Computer Vision 1

THEO GEVERS

MASTER AI

UNIVERSITY OF AMSTERDAM

Lectures/Theory

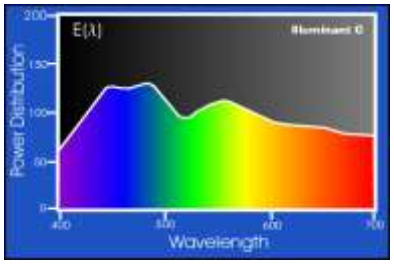
- 06-02-2018, 17:00-19:00, C0.05, **Introduction** (*Szeliski 1*)
- 13-02-2018, 17:00-19:00, C0.05, **Image Formation** (*Szeliski: 2.1.1 + 2.1.2 + 2.2 + 2.3.2 + 2.3.3*)
- 20-02-2018, 17:00-19:00, C0.05, **Color and Image Processing** (*Szeliski: 3.1 + 3.2 + 3.3*)
- 27-02-2018, 17:00-19:00, C0.05, **Feature Detection, Motion and Classification** (*Szeliski: 4, 8.1.1 + 8.1.3 + 8.2.1 + 8.4; Bengio: 4 + 5.1 + 5.2 + 5.3 + 5.7 + 5.8 + 5.9*)
- 06-03-2018, 17:00-19:00, C0.05, **Object Recognition: BoW and ConvNets** (*Szeliski: 5.1.1 + 5.1.4 + 5.1.5 + 5.2 + 5.3 + 5.4, 6.1 + 6.3, 14.1 + 14.2.1 + 14.3 + 14.4.1; Bengio: 7.2 + 7.4 + 9.1 + 9.2 + 9.3*)
- 13-03-2018, 17:00-19:00, C0.05, **Deep Learning, Stereo and 3D Reconstruction** (*Szeliski: 11.1 + 11.2 + 11.3 + 11.4, 12.1 + 12.2: Bengio: 12.1 + 12.2*)
- 20-03-2018, 17:00-19:00, C0.05, **Applications** (*Szeliski: 12.6.2 + 12.6.3 + 12.2.4)*
- 26-03-2018, Monday, 9:00-12:00, **Written Exam**

Today's Class

1. **Reflection Models**
2. **Color Invariance**
3. **Image Processing**

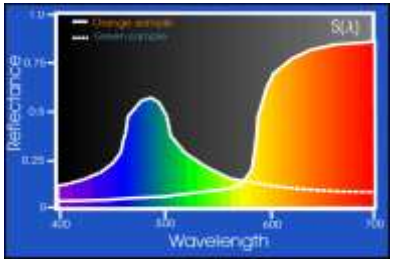
What makes an image

Light source



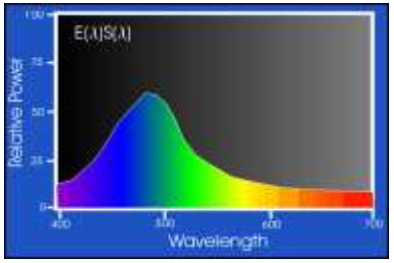
$e(\lambda)$

Object



$\rho(\lambda)$

Sensor



$f_C(\lambda)$

$$R = \int_{\lambda} e(\lambda)\rho(\lambda) f_R(\lambda) d\lambda, \quad G = \int_{\lambda} e(\lambda)\rho(\lambda) f_G(\lambda) d\lambda, \quad B = \int_{\lambda} e(\lambda)\rho(\lambda) f_B(\lambda) d\lambda$$

Recap: What makes an image?
the triplet light-objects-observer

$$R = \int_{\lambda} e(\lambda)\rho(\lambda) f_R(\lambda) d\lambda, \quad G = \int_{\lambda} e(\lambda)\rho(\lambda) f_G(\lambda) d\lambda, \quad B = \int_{\lambda} e(\lambda)\rho(\lambda) f_B(\lambda) d\lambda$$

Basic Principles of Surface Reflectance

Slides from Shree Nayar, Ravi Ramamoorthi, Pat Hanrahan

Surface Appearance

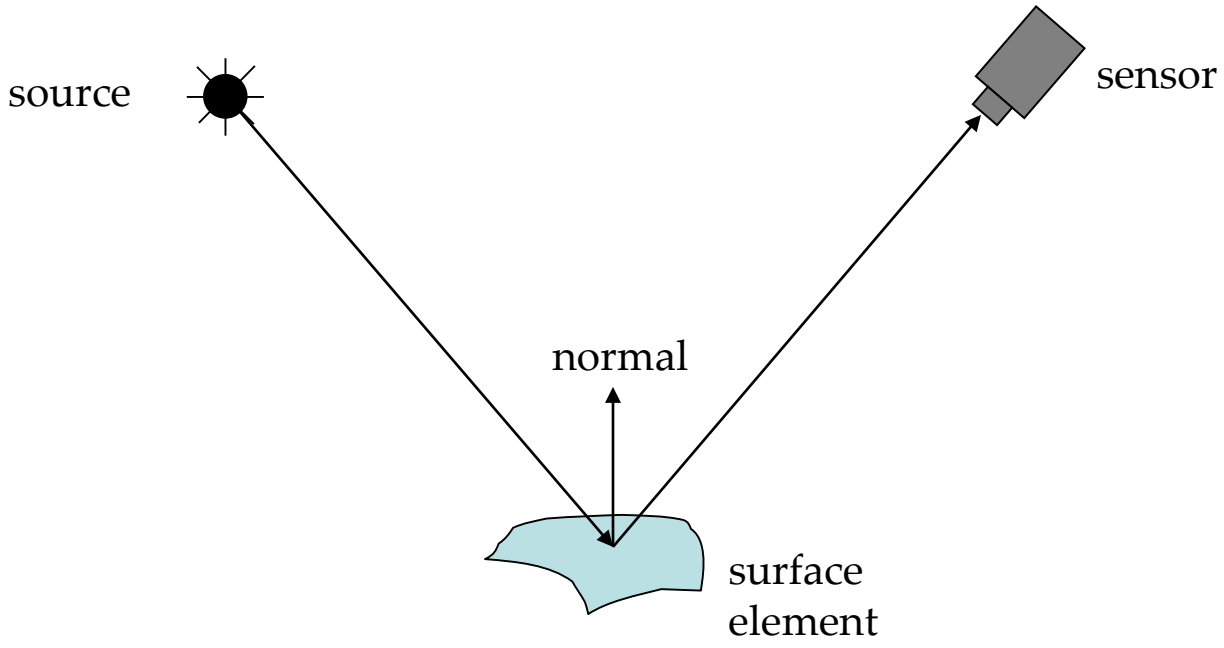
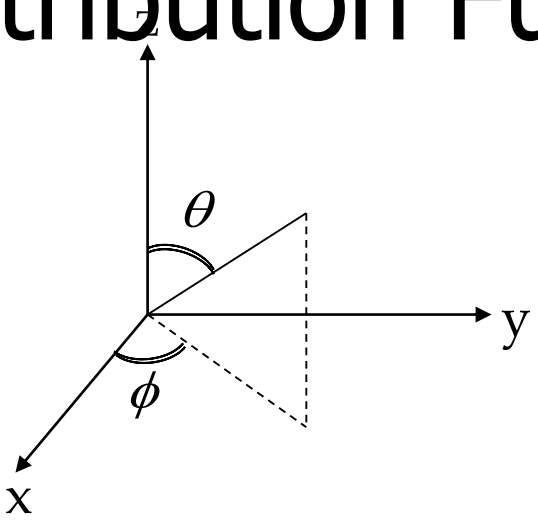
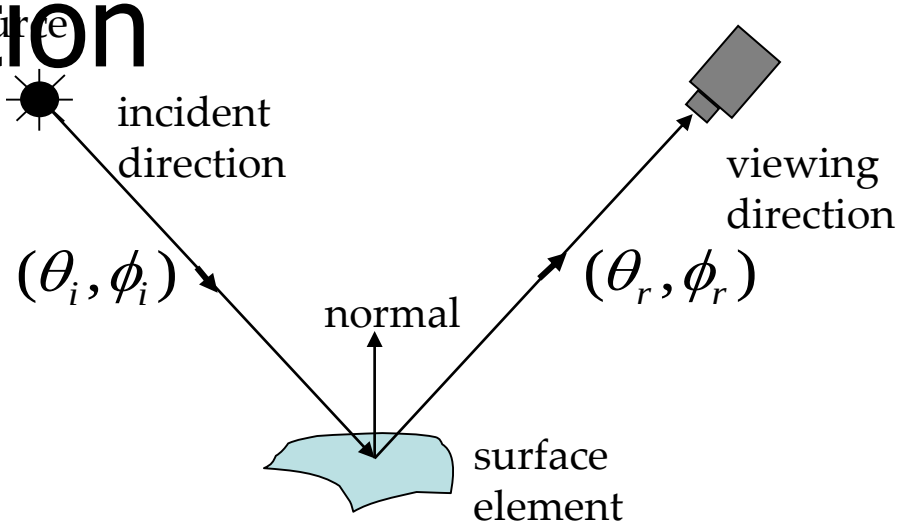


Image intensities = f (normal, surface reflectance, illumination)

Surface Reflection depends on both the viewing and illumination direction.

BRDF: Bidirectional Reflectance Distribution Function



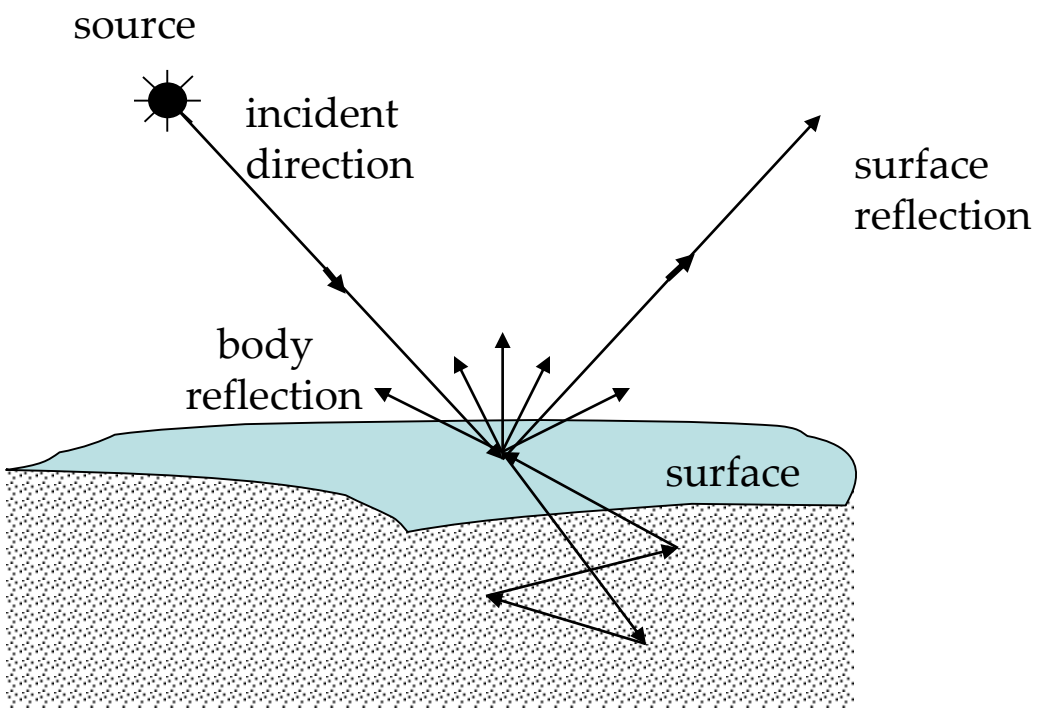


$E^{surface}(\theta_i, \phi_i)$ Irradiance at Surface in direction (θ_i, ϕ_i)

