

Computer Vision 1

Reflectance & Photometric Stereo Image Processing Basics

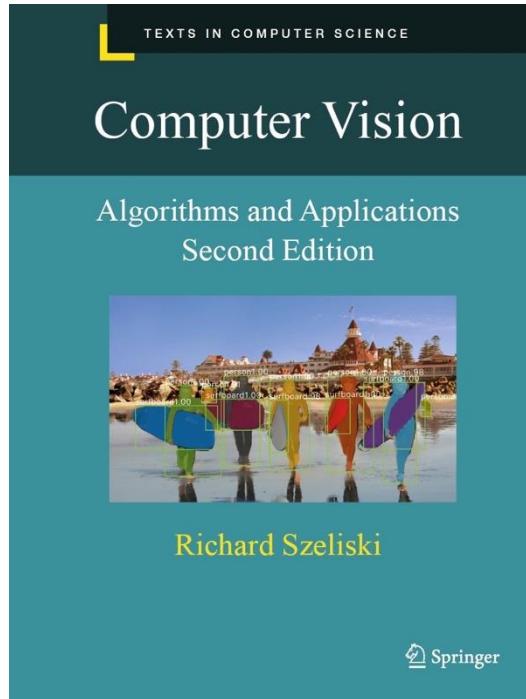
Dr. Martin Oswald, Dr. Dimitris Tzionas, Dr. Arun Mukundan,
[m.r.oswald, d.tzionas, a.mukundan]@uva.nl

Outline

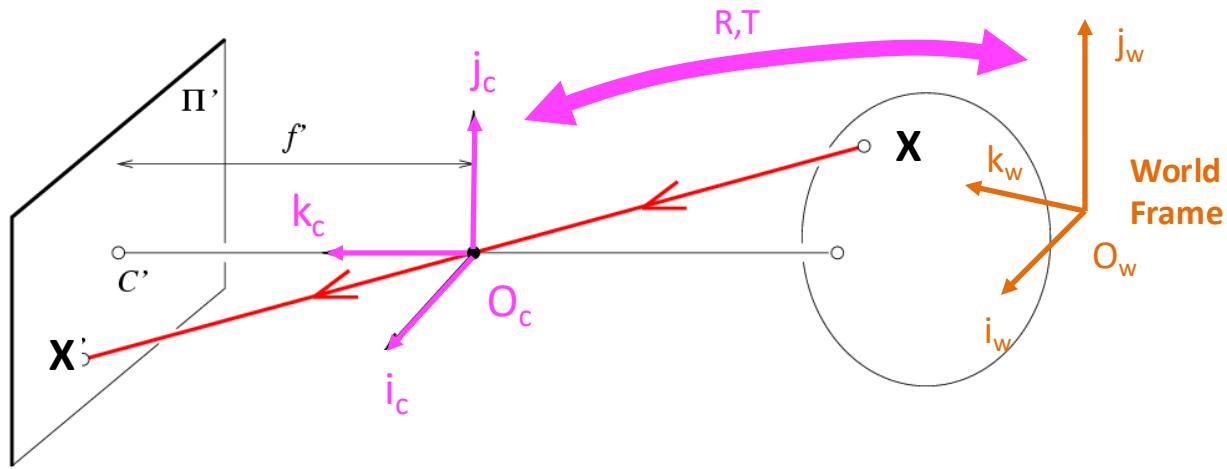
- Reflection and Photometric Stereo
 - Reflection Models
 - Shape from Shading
 - Photometric Stereo
 - Intrinsic Decomposition
 - Applications
- Image Processing Basics
 - Histogram
 - Histogram Modification
 - Histogram Equalization

Textbook

- 3.1.1
- 3.1.2
- 3.1.4
- 2.2.2



Camera Projection Matrix



Camera Frame
posed (R, T) w.r.t.
World Frame

$$\mathbf{x} = \mathbf{K} [\mathbf{I} \quad \mathbf{0}] \mathbf{X}$$

Intrinsic Extrinsic
 Matrix Matrix

\mathbf{O}_w : Center of World Frame
 \mathbf{O}_c : Center of Camera Frame

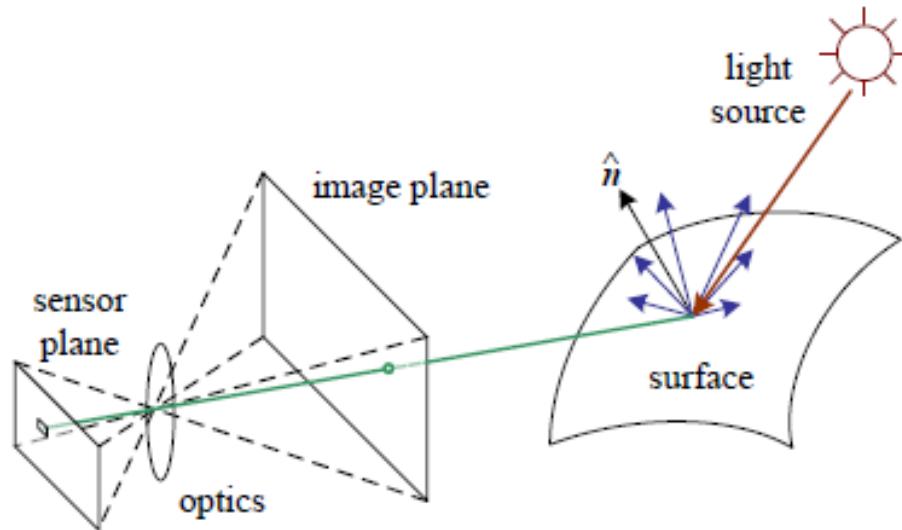
\mathbf{X} : 3D point in World Coordinates: $(X, Y, Z, 1)$
 \mathbf{x} : 2D Image Coordinates: $(u, v, 1)$, up to scale w

\mathbf{K} : Intrinsic Matrix (3x3)
 \mathbf{R} : Rotation (3x3)
 \mathbf{t} : Translation (3x1)

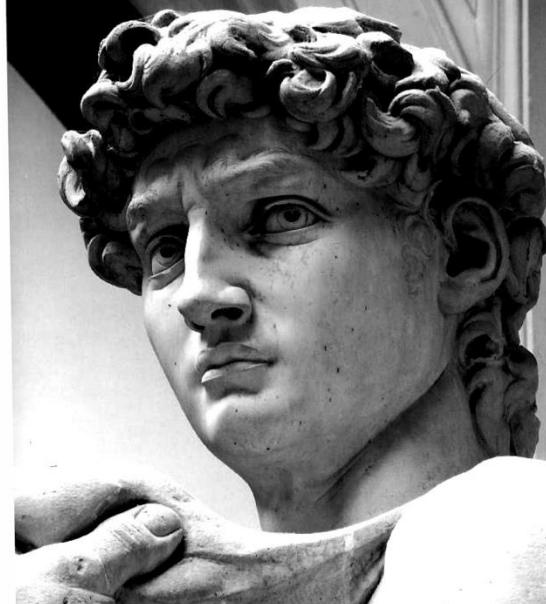
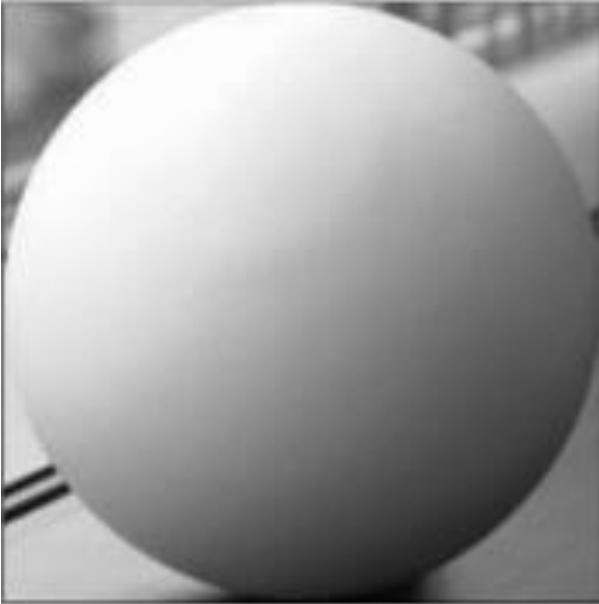
Note: different books
use different notation!

4 main factors which define intensity values

- the geometry of the scene
- the reflectances of the visible surface
- the illumination of the scene
- the viewpoint

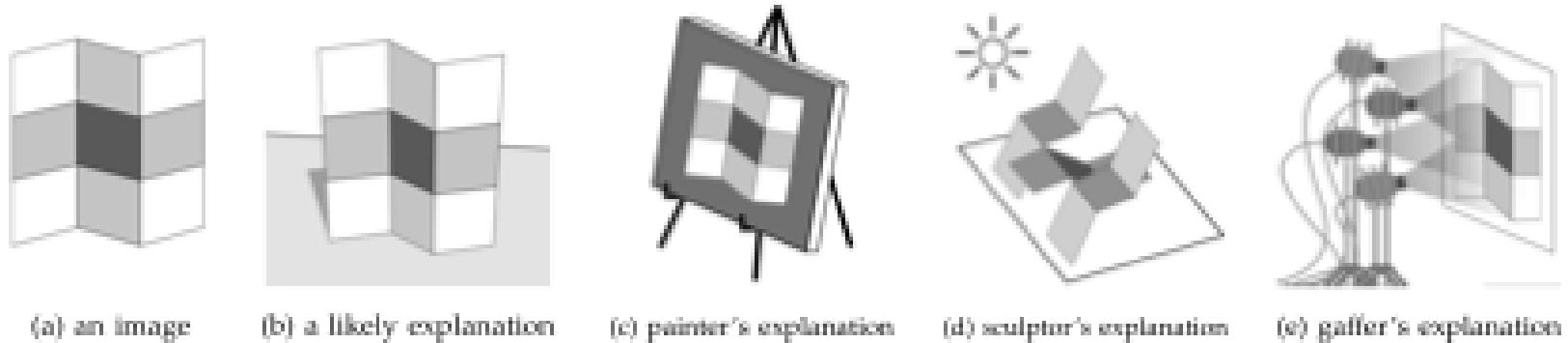


Can we recover shape from shading?



What is the relation between **intensity** and **shape**? Can we recover shape?

Adelson and Pentland's Workshop Metaphor

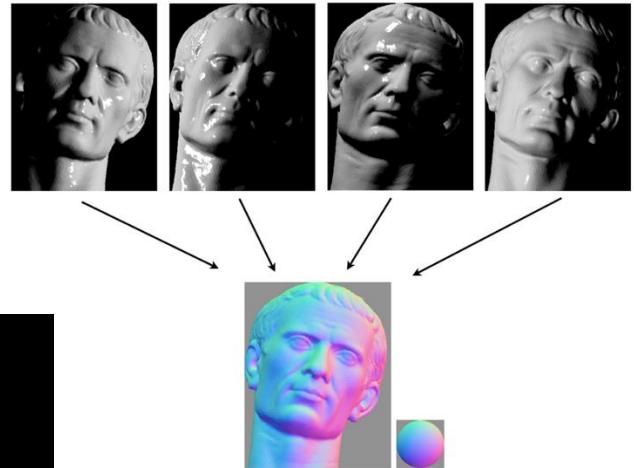
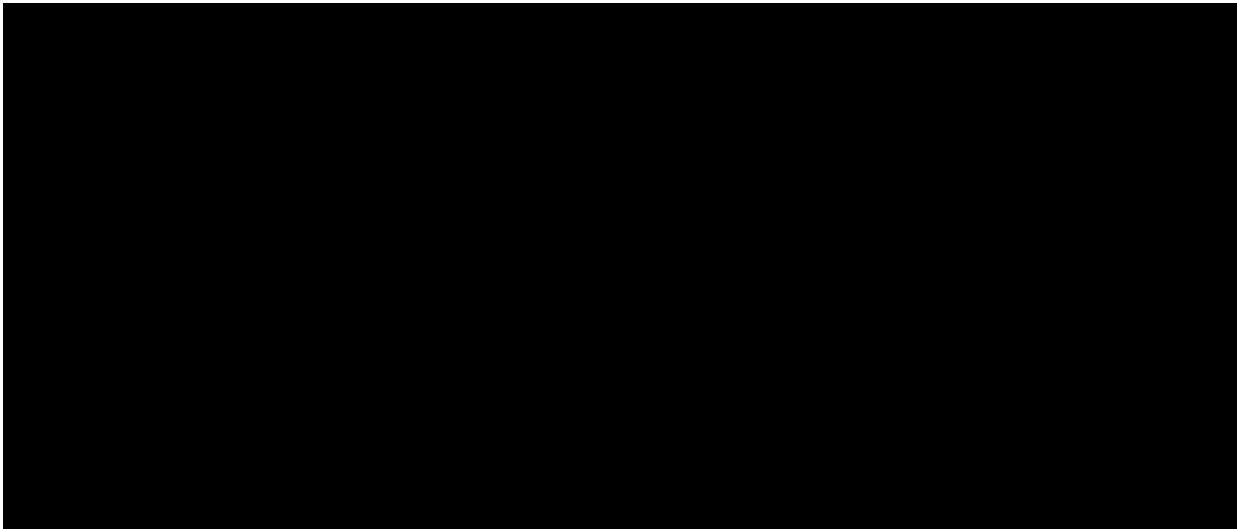


Can we recover **3D** from a single **image**?

