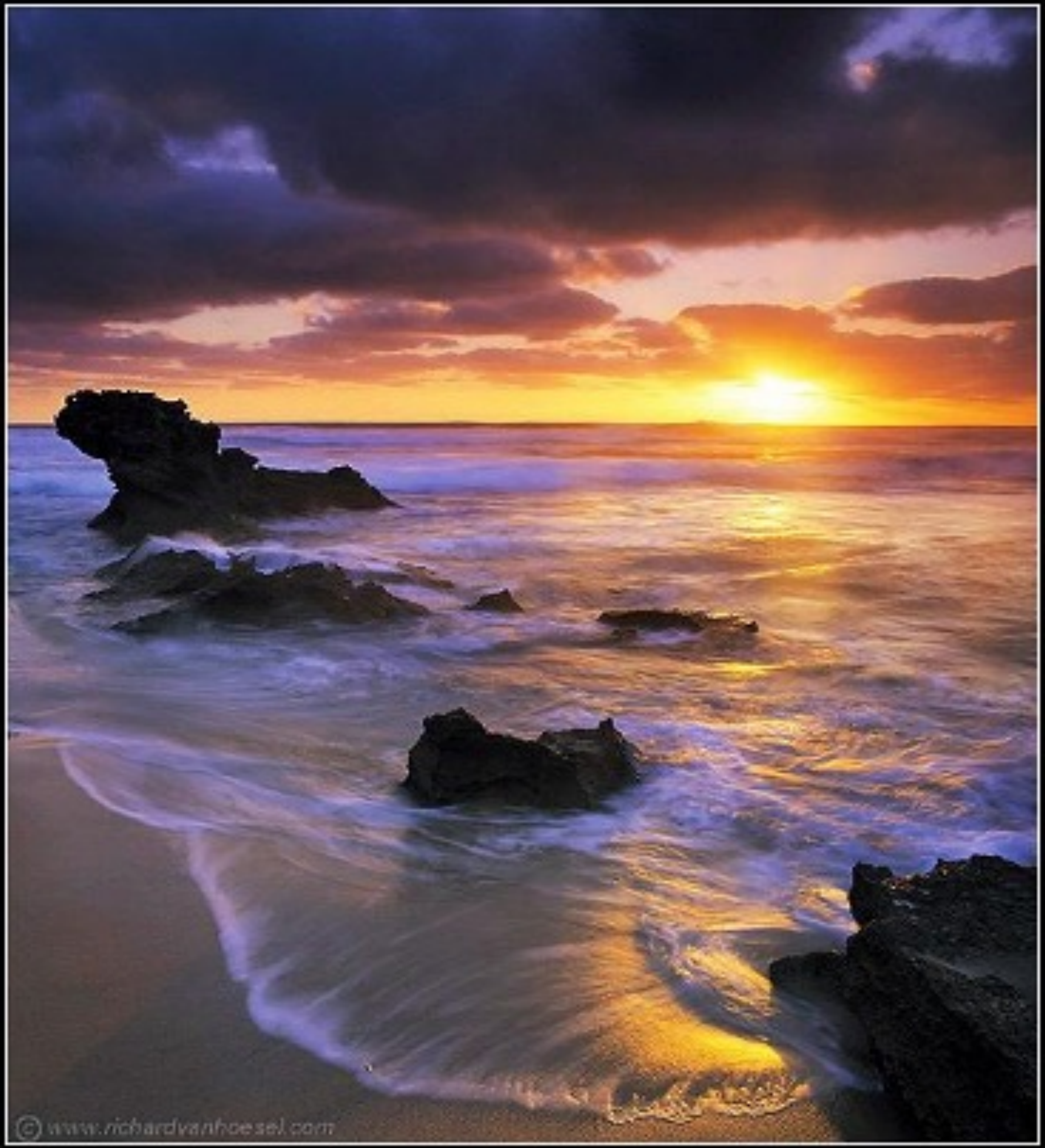
上海科技大学
ShanghaiTech University

# Light and Shading

CS172 Computer Vision I

Instructor: Jiayuan Gu




What determines a pixel’s intensity?

What can we infer about the scene from pixel intensities?

Image Source: A. Efros

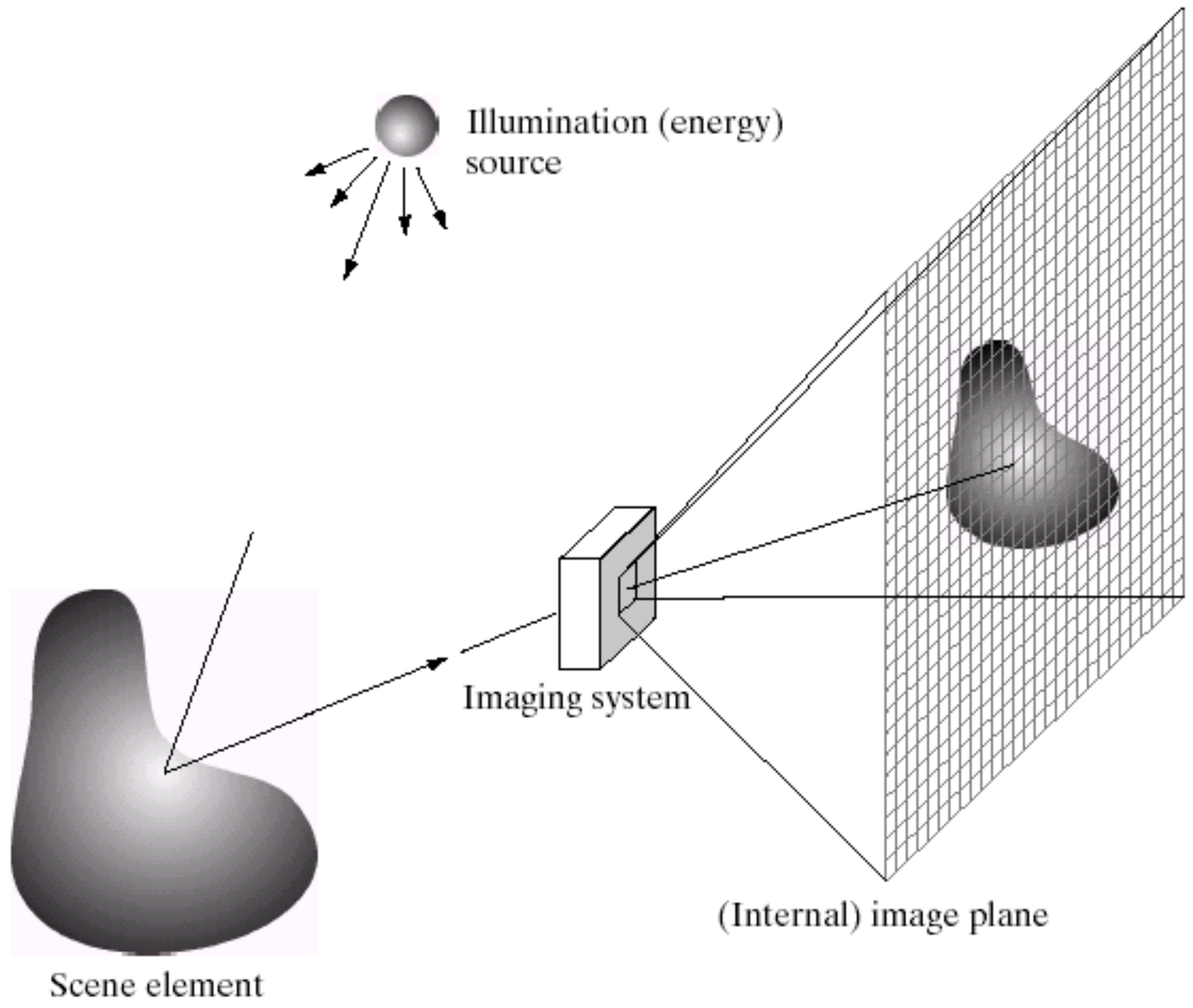
# Agenda

- Light Transport
  - Radiance, irradiance
  - Radiometric relation
  - Image sensing pipeline of digital camera
- Reflectance and Shading
- Photometric Stereo

