CS4670: Computer Vision

Noah Snavely

Lecture 31: Photometric stereo

What happens when a light ray hits an object?

Some of the light gets absorbed

- converted to other forms of energy (e.g., heat)

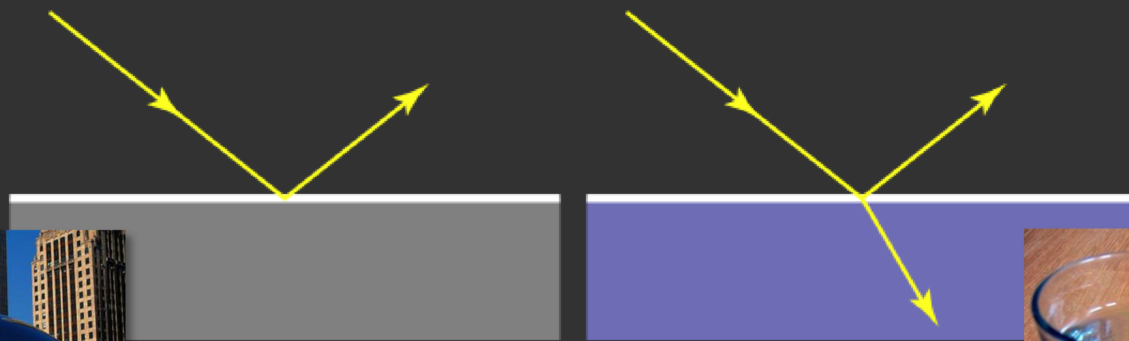
Some gets transmitted through the object

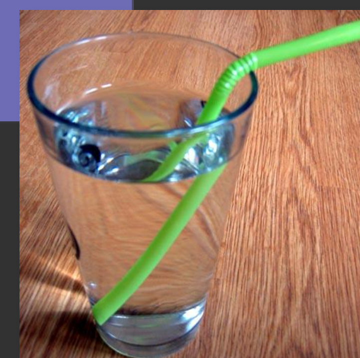
- possibly bent, through “refraction”
- a transmitted ray could possible bounce back

Some gets reflected

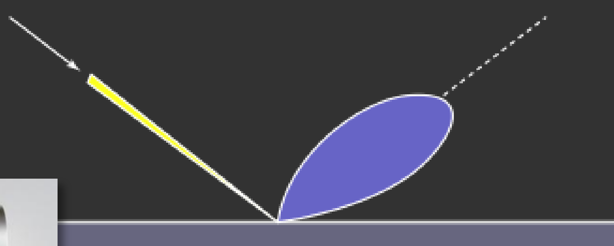
- as we saw before, it could be reflected in multiple directions (possibly all directions) at once

