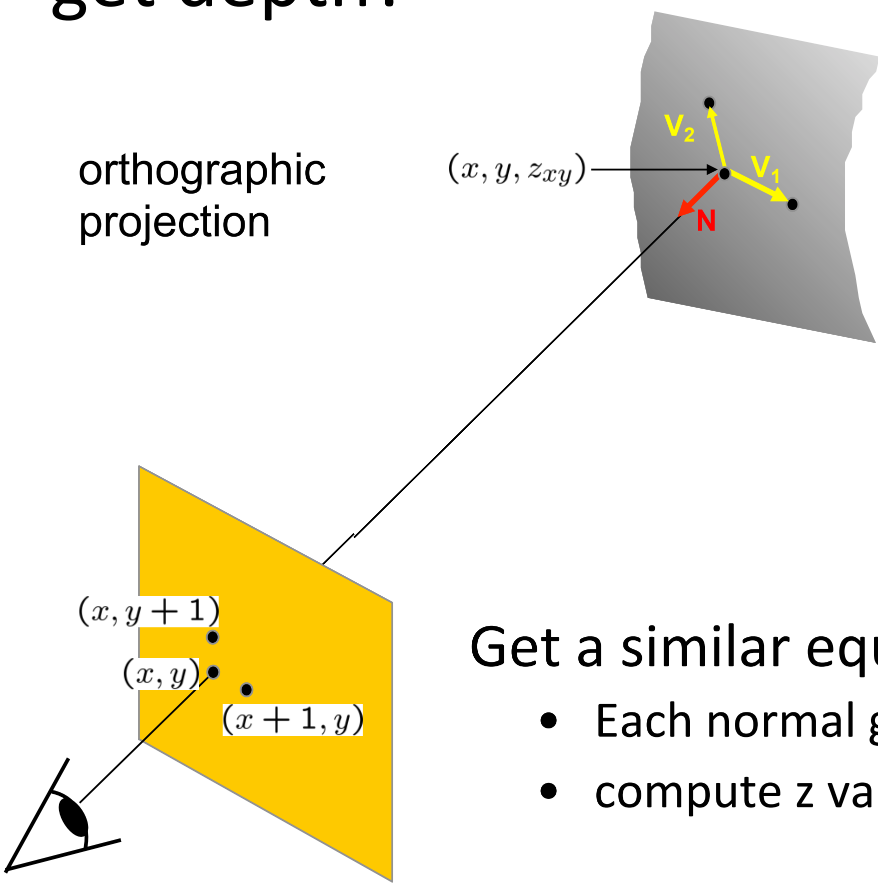
Stack all pixels into one system:




Solve as before.

# Depth Map from Normal Map

- We now have a surface normal, but how do we get depth?




Assume a smooth surface

$$\begin{aligned} V_1 &= (x+1, y, z_{x+1,y}) - (x, y, z_{xy}) \\ &= (1, 0, z_{x+1,y} - z_{xy}) \end{aligned}$$

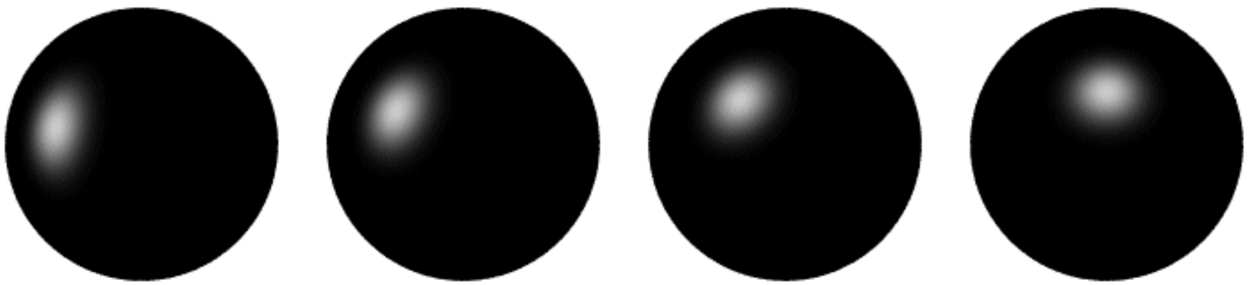
$$\begin{aligned} 0 &= N \cdot V_1 \\ &= (n_x, n_y, n_z) \cdot (1, 0, z_{x+1,y} - z_{xy}) \\ &= n_x + n_z(z_{x+1,y} - z_{xy}) \end{aligned}$$