- Inverting the graphics problem is very challenging
- The space of shapes, paint, and light that exactly reproduce an image is vast
- The image in (a) very likely corresponds to (b), but it could be a painting (c), sculpture (d), or an arrangement of lights (e)
- We are looking for a **simple explanation** and must use **prior knowledge**

Motivation: Photometric Stereo

- Photometric stereo captures **multiple** images from the **same viewpoint**, but with **different light settings**
- Per-pixel estimation of **normal** and **albedo** or **material**
- Assumes far cameras/lights



Total Relighting

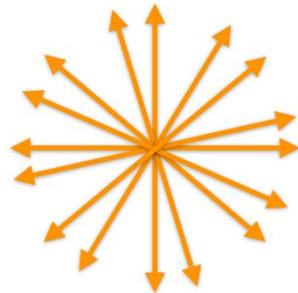
Total Relighting:
Learning to Relight Portraits for Background Replacement



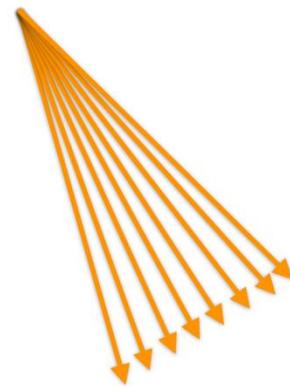
Rohit Pandey*, Sergio Orts Escolano*, Chloe LeGendre*, Christian Haene,
Sofien Bouaziz, Christoph Rhemann, Paul Debevec, and Sean Fanello



Light sources



Point Light



Spot Light

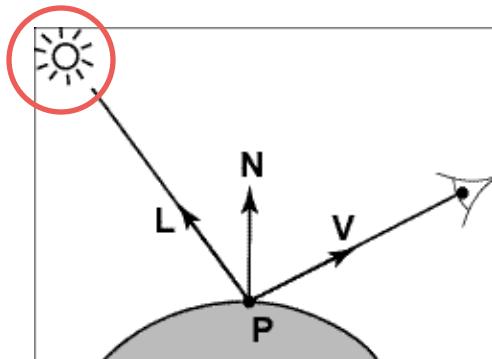


Directional Light

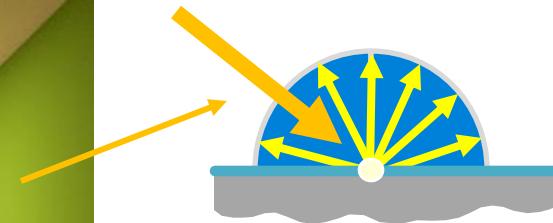
From now we assume directional light

Directional Lighting and Reflectance

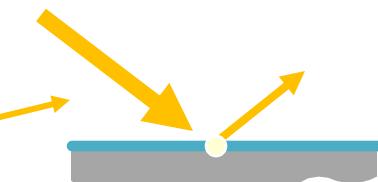
- Key property: all rays are parallel
- Equivalent to an infinitely distant point source



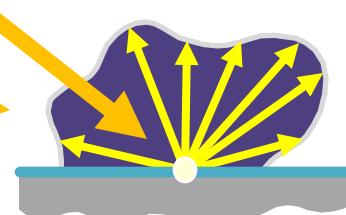
Materials - Three Forms



Ideal diffuse
(Lambertian)

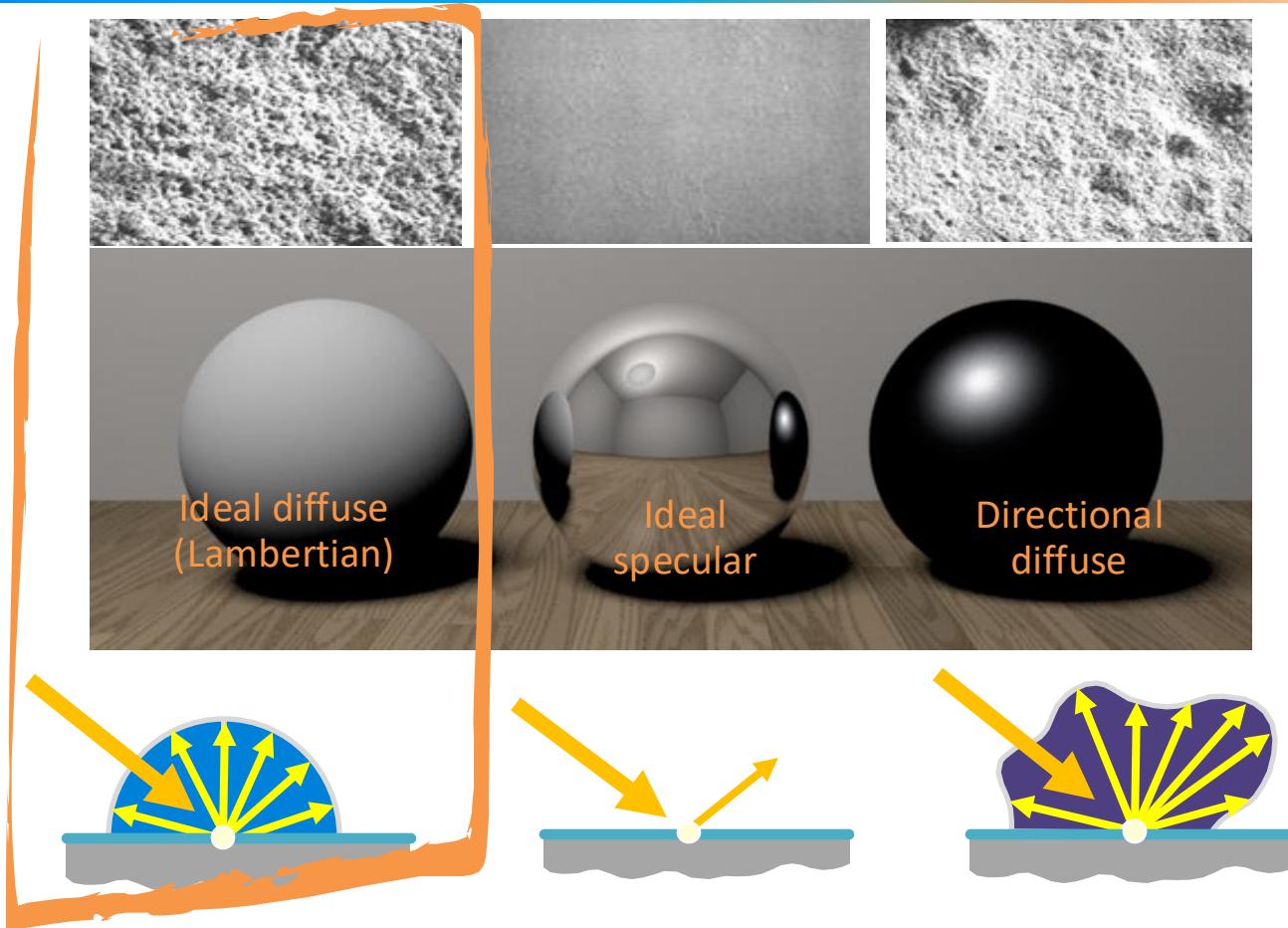


Ideal
specular

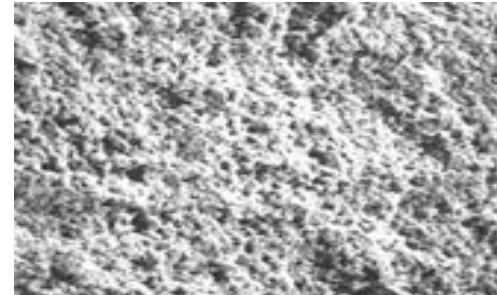
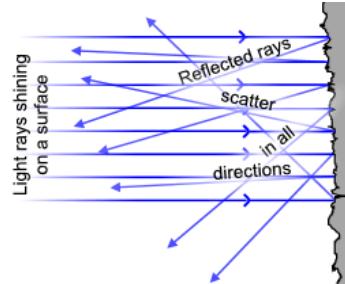
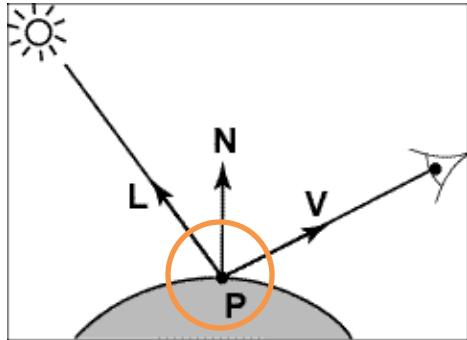


Directional
diffuse

Reflectance - Three Forms

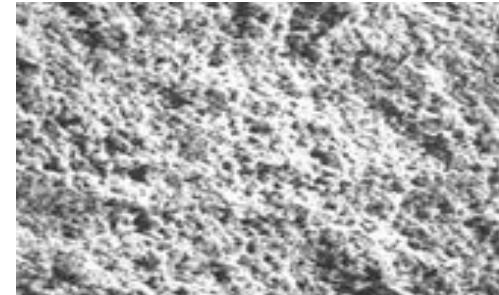
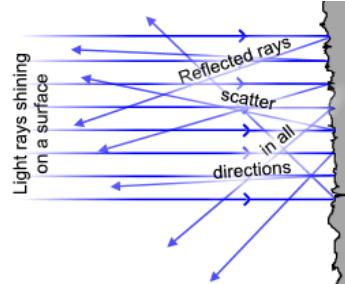
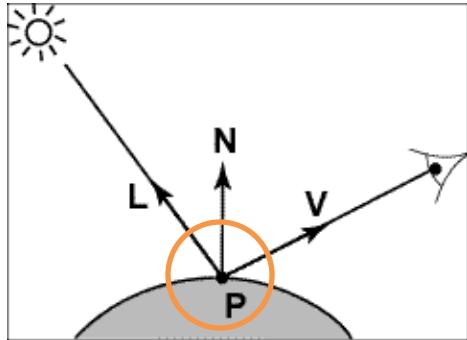


Reflectance - Three Forms



1. Reflected energy is proportional to cosine of angle between L and N (**incoming**)
2. Measured intensity is viewpoint-independent (**outgoing**)

Reflectance - Three Forms



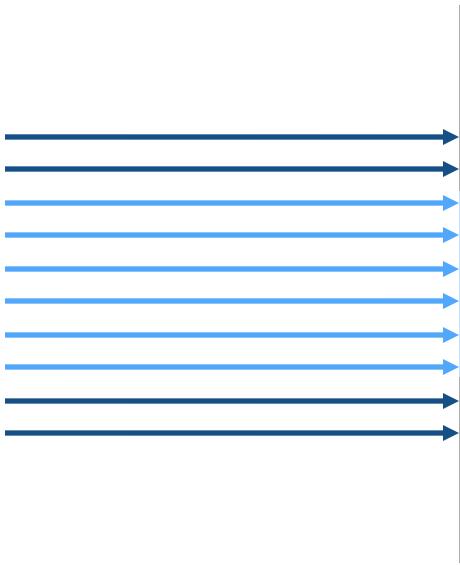
$$I \propto N \cdot L$$

Image intensity \propto Surface normal • Light direction

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

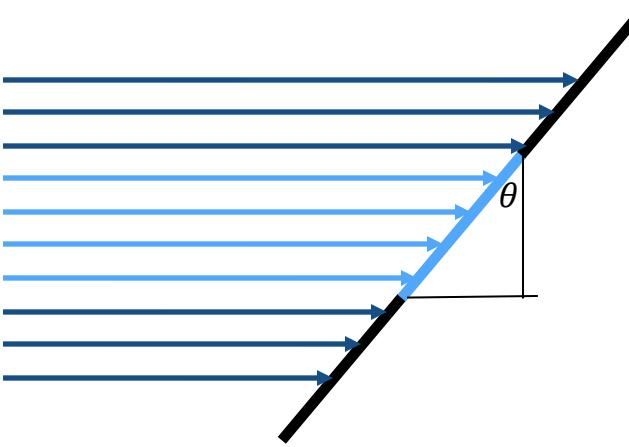
Lambertian Reflectance: Incoming

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



Lambertian Reflectance: Incoming

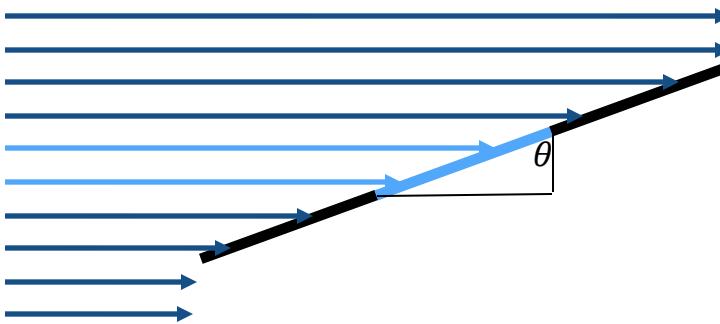
Reflected energy is proportional to cosine of angle between L and N.



$$I \propto \underbrace{N \cdot L}_{\cos(\theta)}$$

Lambertian Reflectance: Incoming

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



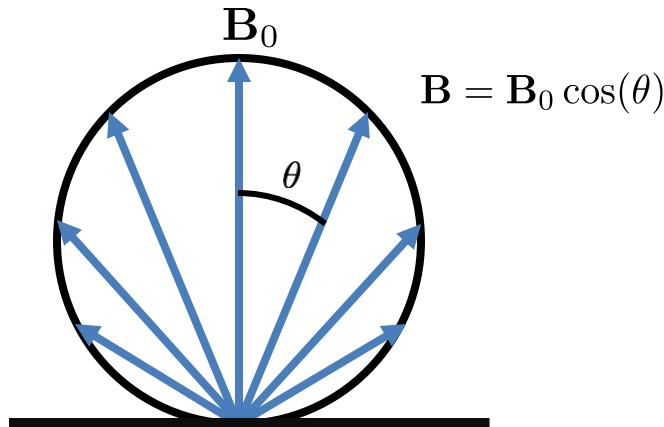
$$I \propto \underbrace{N \cdot L}_{\cos(\theta)}$$

Light hitting surface is proportional to the **cosine**.

Lambertian appearance is view-independent

CVN

- Number of photons reflected to a given angle θ is proportional to $\cos(\theta)$

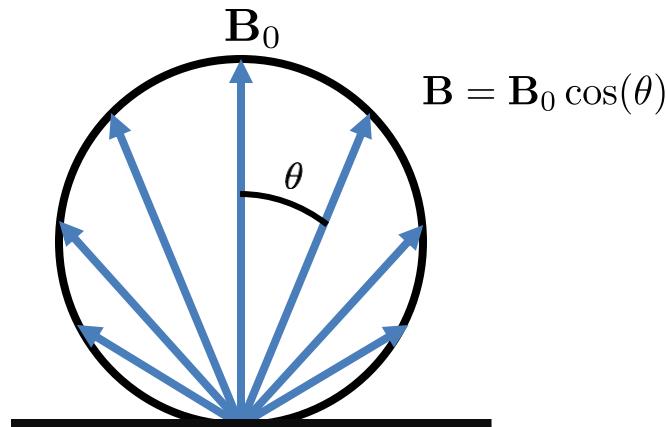


Lambert's cosine law: $B = B_0 \cos(\theta)$

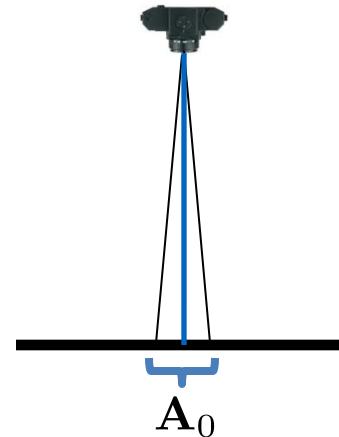
Lambertian appearance is view-independent

CV

- Number of photons reflected to a given angle θ is proportional to $\cos(\theta)$
- But appearance is the same from every angle due to larger pixel footprint at larger angles

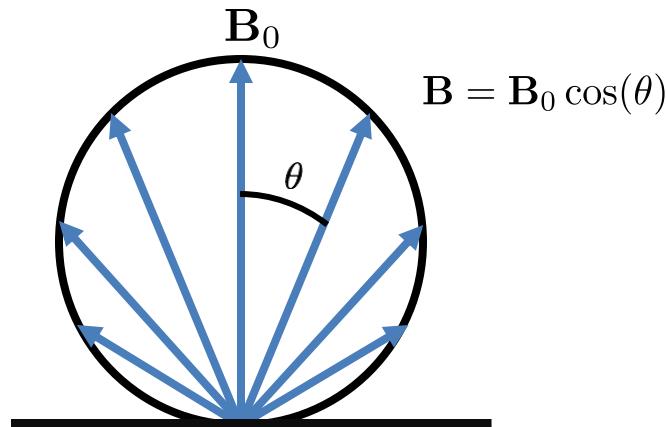


Lambert's cosine law: $B = B_0 \cos(\theta)$

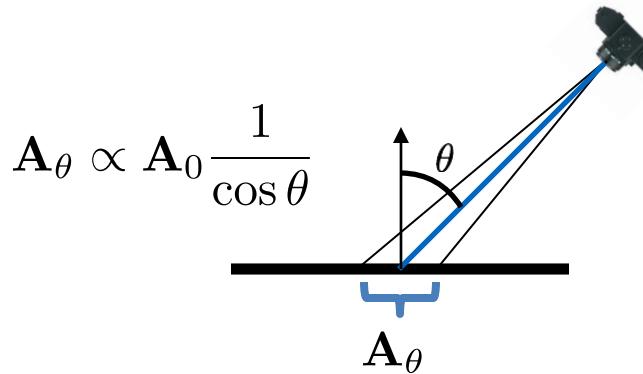


Lambertian appearance is view-independent

- Number of photons reflected to a given angle θ is proportional to $\cos(\theta)$
- But appearance is the same from every angle due to larger pixel footprint at larger angles