Classic reflection behavior

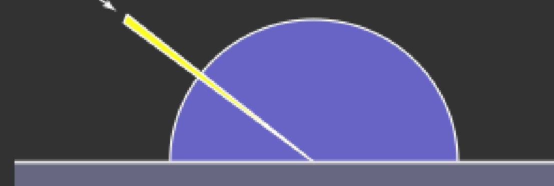


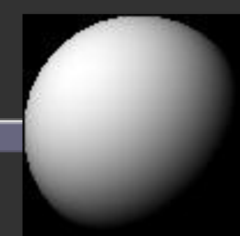


ideal specular



rough specular



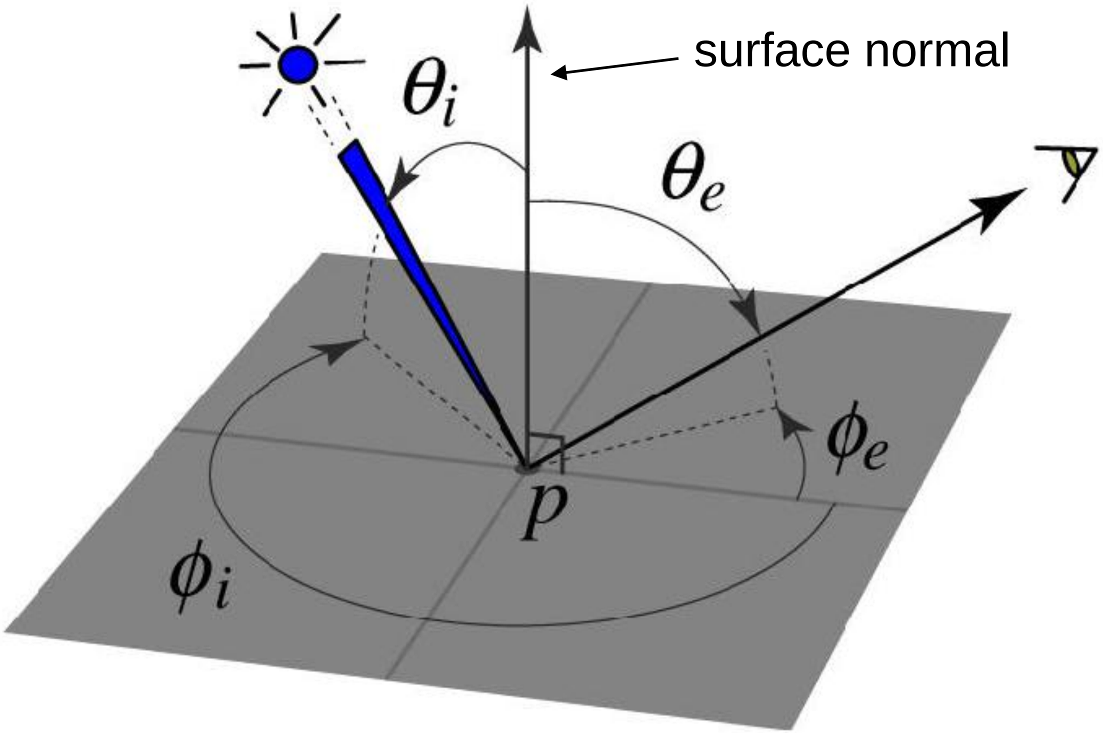


Lambertian

The BRDF

The Bidirectional Reflection Distribution Function

- Given an incoming ray (θ_i, ϕ_i) and outgoing ray (θ_e, ϕ_e) what proportion of the incoming light is reflected along outgoing ray?



Answer given by the BRDF: $\rho(\theta_i, \phi_i, \theta_e, \phi_e)$

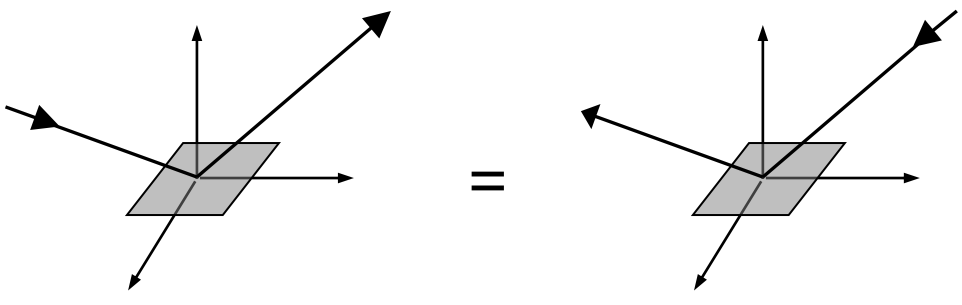
Constraints on the BRDF

Energy conservation

- Quantity of outgoing light ≤ quantity of incident light
  - integral of BRDF ≤ 1