Get a similar equation for $\mathbf{V_2}$

- Each normal gives us two linear constraints on z
- compute z values by solving a matrix equation

# Determining Light Directions

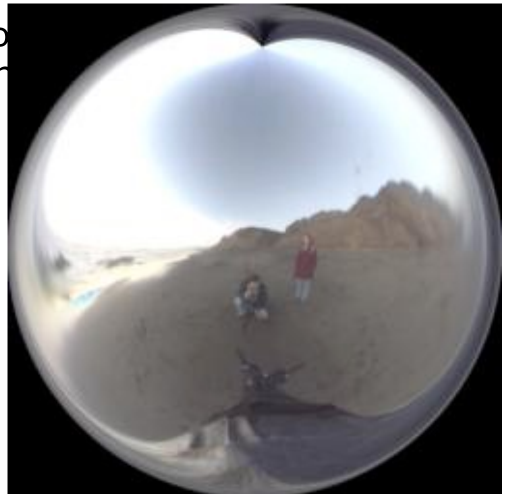
- Trick: Place a mirror ball in the scene.



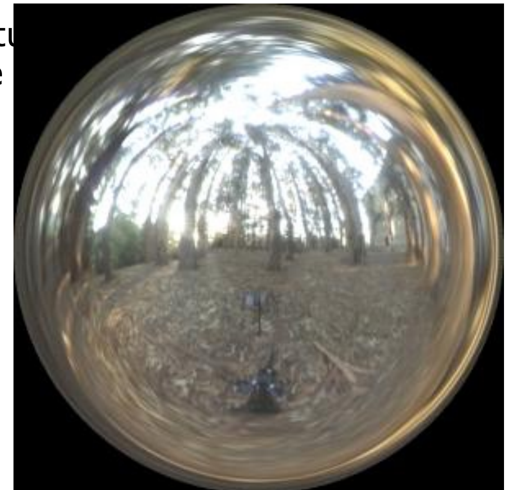
- The location of the highlight is determined by the light source direction.

# Real-World HDR Lighting Environments

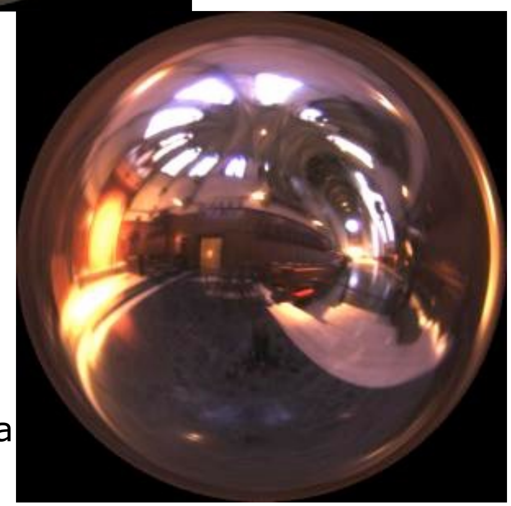
Funsto
Beach



Eucalyptu
Grove



Uffizi
Gallery



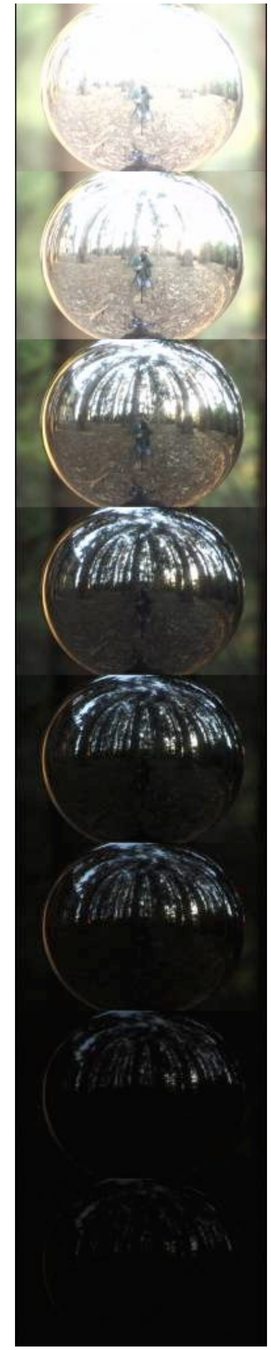
Grace
Cathedra

Lighting Environments from the Light Probe Image Gallery:
http://www.debevec.org/Probes/