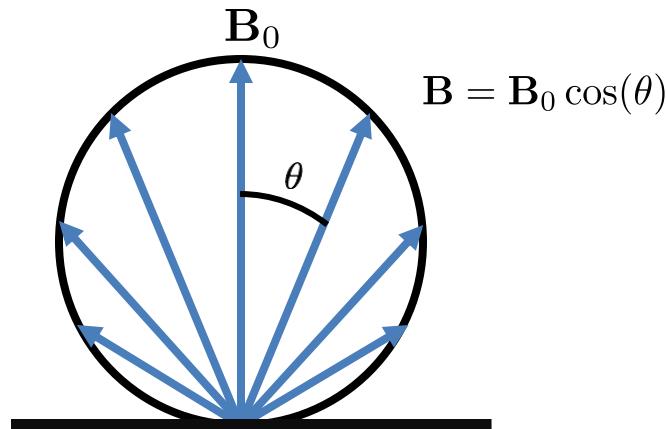


Lambert's cosine law: $B = B_0 \cos(\theta)$

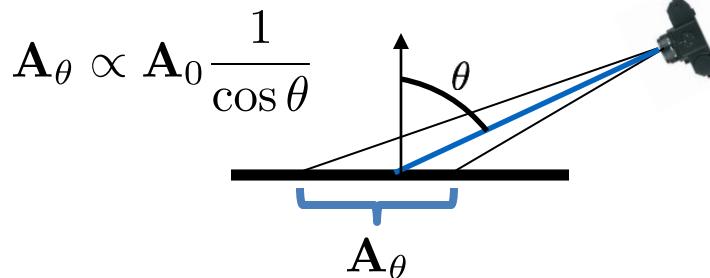


Lambertian appearance is view-independent

- Number of photons reflected to a given angle θ is proportional to $\cos(\theta)$
- But appearance is the same from every angle due to larger pixel footprint at larger angles



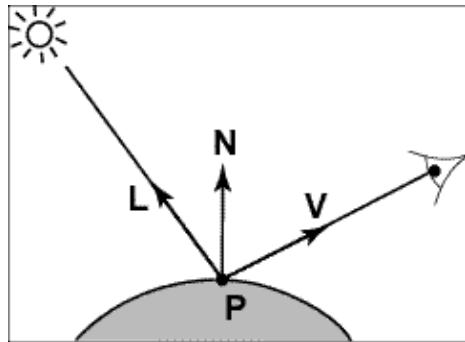
Lambert's cosine law: $B = B_0 \cos(\theta)$


$$A_\theta \propto A_0 \frac{1}{\cos \theta}$$

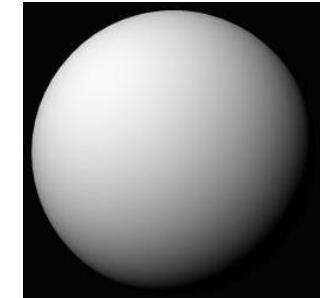
Radiance (what eye sees)

$$\propto B_0 A_0 \cos(\theta) \frac{1}{\cos(\theta)}$$

Final Lambertian Image Formation Model

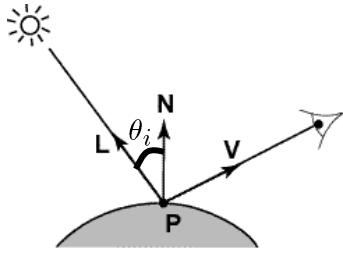
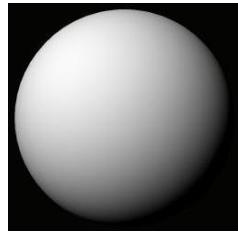


$$I = k_d \mathbf{N} \cdot \mathbf{L}$$



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) κ_d
3. Camera response function
 - Assume pixel value is linearly proportional to incoming energy (perform radiometric calibration if not)

A Single Image: Shape from Shading



Suppose (for now) $k_d = 1$

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

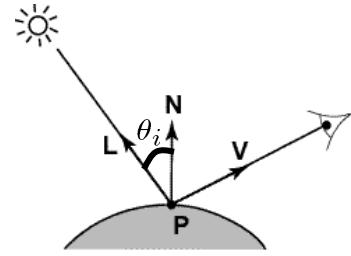
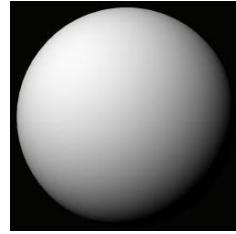
You can directly measure angle between normal and light source

- 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?
 - But, deep learning can help

Shape from Shading

Shape from Shading assumptions:

- Diffuse material with spatially constant albedo
 - Reduces the number of material parameters to 1
- Known point light source at infinity
 - Keeps light direction L constant across all pixels
 - Makes L independent of geometry/depth
- Known camera at infinity (orthography)
 - Keeps view direction V constant across all pixels
 - Makes V independent of geometry/depth



Lightsource Position Ambiguity

Plates and bowls or upside down or not?



rotate the image to
the right

[<https://www.good.is/optical-illusion-plates-and-bowls-upside-down-or-not>]

Application: Detecting composite photos

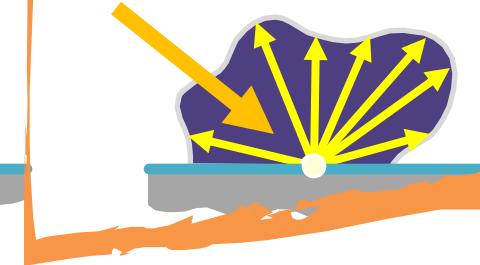
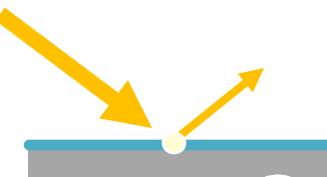
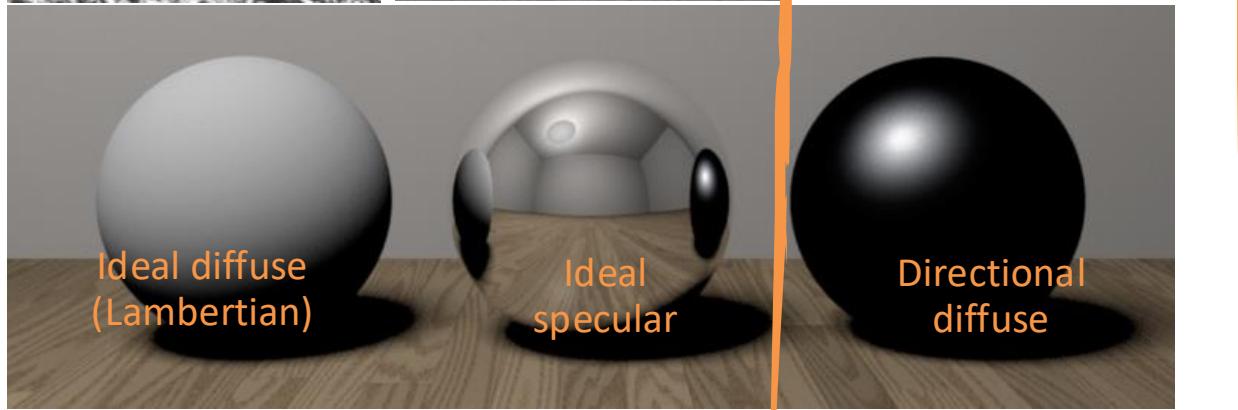
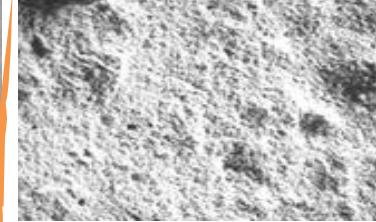
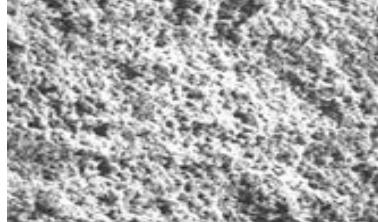
Fake photo



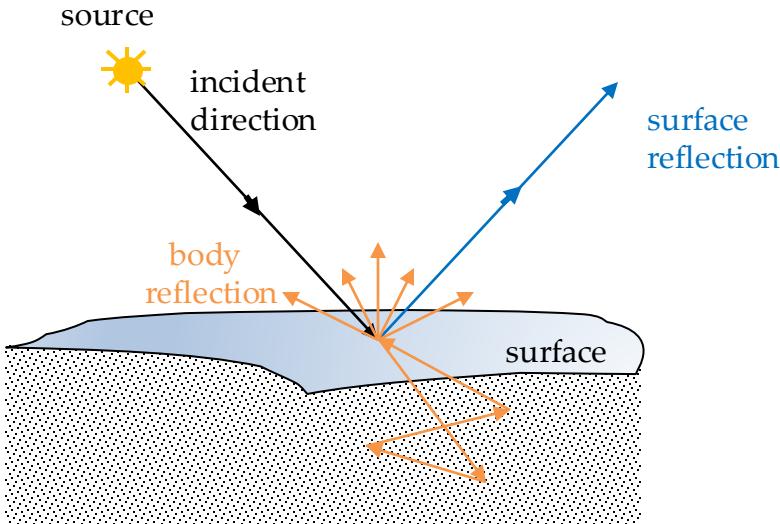
Real photo



Reflectance - Three Forms



Mechanisms of Surface Reflection



Body Reflection:

- Diffuse Reflection
- Matte Appearance
- Non-Homogeneous Medium
- Clay, paper, etc.

Surface Reflection:

- Specular Reflection
- Glossy Appearance
- Highlights
- Dominant for Metals

$$\text{Image Intensity} = \text{Body Reflection} + \text{Surface Reflection}$$

Mechanisms of Surface Reflection

Body Reflection:

- Diffuse Reflection
- Matte Appearance
- Non-Homogeneous Medium
- Clay, paper, etc.



Surface Reflection:

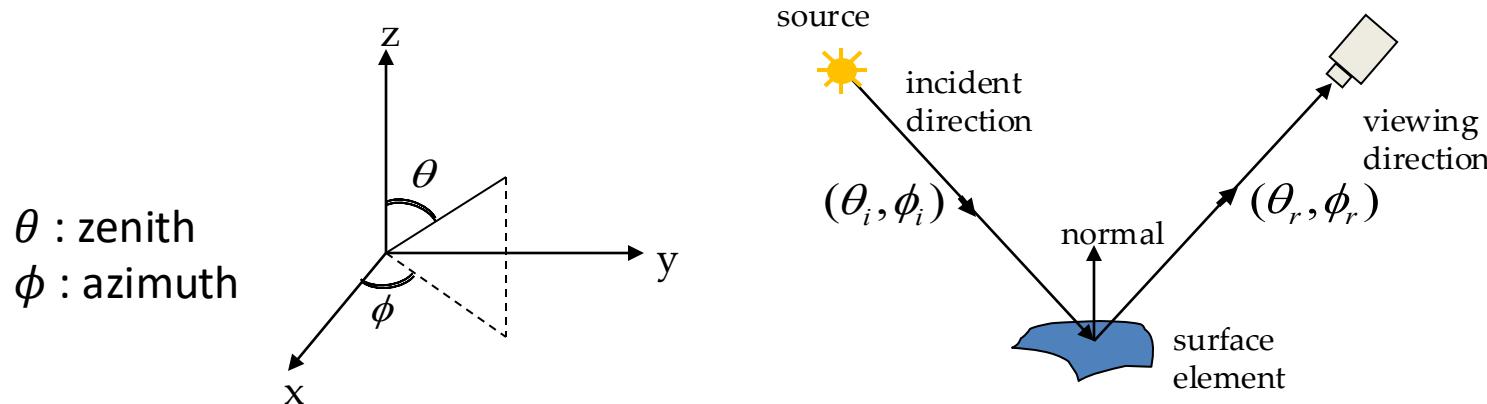
- Specular Reflection
- Glossy Appearance
- Highlights
- Dominant for Metals



Many materials exhibit both Reflections:



BRDF: Bidirectional Reflectance Distribution Function



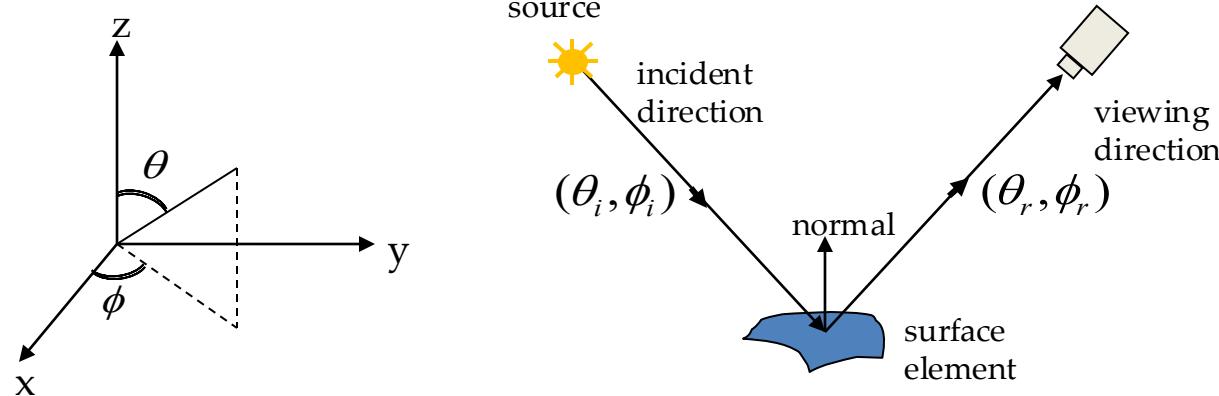
$E^{surface}(\theta_i, \phi_i)$ Irradiance due to source in direction (θ_i, ϕ_i)