Helmholtz reciprocity

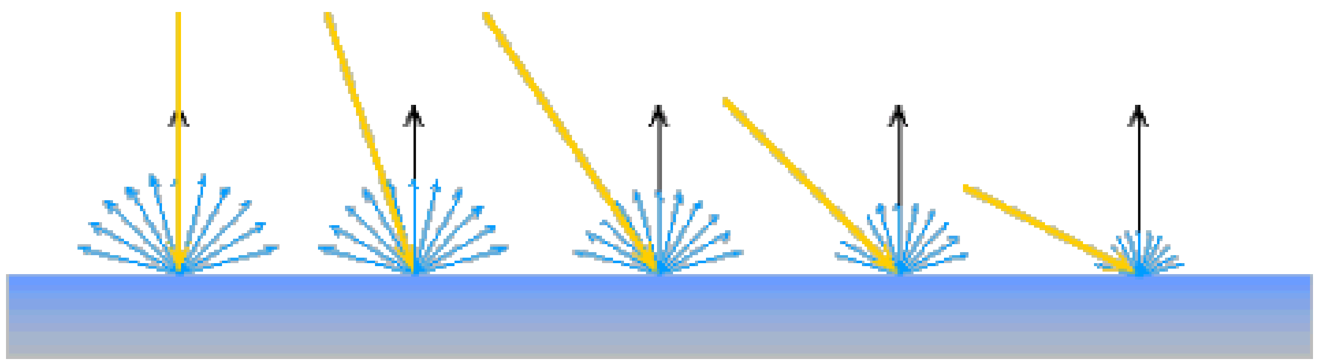
- reversing the path of light produces the same reflectance

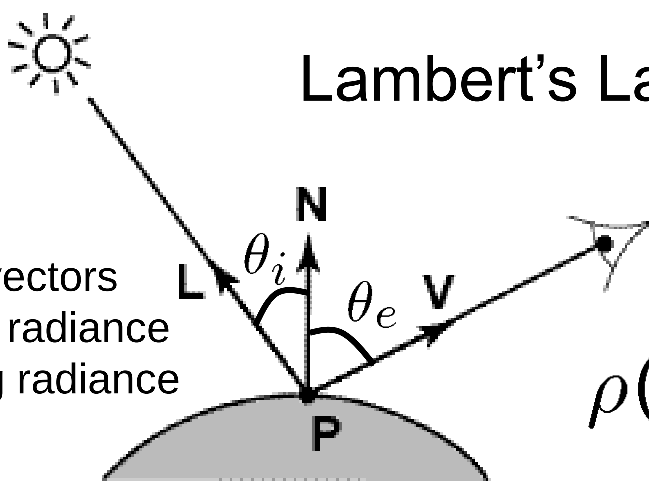


Diffuse reflection

Diffuse reflection governed by **Lambert's law**

- Viewed brightness does not depend on viewing direction
- Brightness *does* depend on direction of illumination
- This is the model most often used in computer vision





L, **N**, **V** unit vectors
I_e = outgoing radiance
I_i = incoming radiance

Lambert's Law: $I_e = k_d \mathbf{N} \cdot \mathbf{L} I_i$

k_d is called **albedo**

BRDF for **Lambertian surface**

$$\rho(\theta_i, \phi_i, \theta_e, \phi_e) = k_d cos\theta_i$$

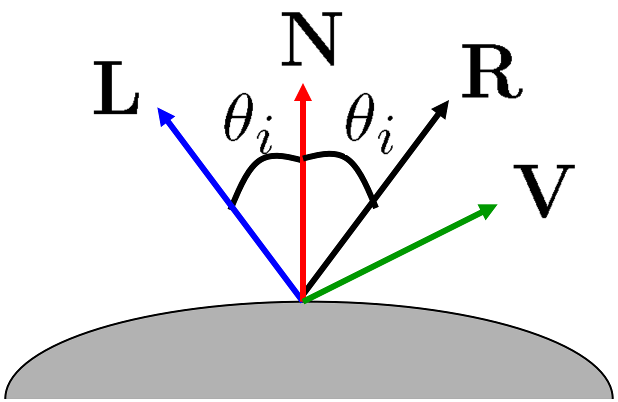
Diffuse reflection

Demo

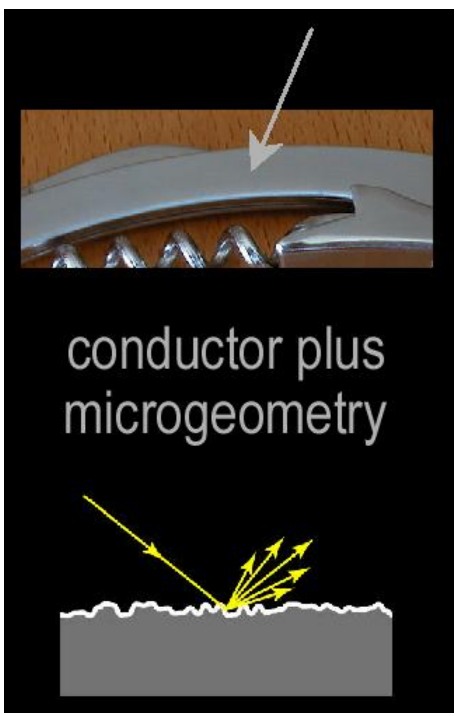
http://www.math.montana.edu/frankw/ccp/multiworld/twothree/lighting/applet1.htm
http://www.math.montana.edu/frankw/ccp/multiworld/twothree/lighting/learn2.htm