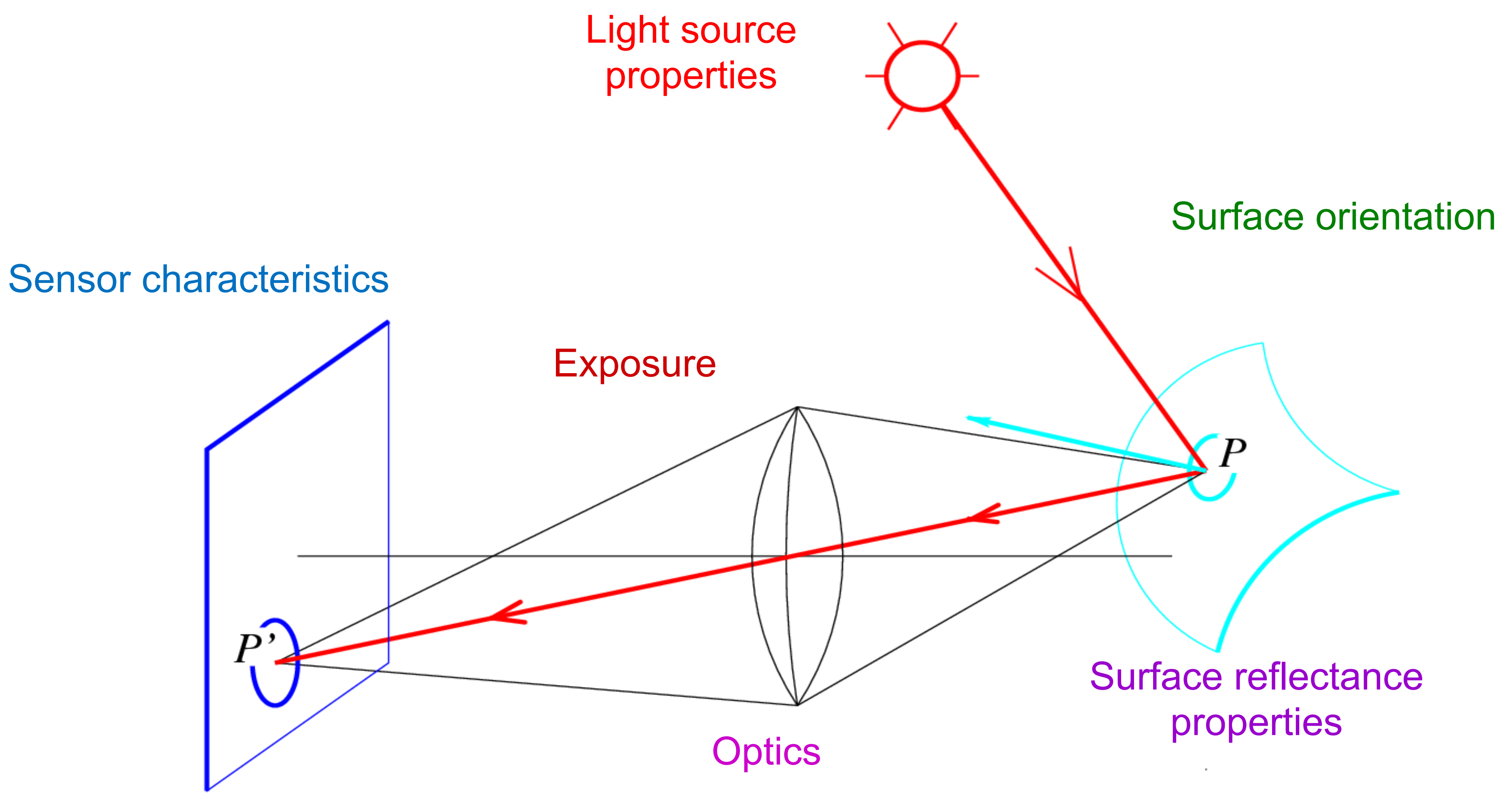
# Light Transport

From light to pixel values

# Review: Image Formation



# What determines the brightness of an image pixel?



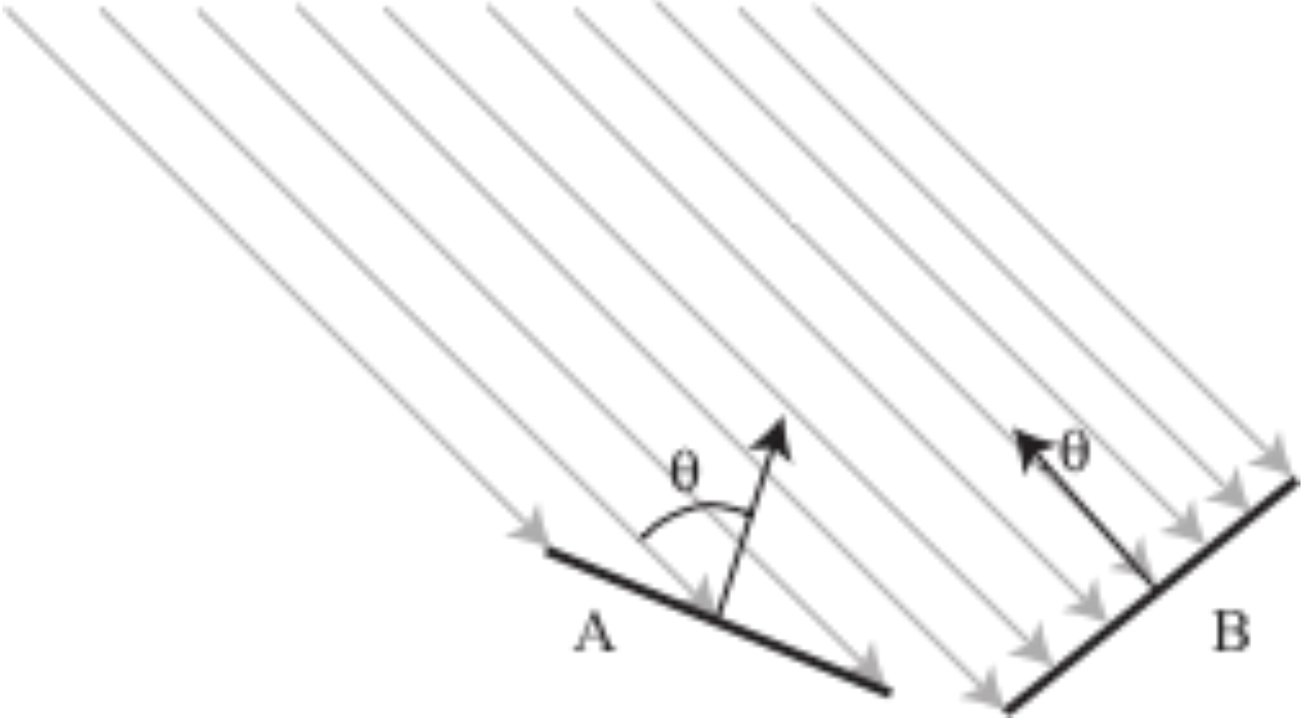

Credit to Fei-Fei Li

# Computing Reflected Intensity

$$I(x) = \rho(x)(\boldsymbol{S} \cdot \boldsymbol{N}(x))$$

Lambert's Law

$\rho$: albedo
$\boldsymbol{S}$: light source direction
$\boldsymbol{N}$: surface normal
$\mathrm{I}$: reflected intensity

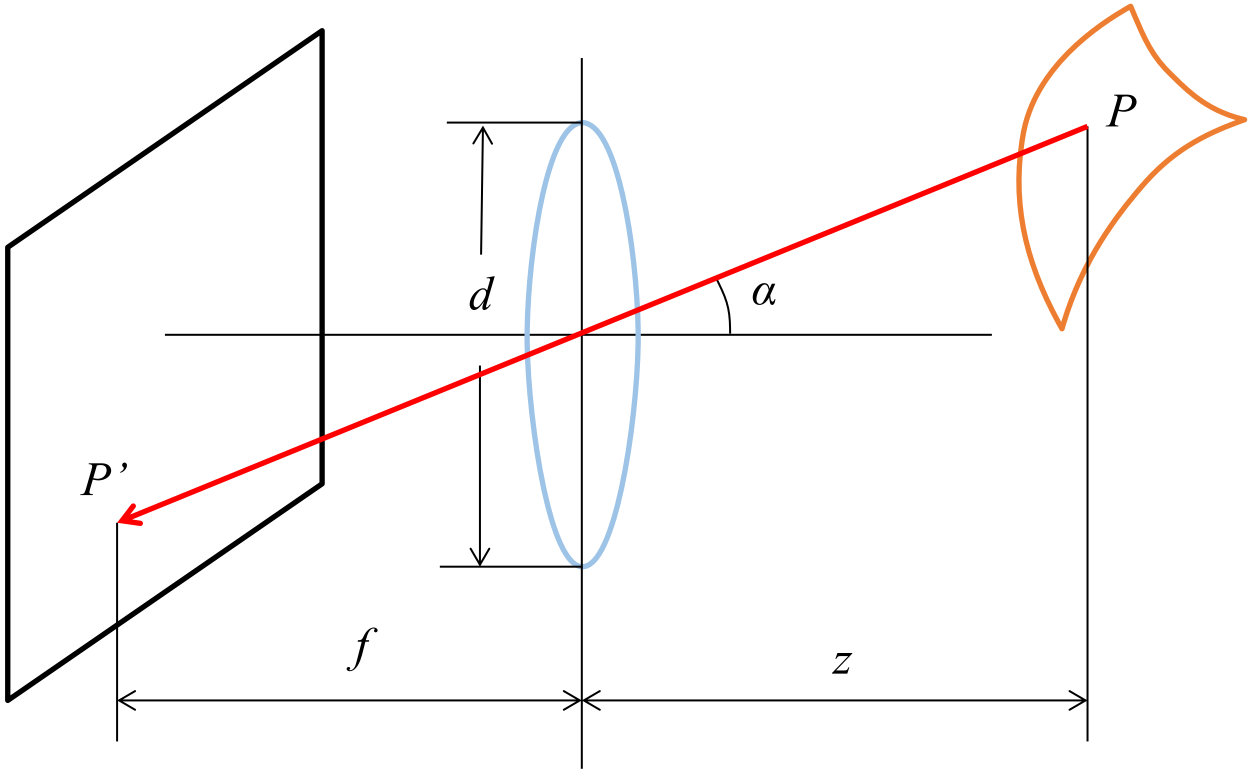



Intensity and surface orientation

Credit to David Forsyth

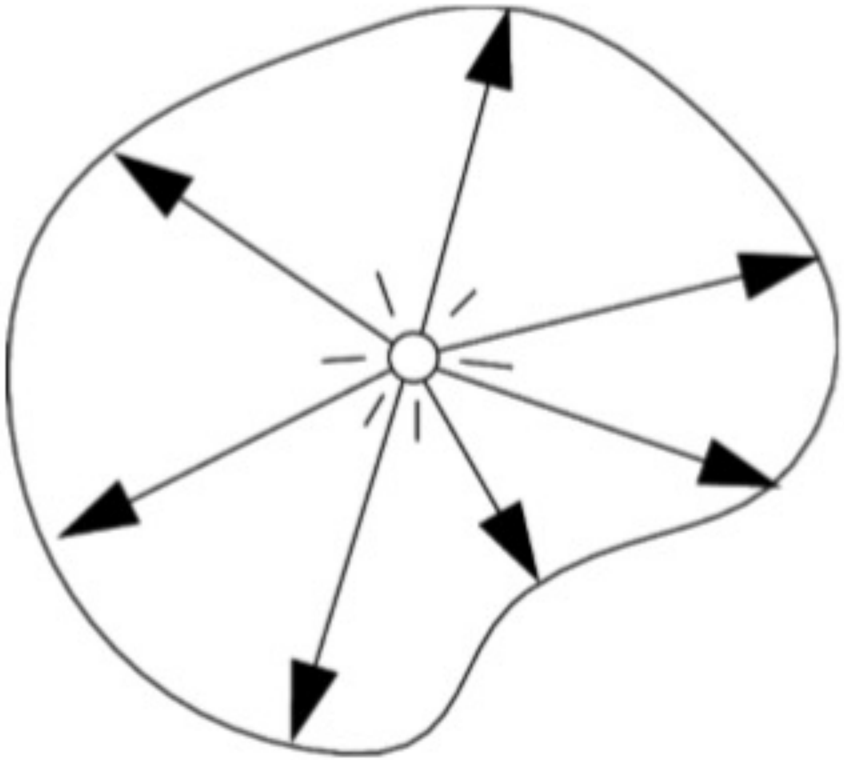
# Fundamental Radiometric Relation

- *L*: *Radiance* emitted from *P* toward *P*'
  - Energy carried by a ray
  - Unit: Watts per square meter per steradian
- *E*: *Irradiance* falling on *P*' from the lens
  - Energy arriving at a surface
  - Unit: Watts per square meter



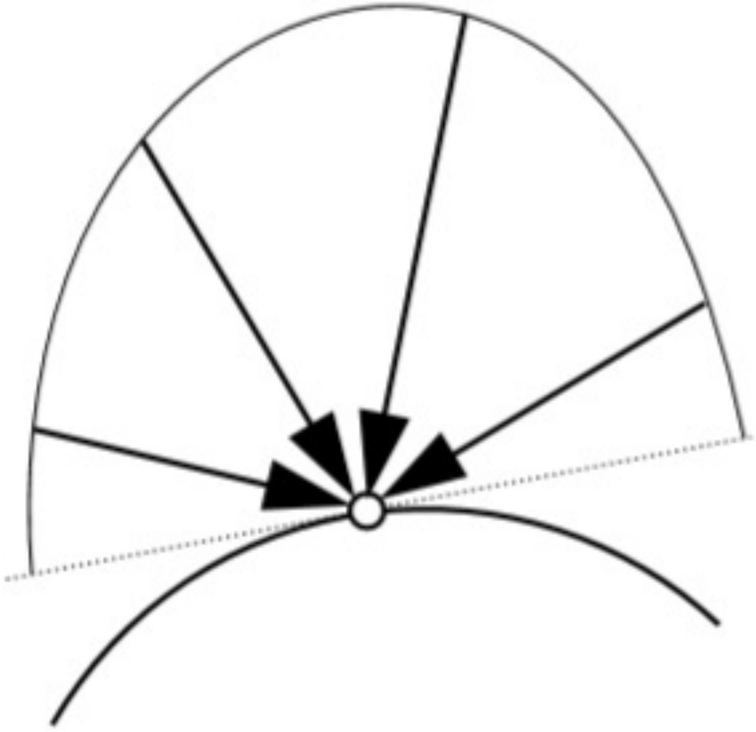

What is the relationship between *E* and *L*?