$L^{surface}(\theta_r, \phi_r)$ Radiance of Surface in direction (θ_r, ϕ_r)

$$\text{BRDF: } f(\theta_i, \phi_i; \theta_r, \phi_r) = \frac{L^{surface}(\theta_r, \phi_r)}{E^{surface}(\theta_i, \phi_i)}$$

describe how incoming and outcoming lights behaves

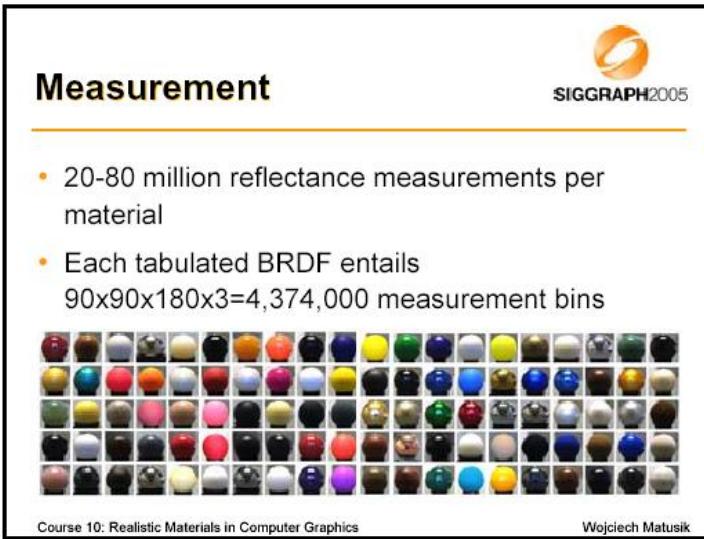
BRDF Properties



- BRDF is a 4D function
- Non-negative: $f(\theta_i, \phi_i, \theta_r, \phi_r) > 0$
- Helmholtz Reciprocity: $f(\theta_i, \phi_i, \theta_r, \phi_r) = f(\theta_r, \phi_r, \theta_i, \phi_i)$

BRDF databases

- MERL ([Matusik](#) et al.): 100 isotropic, 4 nonisotropic, dense



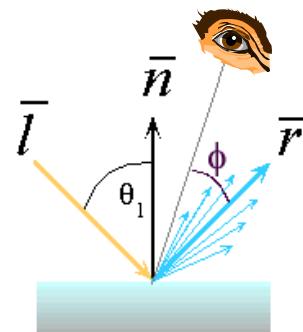
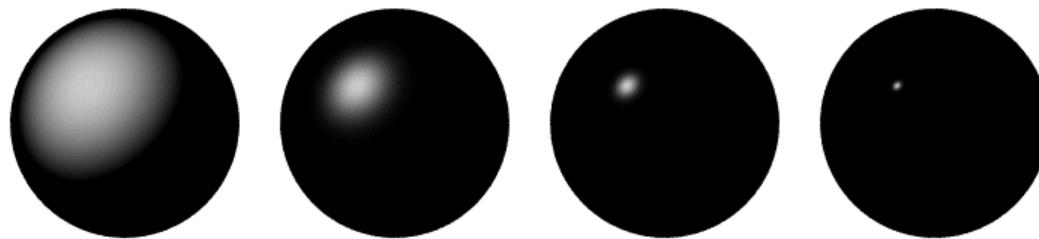
- [CURET](#) (Columbia-Utrecht): 60 samples, more sparsely sampled, but also bidirectional texture functions (BTF)

Sample albedos	
Surface	Typical albedo
Fresh asphalt	0.04 ^[6]
Open ocean	0.06 ^[7]
Worn asphalt	0.12 ^[6]
Conifer forest, summer	0.08, ^[8] 0.09 to 0.15 ^[9]
Deciduous forest	0.15 to 0.18 ^[9]
Bare soil	0.17 ^[10]
Green grass	0.25 ^[10]
Desert sand	0.40 ^[11]
New concrete	0.55 ^[10]
Ocean ice	0.50 to 0.70 ^[10]
Fresh snow	0.80 ^[10]
Aluminium	0.85 ^{[12][13]}

[Wikipedia: [Albedo](#)]

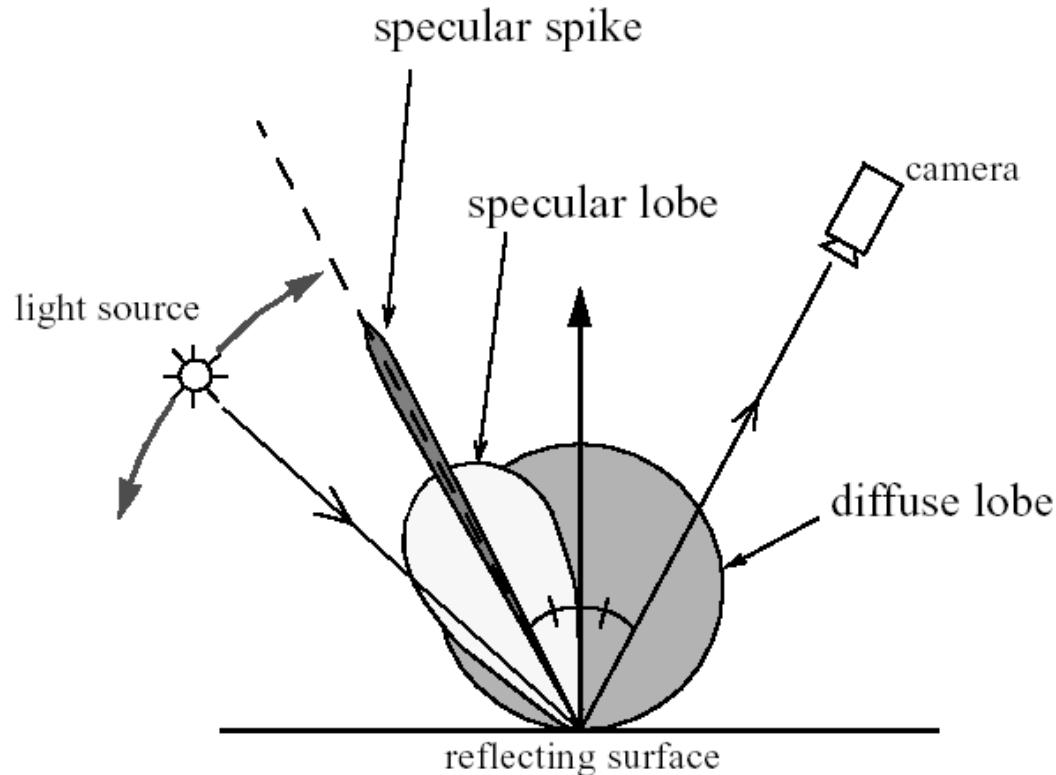
Phong Model

- These spheres illustrate the Phong model as *lighting direction* and n_{shiny} are varied:

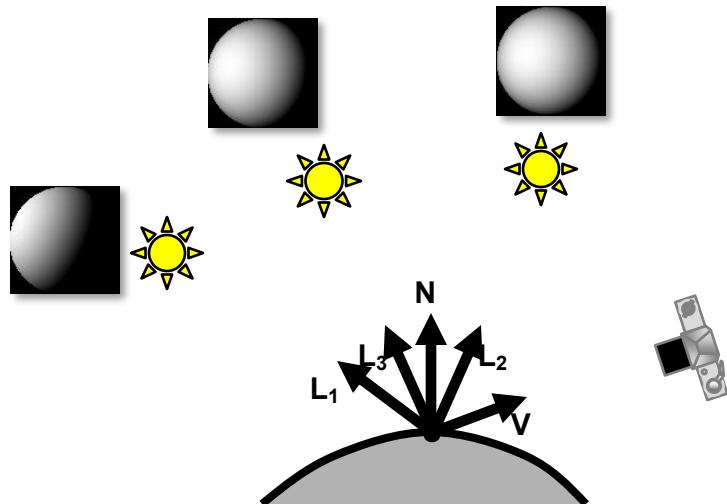


$$L = I \rho_s (\cos \phi)^{n_{shiny}}$$

All components of Surface Reflection



Multiple Images: Photometric Stereo



$$I_1 = k_d \mathbf{N} \cdot \mathbf{L}_1$$

$$I_2 = k_d \mathbf{N} \cdot \mathbf{L}_2$$

$$I_3 = k_d \mathbf{N} \cdot \mathbf{L}_3$$

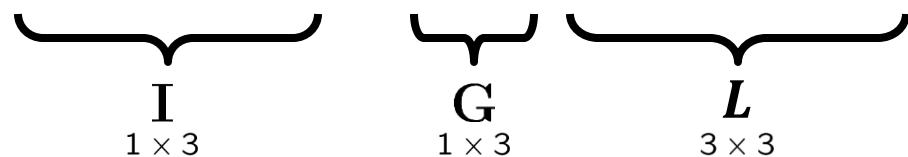
\mathbf{N} is the unknown that is

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

$$\left[\begin{array}{ccc} I_1 & I_2 & I_3 \end{array} \right] = k_d \mathbf{N}^T \left[\begin{array}{ccc} \mathbf{L}_1 & \mathbf{L}_2 & \mathbf{L}_3 \end{array} \right]$$



Solving:

$$\mathbf{G} = \mathbf{IL}^{-1}$$

Normalized the \mathbf{G} , separate it into mode k_d and normal \mathbf{N}

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

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

Solving the Equations

$$\left[\begin{array}{ccc} I_1 & I_2 & I_3 \end{array} \right] = k_d \mathbf{N}^T \left[\begin{array}{ccc} \mathbf{L}_1 & \mathbf{L}_2 & \mathbf{L}_3 \end{array} \right]$$

$\underbrace{}_{\mathbf{I}}$ $\underbrace{\phantom{\mathbf{L}_1}}_{\mathbf{G}}$ $\underbrace{\phantom{\mathbf{L}_1 \mathbf{L}_2}}_{\mathcal{L}}$

1×3 1×3 3×3

$$\mathbf{G} = \mathbf{IL}^{-1}$$

- 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

$$\begin{bmatrix} I_1 & \dots & I_n \end{bmatrix} = k_d \mathbf{N}^T \begin{bmatrix} \mathbf{L}_1 & \dots & \mathbf{L}_n \end{bmatrix}$$

- Solve using least squares (normal equations):

$$\mathbf{I} = \mathbf{G}\mathbf{L}$$

Find \mathbf{G} s.t. minimizes $\|\mathbf{I} - \mathbf{G}\mathbf{L}\|^2$

$$\mathbf{I}\mathbf{L}^T = \mathbf{G}\mathbf{L}\mathbf{L}^T$$

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

- Given \mathbf{G} , solve for \mathbf{N} and k_d as before.

More than one pixel

Previously:

$$\begin{matrix} 1 \times \# \text{ images} \\ \boxed{I} \end{matrix} = \boxed{N} * \begin{matrix} 1 \times 3 \\ 3 \times \# \text{ images} \\ \boxed{L} \end{matrix}$$

More than one pixel

Stack all pixels into one system:

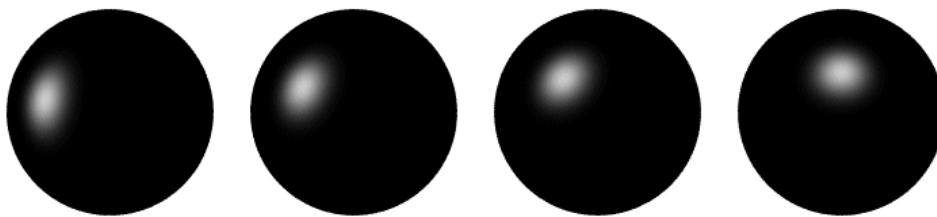
$$\begin{matrix} p \times \# \text{ images} \\ | \\ \boxed{\quad} \end{matrix} = \boxed{N} * \boxed{L}$$

The equation illustrates the decomposition of a full image matrix I into a product of two matrices N and L . Matrix I has dimensions $p \times \# \text{ images}$. Matrix N has dimensions $p \times 3$, and matrix L has dimensions $3 \times \# \text{ images}$.

Solve as before.

Computing Light Source Directions

- Trick: Place a mirror ball in the scene.



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

HDRI Sky Probe



Real-World HDR Lighting Environments

Funston
Beach



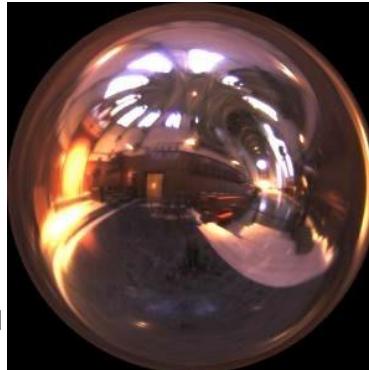
Eucalyptus
Grove



Uffizi
Gallery



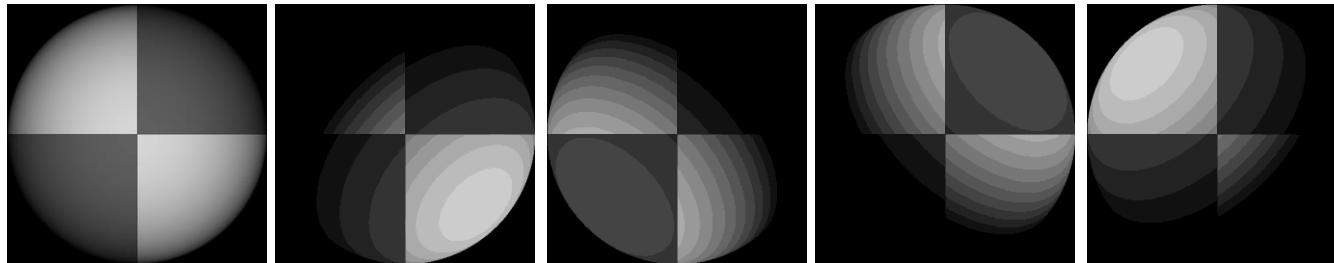
Grace
Cathedral



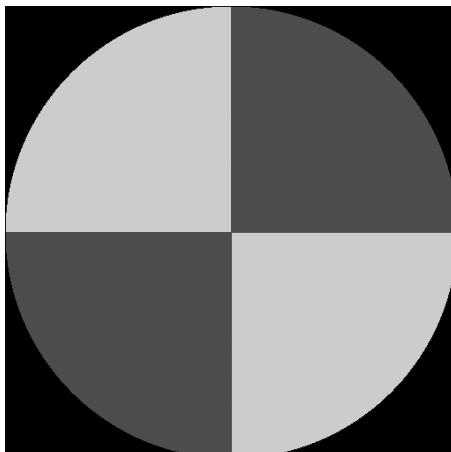
Lighting Environments from the Light Probe Image Gallery:
<http://www.debevec.org/Probes/>