Specular reflection

For a perfect mirror, light is reflected about **N**

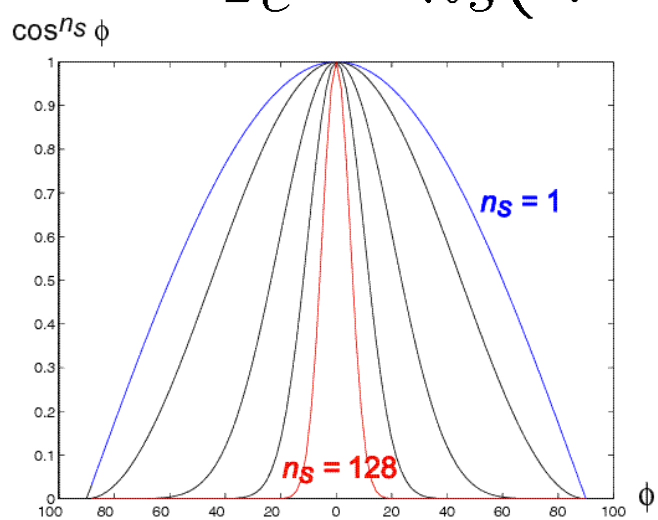


$$I_e = \begin{cases} I_i & \text{if } \mathbf{V} = \mathbf{R} \\ 0 & \text{otherwise} \end{cases}$$

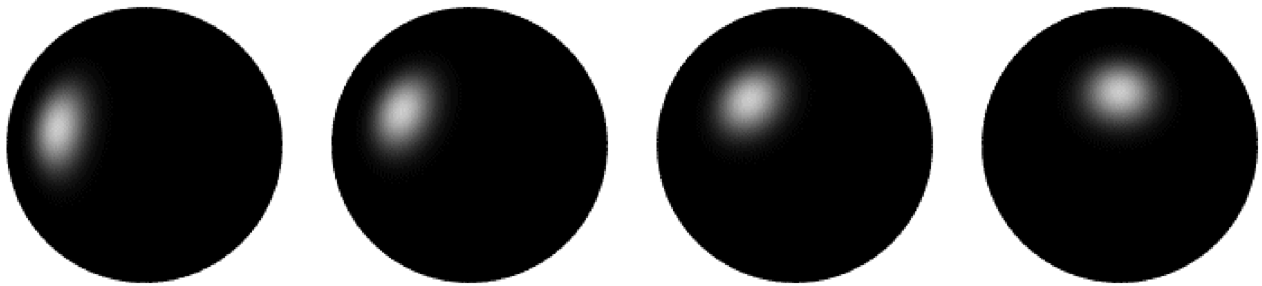


Near-perfect mirrors have a **highlight** around **R**

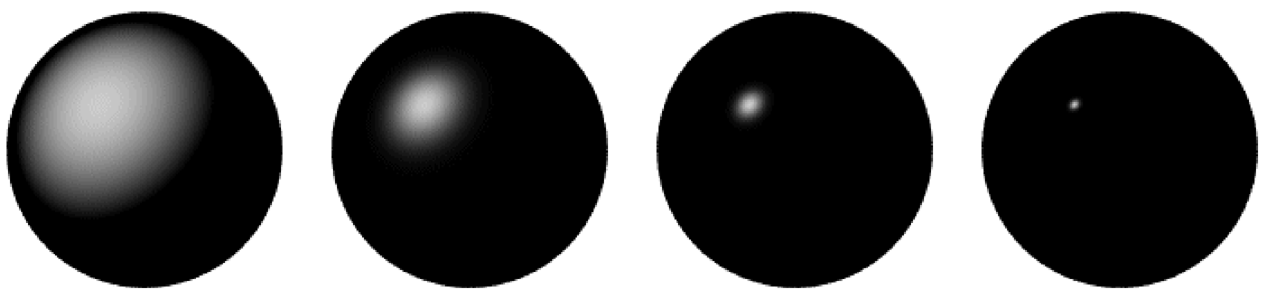
- common model: $I_e = k_s(\mathbf{V} \cdot \mathbf{R})^{n_s} I_i$