Szeliski, Ch 2.2.3



Light Emitted
From A Source

"Radiant Intensity"



Light Falling
On A Surface

"Irradiance"

Light Traveling
Along A Ray

"Radiance"

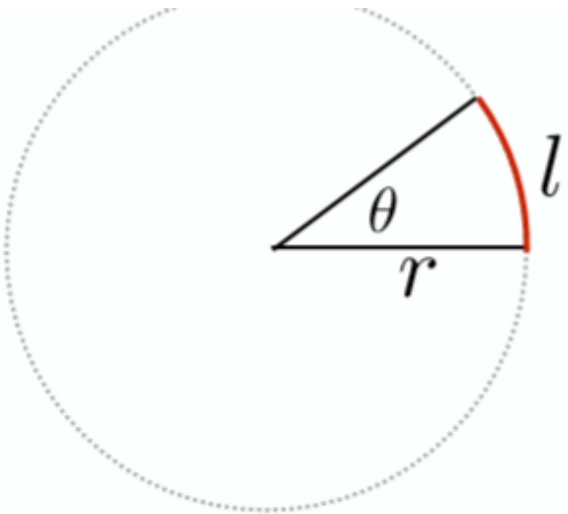
**Self-reading**

From [Zhihu blog]

# Steradian: Unit of Solid Angle

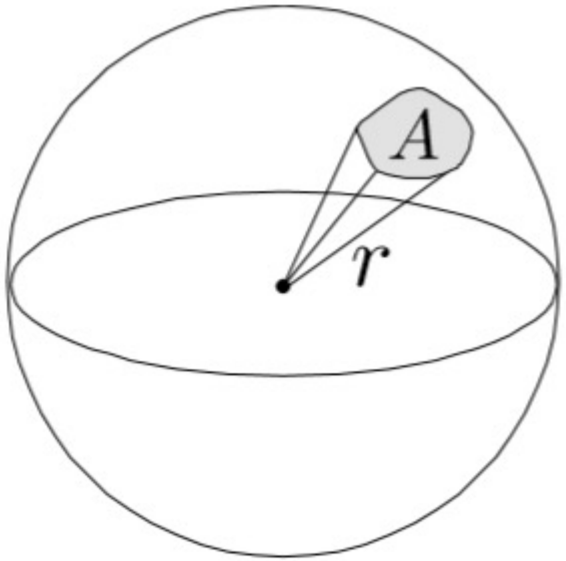
**Angle: ratio of subtended arc length on circle to radius**

- $\theta = \frac{l}{r}$
- **Circle has** $2\pi$ **radians**




**Solid angle: ratio of subtended area on sphere to radius squared**

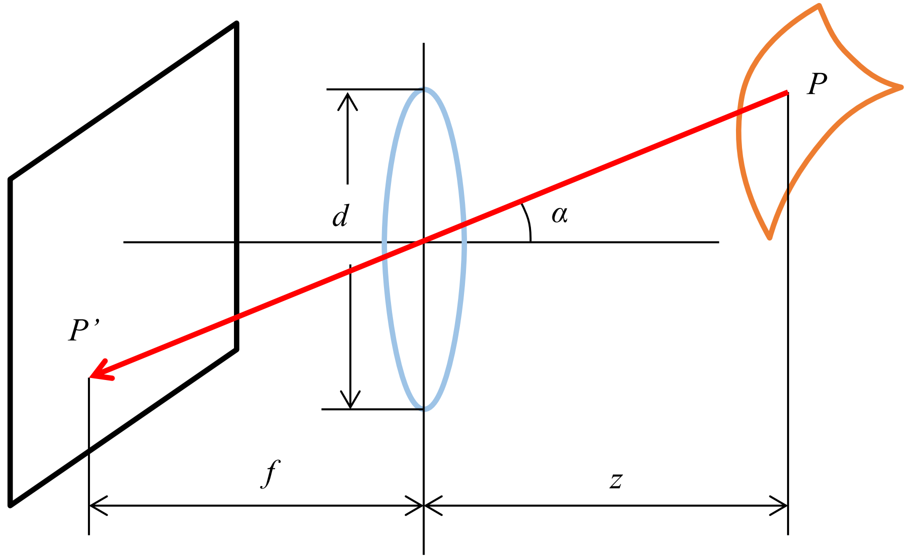
- $\Omega = \frac{A}{r^2}$
- **Sphere has** $4\pi$ **steradians**




Self-reading

From Zhihu blog

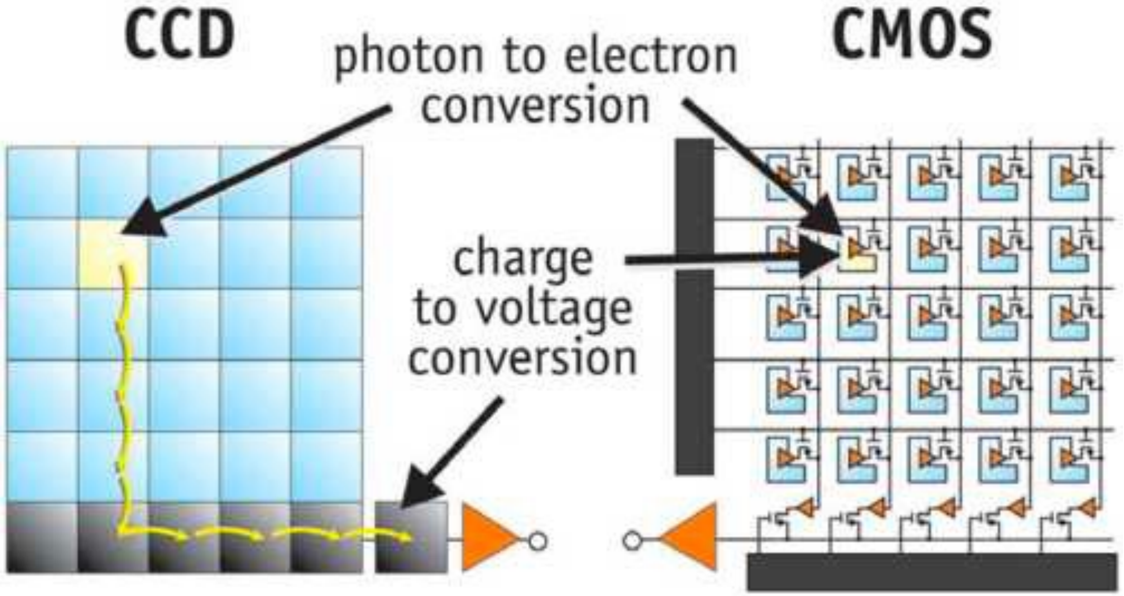
# Fundamental Radiometric Relation




$$E = \left[\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4\alpha\right] L$$

- Image irradiance is linearly related to scene radiance $L$
- Irradiance is proportional to the area of the lens and inversely proportional to the squared distance between the lens and the image plane
- Irradiance falls off as the angle $\alpha$ between the viewing ray and the optical axis increases (natural vignetting)

# Digital Camera


## A digital camera replaces film with a sensor array

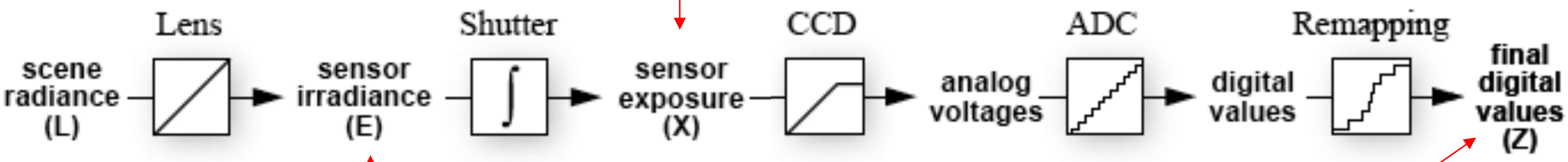
- Each cell in the array is light-sensitive diode that converts photons to electrons
- Two common types: Charge Coupled Device (CCD) and complementary metal oxide on silicon (CMOS)

http://electronics.howstuffworks.com/digital-camera.htm

Credit to Steve Seitz

# From Light Rays to Pixel Values

$$\boxed{X = E \cdot \Delta t}$$

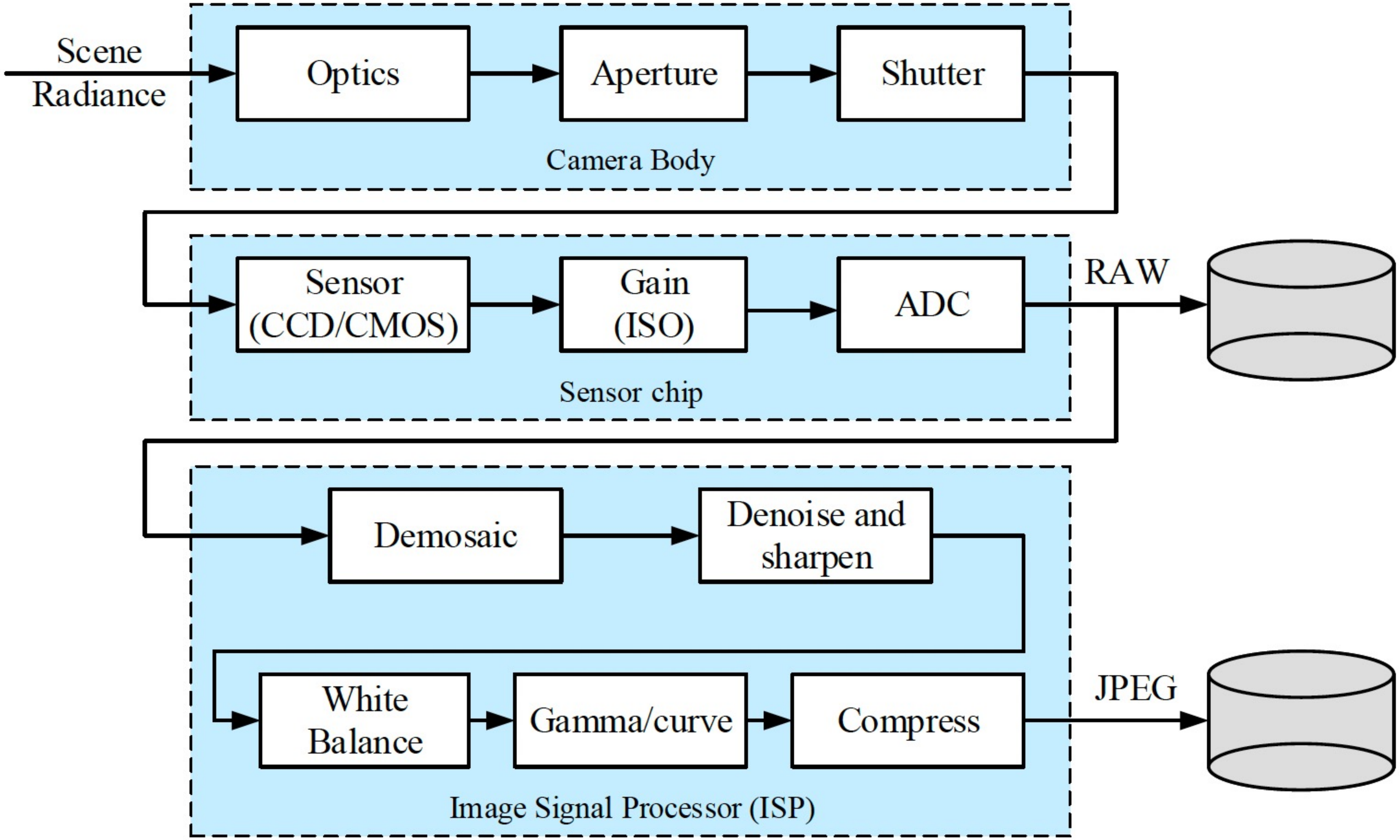



$$\boxed{E = \left[ \frac{\pi}{4} \left( \frac{d}{f} \right)^2 \cos^4 \alpha \right] L}$$

$$\boxed{Z = f\left(E \cdot \Delta t\right)}$$

- Camera response function: the mapping $f$ from irradiance to pixel values
  - Enables us to create high dynamic range images
  - Useful if we want to estimate material properties
  - For more info: P. E. Debevec and J. Malik, [*Recovering High Dynamic Range Radiance Maps from Photographs*](), *SIGGRAPH 97*