Example

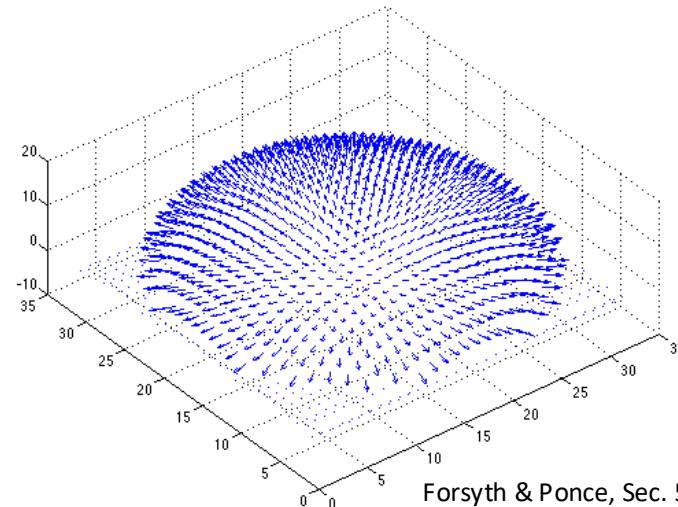
Input views



Recovered albedo



Recovered normal field



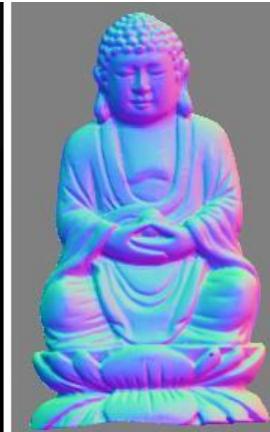
Forsyth & Ponce, Sec. 5.4

Depth from normals

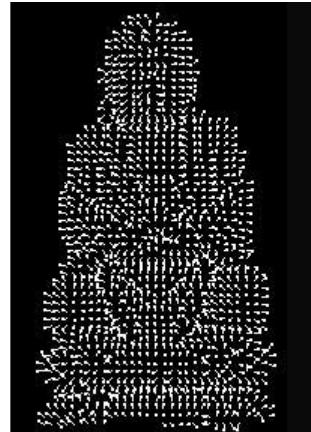
- Solving the linear system per-pixel gives us an estimated surface normal for each pixel
- How can we compute depth from normals?
 - Normals are like the “derivative” of the true depth



Input photo

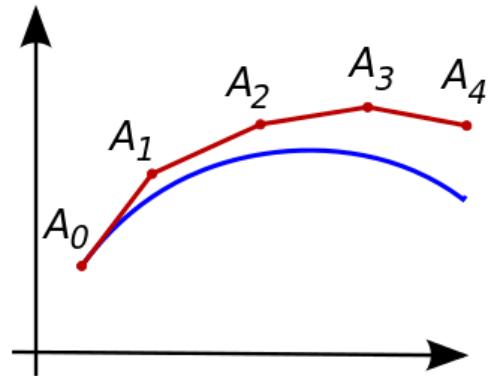


Estimated normals

Estimated normals
(needle diagram)

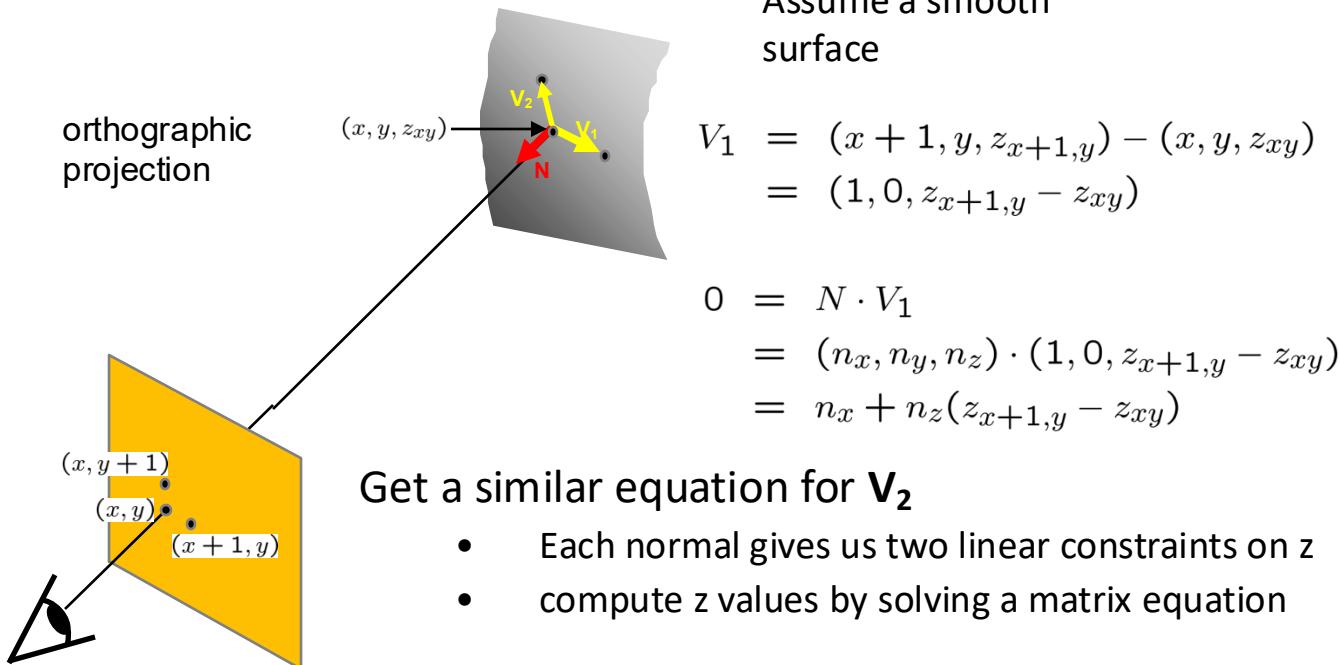
Normal Integration

- Integrating a set of derivatives is easy in 1D
 - (similar to Euler's method from diff. eq. class)
- Could integrate normals in each column / row separately
 - Wouldn't give a good surface
- Instead, we formulate as a linear system and solve for depths that *best agree with the surface normals*

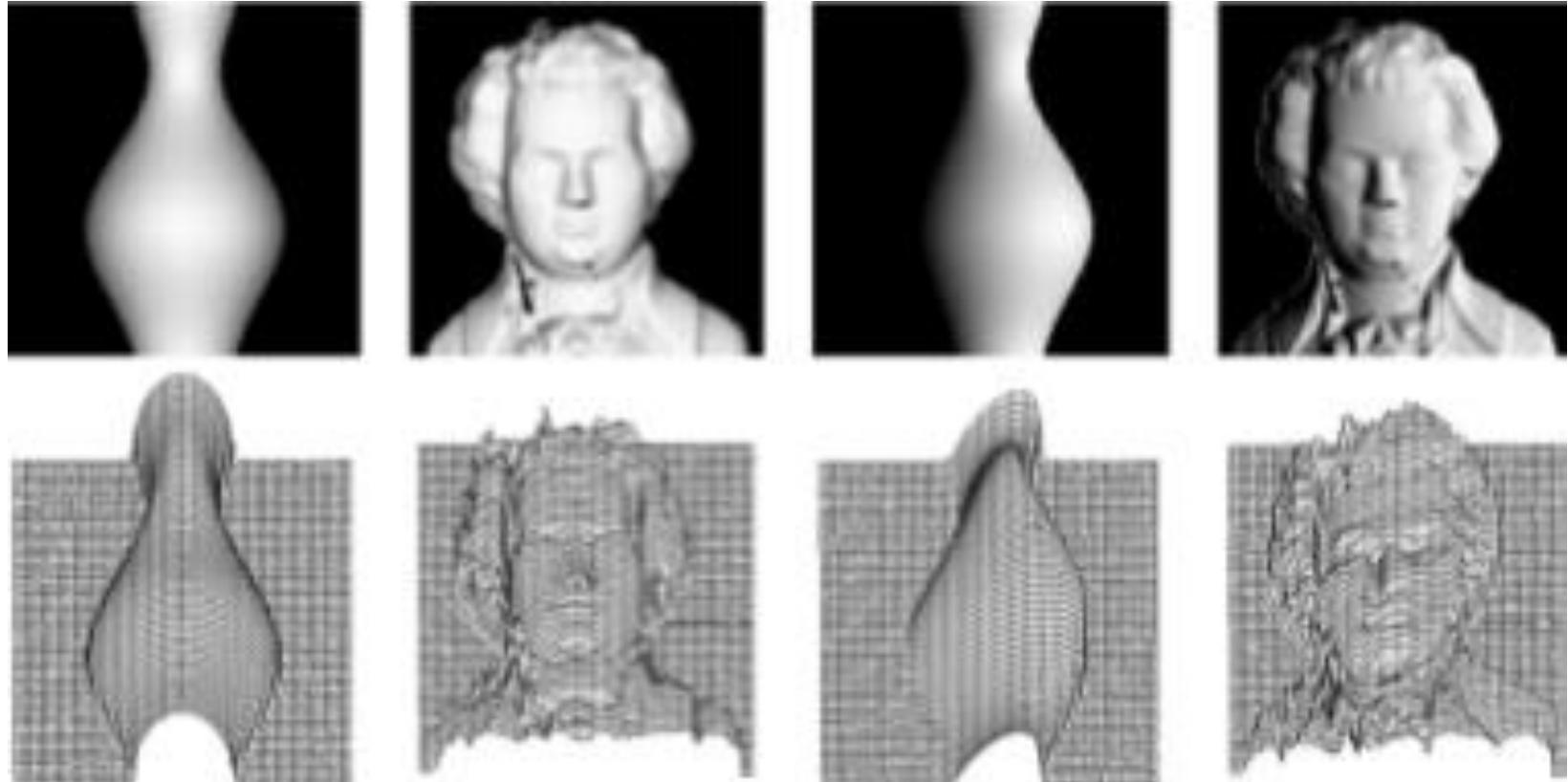


Depth Map from Normal Map

- Given the surface normals we can iteratively compute depth?

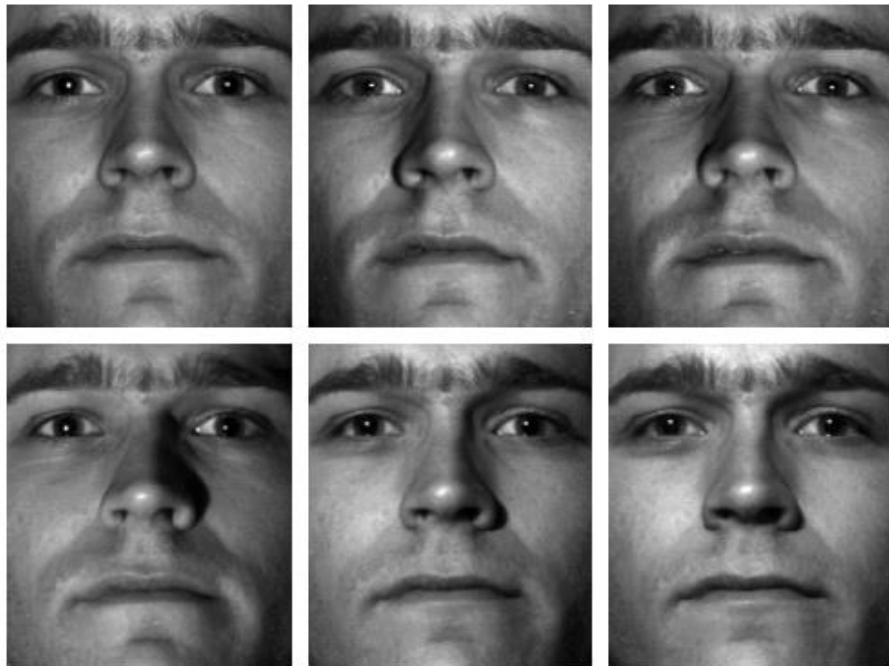


Results



[Slide credit: Andreas Geiger] [Frankot and Chellappa: A Method for enforcing integrability in shape from shading algorithms. TPAMI, 1988]

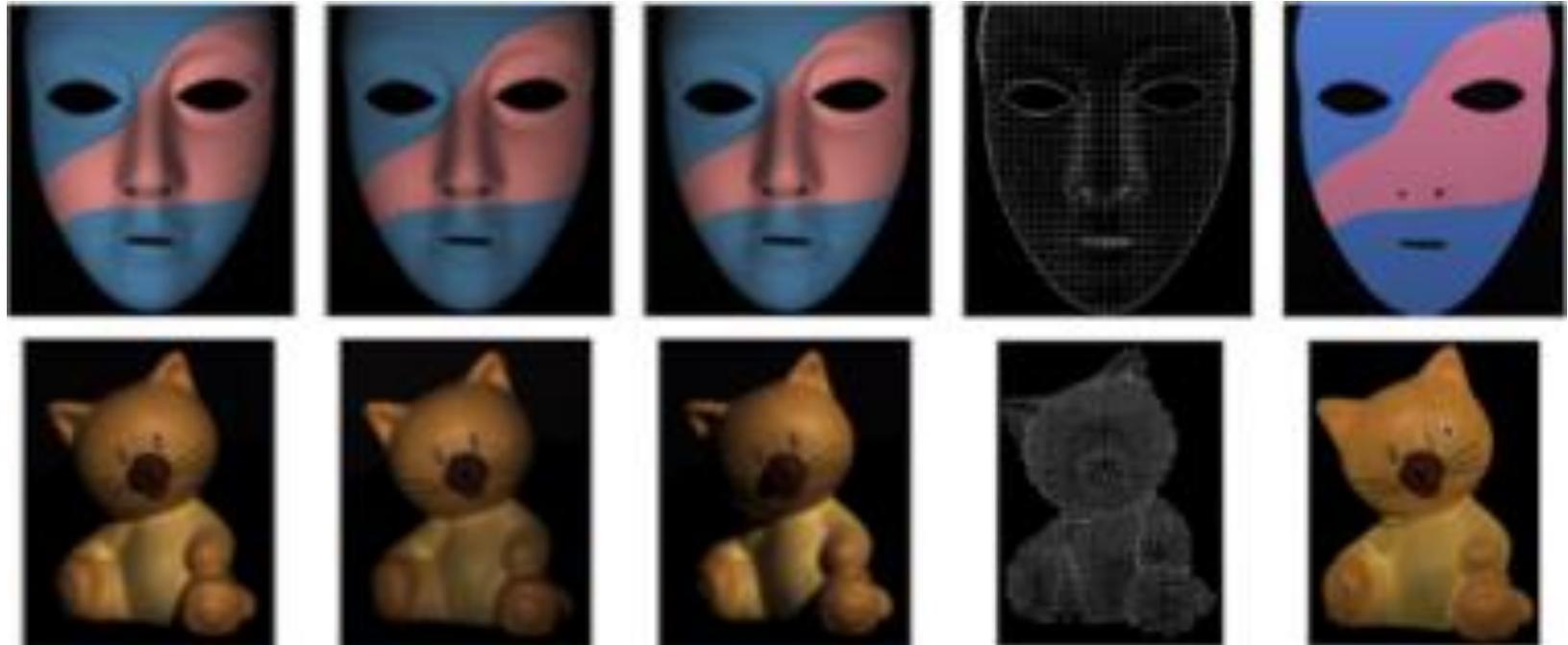
Results



Results



Results



- For color images, apply PS to each channel separately to obtain color albedo.
- Deviations from Lambertian assumption and global illumination cause errors.