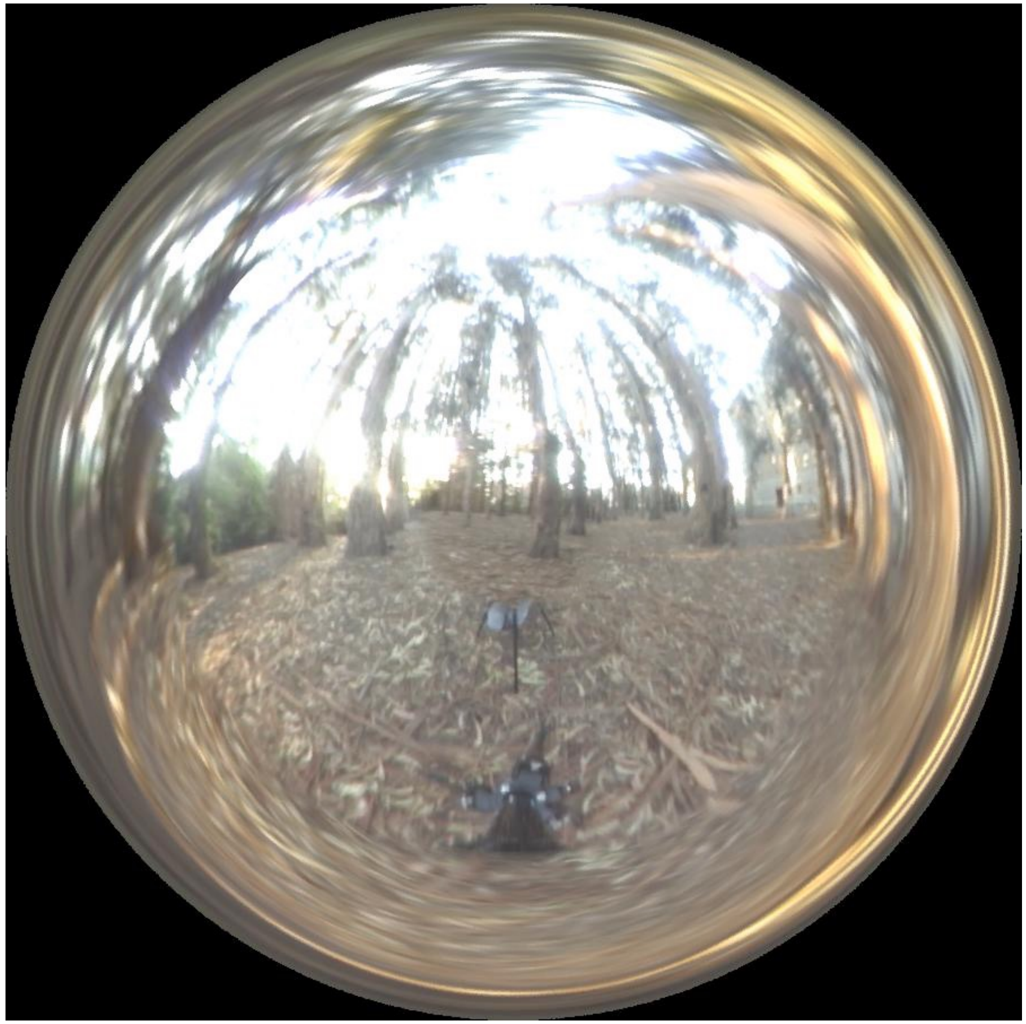

Mirrored Sphere


Canon
Canon
REMOTE SWITCH

# Acquiring the Light Probe



# Assembling the Light Probe



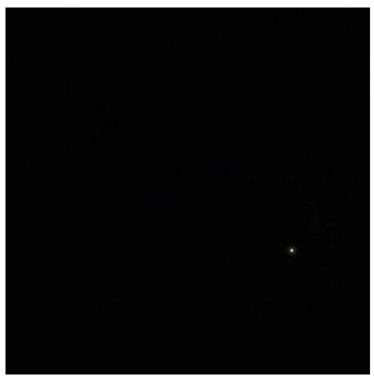
# Extreme HDR Image Series

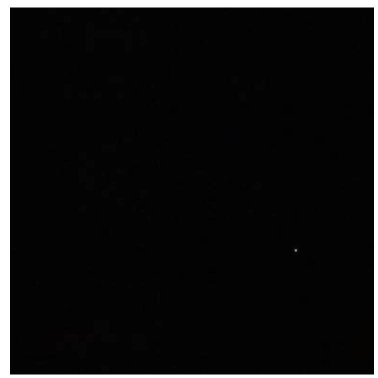