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

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

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

Image Intensity = Body Reflection + Surface Reflection

Mechanisms of Surface Reflection

Body Reflection:

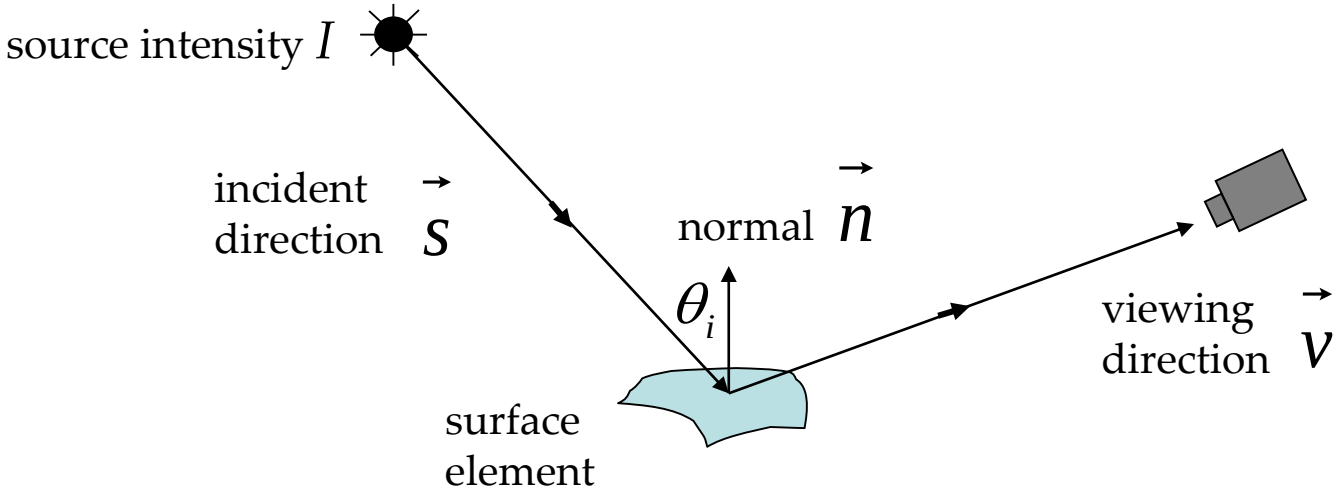
Diffuse Reflection
Matte Appearance
Non-Homogeneous Medium
Clay, paper, etc

Surface Reflection:

Specular Reflection
Glossy Appearance
Highlights
Dominant for Metals

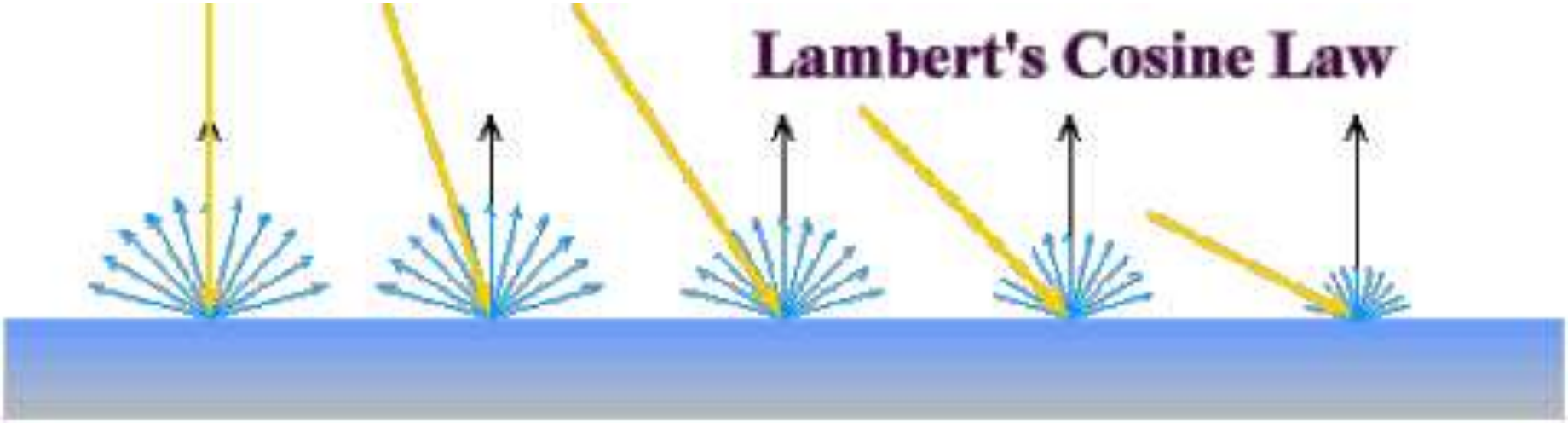
Many materials exhibit both Reflections:

Diffuse Reflection and Lambertian BRDF



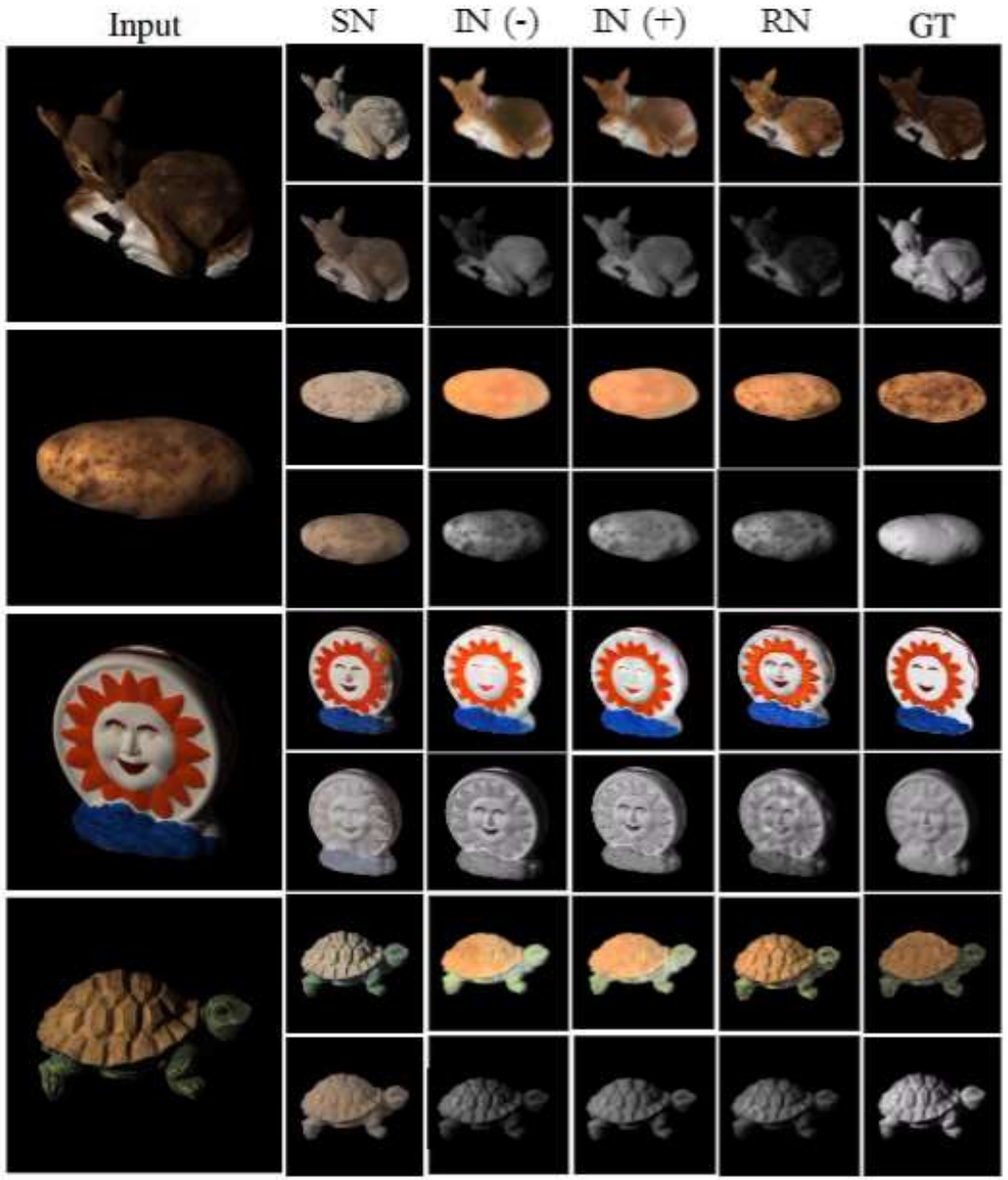
- Surface appears equally bright from ALL directions! (independent of $\vec{v}$)
- Lambertian BRDF is simply a constant : $f(\theta_i, \phi_i; \theta_r, \phi_r) = \frac{\rho_d}{\pi}$ (ρ_d: albedo)
- Surface Radiance : $L = \frac{\rho_d}{\pi} I \cos\theta_i = \frac{\rho_d}{\pi} I \, \vec{n} . \vec{s}$ (I: source intensity)
- Commonly used in Vision and Graphics!