Specular reflection

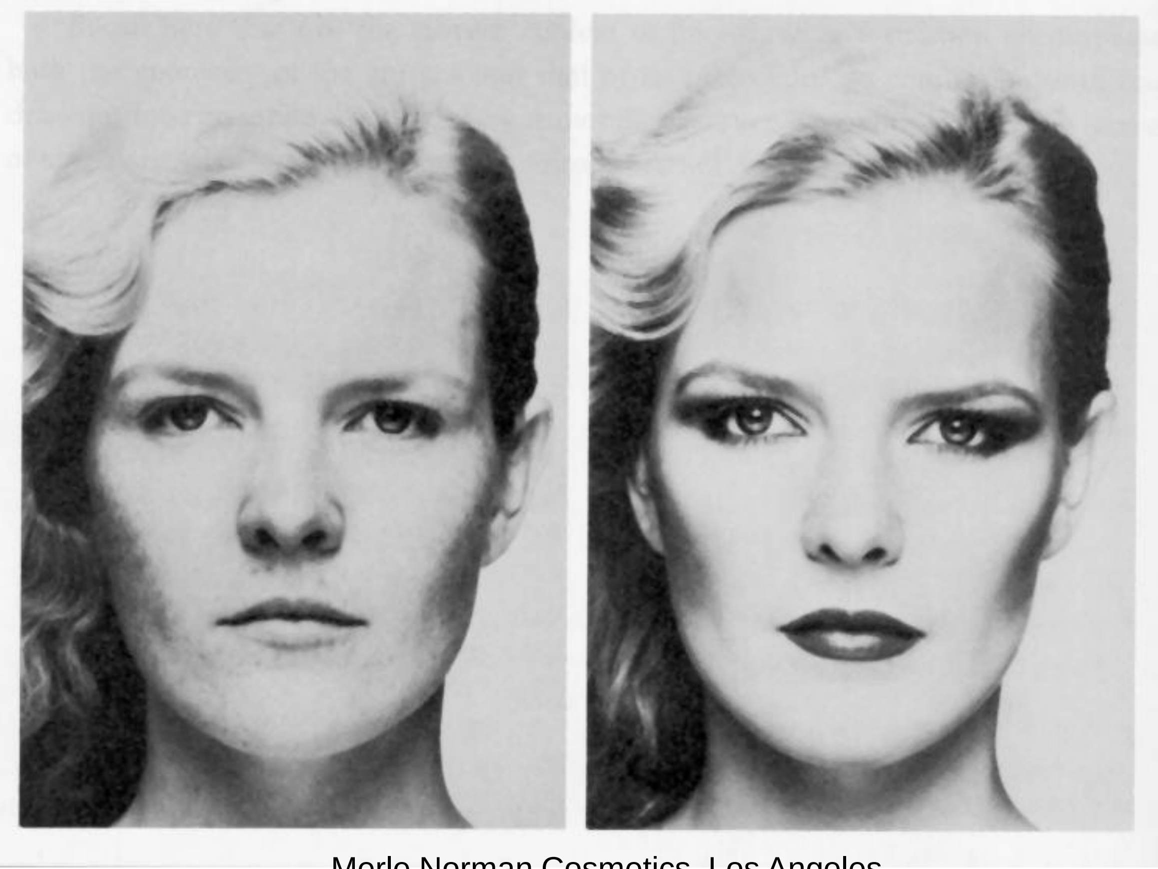


Moving the light source



Changing n_s

Photometric Stereo

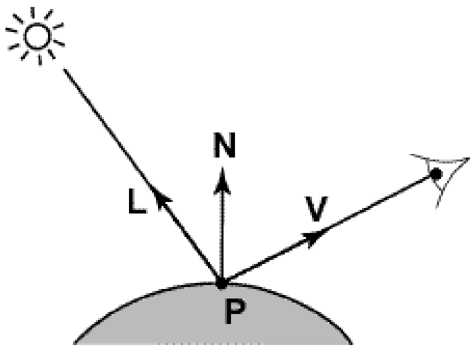


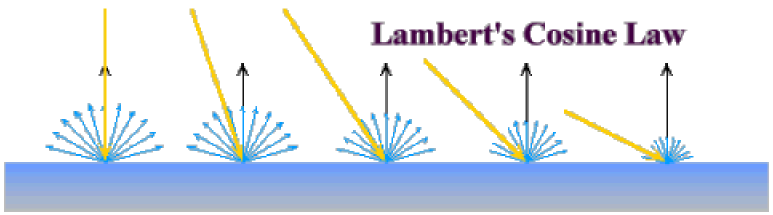
Merle Norman Cosmetics, Los Angeles

Readings

- R. Woodham, *Photometric Method for Determining Surface Orientation from Multiple Images*. Optical Engineering 19(1)139-144 (1980). (PDF)

Diffuse reflection





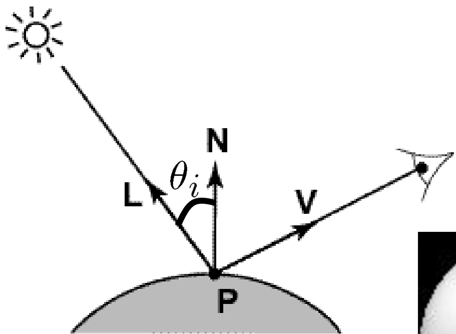
$$R_e = k_d \mathbf{N} \cdot \mathbf{L} R_i$$

image intensity of **P** → $$I = k_d \mathbf{N} \cdot \mathbf{L}$$

Simplifying assumptions

- $I = R_e$: camera response function is the identity function:
- $R_i = 1$: light source intensity is 1