Extension

- Photometric Stereo from Colored Lighting

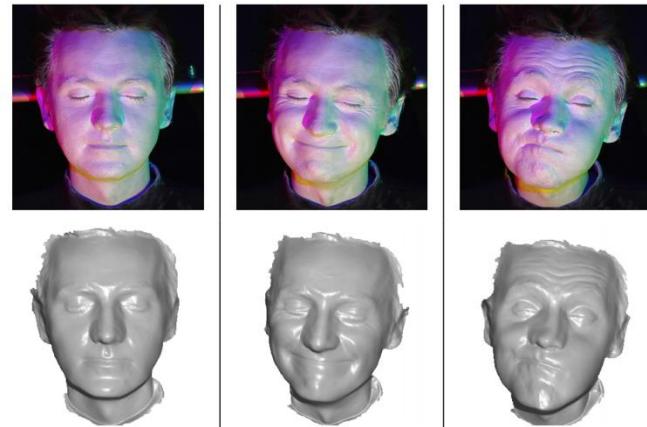
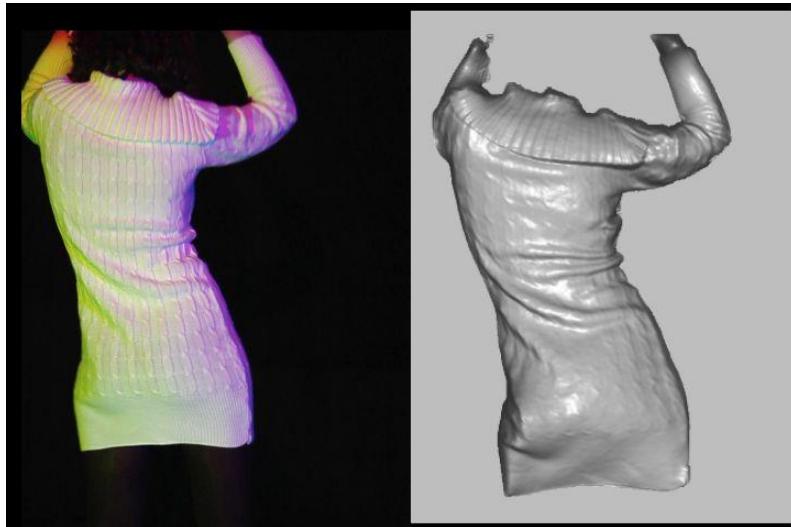


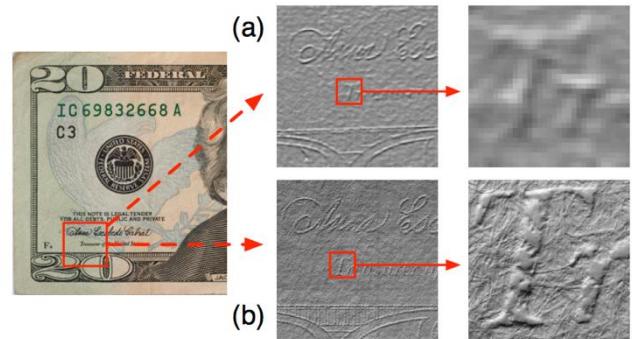
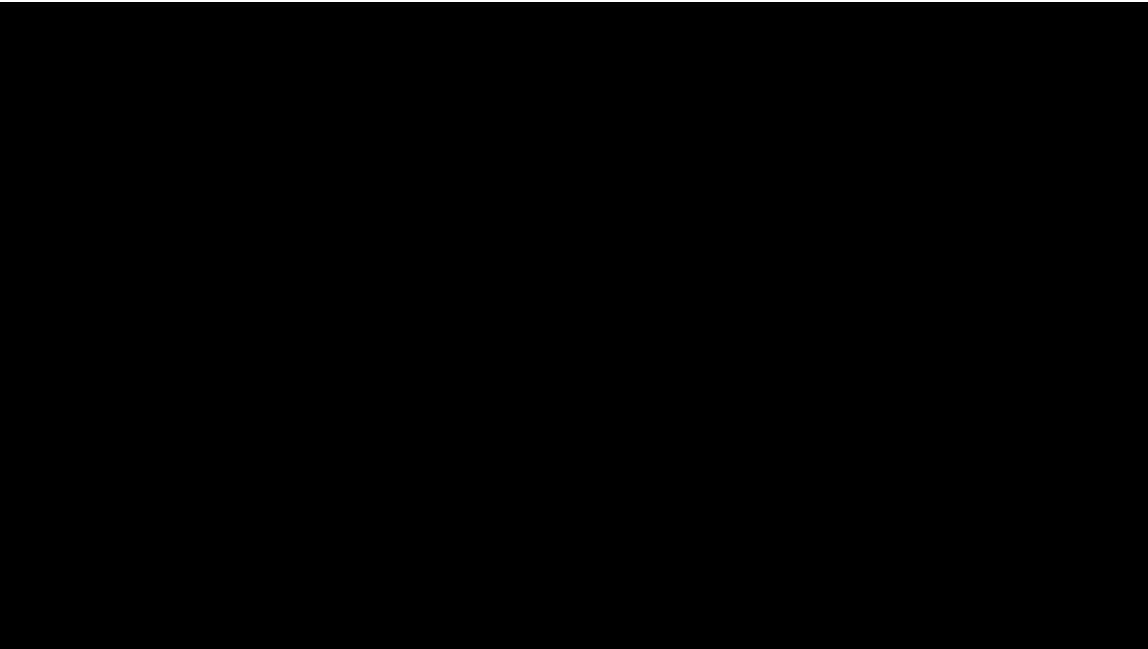
Fig. 2. Applying the original algorithm to a face with white makeup.
Top: example input frames from video of an actor smiling and grimacing.
Bottom: the resulting integrated surfaces.

Video Normals from Colored Lights

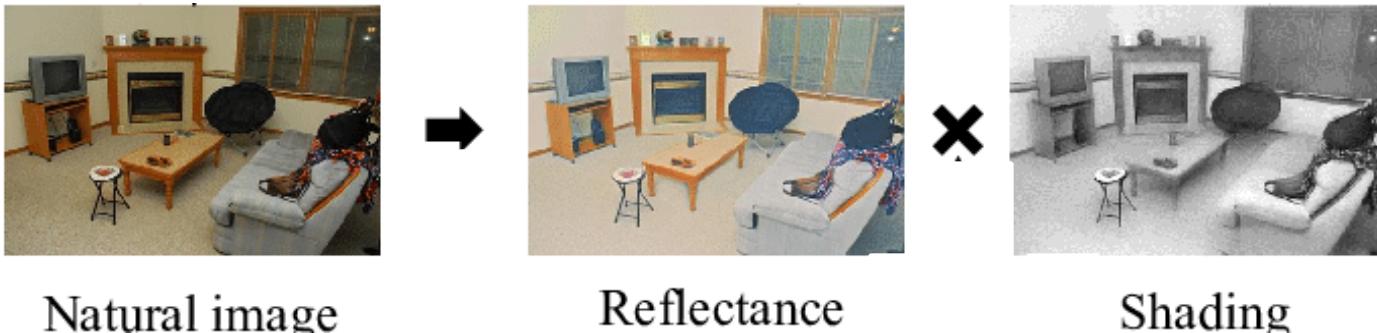
Gabriel J. Brostow, Carlos Hernández, George Vogiatzis, Björn Stenger, Roberto Cipolla
[IEEE TPAMI](#), 2011.

Specialized High-Resolution Scanners

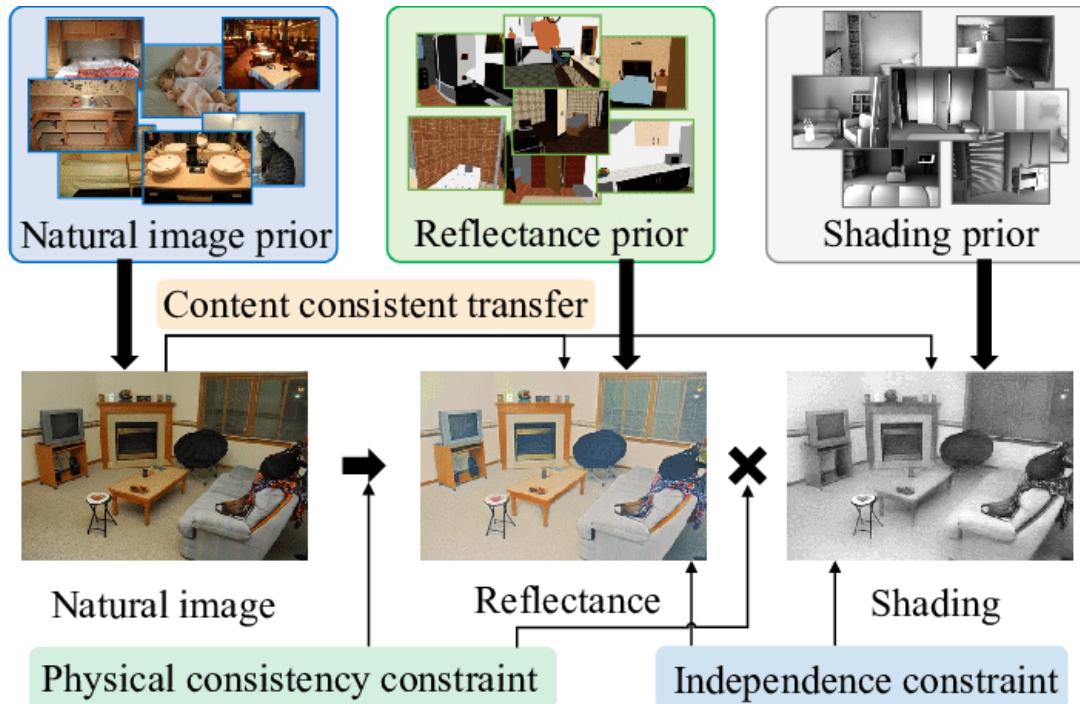
- Recovers extremely subtle details with high accuracy
- Photometric stereo yields unmatched performance



Intrinsic Image Decomposition

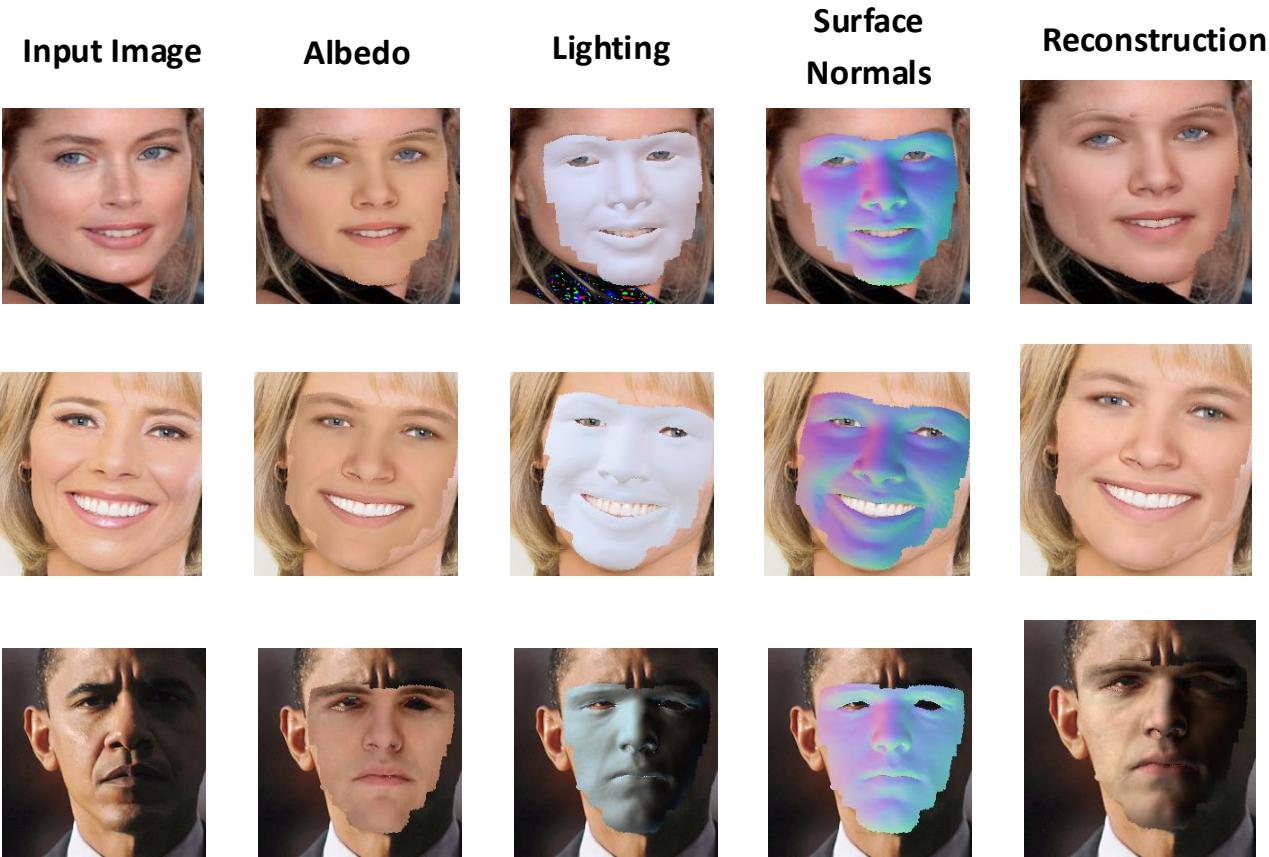


Intrinsic Image Decomposition



Unsupervised Learning for Intrinsic Image Decomposition from a Single Image
Yunfei Liu, Yu Li, Shaodi You and Feng Lu, CVPR2020

Diffuse Reflection



Total Relighting

Total Relighting:
Learning to Relight Portraits for Background Replacement



Rohit Pandey*, Sergio Orts Escolano*, Chloe LeGendre*, Christian Haene,
Sofien Bouaziz, Christoph Rhemann, Paul Debevec, and Sean Fanello



Total Relighting

Deep Relighting Module - Design

Several CNNs with U-Net architecture in series

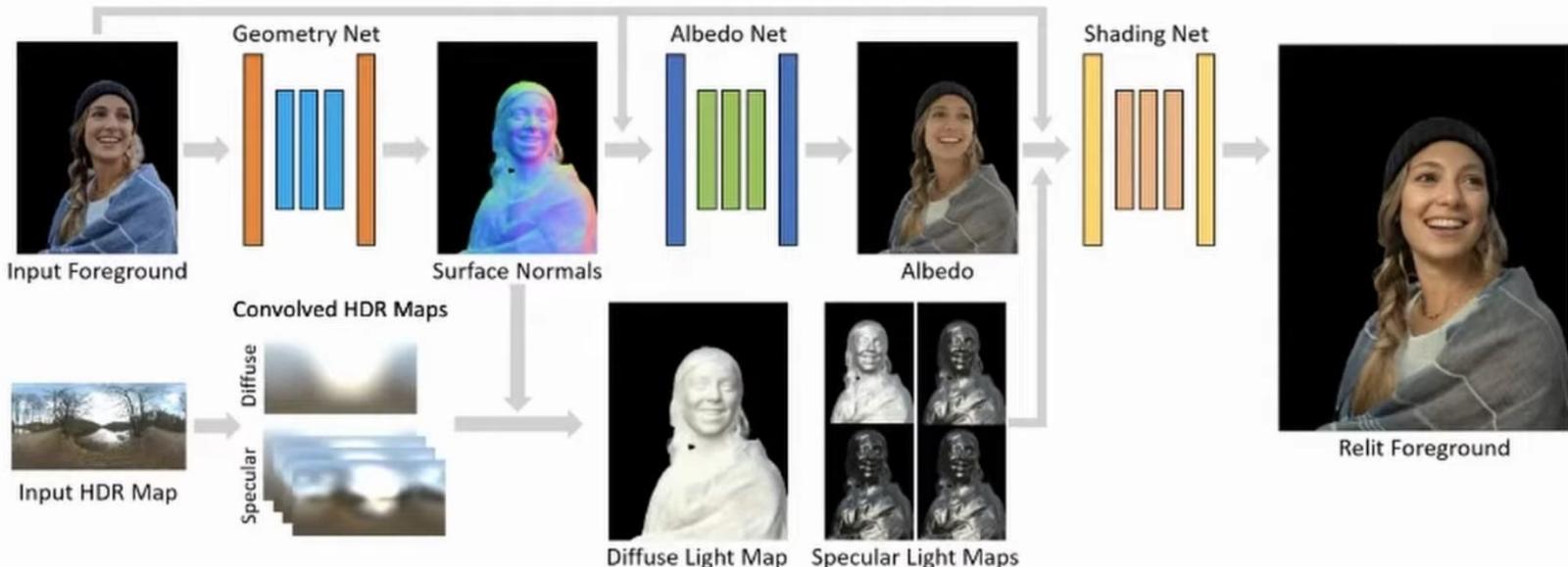


Image processing operations

- Point-wise operations
 - histogram operation
- Neighborhood operations
 - Spatial domain (image convolution)
 - Frequency domain (Fourier transform)
 - Binary image analysis
- Geometric operations
 - Rotation, translation, affine...