Diffuse Reflection and Lambertian BRDF



Diffuse Reflection and Lambertian BRDF

MIT Dataset (Small)



Surface Radiance:

$$I = \rho\, I_i \cos\theta = \rho I_i \vec{n} \cdot \vec{s}$$

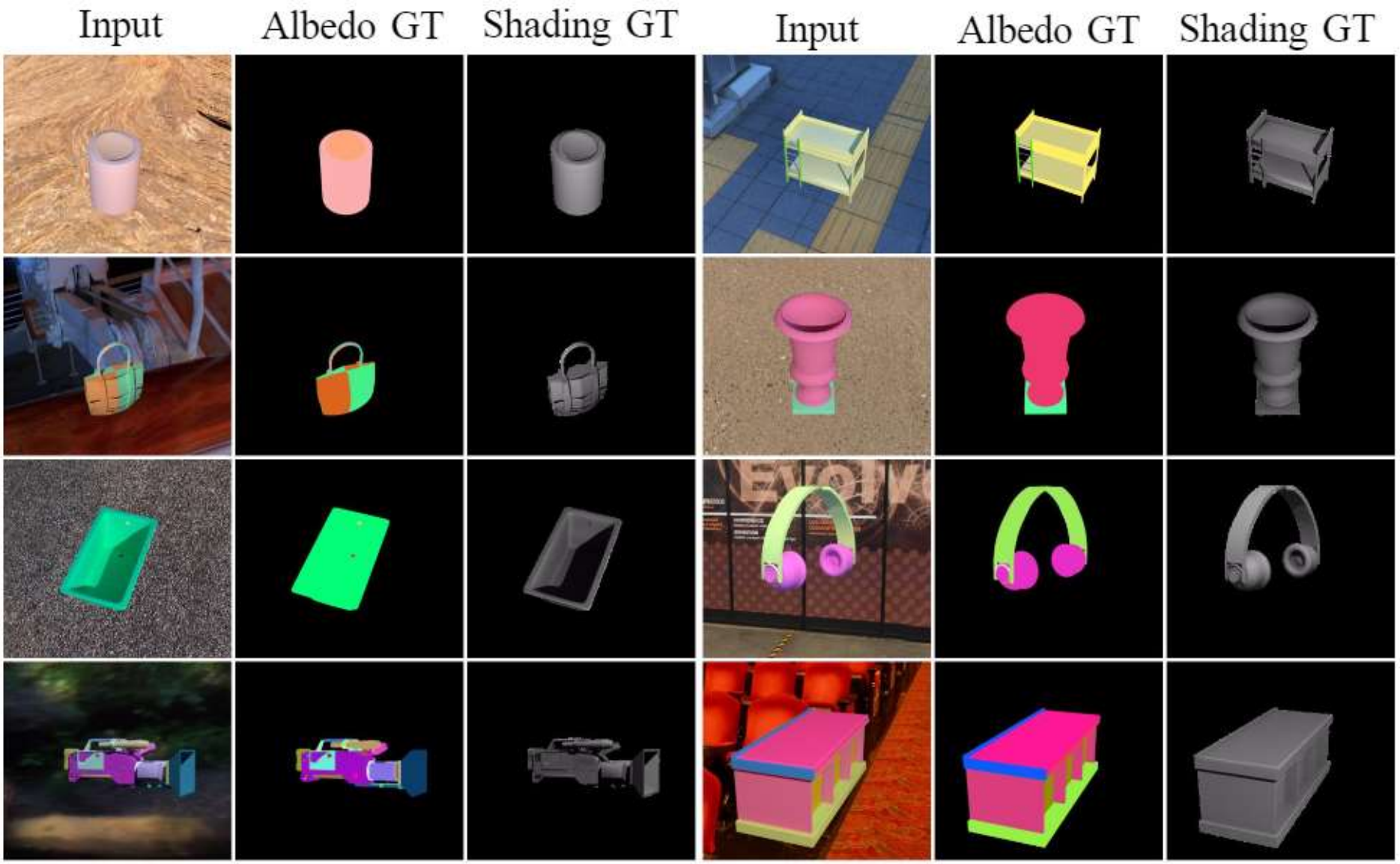
$$I = \rho\, S, \text{where S} = I_i \vec{n} \cdot \vec{s}$$

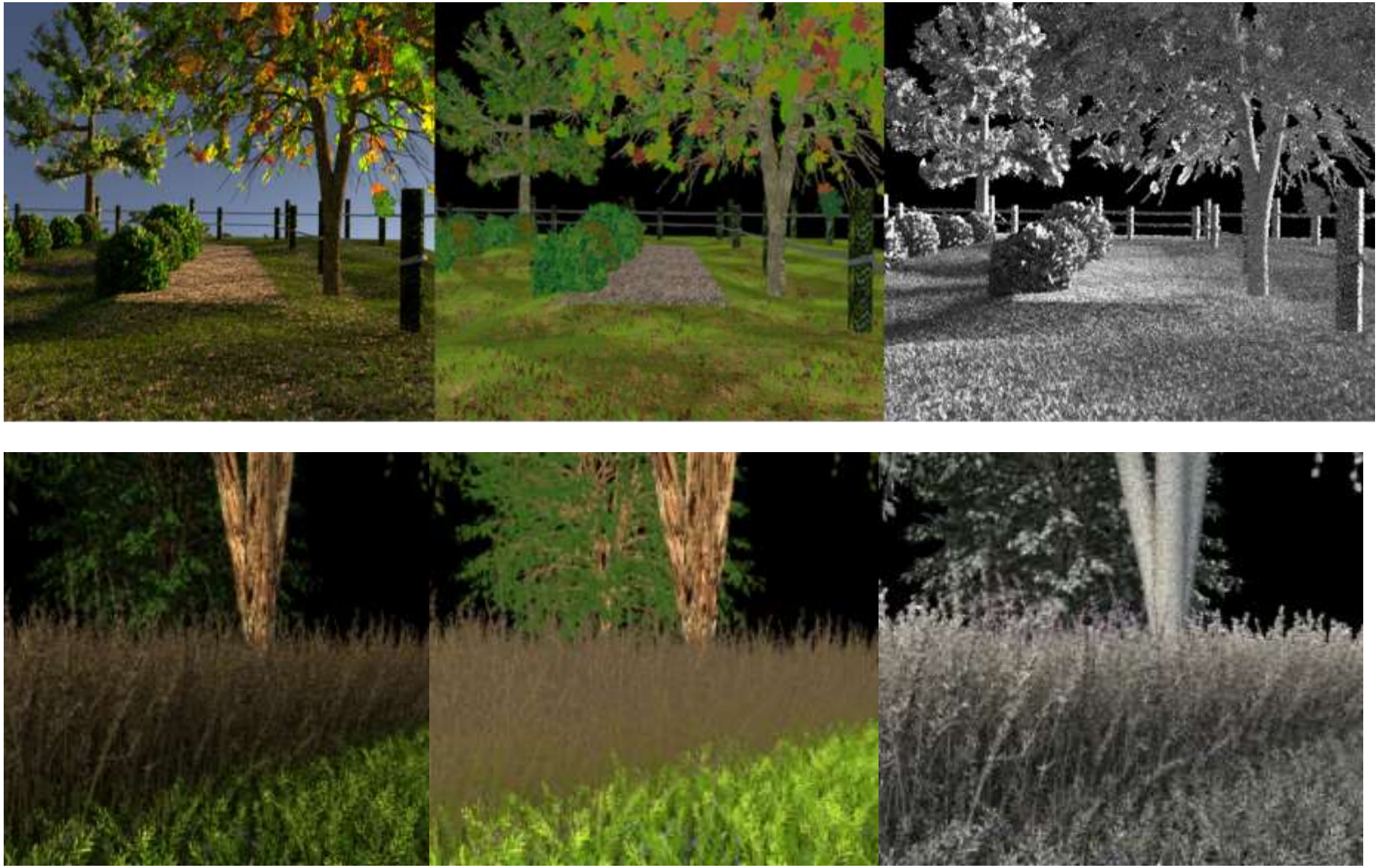
$$I = R\, S, \text{where R} = \rho$$

$$I = R\, S$$

Intensity = reflectance x shading

ShapeNet Rendered Dataset (1M)





Video link: https://youtu.be/l5U85sGdanA

Face Analysis

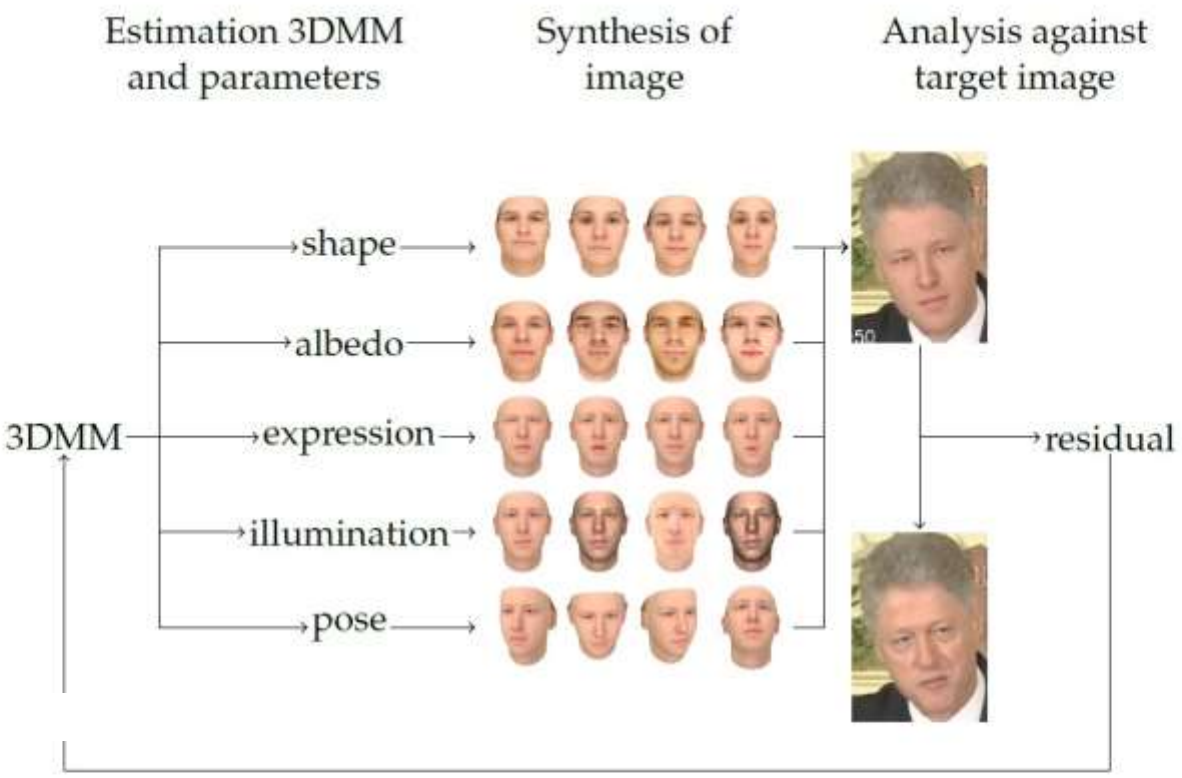


FIGURE 1.1: Example of 3DMM fitting following an analysis-by-synthesis approach. From left to right: The 3D Morphable Model, prediction of parameters for the different components of the optimization problem and rendered results, analyses against target image, and feedback loop.