  - can achieve this by dividing each pixel in the image by R_i

Shape from shading



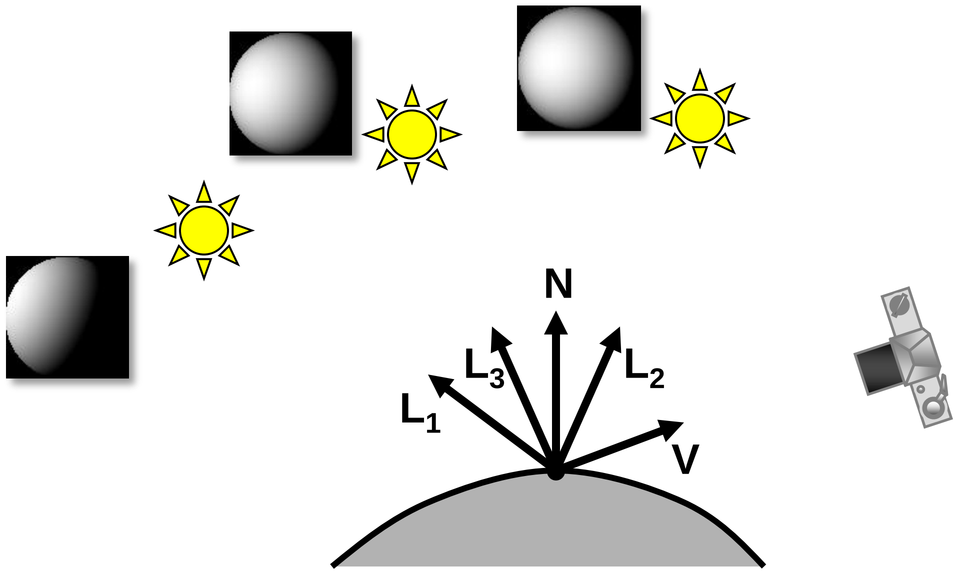
Suppose $k_d = 1$

$$
\begin{aligned}
I &= k_d \mathbf{N} \cdot \mathbf{L} \\
&= \mathbf{N} \cdot \mathbf{L} \\
&= cos\ \theta_i
\end{aligned}
$$

You can directly measure angle between normal and light source

- Not quite enough information to compute surface shape
- But can be if you add some additional info, for example
  - assume a few of the normals are known (e.g., along silhouette)
  - constraints on neighboring normals—"integrability"
  - smoothness
- Hard to get it to work well in practice
  - plus, how many real objects have constant albedo?