# Image Sensing Pipeline

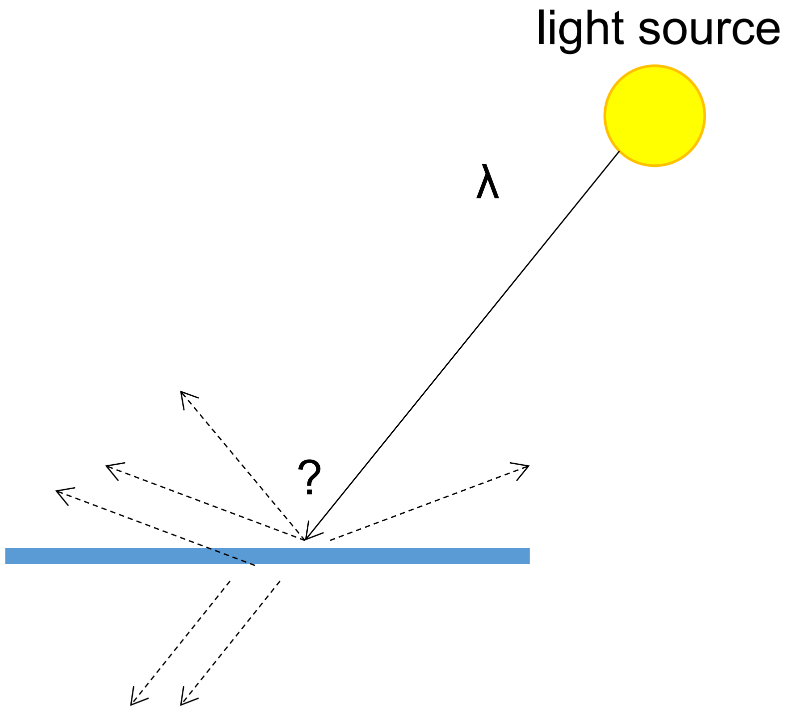



Szeliski, Ch 2.3

# Reflectance and Shading

# A Photon's Life Choices

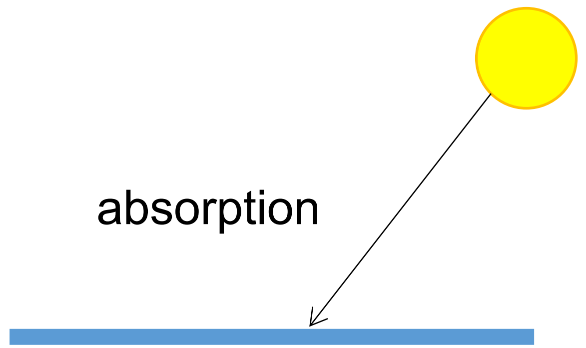
- Absorption
- Diffusion
- Reflection
- Transparency
- Refraction
- Fluorescence
- Phosphorescence
- Subsurface scattering
- Interreflection

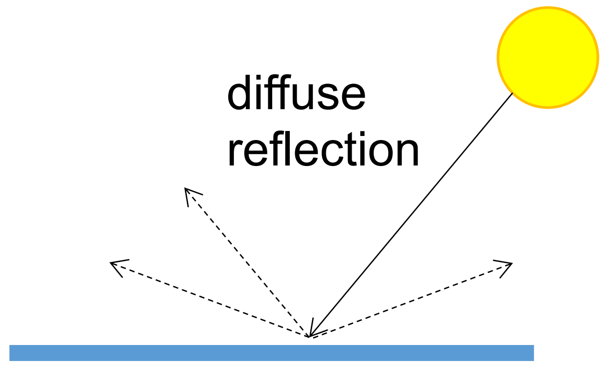


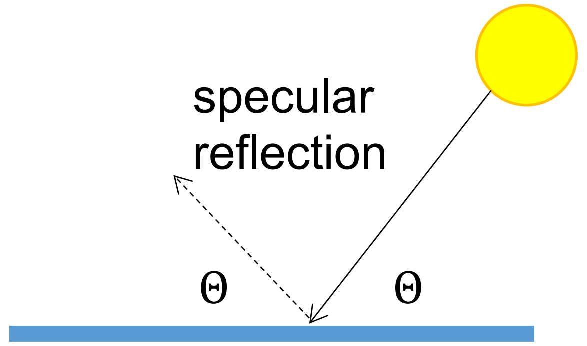
# Common Effects

When light hits a typical surface

- Some light is absorbed ($1-\rho$)
  - More absorbed for low albedos
- Some light is reflected diffusely
  - Independent of viewing direction
- Some light is reflected specularly
  - Light bounces off (like a mirror), depends on viewing direction



# Bidirectional Reflectance Distribution Function (BRDF)

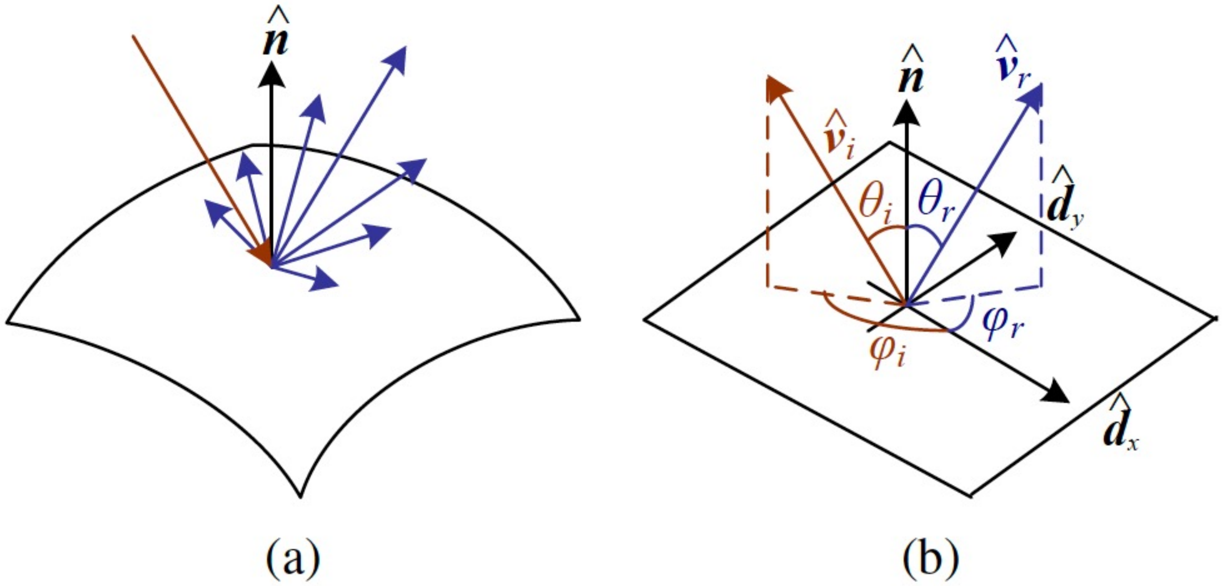
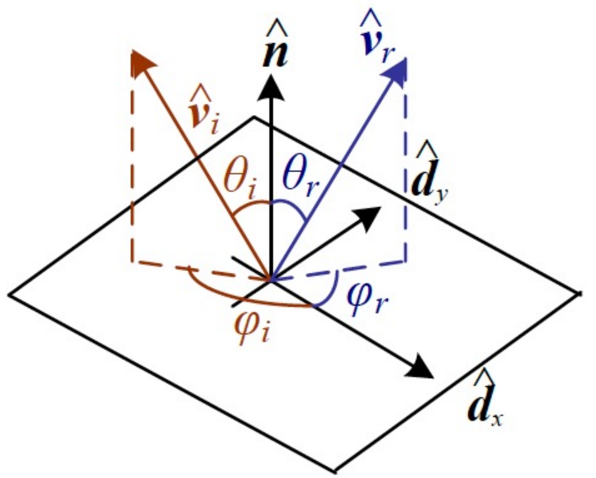



**Figure 2.15** *(a) Light scatters when it hits a surface. (b) The bidirectional reflectance distribution function (BRDF) $f(\theta_i, \phi_i, \theta_r, \phi_r)$ is parameterized by the angles that the incident, $\hat{\mathbf{v}}_i$, and reflected, $\hat{\mathbf{v}}_r$, light ray directions make with the local surface coordinate frame* $(\hat{\mathbf{d}}_x, \hat{\mathbf{d}}_y, \hat{\mathbf{n}})$.

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r; \lambda)$$

Szeliski, Ch 2.2.2

# Bidirectional Reflectance Distribution Function (BRDF)




The amount of light exiting a surface point in a direction $\hat{v}_r$

$$L_r(\hat{\mathbf{v}}_r;\lambda) = \int L_i(\hat{\mathbf{v}}_i;\lambda) f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}};\lambda) \cos^+\theta_i \, d\hat{\mathbf{v}}_i$$

incoming light

foreshortening factor

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r; \lambda)$$

$$\cos^+\theta_i = \max(0, \cos\theta_i)$$

$f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}};\lambda)$ (isotropic surface)

For a finite number of (point) light sources

$\hat{v}_i$: incident direction
$\hat{v}_r$: reflected direction
$\hat{n}$: normal direction
$\lambda$: wavelength

$$L_r(\hat{\mathbf{v}}_r;\lambda) = \sum_i L_i(\lambda) f_r(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}};\lambda) \cos^+\theta_i$$

Szeliski, Ch 2.2.2

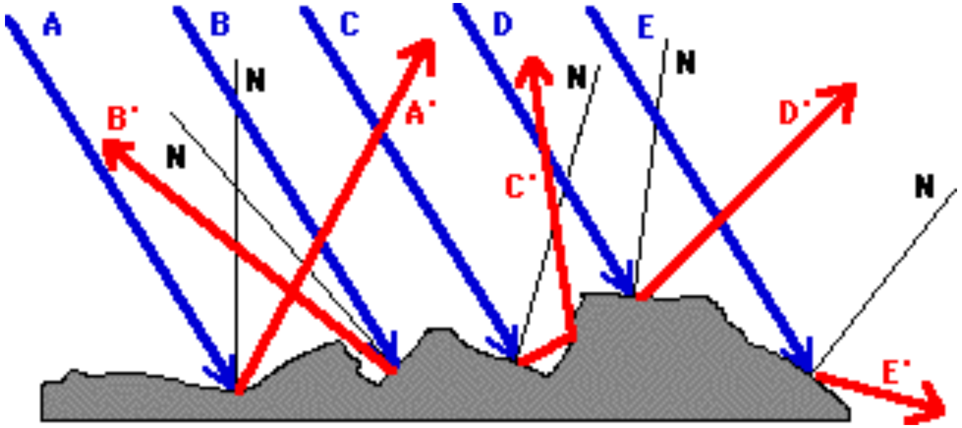
# Diffusion Reflection

$$f_d(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_r, \hat{\mathbf{n}}; \lambda) = f_d(\lambda)$$

The BRDF is constant

$$L_d(\hat{\mathbf{v}}_r; \lambda) = \sum_i L_i(\lambda) f_d(\lambda) \cos^+ \theta_i = \sum_i L_i(\lambda) f_d(\lambda) [\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}]^+$$

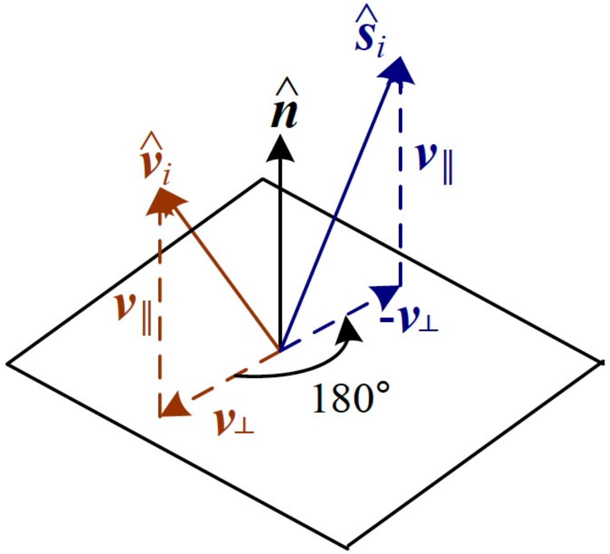
Shading equation




Why Does a Rough Surface Diffuse A Beam of Light?

https://www.physicsclassroom.com/class/refln/lesson-1/specular-vs-diffuse-reflection

# Specular Reflection




specular reflection direction

$$\hat{\mathbf{s}}_i = \mathbf{v}_\parallel - \mathbf{v}_\perp = (2\hat{\mathbf{n}}\hat{\mathbf{n}}^T - \mathbf{I})\mathbf{v}_i.$$