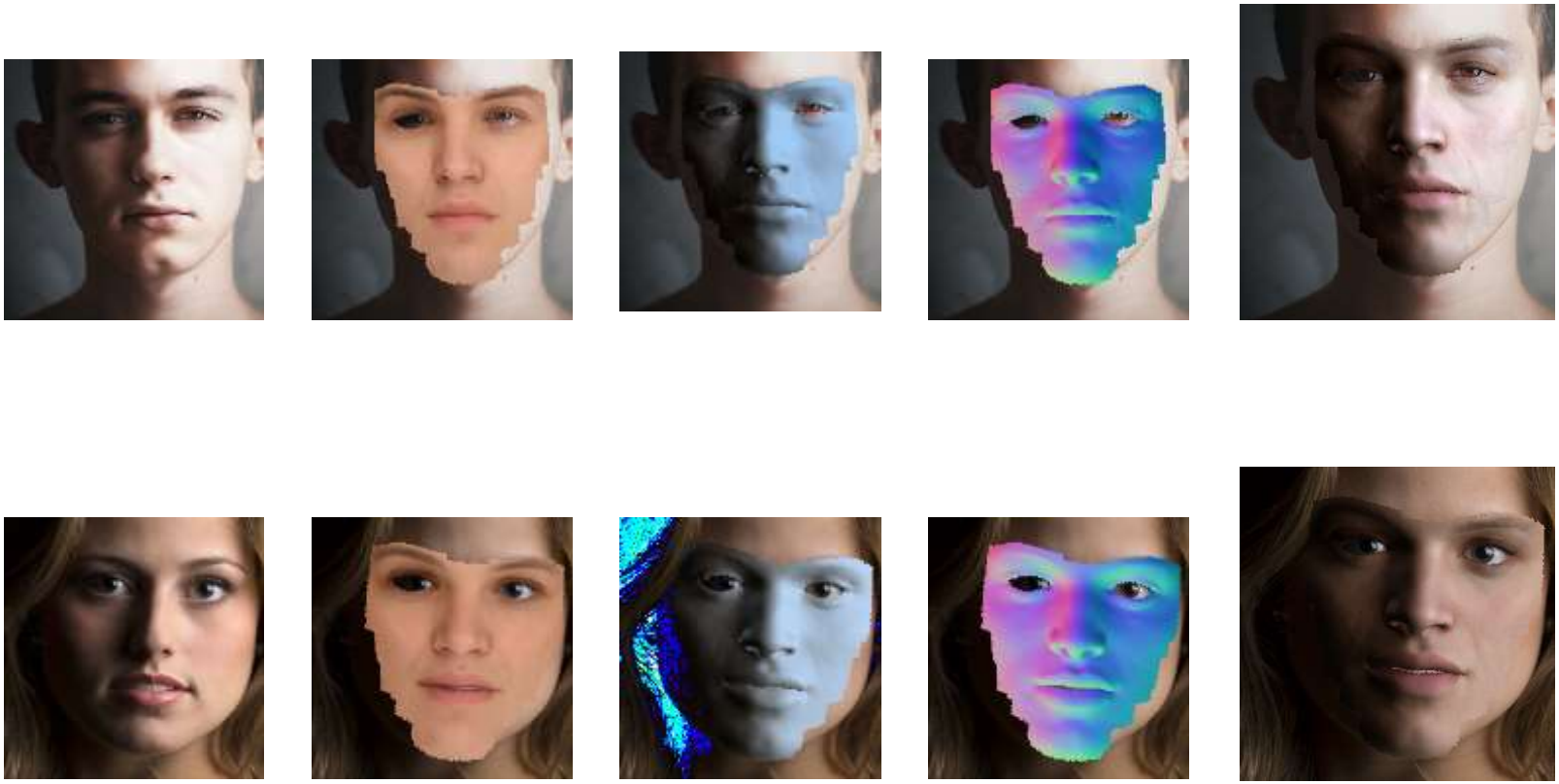










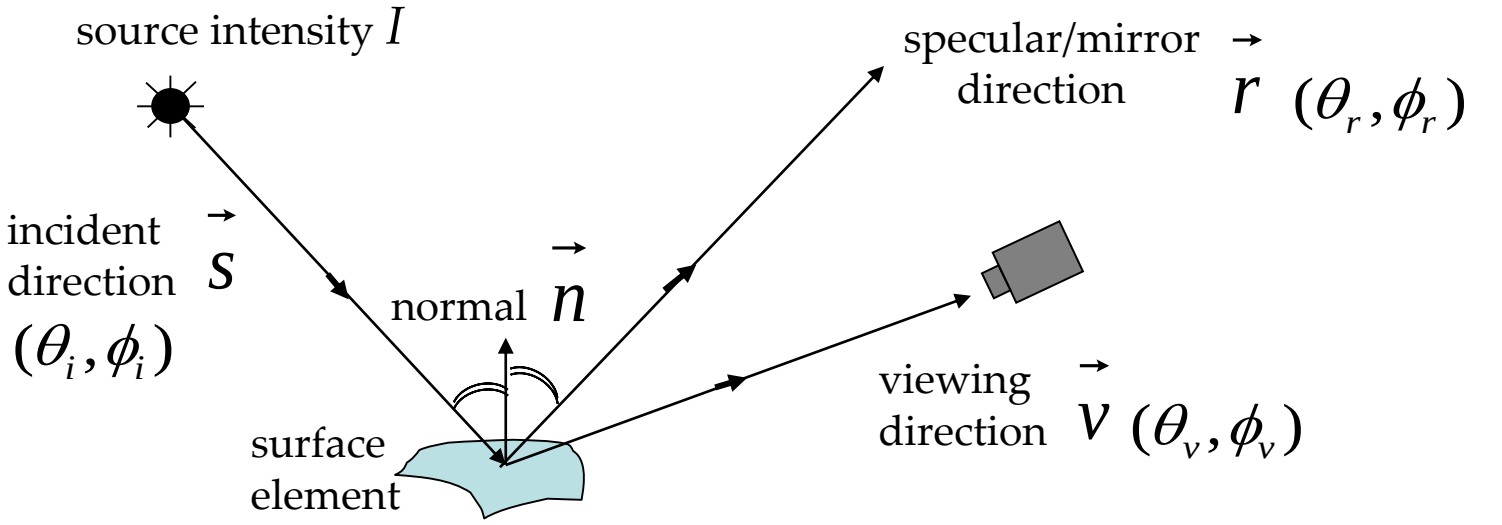



albedo_0


mask_03


out_0300

Specular Reflection and Mirror BRDF



- Very smooth surface.
- All incident light energy reflected in a SINGLE direction. (only when $\vec{v} = \vec{r}$)
- Mirror BRDF is simply a double-delta function :

specular albedo

$$f(\theta_i, \phi_i; \theta_v, \phi_v) = \rho_s \, \delta(\theta_i - \theta_v) \, \delta(\phi_i + \pi - \phi_v)$$

- Surface Radiance : $L = I \, \rho_s \, \delta(\theta_i - \theta_v) \, \delta(\phi_i + \pi - \phi_v)$

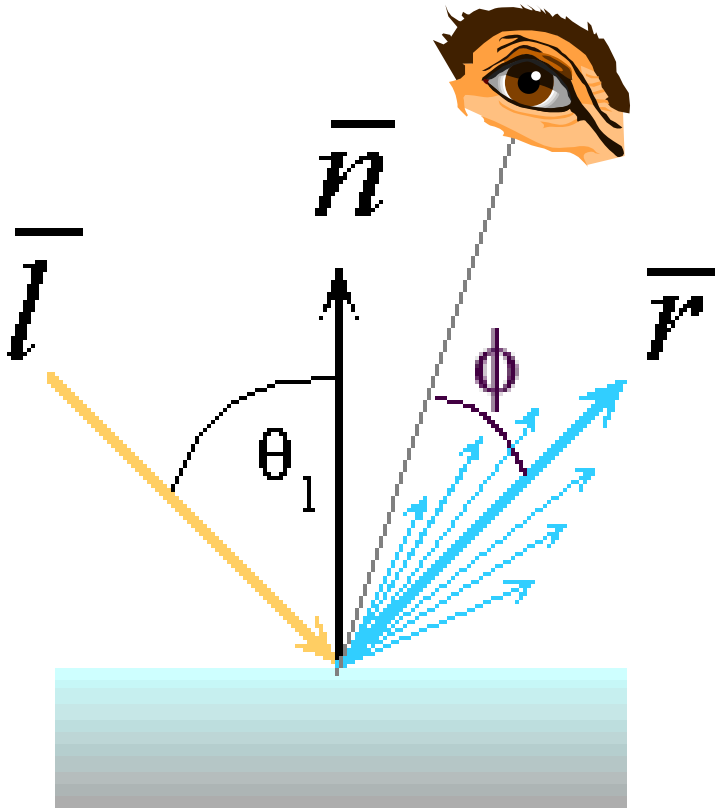
Glossy Surfaces

- Many glossy surfaces show broader highlights in addition to specular reflection.

- Example Models : Phong Model (no physical basis, but sort of works (empirical)

Phong Model: An Empirical Approximation

- An illustration of the angular falloff of highlights:

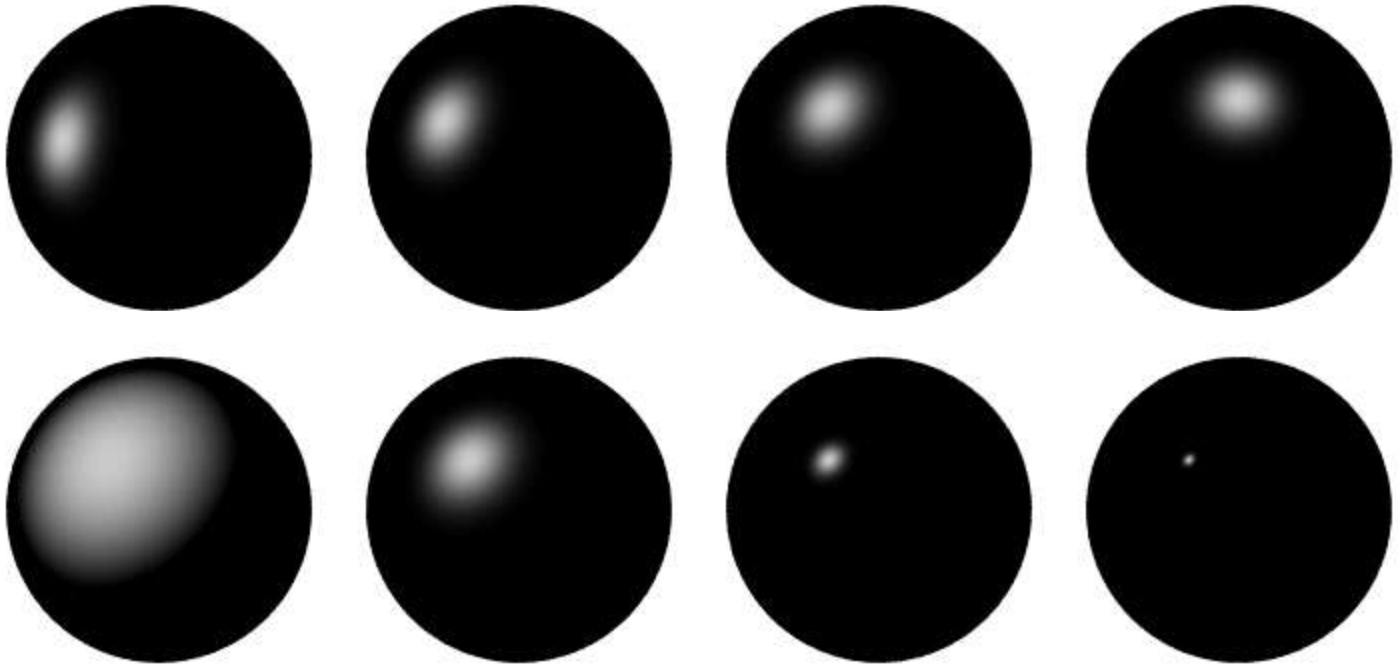


$$L = I\ \rho_s\ (\cos\phi)^{n_{shiny}}$$

- Very commonly used in Computer Graphics

Phong Examples

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

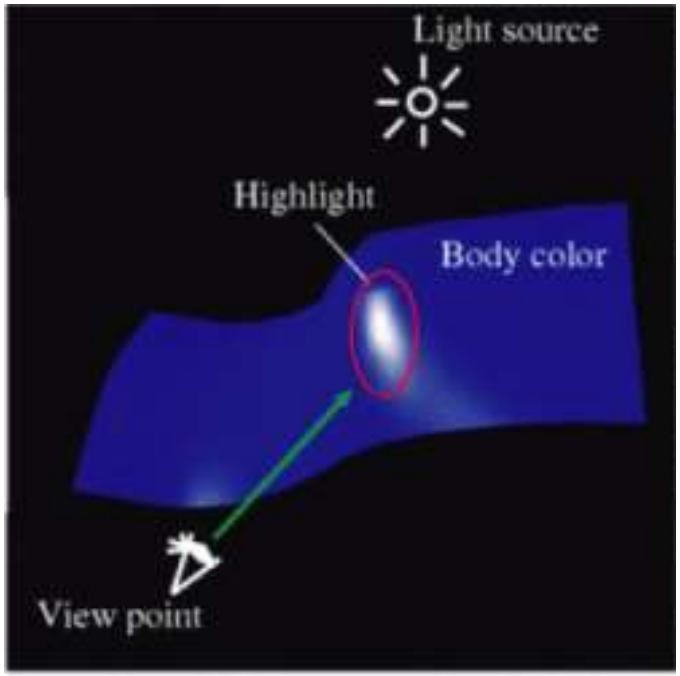


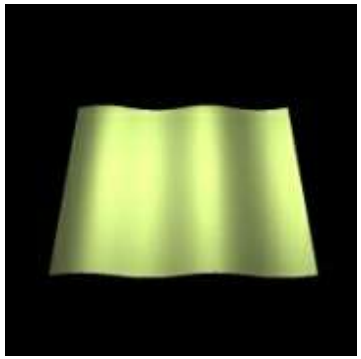
Surface Reflection

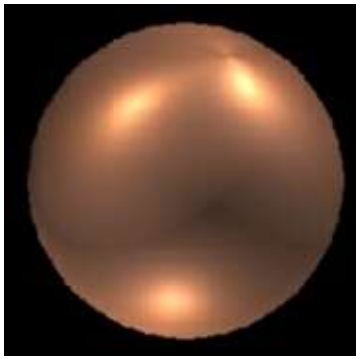
A Simple Reflection Model - Dichromatic Reflection

Observed Image Color = a x Body Color + b x Specular Reflection Color

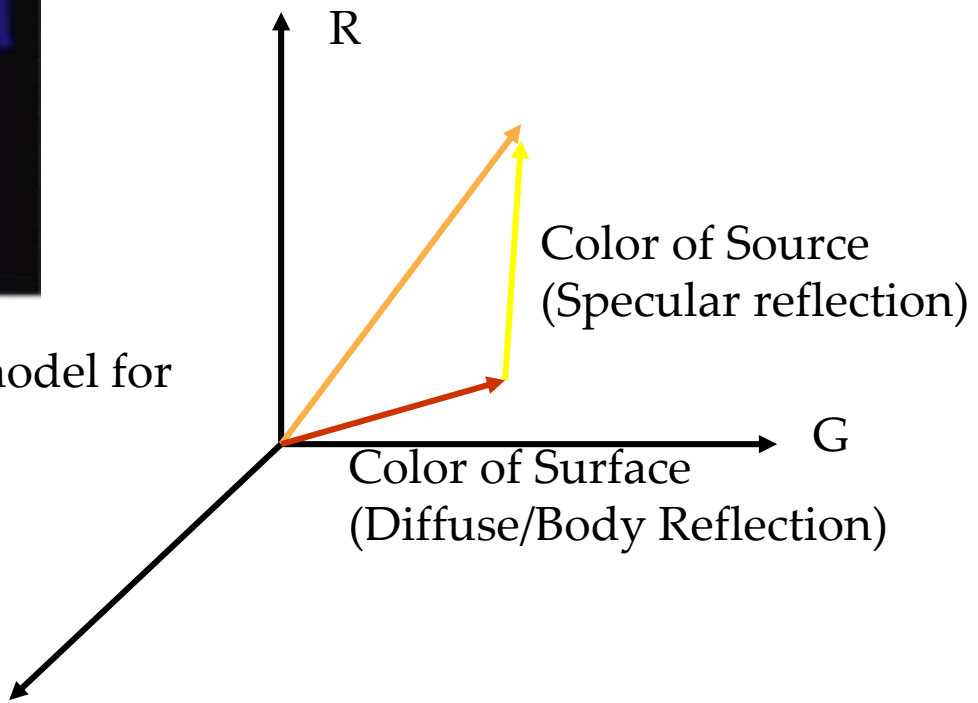
Klinker-Shafer-Kanade 1988





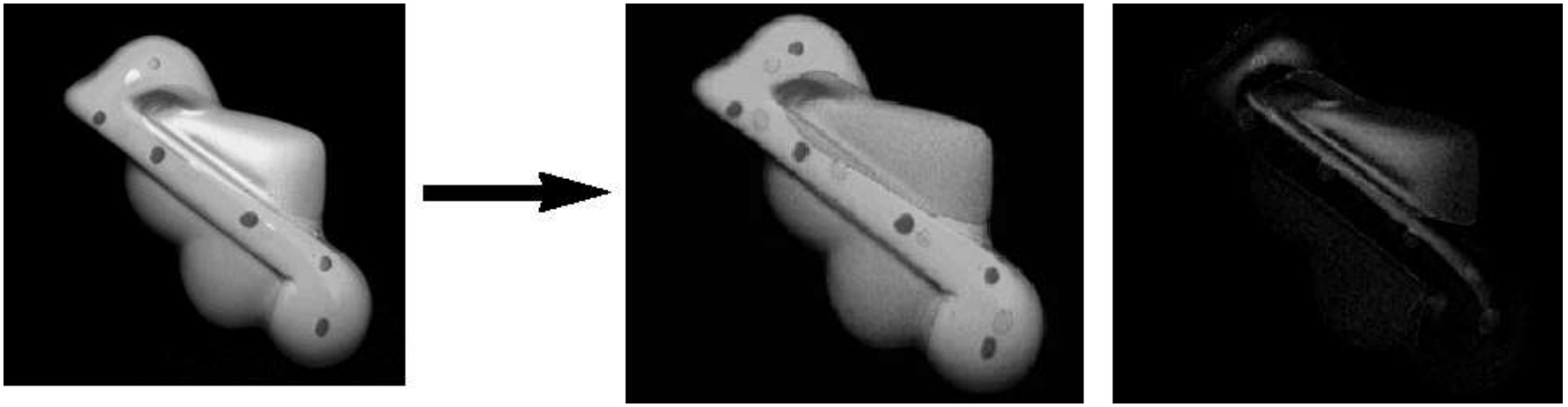


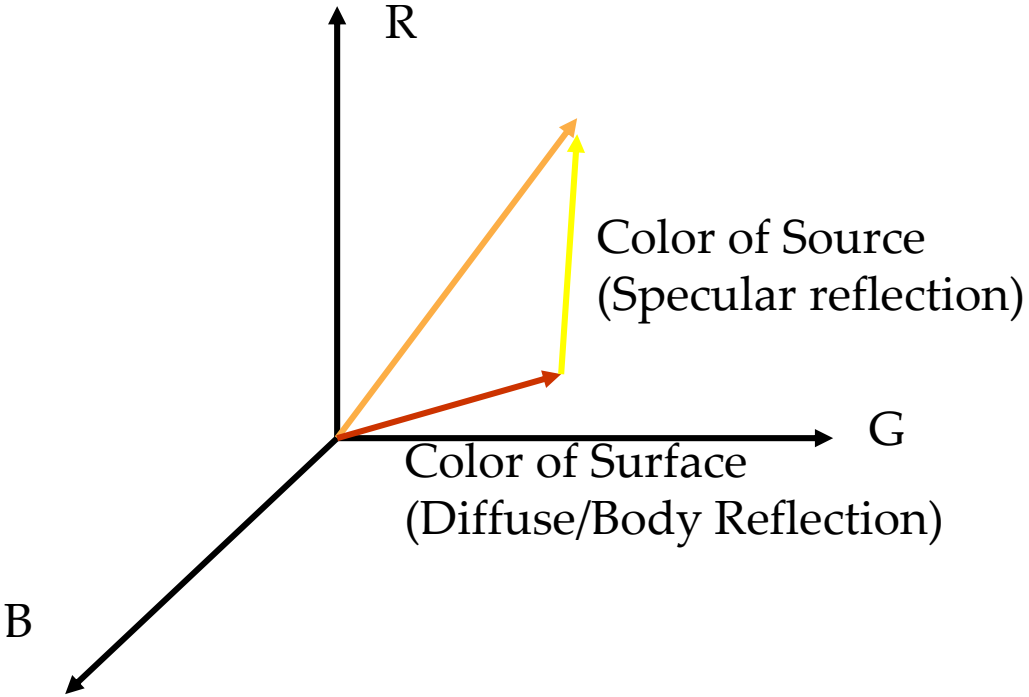
Does not specify any specific model for Diffuse/specular reflection