Histograms

How do we ‘summarize’ images?



← https://rvdboomgaard.github.io/ComputerVision_LectureNotes/LectureNotes/IP/Images/ImageHistograms.htm
https://rvdboomgaard.github.io/ComputerVision_LectureNotes/LectureNotes/IP/PointOperators/index.html →



Image Histograms

1D / Univariate Histogram

Summarize image f via **Histogram** of its **values**
 (over all possible values)

$$h_f[i] = \sum_{\substack{\text{for} \\ i \\ = 0, \dots, k-1}}_{x \in E} [e_i \leq f(x) < e_{i+1}]$$

Iverson Bracket Is pixel value inside bin?
 $f(x) \in [e_i, e_{i+1}]$
 Bin edges
 $e_i \text{ for } i = 0, \dots, k$

$$[c] = \begin{cases} 1 & \text{for } c = \text{true} \\ 0 & \text{for } c = \text{false} \end{cases}$$



NumPy: **histogram()**



<https://numpy.org/doc/stable/reference/generated/numpy.histogram.html>

How choose **bin size/number?**

- No clear answer
- Empirically → More art than science
- **Sturges' rule** (assumes *Gaussian* data):

$$k = \lceil \log_2 n \rceil + 1$$

k
 bins number pixels number

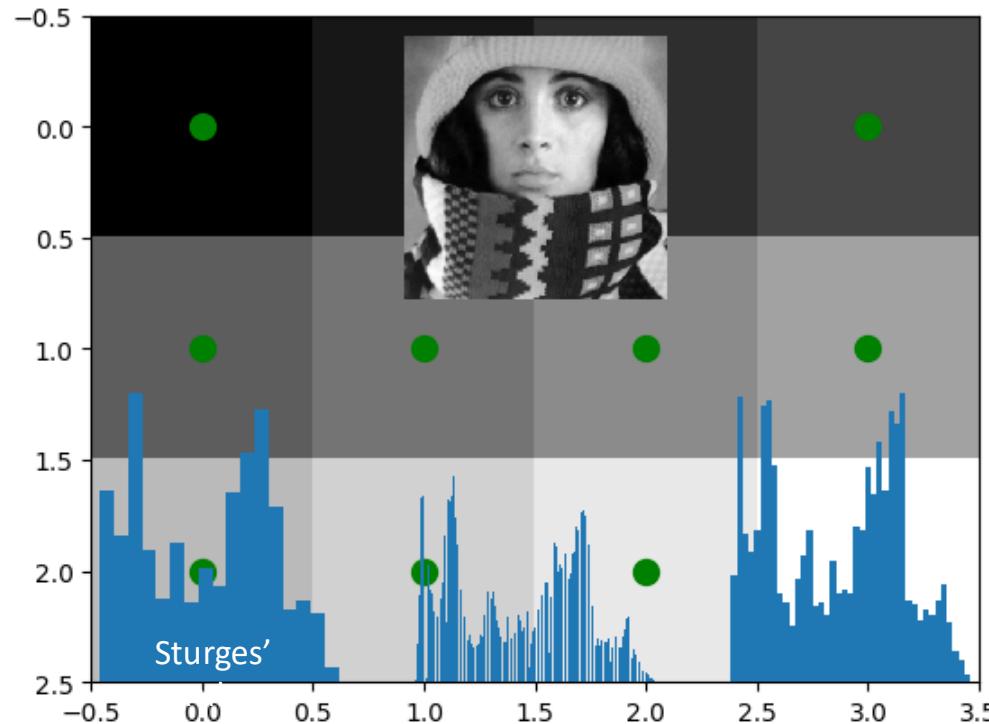


histogram_bin_edges()

https://numpy.org/doc/stable/reference/generated/numpy.histogram_bin_edges.html

Image Histograms

1D / Univariate Histogram



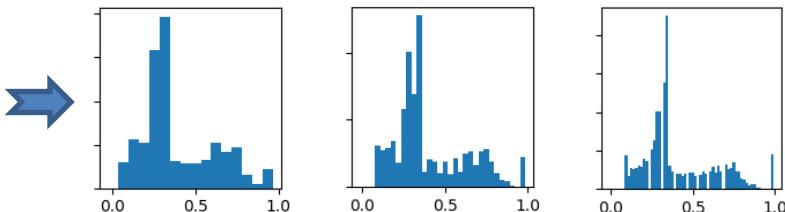
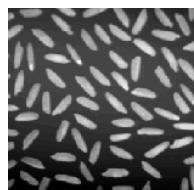
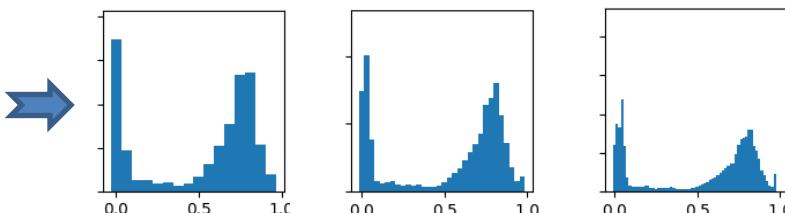
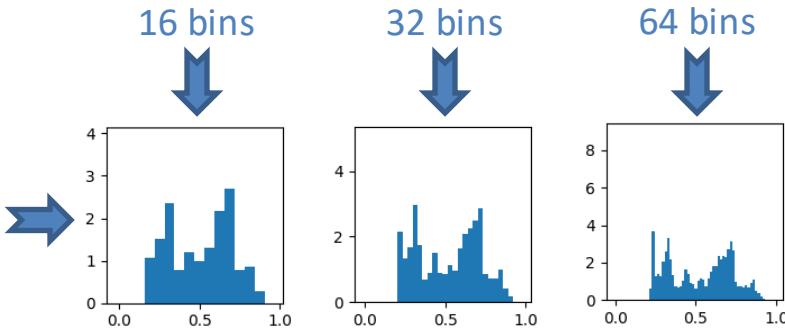
Too many (and too small) bins?

- Many bins will be empty
- 'High-frequency' info

Too few (and too big) bins?

- Will not encode 'well enough'
- the distribution of values

Image Histograms



Too many (and too small) bins?

- Many bins will be **empty**
- 'High-frequency' info

Too few (and too big) bins?

- Will **not encode 'well enough'**
the distribution of values

Image Histograms

Histogram →

$h_f(u)$ counts pixels with value $f(x) = u$

PDF – Probability-density function

normalized such that: $\sum h_f(u) = 1$

Cumulative Histogram →

$H_f(u)$ counts pixels with value $f(x) \leq u$

Cumulative distribution function

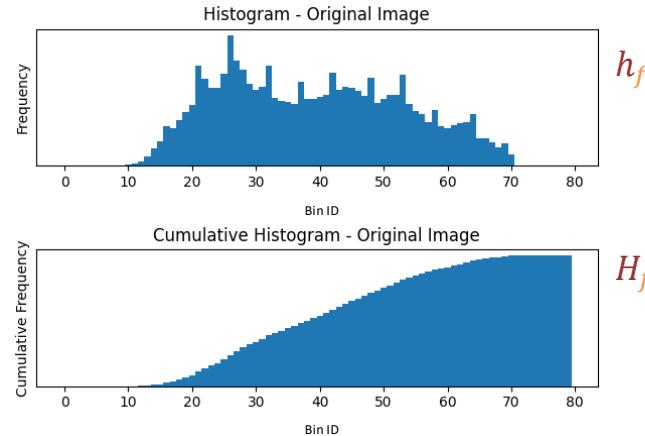


Image Histograms

3D / Multivariate Histogram

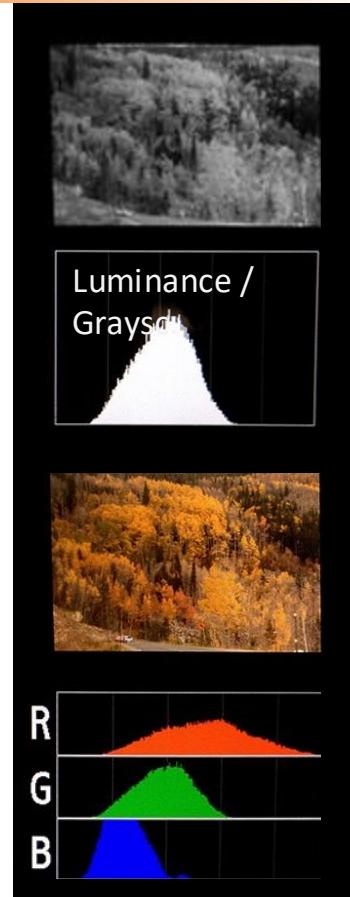
Grayscale images

(each pixel → Scalar value)



1D Histogram

(Univariate)



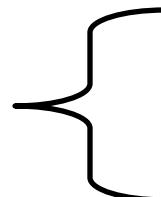
What about **Color** images?

(each pixel → Triplet of BGR value)

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} b(\mathbf{x}) \\ g(\mathbf{x}) \\ r(\mathbf{x}) \end{bmatrix}$$



Ignores color
Correlations!



Trivial
solution:
3x separate
1D histograms
per channel

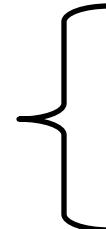


Image Histograms

3D / Multivariate Histogram

Better solution → *3D Histogram*

$$h_f[i, j, k] = \sum_{x \in E} [e_{1,i} \leq f_1(x) < e_{1,i+1}] [e_{2,j} \leq f_2(x) < e_{2,j+1}] [e_{3,k} \leq f_3(x) < e_{3,k+1}]$$



Is pixel value inside
3D bin
with indices (i, j, k) ?

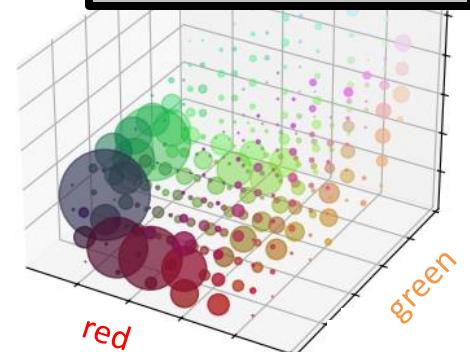


histogramdd()

<https://numpy.org/doc/stable/reference/generated/numpy.histogramdd.html>



3D bins: 8x8x8 cubes



b
l
u
e

green

red

Image Histograms

Online Demo - 1D Histograms

'This tool basically creates 256 bins for each color (red, green, blue) and greyscale (luma) intensity. The number of bins is shown on the horizontal axis.'

'The tool then loops through every image pixel and counts the occurrence of each intensity. The counts of occurrences in each bin are then displayed on vertical axis.'

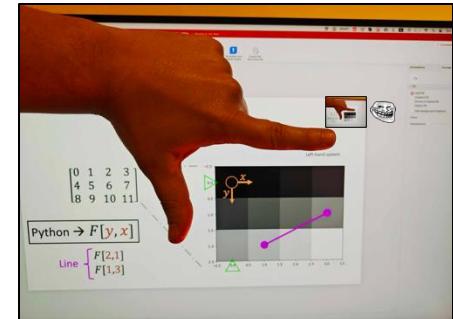
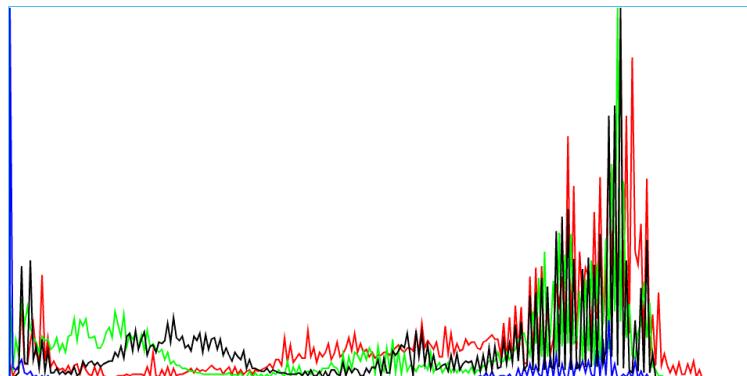
'Counts for each pixel-intensity are normalized to range 0 to 255 before they are displayed on the graph.'

'Used JavaScript in combination with WebAssembly to create this tool.'

SISIK

Export to csv

Red Green Blue Luma Frame Fill



<https://sisik.eu/histo>



Point Operators

Histograms

$$\left. \begin{array}{l} f: \mathcal{D} \rightarrow ([0,1] \in \mathbb{R}) \\ \varphi: ([0,1] \in \mathbb{R}) \rightarrow ([0,1] \in \mathbb{R}) \end{array} \right\} (\varphi f)(x) = \varphi(f(x))$$

Monadic
'in-place'
Point Operator

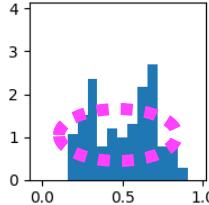
Replaces pixel value $f(x)$ **independent** of:

- Position x
- All other pixel values in neighborhood

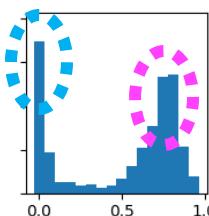
How can we 'manipulate' histograms with Monadic Point Operators?

Point Operators

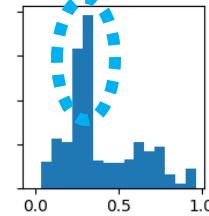
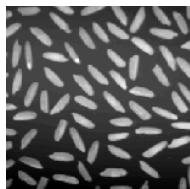
Histograms (issues)



Only *mid-range* gray
No extremes



Spikes for *bright* and
very-dark areas



Mostly *dark-ish* areas

Not using full range
of luminance
might be sub-optimal

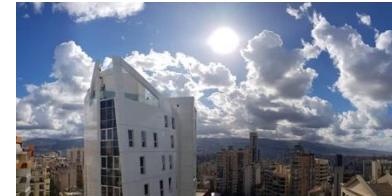


Missing *large (bright)*
values?
Missing *low (dark)* values?