The Phong model uses a power of the cosine of the angle

$$f_s(\theta_s;\lambda) = k_s(\lambda)\cos^{k_e}\theta_s$$

The Torrance and Sparrow micro-facet model uses a Guassian

$$f_s(\theta_s;\lambda) = k_s(\lambda)\exp(-c_s^2\theta_s^2)$$

# Phong Shading and Ambient Illumination


Photo credit: Jessica Andrews, flickr

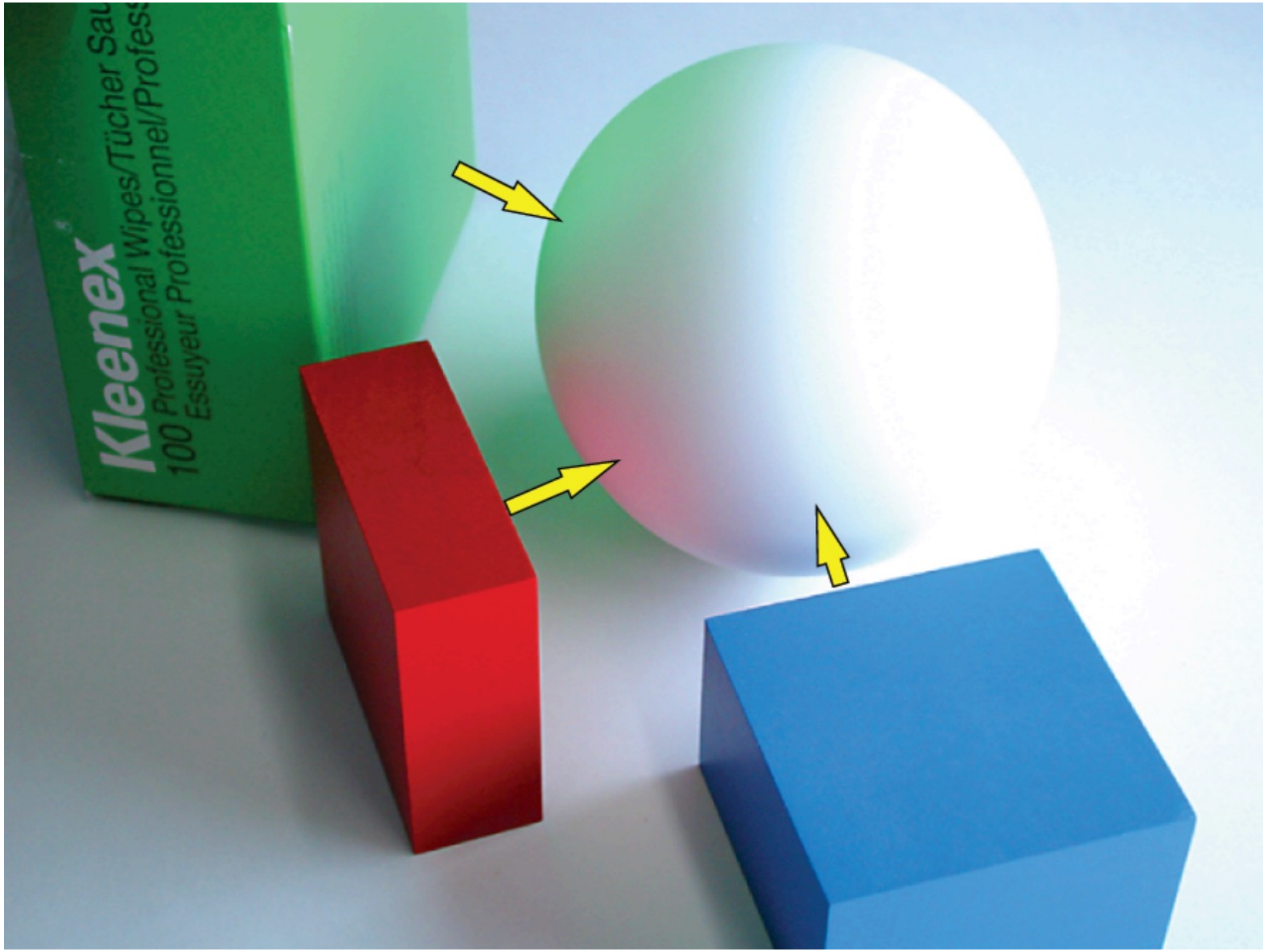
Why is the back of cup illuminated?

$$f_a(\lambda) = k_a(\lambda) L_a(\lambda)$$

$$L_r(\hat{\mathbf{v}}_r; \lambda) = k_a(\lambda) L_a(\lambda) + k_d(\lambda) \sum_i L_i(\lambda) [\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}]^+ + k_s(\lambda) \sum_i L_i(\lambda) (\hat{\mathbf{v}}_r \cdot \hat{\mathbf{s}}_i)^{k_e}$$