1 sec
f/4


1/4 sec
f/4


1/30 sec
f/4


1/30 sec
f/16


1/250 sec
f/16


1/1000 sec
f/16

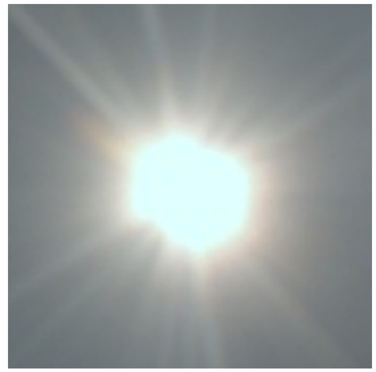

1/8000 sec f/16

# Extreme HDR Image Series

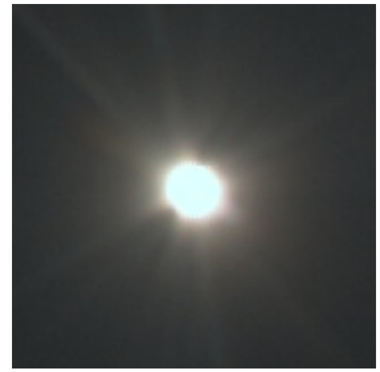
## sun closeup



1 sec
f/4

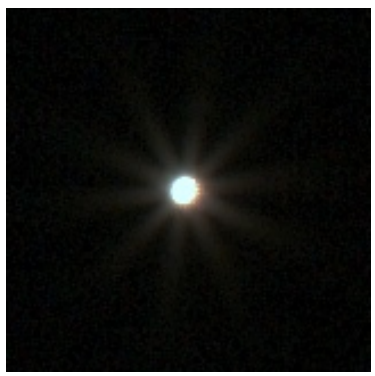


1/4 sec
f/4



1/30 sec
f/4

1/30 sec
f/16

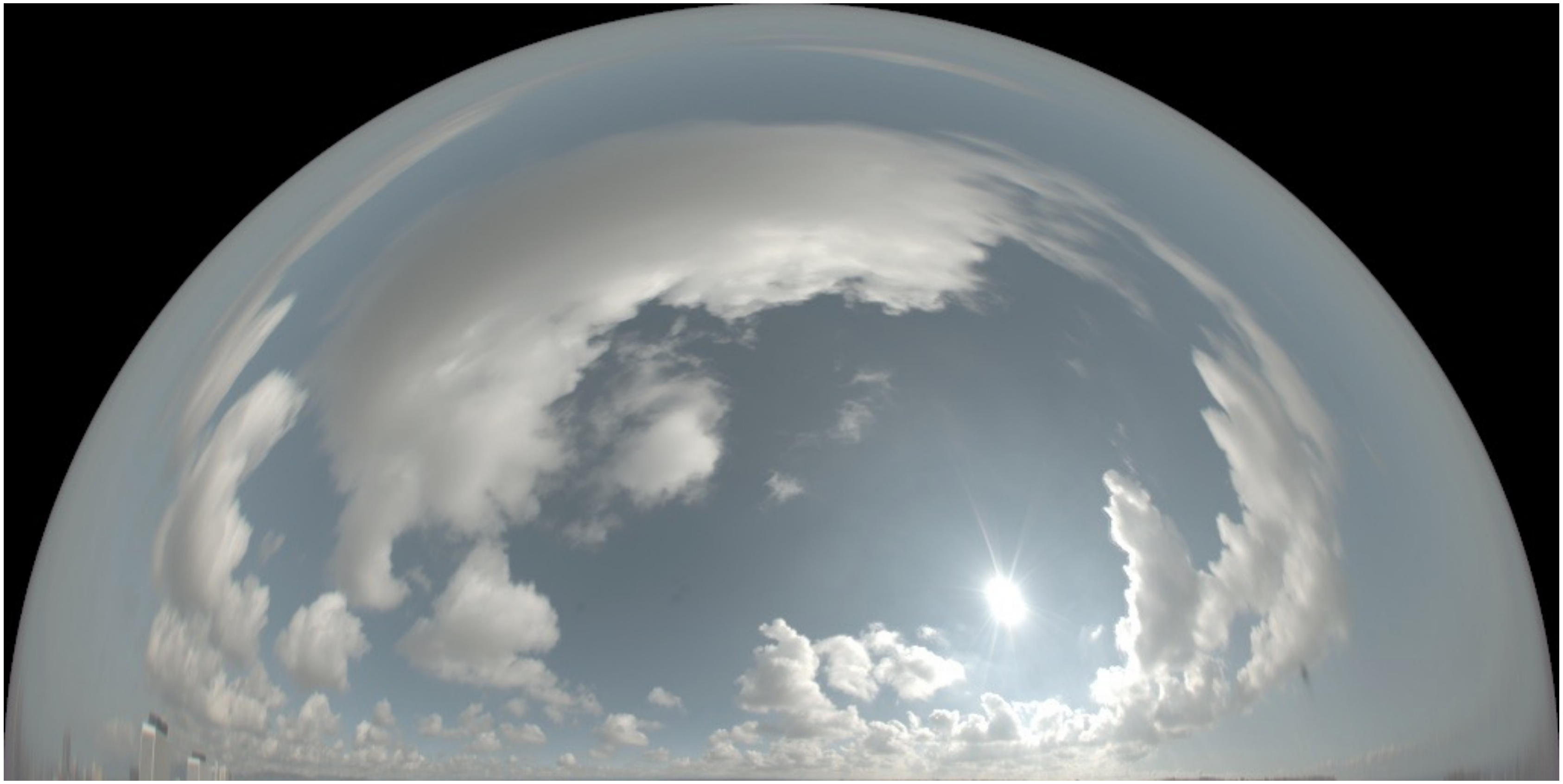


1/250 sec
f/16

1/1000 sec
f/16

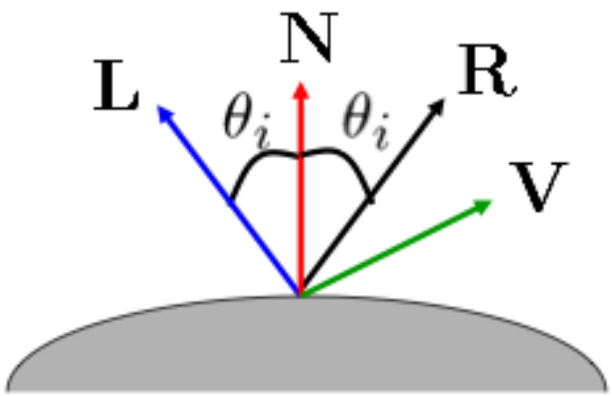
1/8000 sec f/16
only image that does not saturate!