Separating Diffuse and Specular Reflections

Observed Image Color = a x Body Color + b x Specular Reflection Color





Today's Class

1. Reflection Models
2. **Color Invariance**
3. Image Processing

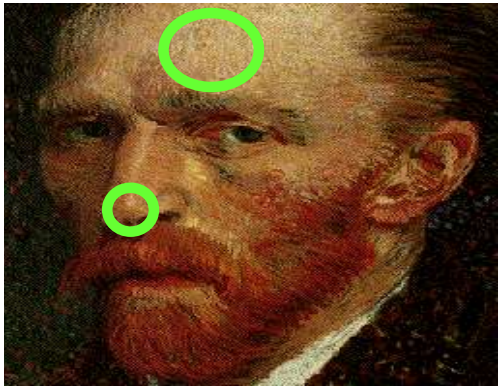
Reflection Model

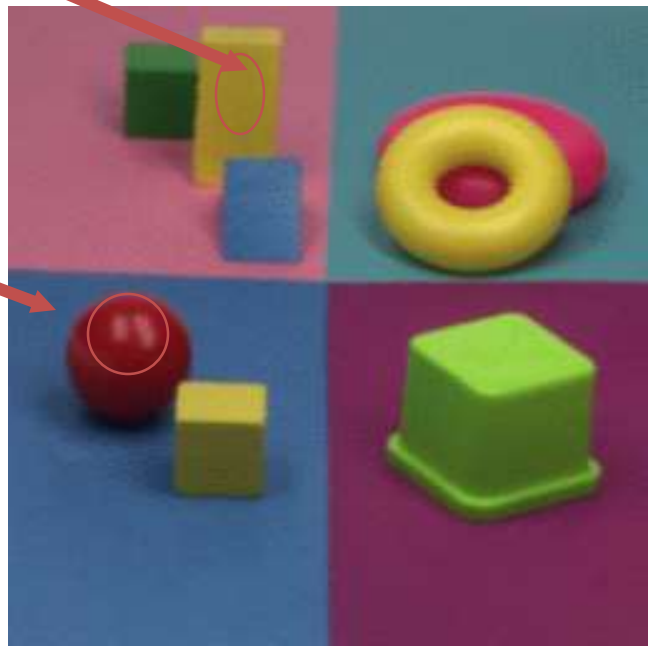
Dichromatic reflection model [Shafer]

body = $$m_b(\mathbf{n},\mathbf{s})\int_\lambda e(\lambda)\rho(\lambda)\,f_C(\lambda)d\lambda$$

surface = $$m_s(\mathbf{n},\mathbf{s},\mathbf{v})\int_\lambda e(\lambda)c(\lambda)\,f_C(\lambda)d\lambda$$

for {R,G,B} giving an R-, B-, G-sensor response





Reflection Model

$$C = m_b(\mathbf{n},\mathbf{s})\int_{\lambda} e(\lambda)\rho(\lambda)\, f_C(\lambda)d\lambda + m_s(\mathbf{n},\mathbf{s},\mathbf{v})\int_{\lambda} e(\lambda)c(\lambda)\, f_C(\lambda)d\lambda$$

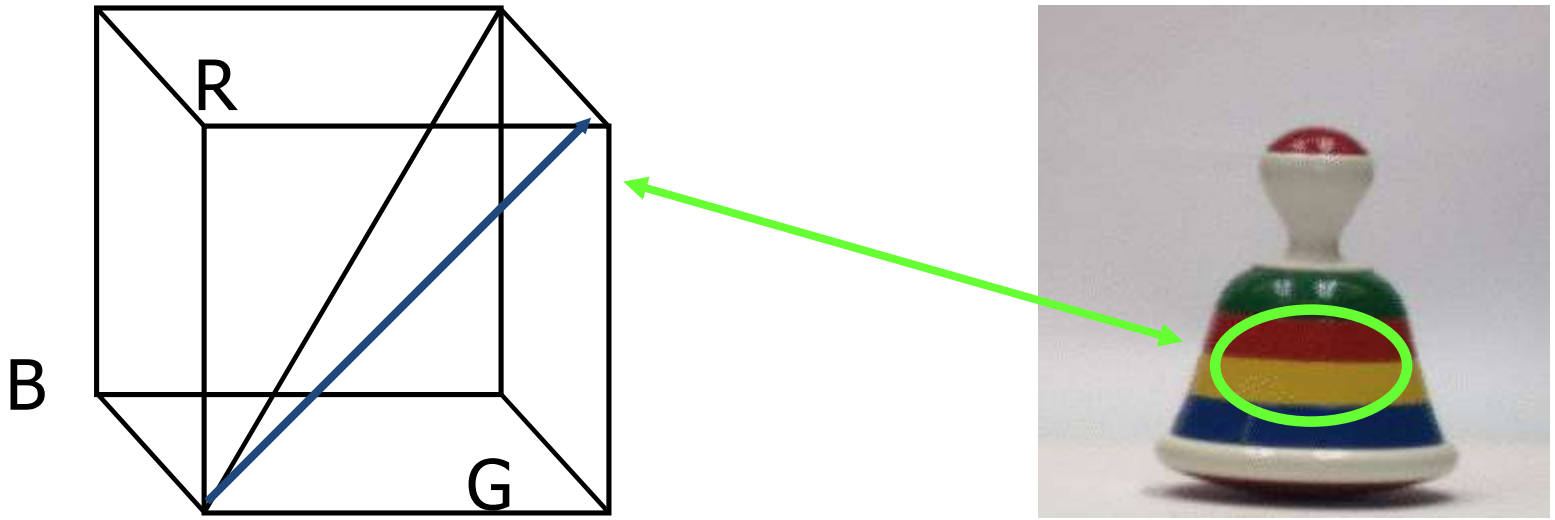
$\rho(\lambda)$	surface albedo	scene & viewpoint invariant
$e(\lambda)$	illumination	scene dependent
$\mathbf{n}$	object surface normal	object shape variant
$\mathbf{s}$	illumination direction	scene dependent
$\mathbf{v}$	viewer's direction	viewpoint variant
$f_C(\lambda)$	sensor sensitivity	camera dependent
$c(\lambda)$	specular	surface dependent

Body Reflectance in RGB - Space

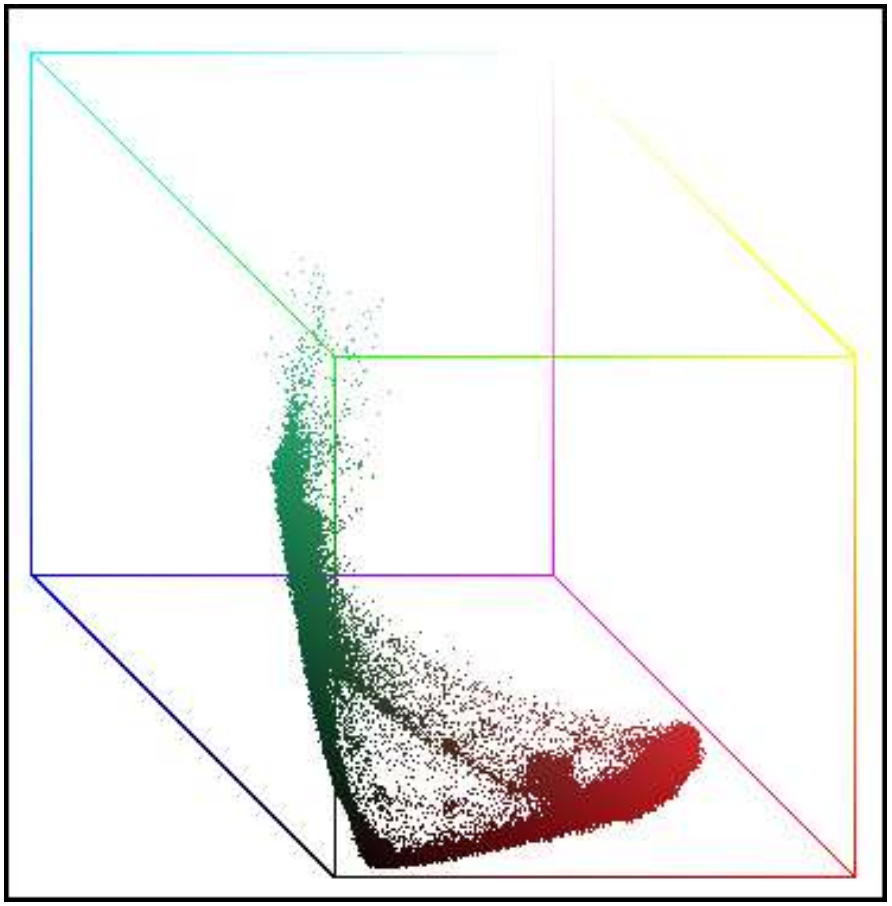
Consider the body reflection term:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} m_b(\mathbf{n},\mathbf{s}) \int_\lambda e(\lambda)\rho(\lambda) f_R(\lambda) d\lambda \\ m_b(\mathbf{n},\mathbf{s}) \int_\lambda e(\lambda)\rho(\lambda) f_G(\lambda) d\lambda \\ m_b(\mathbf{n},\mathbf{s}) \int_\lambda e(\lambda)\rho(\lambda) f_B(\lambda) d\lambda \end{bmatrix}$$

for example the geometric term is Lambertian i.e. $m_b(\mathbf{n},\mathbf{s}) = \mathbf{n} \cdot \mathbf{s} = \cos\theta$



Body Reflectance in RGB - Matte Surfaces



An object with its representation in RGB-colour space.

Reflectance under White Light

Dichromatic reflection:

$$C = m_b(\mathbf{n},\mathbf{s})\int_\lambda e(\lambda)\rho(\lambda)\, f_C(\lambda)d\lambda + m_s(\mathbf{n},\mathbf{s},\mathbf{v})\int_\lambda e(\lambda)c(\lambda)\, f_C(\lambda)d\lambda$$

Considering dichromatic reflectance and white illumination, then $e = e(\lambda)$ and $c = c(\lambda)$. Then,

for $C_W = \{R,G,B\}$ giving the red, green and blue sensor response under white light. This gives:

$$\begin{bmatrix} R_W \\ G_W \\ B_W \end{bmatrix} = \begin{bmatrix} em_b(\mathbf{n},\mathbf{s})\int_\lambda \rho(\lambda)\, f_R(\lambda)d\lambda + em_s(\mathbf{n},\mathbf{s},\mathbf{v})cf_R \\ em_b(\mathbf{n},\mathbf{s})\int_\lambda \rho(\lambda)\, f_G(\lambda)d\lambda + em_s(\mathbf{n},\mathbf{s},\mathbf{v})cf_G \\ em_b(\mathbf{n},\mathbf{s})\int_\lambda \rho(\lambda)\, f_B(\lambda)d\lambda + em_s(\mathbf{n},\mathbf{s},\mathbf{v})cf_B \end{bmatrix}$$

rgb – Photometric Invariance: Proof

Consider the body reflection term:

$$C_{WB} = em_b(\mathbf{n},\mathbf{s})\int_{\lambda} \rho(\lambda)\, f_C(\lambda) d\lambda$$

For $C_{WB} = \{R_{WB}, G_{WB}, B_{WB}\}$ giving the red, green and blue sensor response under white light. Consider normalized color:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}$$

Then:

$$r = \frac{\cancel{em_b(\mathbf{n},\mathbf{s})}\int_{\lambda} \rho(\lambda)\, f_R(\lambda) d\lambda}{\cancel{em_b(\mathbf{n},\mathbf{s})}\left(\int_{\lambda} \rho(\lambda)\, f_R(\lambda) d\lambda + \int_{\lambda} \rho(\lambda)\, f_G(\lambda) d\lambda + \int_{\lambda} \rho(\lambda)\, f_B(\lambda) d\lambda\right)}$$