Phong Shading Model

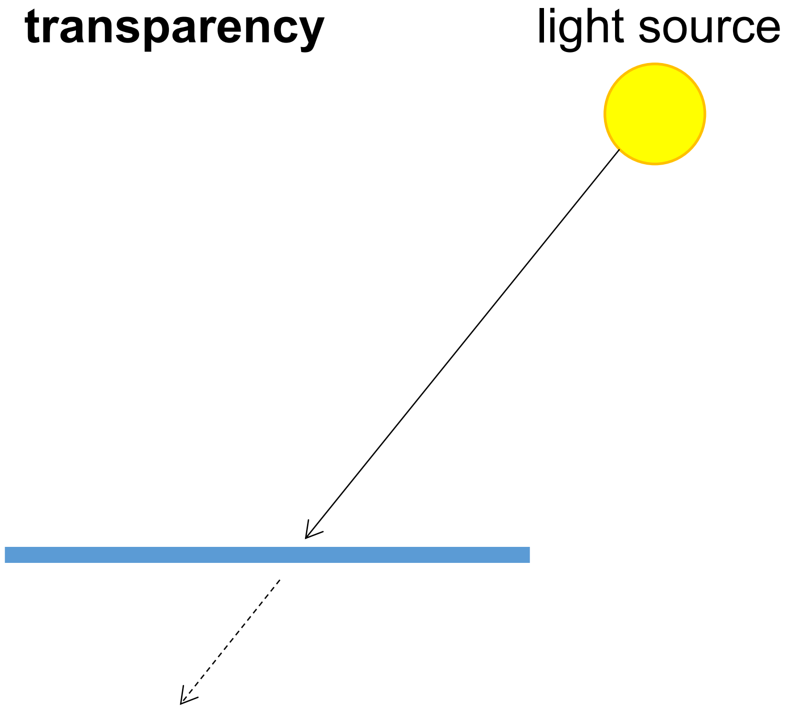
# Interreflection



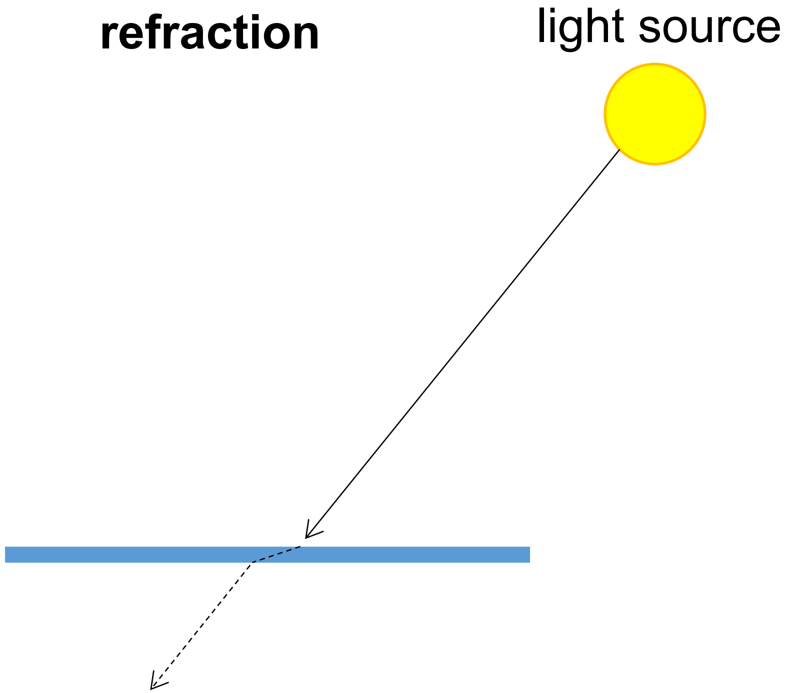

Credit to Koenderink and David Forsyth

# Other Effects

**transparency**

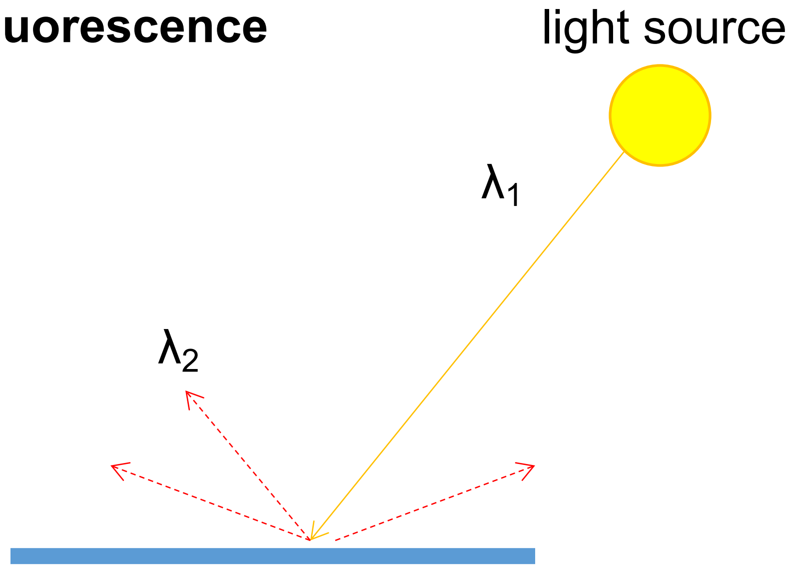



**refraction**

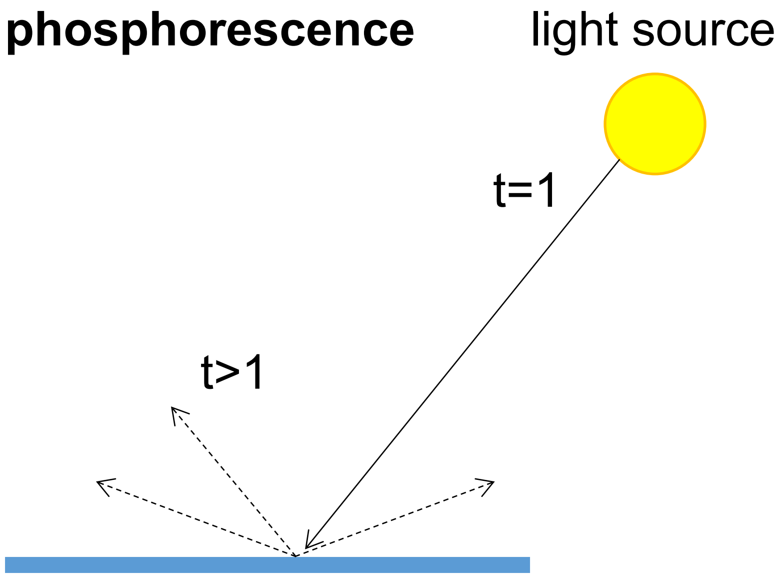


# Other Effects

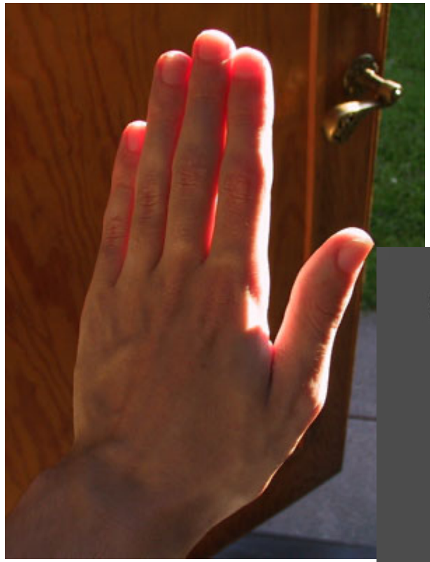
**fluorescence**




**phosphorescence**



# Other Effects



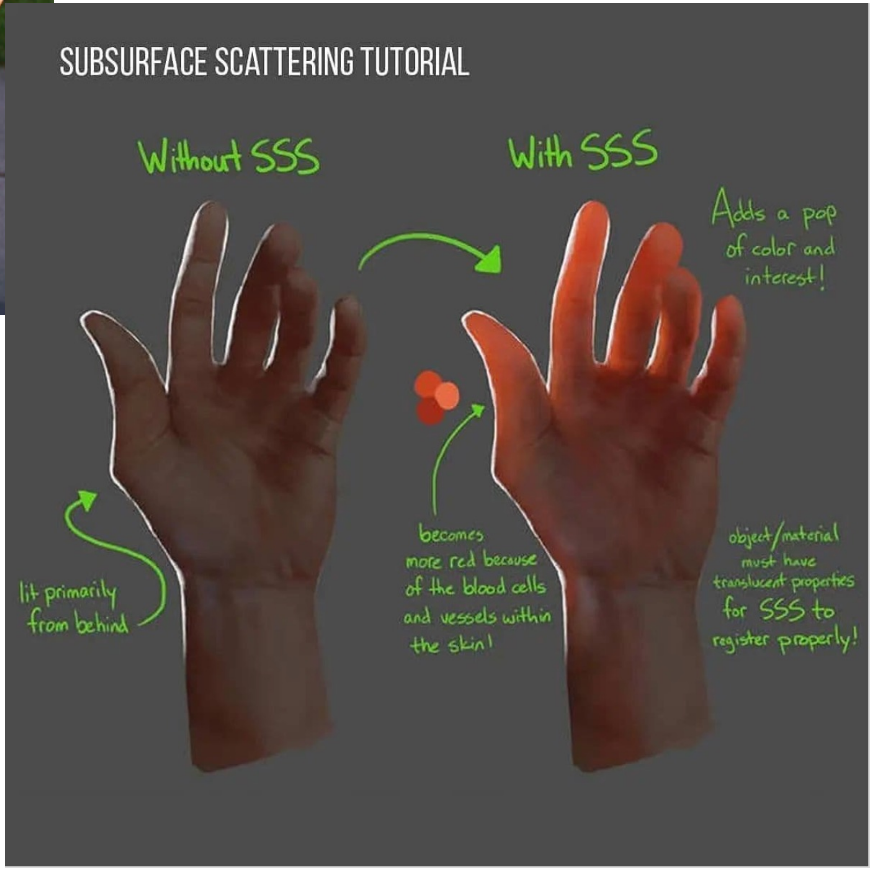
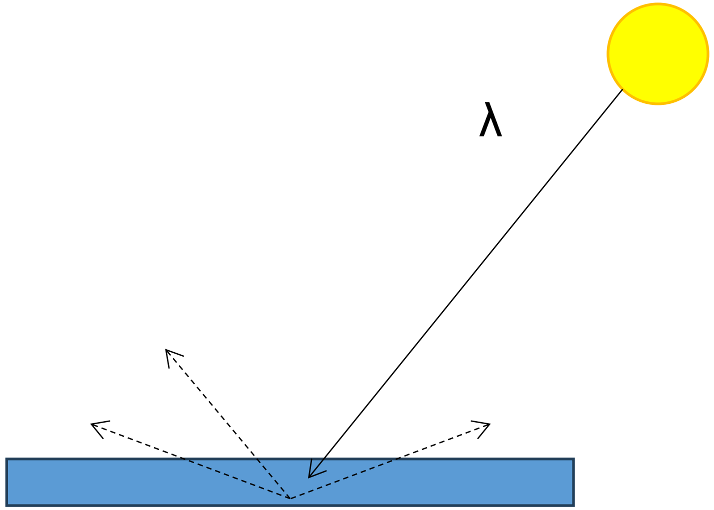
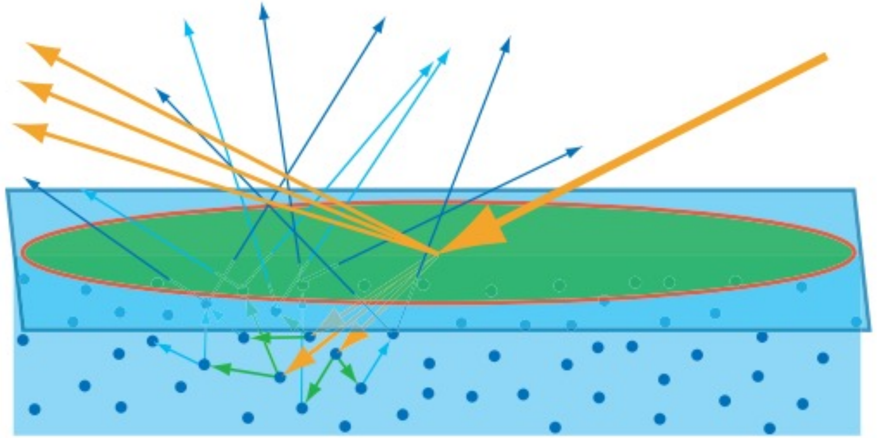



Figure from post

**subsurface scattering (SSS)**

light source



https://therealmjp.github.io/posts/sss-intro/

# Photometric Stereo

Shape from Shading

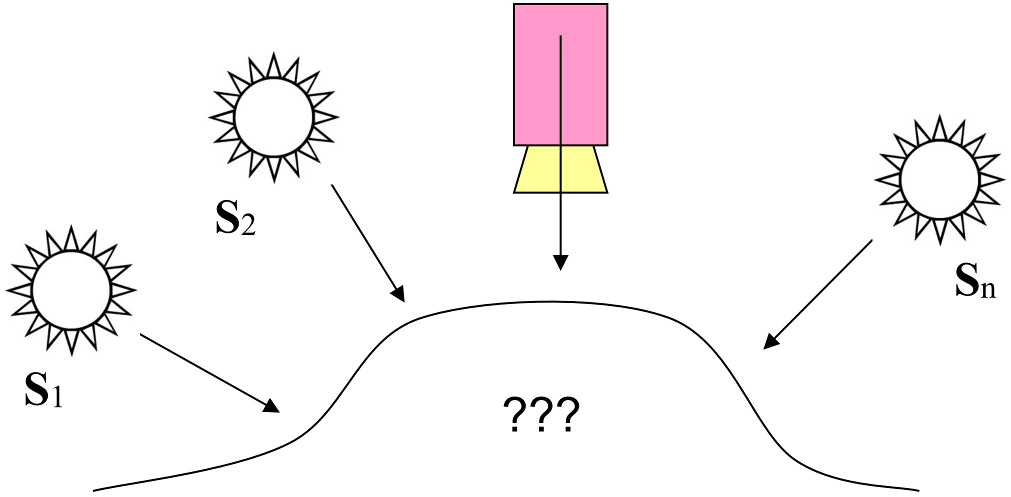
# Shape from Shading

Luca della Robbia, *Cantoria*, 1438

Can we reconstruct the shape of an object based on shading cues?

# Photometric stereo

- Assume:
  - A *Lambertian* object
  - A *local shading model*
    - Each point on a surface receives light only from sources visible at that point
  - A set of *known* light source directions
  - A set of pictures of an object, obtained in exactly the same camera/object configuration but using different sources
- Goal: reconstruct object shape and albedo



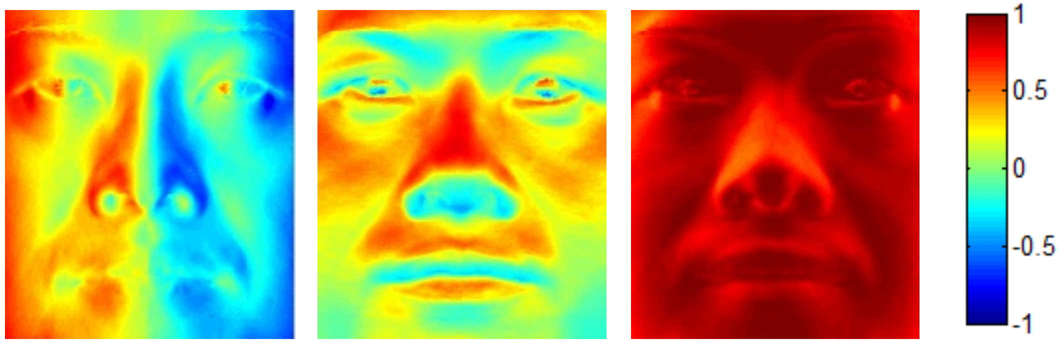

F&P 2nd edition, sec. 2.2.4

# Example

Input

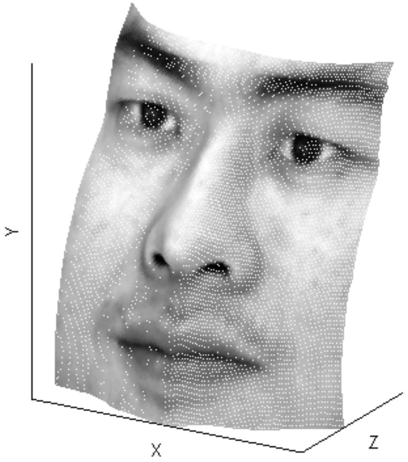
Recovered albedo

Recovered normal field



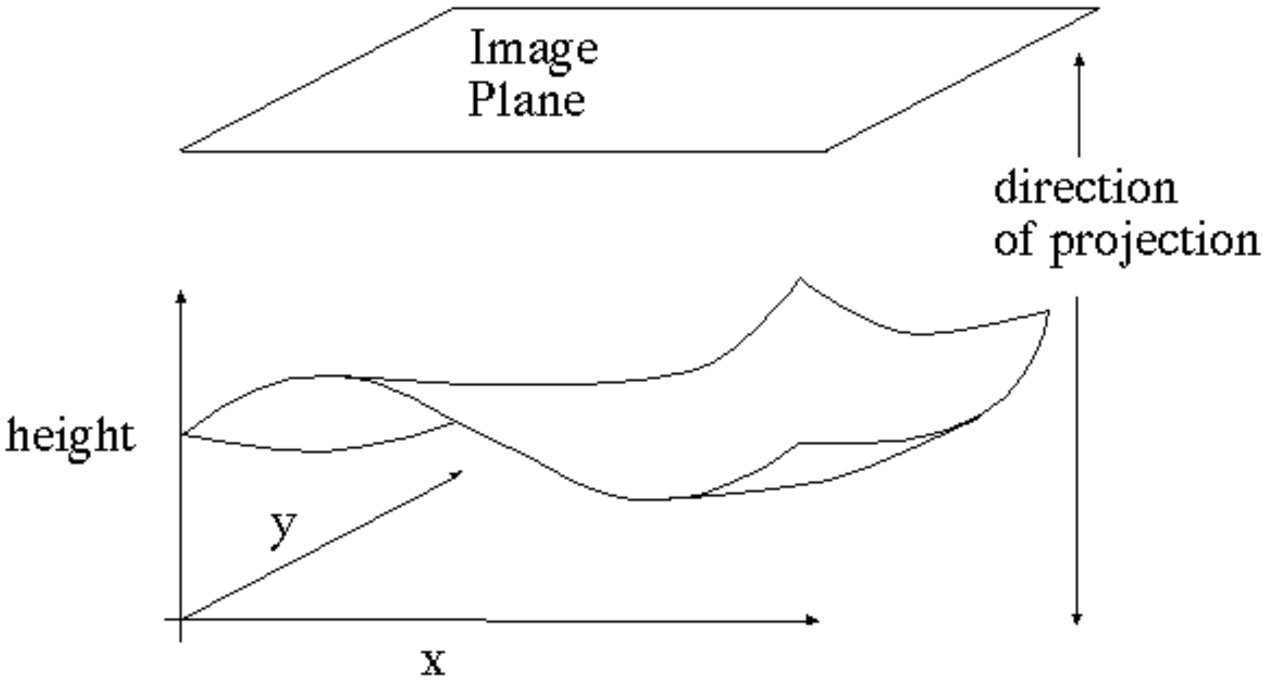

x y z

Recovered surface model



# Image model

- **Known:** source vectors $\boldsymbol{S}_j$ and pixel values $\boldsymbol{I}_j(x, y)$
- **Unknown:** surface normal $\boldsymbol{N}(x, y)$ and albedo $\rho(x, y)$