→ ... too *dark*
→ ... too *bright*

Image might be ... :

Changing illumination conditions
for same scene → very different histograms!



Point Operators

Histogram Contrast Stretching

$$f(x): \mathbb{R}^2 \rightarrow [a, b] \subset [0, 1] \in \mathbb{R}$$

Contrast Stretching { Construct new image using full range of luminance

Monadic Operator

$$g(x) = \varphi(f(x)) = \frac{f(x) - a}{b - a}$$

'Distance' from min luminance in f

Stretches luminance range to $[0, 1]$

$$g = \frac{f - f_{\min}}{f_{\max} - f_{\min}}$$

Lifted arithmetical operators

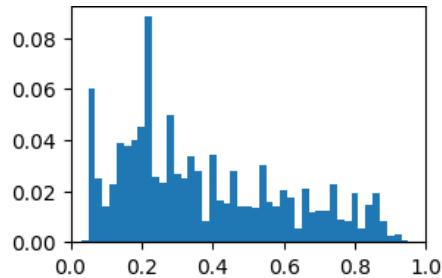
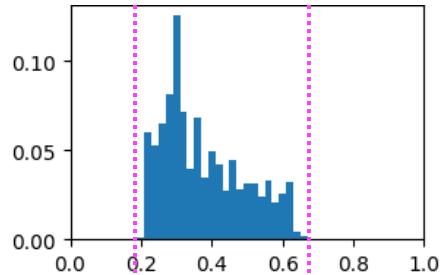
Input:
Low Contrast



Output:
Contrast Stretched

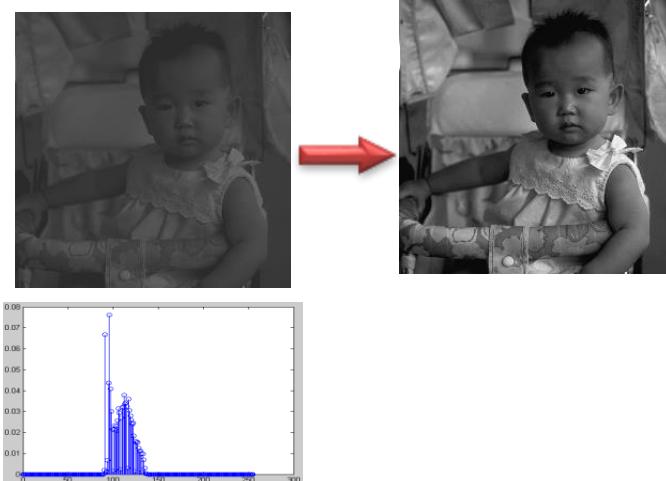
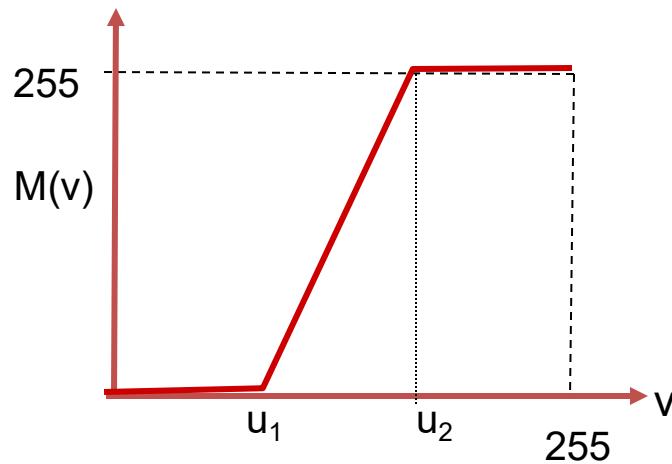


X axis: luminance values (bins)
Y axis: frequency for each value



Contrast Enhancement

- If most of the gray-levels in the image are in $[u_1 \ u_2]$, the following mapping increases the image contrast.
- The values u_1 and u_2 can be found by using the image's accumulated histogram.



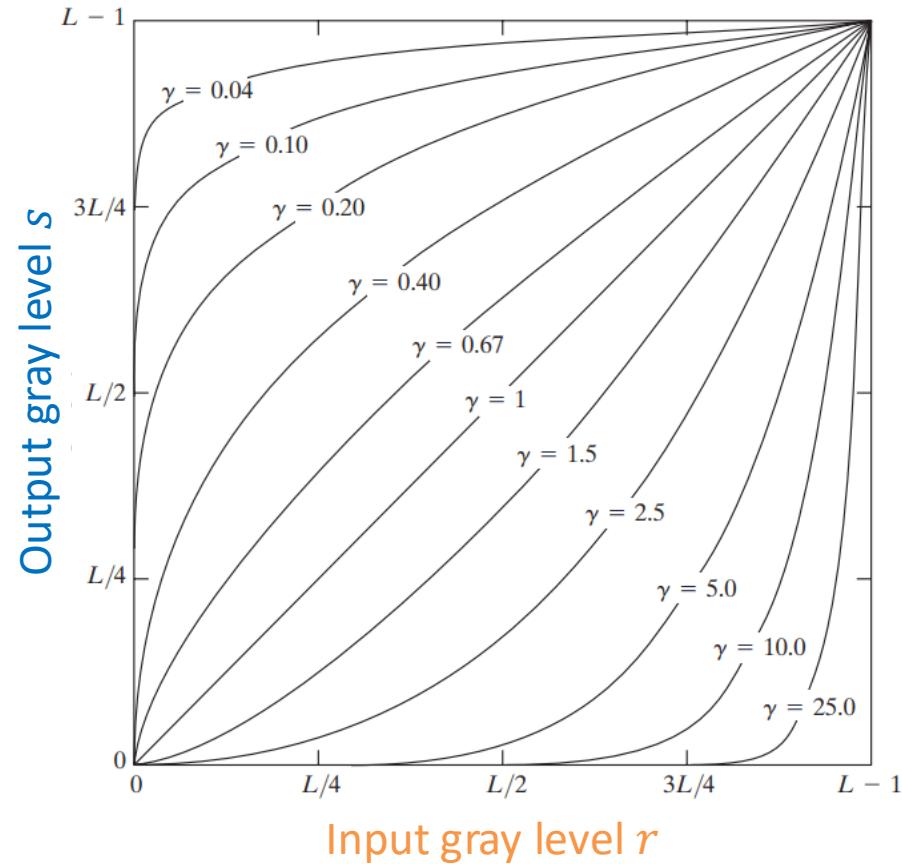
Gamma Correction

Non-linear scaling (power-law transformation):

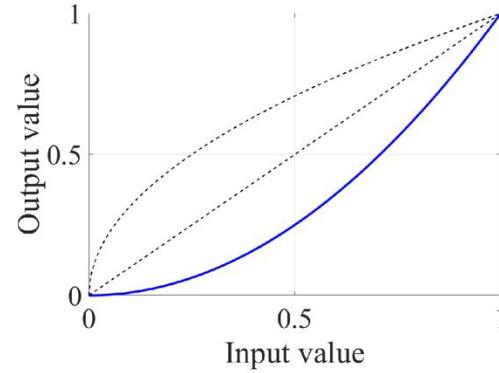
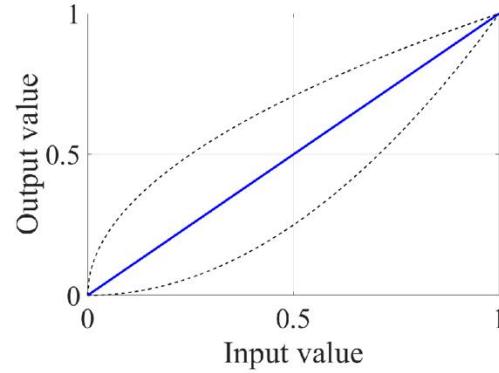
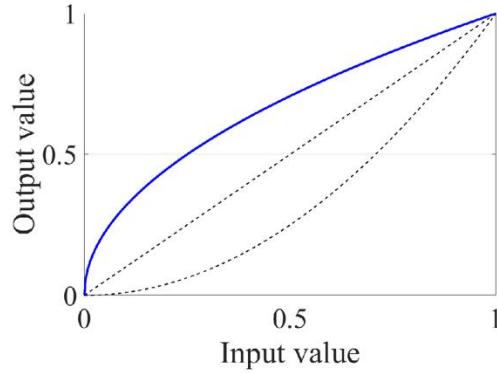
$$\text{Output gray level } s(r) = c r^\gamma$$

Normalization constant Input gray level

- Note: $s(r)$ is linear for $\gamma = 1$.



Gamma Correction



[Ngo and Kang, Electronics, 2021]

Gamma Correction

gamma=1.0



gamma=3.0



gamma=5.0



gamma=5.0

Point Operators

Histogram Equalization (idealized)

Produce image $g = \varphi f$ with '**constant' (flat) histogram** h_g

All luminance values should be 'equally' probable
(as 'equal' as possible for finite number of pixels)

Useful consequence: **cumulative histogram** $H_g(u) = u$!

Why? Think: integrate a **constant**

Luminance value: $u = f(x)$

Preserve **order**: $u_1 \leq u_2 \Rightarrow \varphi(u_1) \leq \varphi(u_2)$

So, the p^{th} percentile of f & g should be the same:

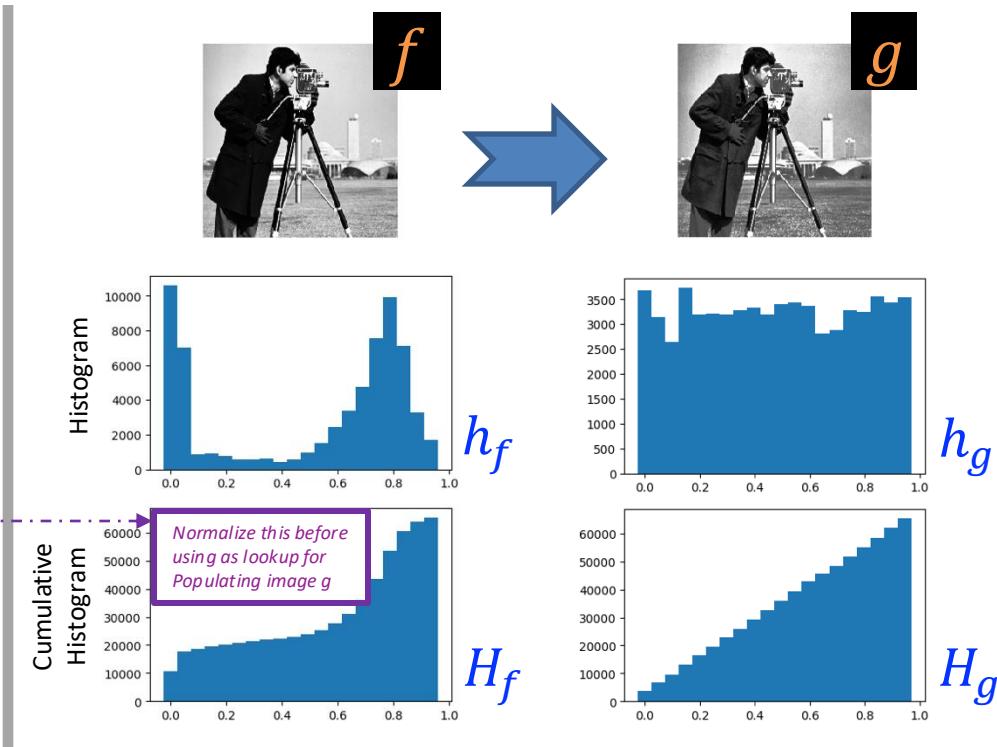
$$H_f(u) = H_g(\varphi(u)) = \varphi(u)$$

Visit each pixel of f
replace its value u with:

$$\varphi(u) = H_f^{\text{norm}}(u)$$

To populate a pixel of g :

- visit the corresponding pixel location of f
- read its value u (i.e., $f(x)$)
- use u as key to 'query' (the *normalized*) H_f as "lookup table"
- Copy value from "lookup table" as value of pixel of g



Point Operators

Histogram Thresholding

Segment foreground (objects of interests) from background

- Manually ☺ choose threshold t → looking at histogram (bimodal distribution)
- Automatically** choose threshold t → *IsoData* thresholding

$m_L(t)$ → Mean of pixels with value: $\leq t$

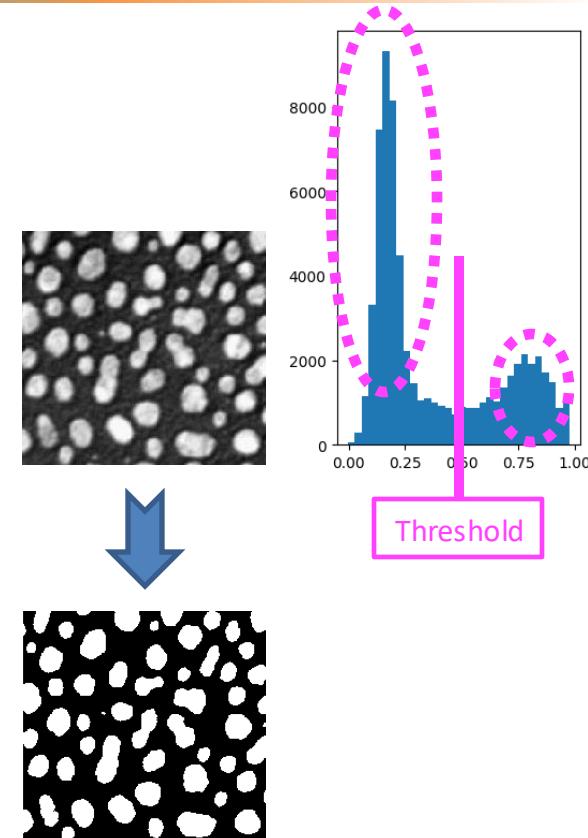
$m_H(t)$ → Mean of pixels with value: $> t$

$$t = \frac{m_L(t) + m_H(t)}{2} = m(t) \quad \text{Function of itself!}$$



Iterative Solution

- Start with initial guess t_i for $i = 0$
- Compute $t_{i+1} = m(t_i)$
- Repeat until convergence
(i.e. repeat as long as t_{i+1} and t_i are not too similar)



Resources for Histogram Slides



https://rvdboomgaard.github.io/ComputerVision_LectureNotes/LectureNotes/IP/Images/index.html



https://rvdboomgaard.github.io/ComputerVision_LectureNotes/LectureNotes/IP/PointOperators/index.html

Additional Resources

- Photometric Stereo
 - Shree Nayar: [YouTube](#)
 - Andreas Geiger: [YouTube](#)
- Specular and Diffuse Reflection
 - Steve Seitz: [YouTube](#) / [YouTube](#)

Disclaimer

Many of the slides used here are obtained from online resources (including many open lecture materials) without appropriate acknowledgement. They are used here for the sole purpose of classroom teaching. All the credit and all the copyrights belong to the original authors. You should not copy it, redistribute it, put it online, or use it for any other purposes than for this course.