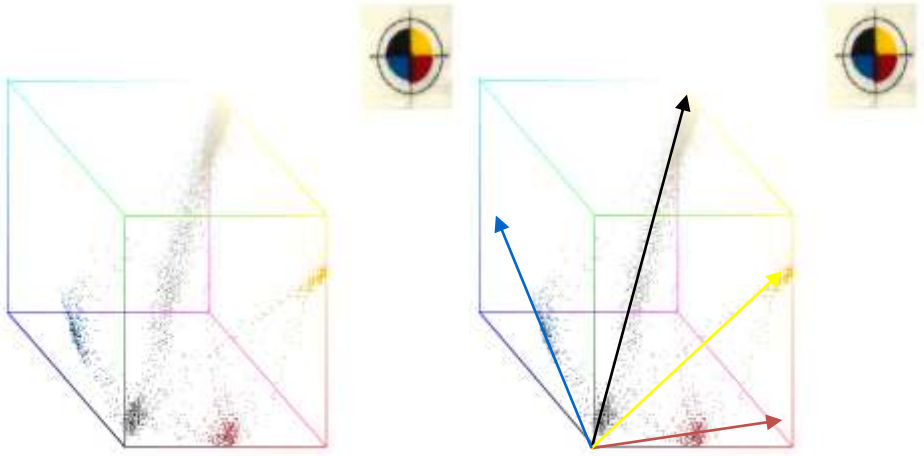
Also holds for g and b. In the sequel: $k_C = \int_{\lambda} \rho(\lambda)\, f_C(\lambda) d\lambda$

Colour Invariance / c1c2c3 space

$$c_1(R,G,B) = \arctan\frac{R}{\max\{G,B\}}$$

$$c_2(R,G,B) = \arctan\frac{G}{\max\{R,B\}}$$

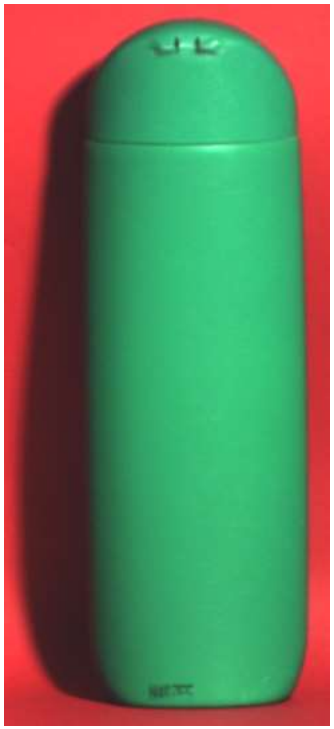
$$c_3(R,G,B) = \arctan\frac{B}{\max\{R,G\}}$$



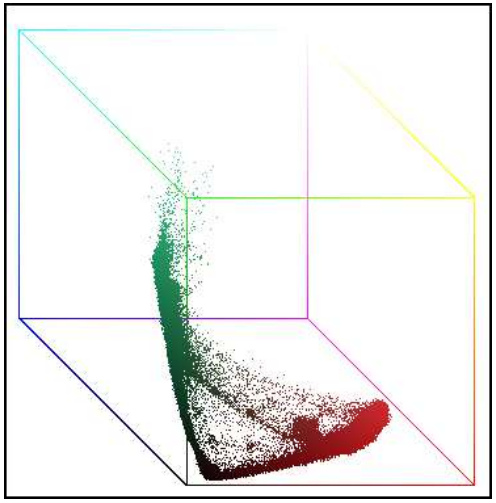
$$c_1(R_b,G_b,B_b) = \arctan\left(\frac{em_b(\mathbf{n},\mathbf{s})k_R}{\max\{em_b(\mathbf{n},\mathbf{s})k_G, em_b(\mathbf{n},\mathbf{s})k_B\}}\right) = \arctan\left(\frac{k_R}{\max\{k_G,k_B\}}\right)$$

$$\text{Where}: k_C = \int_\lambda \rho(\lambda) f_C(\lambda) d\lambda$$

Colour Invariance / rgb space

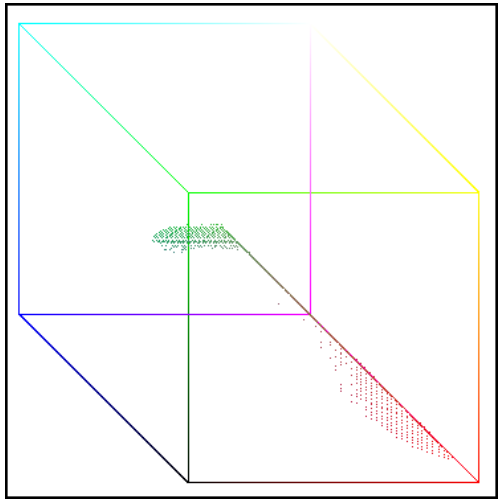


Original RGB image



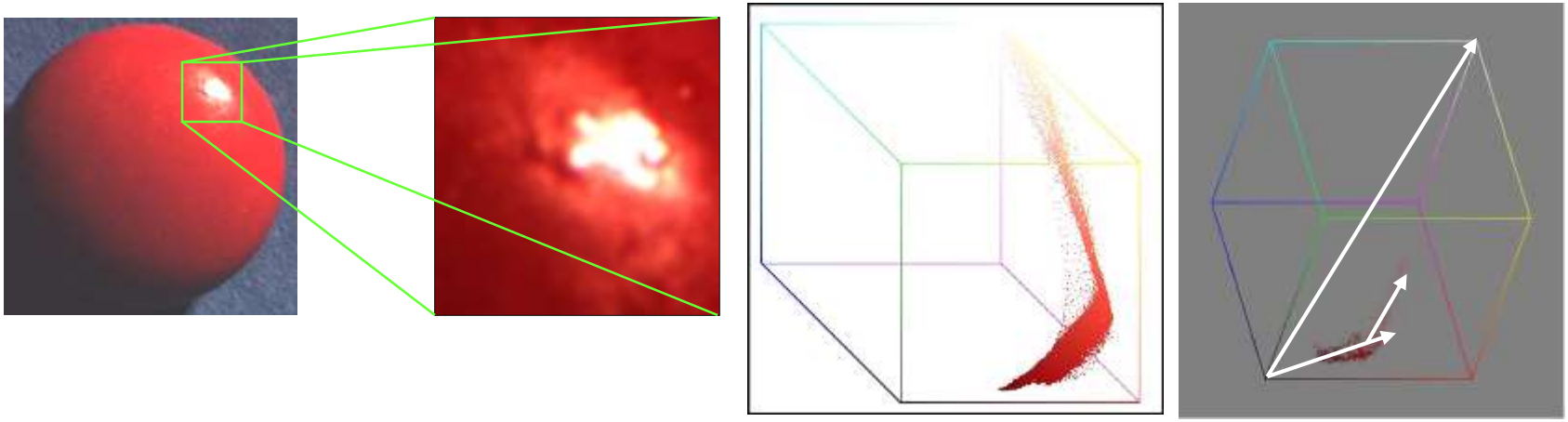
3D plot of RGB image

rgb image



3D plot of rgb image

Body Reflectance in RGB - space



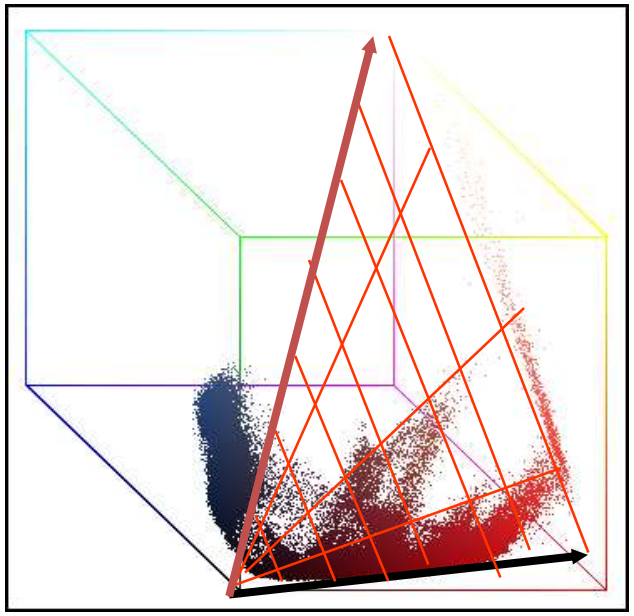
Consider the surface reflection term:

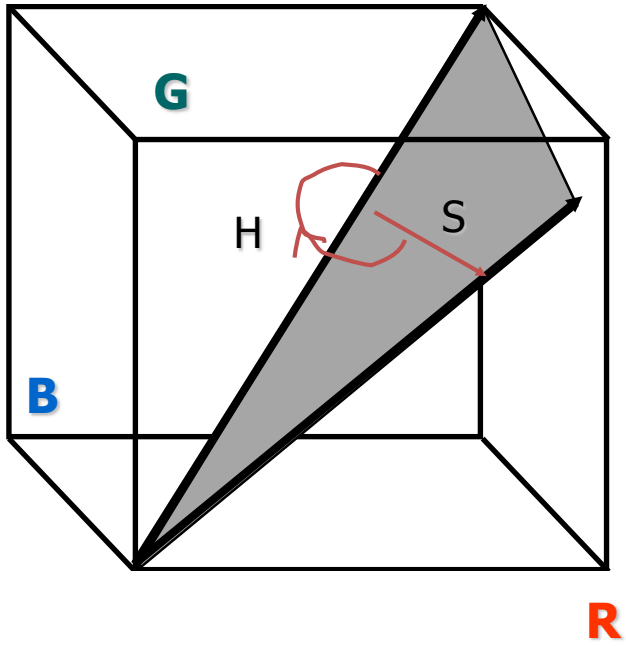
$$m_s(\mathbf{n},\mathbf{s},\mathbf{v})\int_{\lambda} e(\lambda)c(\lambda)\,f_C(\lambda)d\lambda$$

Where the geometric term is for example the phong model:

$$\cos^n \alpha$$

Colour Invariance: Hue





Hue: $H(R,G,B) = \arctan\left(\frac{\sqrt{3}(G-B)}{(R-G)+(R-B)}\right)$

Colour invariance

$$l1(R,G,B) = \frac{(R-G)^2}{(R-G)^2 + (R-B)^2 + (G-B)^2}$$

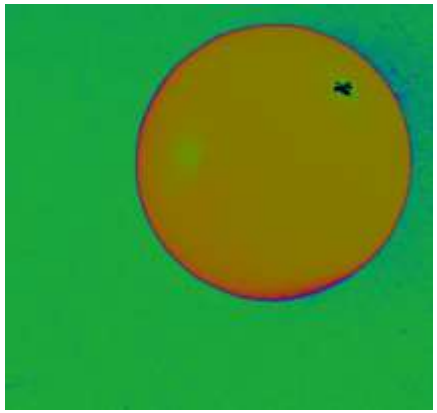
$$l2(R,G,B) = \frac{(R-B)^2}{(R-G)^2 + (R-B)^2 + (G-B)^2}$$

$$l3(R,G,B) = \frac{(G-B)^2}{(R-G)^2 + (R-B)^2 + (G-B)^2}$$

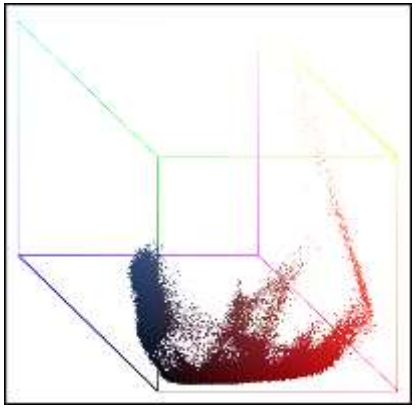
Colour Invariance: Hue

Original colour image

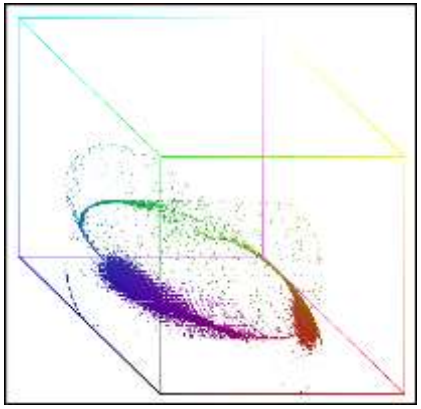
L1l2l3 transformation



L1l2l3 image



3D plot of colour image



3D plot of l1l2l3 image

Colour Invariance - Summary

	shadows	shading	highlights	ill. intensity	ill. Colour
I	-	-	-	-	-
R,G,B	-	-	-	-	-
r,g,b	+	+	-	+	-
c1,c2,c3	+	+	-	+	-
Hue	+	+	+	+	-
l1,l2,l3	+	+	+	+	-
m1m2m3	+	+	-	+	+