F&P 2nd ed., sec. 2.2.4

# Image model

- **Known:** source vectors $\boldsymbol{S}_j$ and pixel values $\boldsymbol{I}_j(x, y)$
- **Unknown:** surface normal $\boldsymbol{N}(x, y)$ and albedo $\rho(x, y)$
- Assume that the response function of the camera is a linear scaling by a factor of $k$
- Lambert's law:

$$
\begin{aligned}
I_j(x, y) &= k\,\rho(x, y)\left(\mathbf{N}(x, y) \cdot \mathbf{S}_j\right) \\
&= \left(\rho(x, y)\mathbf{N}(x, y)\right) \cdot (k\mathbf{S}_j) \\
&= \mathbf{g}(x, y) \cdot \mathbf{V}_j
\end{aligned}
$$

F&P 2nd ed., sec. 2.2.4

# Least squares problem

For each pixel, set up a linear system:

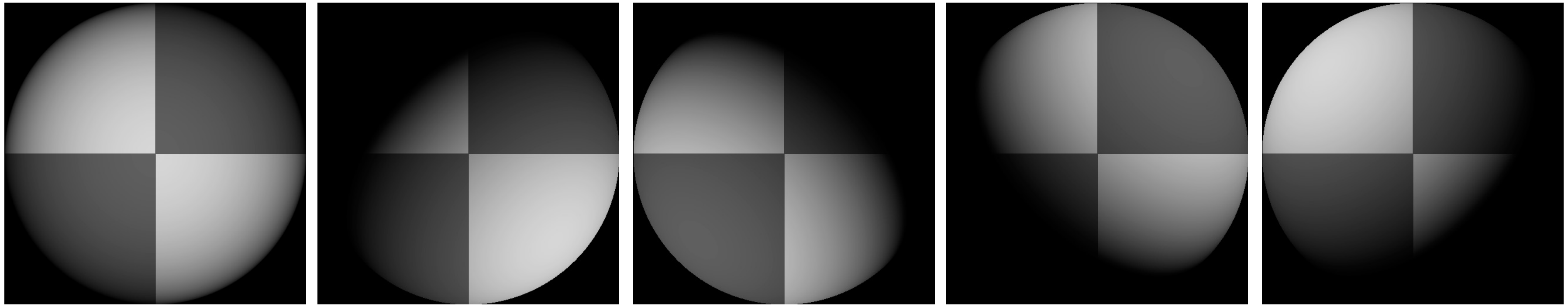
$$\begin{bmatrix} I_1(x,y) \\ I_2(x,y) \\ \vdots \\ I_n(x,y) \end{bmatrix} = \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \\ \vdots \\ \mathbf{V}_n^T \end{bmatrix} \mathbf{g}(x,y)$$

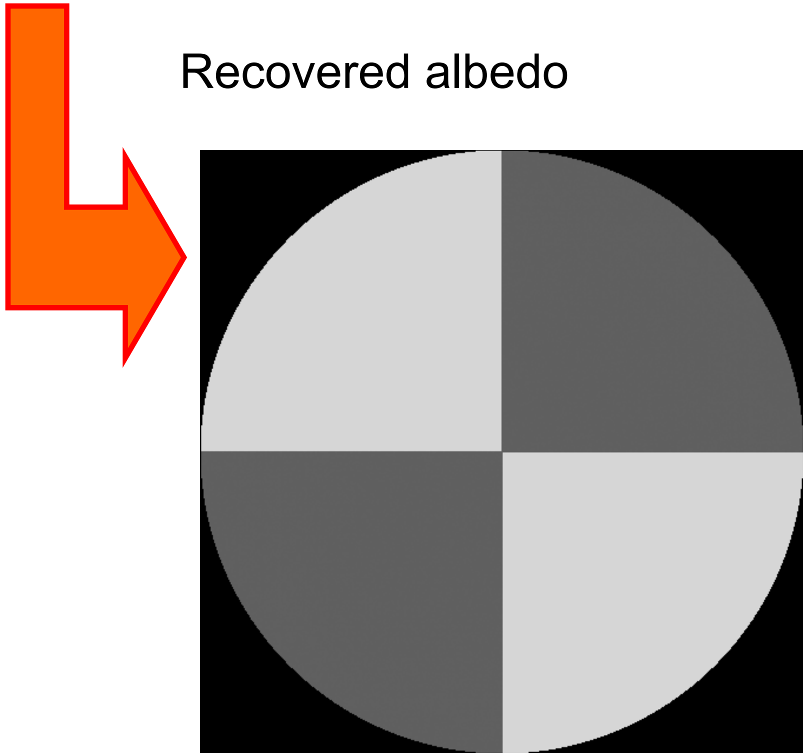
| $(n \times 1)$ | $(n \times 3)$ | $(3 \times 1)$ |
|---|---|---|
| known | known | unknown |

- Obtain least-squares solution for $\boldsymbol{g}(x,y)$, which we defined as $\rho(x,y)\boldsymbol{N}(x,y)$
- Since $\boldsymbol{N}(x,y)$ is the unit normal, $\rho(x,y)$ is given by the magnitude of $\boldsymbol{g}(x,y)$
- Finally, $\boldsymbol{N}(x,y) = \boldsymbol{g}(x,y)/\rho(x,y)$

F&P 2nd ed., sec. 2.2.4

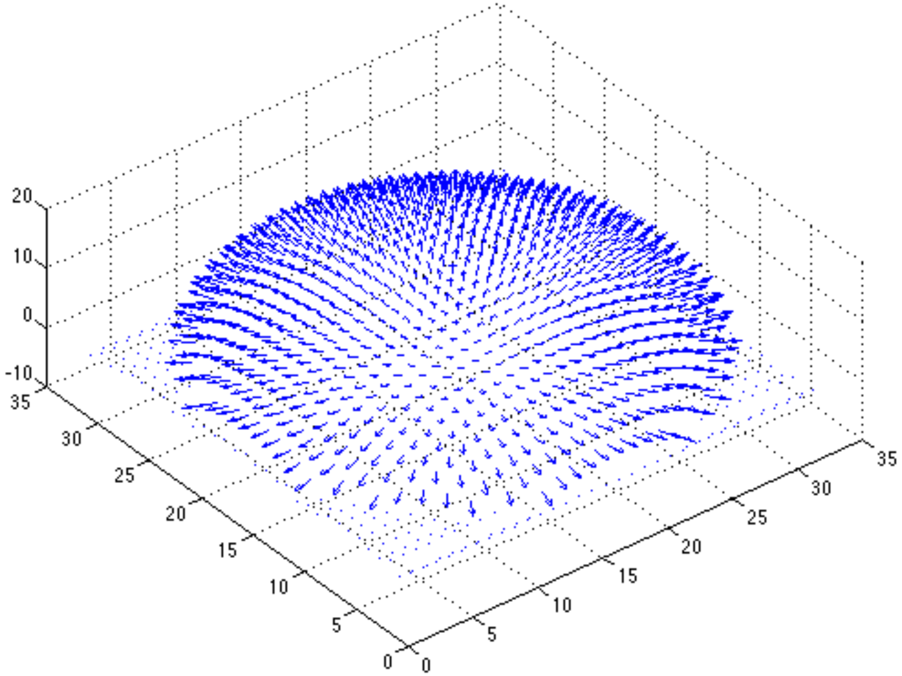
# Synthetic Example





Recovered albedo

Recovered normal field




F&P 2nd ed., sec. 2.2.4

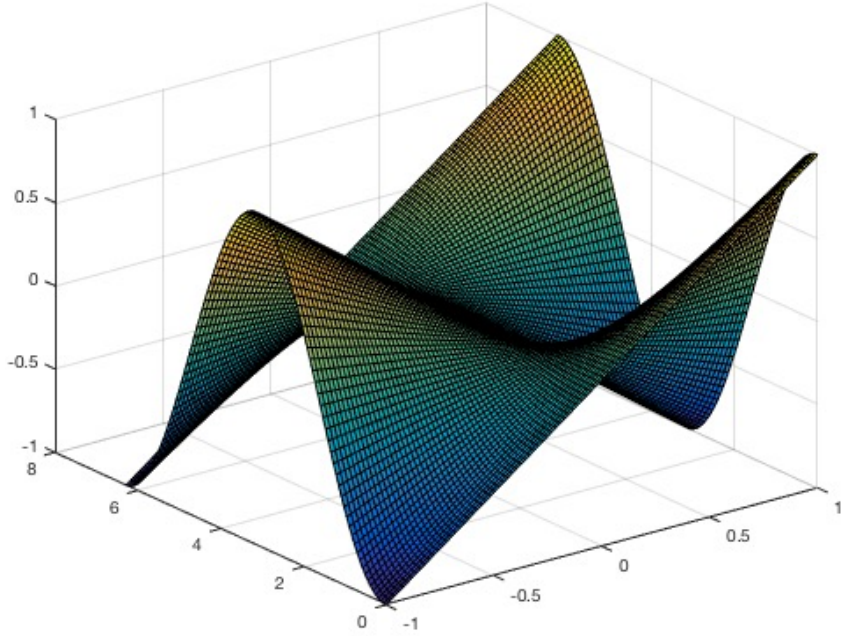
# Recovering a Surface from Normals

- Recall: the surface is written as $\left(x, y, f(x, y)\right)$

- The tangent plane is spanned by $\left(1, 0, \frac{\partial f}{\partial x}\right)$ and $\left(0, 1, \frac{\partial f}{\partial y}\right)$

- The normal is orthogonal to the tangent plane and a unit vector,
- so it is the normalized version of $\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, 1\right)$

- that is $\boldsymbol{N} = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, 1\right) \frac{1}{\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 + 1}}$



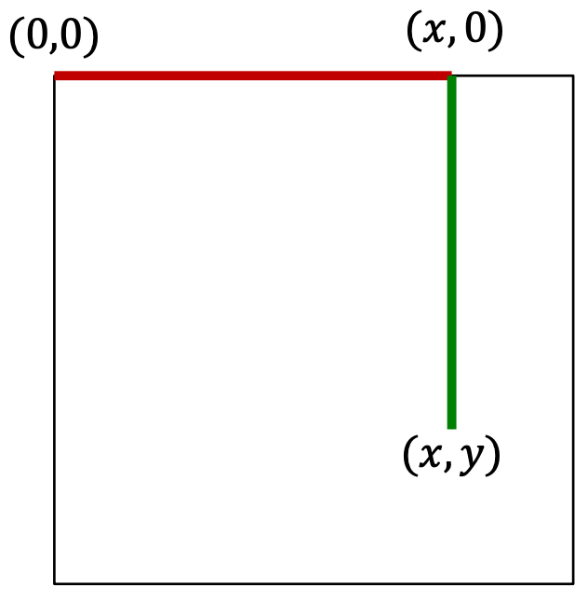
# Recovering a Surface from Normals

- Recall: the estimated vector g is written as $g(x,y) = \left(g_1(x,y), g_2(x,y), g_3(x,y)\right)$

- And we have already known $\boldsymbol{N} = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, 1\right) \frac{1}{\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 + 1}}$

- Thus, we have $\frac{\partial f}{\partial x} = \frac{g_1(x,y)}{g_3(x,y)}, \frac{\partial f}{\partial y} = \frac{g_2(x,y)}{g_3(x,y)}$

- Then, we can recover the height z by integration
  - $f(x,y) = \int_0^x \frac{\partial f}{\partial x}(u,0)du + \int_0^y \frac{\partial f}{\partial y}(x,v)dv + f(0,0)$