Photometric stereo



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

Can write this as a matrix equation:

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

Solving the equations

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

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

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

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

More than three lights

Get better results by using more lights

$$\begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} \mathbf{L_1} \\ \vdots \\ \mathbf{L_n} \end{bmatrix} k_d \mathbf{N}$$

Least squares solution:

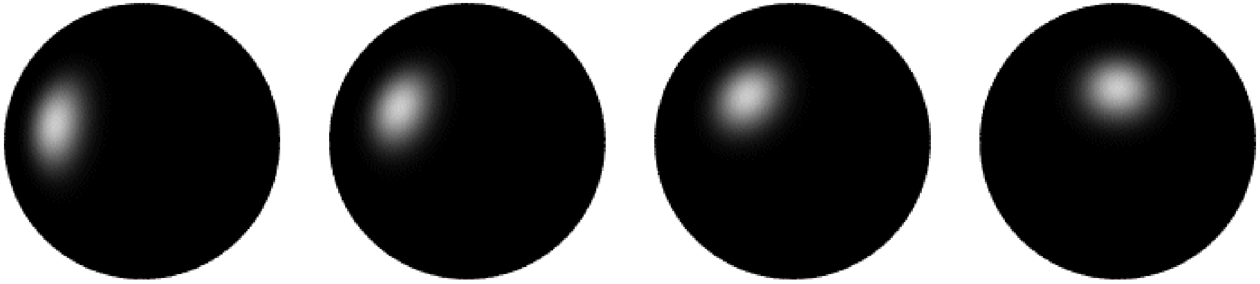
$$\begin{aligned} \mathbf{I} &= \mathbf{LG} \\ \mathbf{L^T I} &= \mathbf{L^T L G} \\ \mathbf{G} &= (\mathbf{L^T L})^{-1} (\mathbf{L^T I}) \end{aligned}$$

Solve for N, k_d as before

What's the size of $\mathbf{L^T L}$?