# HDRI Sky Probe



# Determining Light Directions

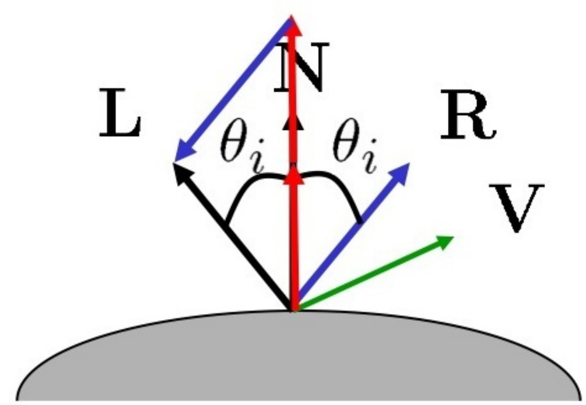
- For a perfect mirror, the light is reflected across N:




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

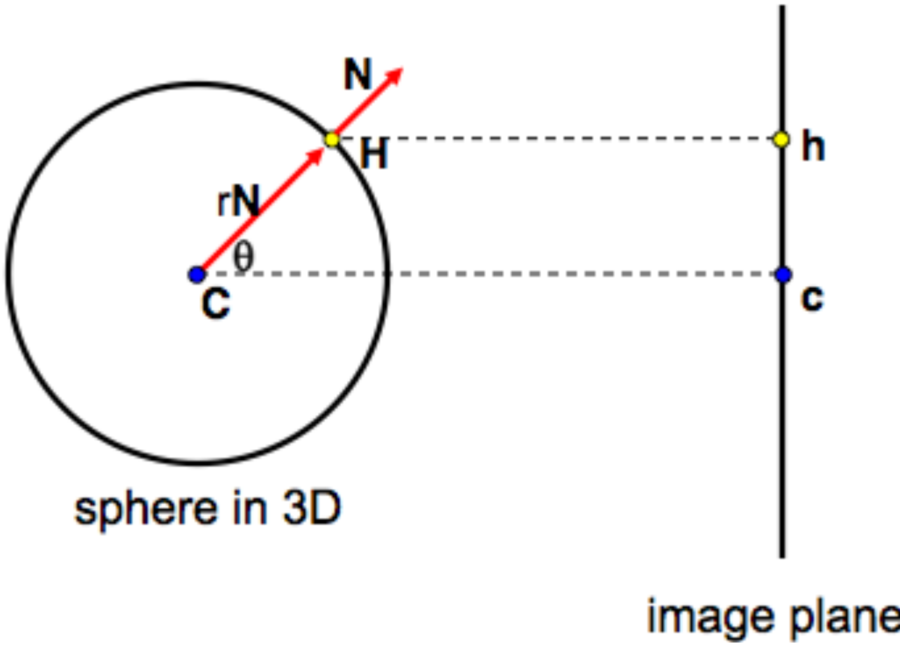
- So the light source direction is given by:

$$L = 2(N \cdot R)N - R$$



# Determining Light Directions

- For a sphere with highlight at point H:




Compute N:

$$N_x = \frac{x_h - x_c}{r}$$

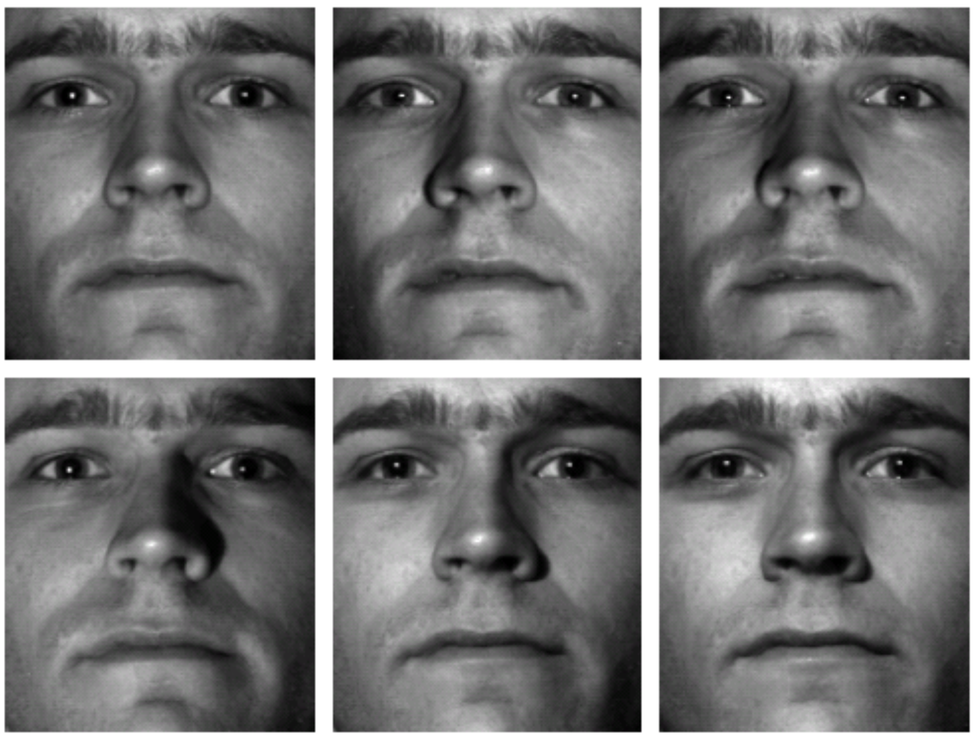
$$N_y = \frac{y_h - y_c}{r}$$

$$N_z = \sqrt{1 - x^2 - y^2}$$

- R = direction of the camera from C = $[0\ 0\ 1]^T$

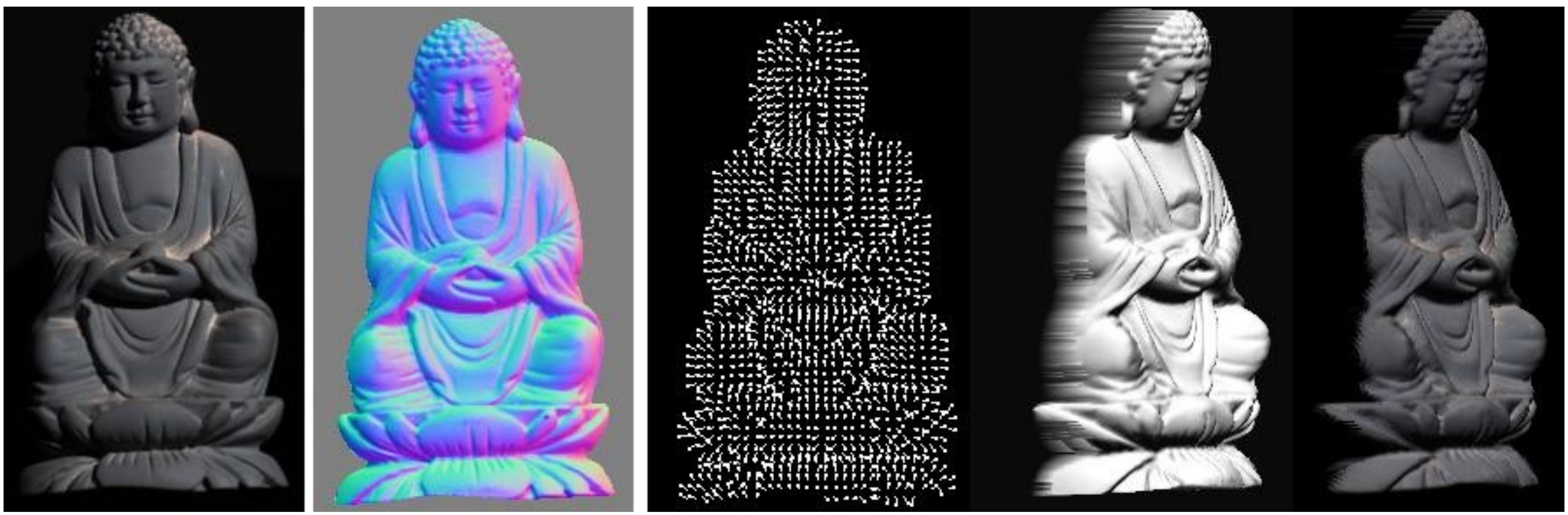
$$L = 2(N \cdot R)N - R$$

# Results



from Athos Georghiades

# Results



Input (1 of 12) | Normals (RGB colormap) | Normals (vectors) | Shaded 3D rendering | Textured 3D rendering

# Questions?