# Pseudo Codes

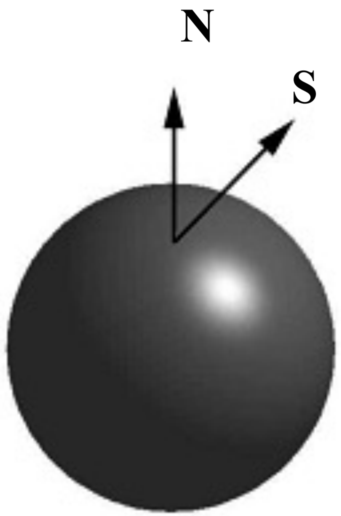
```
Input: Normal fields Nx, Ny, Nz
Output: Height map Z

# Normalize the normal vectors
Nx_normalized = Nx / Nz
Ny_normalized = Ny / Nz

# Initialize height map Z
Z = zeros_like(Nx)

# Perform numerical integration
for x in range(width):
    for y in range(height):
        if x > 0:
            Z[x, y] += Z[x-1, y] + Nx_normalized[x, y]
        if y > 0:
            Z[x, y] += Z[x, y-1] + Ny_normalized[x, y]
```

# Limitations

- Orthographic camera model
- Simplistic reflectance and lighting model
- No shadows
- No interreflections
- No missing data
- Integration is tricky

# Finding the Direction of the Light Source

$$\boxed{I(x,y) = \mathbf{N}(x,y) \cdot \mathbf{S}(x,y)}$$

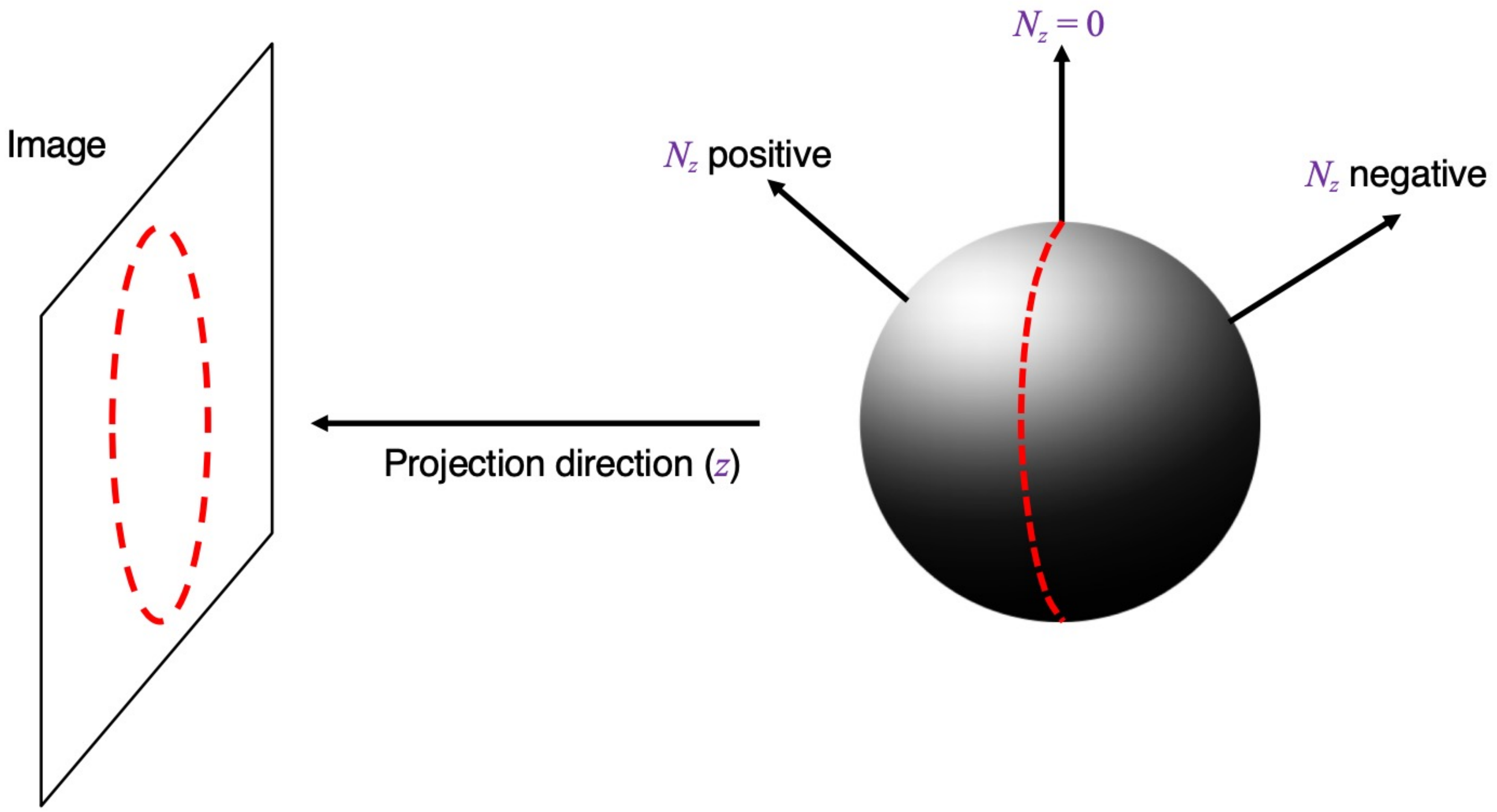



Full 3D case:

$$\begin{pmatrix} N_x(x_1,y_1) & N_y(x_1,y_1) & N_z(x_1,y_1) \\ N_x(x_2,y_2) & N_y(x_2,y_2) & N_z(x_2,y_2) \\ \vdots & \vdots & \vdots \\ N_x(x_n,y_n) & N_y(x_n,y_n) & N_z(x_n,y_n) \end{pmatrix} \begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix} = \begin{pmatrix} I(x_1,y_1) \\ I(x_2,y_2) \\ \vdots \\ I(x_n,y_n) \end{pmatrix}$$

P. Nillius and J.-O. Eklundh, "Automatic estimation of the projected light source direction," CVPR 2001

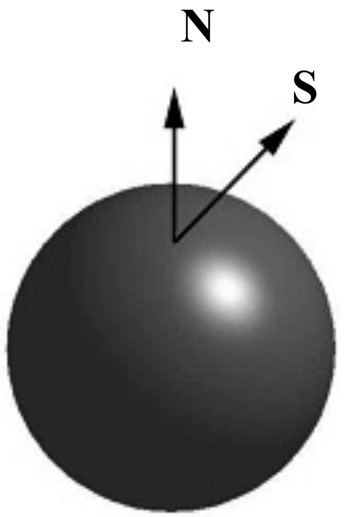
# Occluding Contour




P. Nillius and J.-O. Eklundh, “Automatic estimation of the projected light source direction,” CVPR 2001

# Finding the Direction of the Light Source

$\boxed{I(x,y) = \mathbf{N}(x,y) \cdot \mathbf{S}(x,y)}$




Full 3D case:

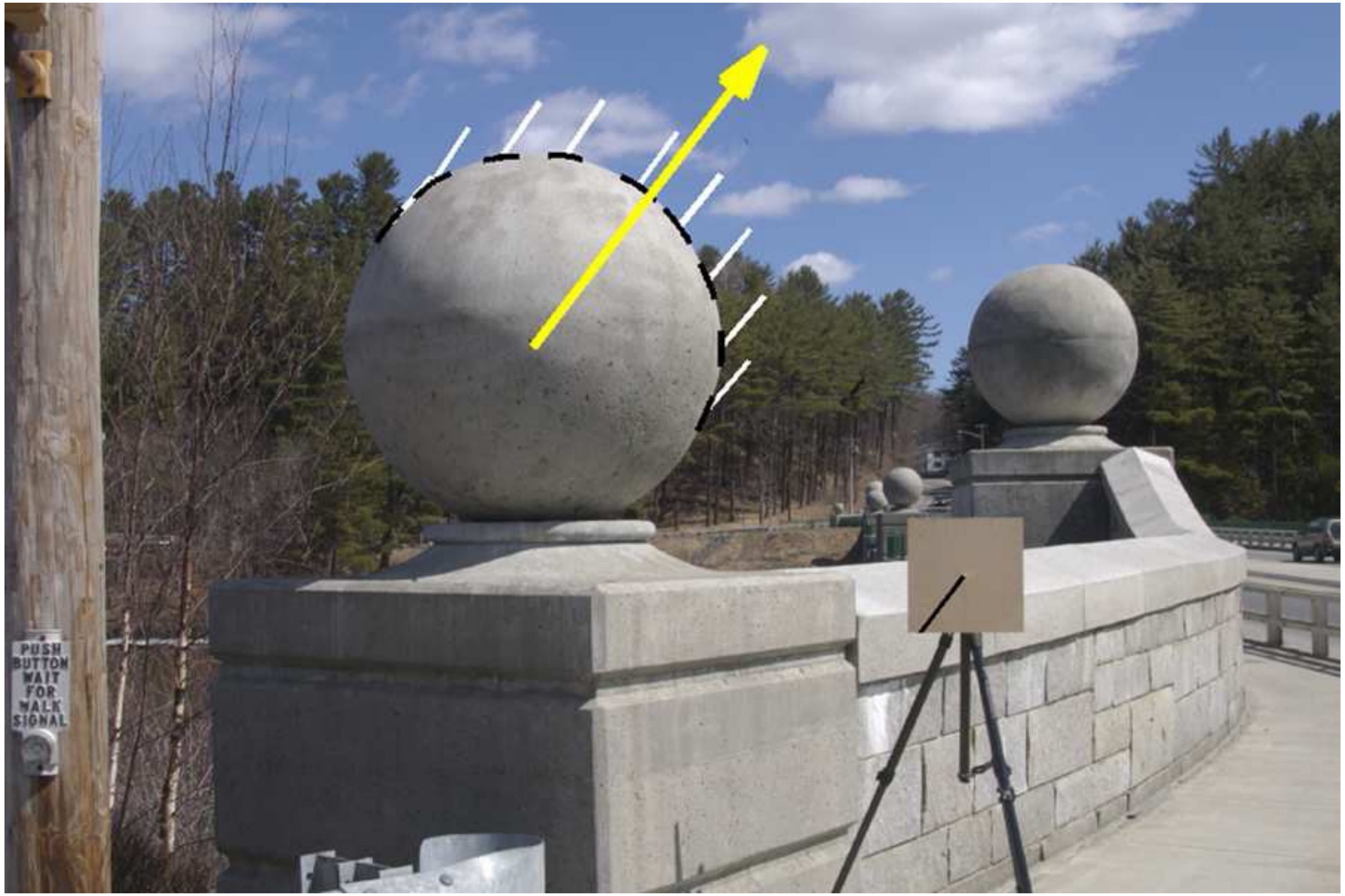
$$\begin{pmatrix} N_x(x_1,y_1) & N_y(x_1,y_1) & N_z(x_1,y_1) \\ N_x(x_2,y_2) & N_y(x_2,y_2) & N_z(x_2,y_2) \\ \vdots & \vdots & \vdots \\ N_x(x_n,y_n) & N_y(x_n,y_n) & N_z(x_n,y_n) \end{pmatrix} \begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix} = \begin{pmatrix} I(x_1,y_1) \\ I(x_2,y_2) \\ \vdots \\ I(x_n,y_n) \end{pmatrix}$$

For points on the *occluding contour*:

$$\begin{pmatrix} N_x(x_1,y_1) & N_y(x_1,y_1) \\ N_x(x_2,y_2) & N_y(x_2,y_2) \\ \vdots & \vdots \\ N_x(x_n,y_n) & N_y(x_n,y_n) \end{pmatrix} \begin{pmatrix} S_x \\ S_y \end{pmatrix} = \begin{pmatrix} I(x_1,y_1) \\ I(x_2,y_2) \\ \vdots \\ I(x_n,y_n) \end{pmatrix}$$

P. Nillius and J.-O. Eklundh, “Automatic estimation of the projected light source direction,” CVPR 2001

# Finding the Direction of the Light Source




P. Nillius and J.-O. Eklundh, “Automatic estimation of the projected light source direction,” CVPR 2001

# Application: Detecting composite photos

Fake photo

Real photo

M. K. Johnson and H. Farid, [Exposing Digital Forgeries by Detecting Inconsistencies in Lighting](), ACM Multimedia and Security Workshop, 2005.