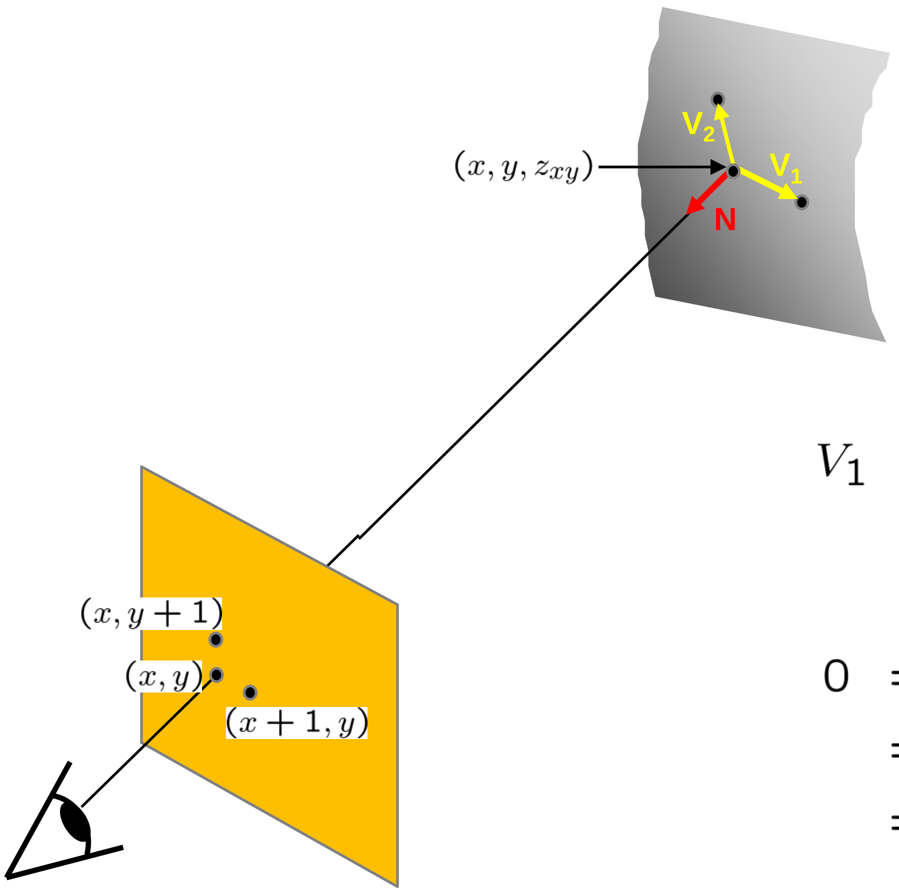
Computing light source directions

Trick: place a chrome sphere in the scene



- the location of the highlight tells you where the light source is

Depth from normals



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

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

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

Example

What if we don’t have mirror ball?

Hayakawa, Journal of the Optical Society of America, 1994, Photometric stereo under a light source with arbitrary motion.

Limitations

Big problems

- doesn't work for shiny things, 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

Newer work addresses some of these issues

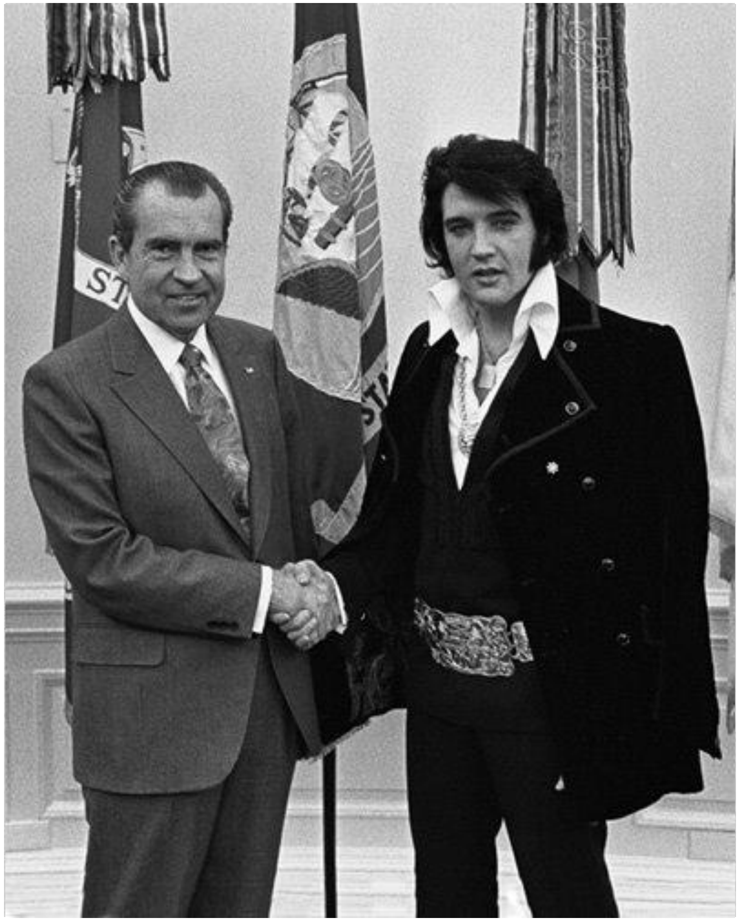
Some pointers for further reading:

- Zickler, Belhumeur, and Kriegman, *"Helmholtz Stereopsis: Exploiting Reciprocity for Surface Reconstruction."* IJCV, Vol. 49 No. 2/3, pp 215-227.
- Hertzmann & Seitz, "*Example-Based Photometric Stereo: Shape Reconstruction with General, Varying BRDFs*." IEEE Trans. PAMI 2005

Application: Detecting composite photos

Which is the real photo?

Fake photo



Real photo