- no invariance

+ invariance

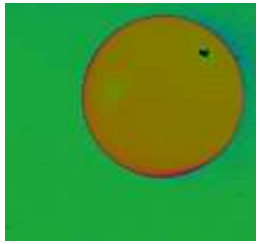
Summary

- Reflection models help to understand the image formation process.

$$C = m_b(\mathbf{n},\mathbf{s})\int_\lambda e(\lambda)\rho(\lambda)\,f_C(\lambda)d\lambda + m_s(\mathbf{n},\mathbf{s},\mathbf{v})\int_\lambda e(\lambda)c(\lambda)\,f_C(\lambda)d\lambda$$

- Color invariance at the pixel. White light source!

- Be aware of instabilities



Today's Class

1. Reflection Models
2. **Color Invariance : Constancy**
3. Image Processing

Color constancy

Gray-world

Training

Unknown

Test

Unknown

Algorithm

Canonical

Canonical

Mean color = constant

Scale-by-max

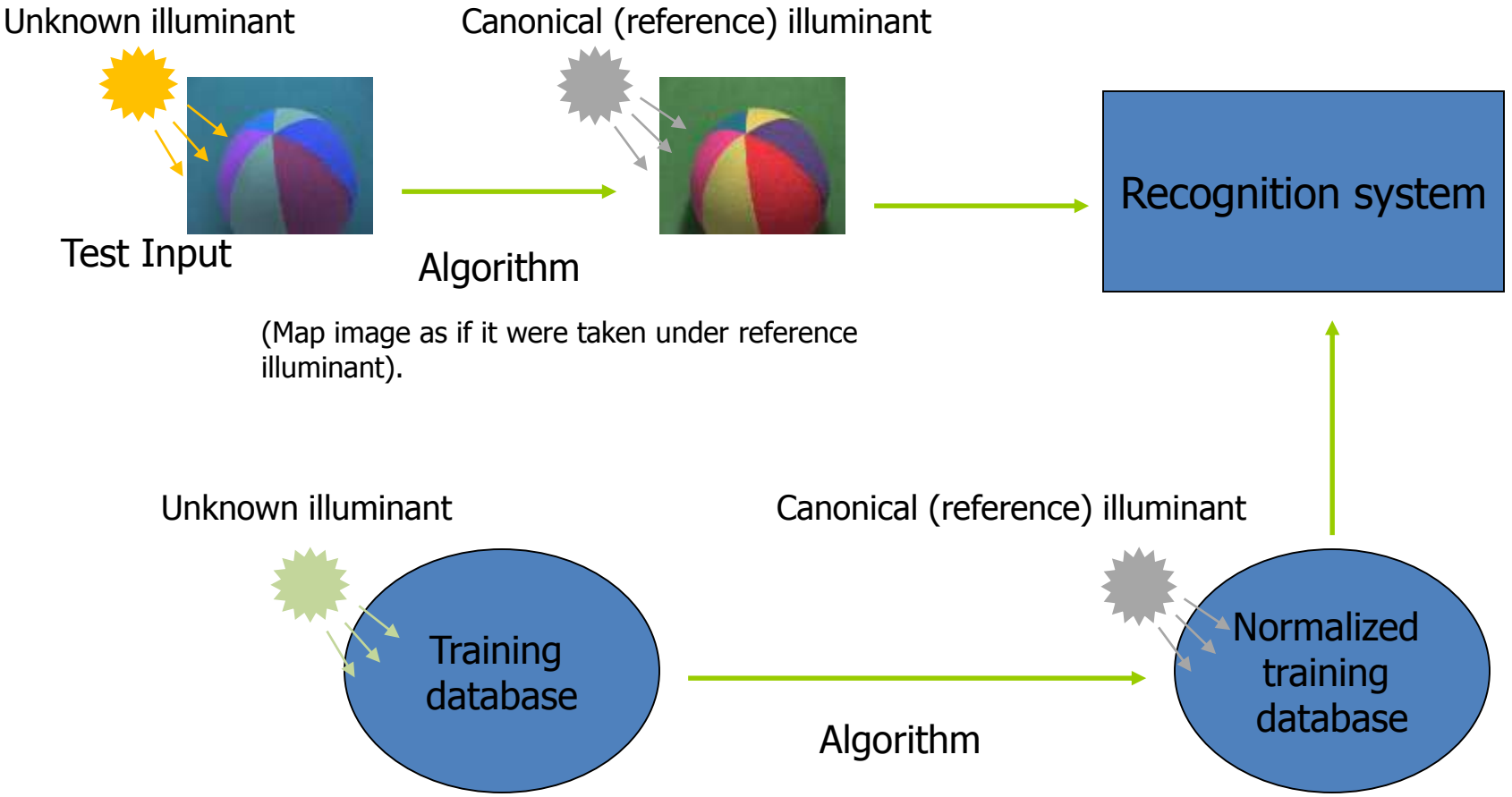
Training

Test

Algorithm

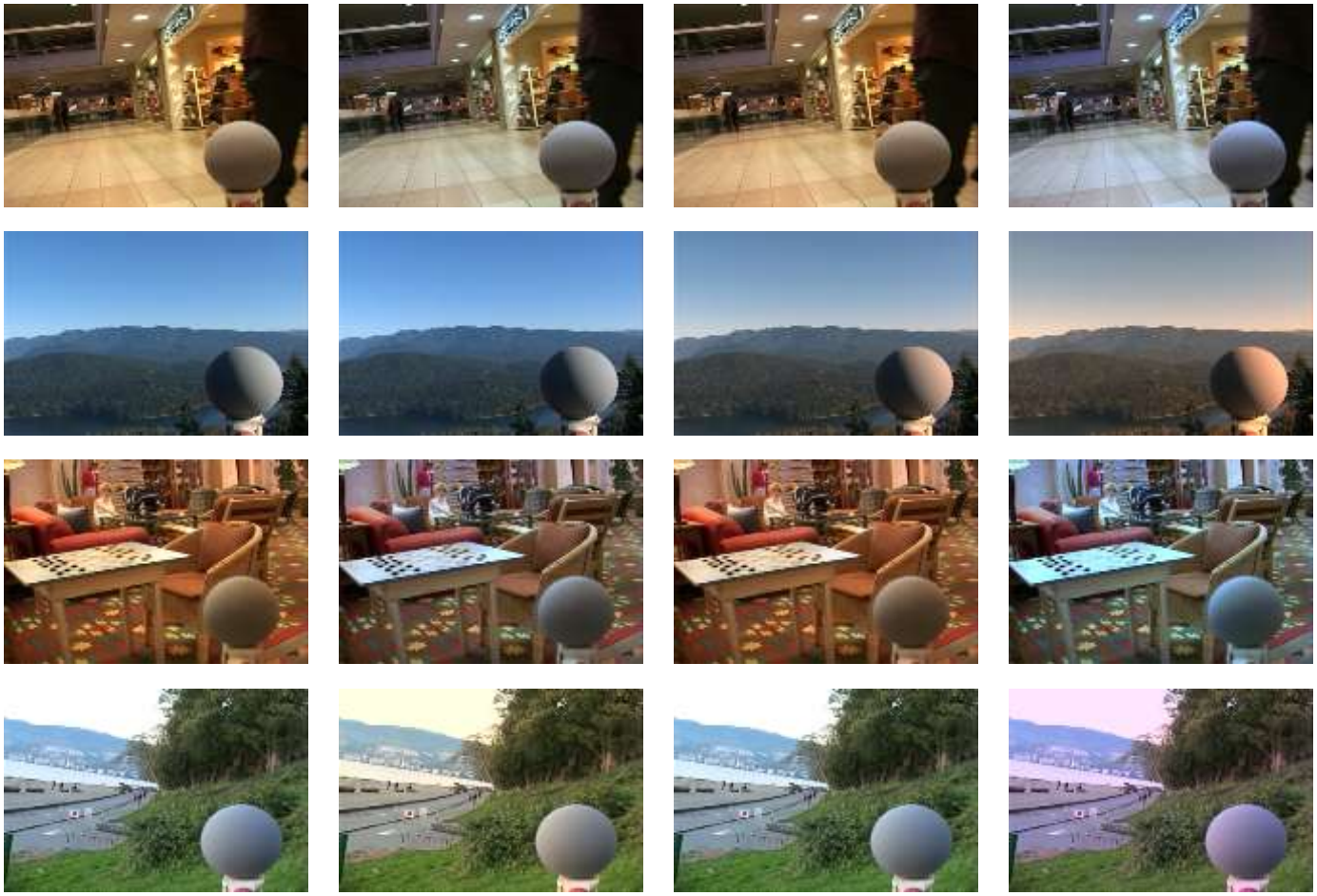
Max color = constant

Color normalization



White Patch and Grey-world Examples

Original image	Ideal correction	White-Patch	Grey-world



Images from “A large image data set for color constancy research”, by F. Ciurea and B. Funt in CIC 2003.

Color Constancy

- Model images assuming Lambertian reflectance:

$$C = m_b(\mathbf{n}, \mathbf{s}) \int_{\lambda} e(\lambda)\rho(\lambda) f_C(\lambda) d\lambda$$

- Image transformation using von Kries model[1]:

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix}$$

(1) - J. von Kries. "Influence of adaptation on the effects produced by luminous stimuli." In D. MacAdam, editor, *Sources of Color Vision*, pages 109–119. MIT Press, 1970.

Color : Von Kries Model

Assuming body reflection

$$C = m_b(\mathbf{n},\mathbf{s})\int e(\lambda)\rho(\lambda) f_C(\lambda)d\lambda \quad \text{for} \quad C = \{R,G,B\}$$

And narrow-band filters: $f_C(\lambda) = \delta(\lambda - \lambda_C)$

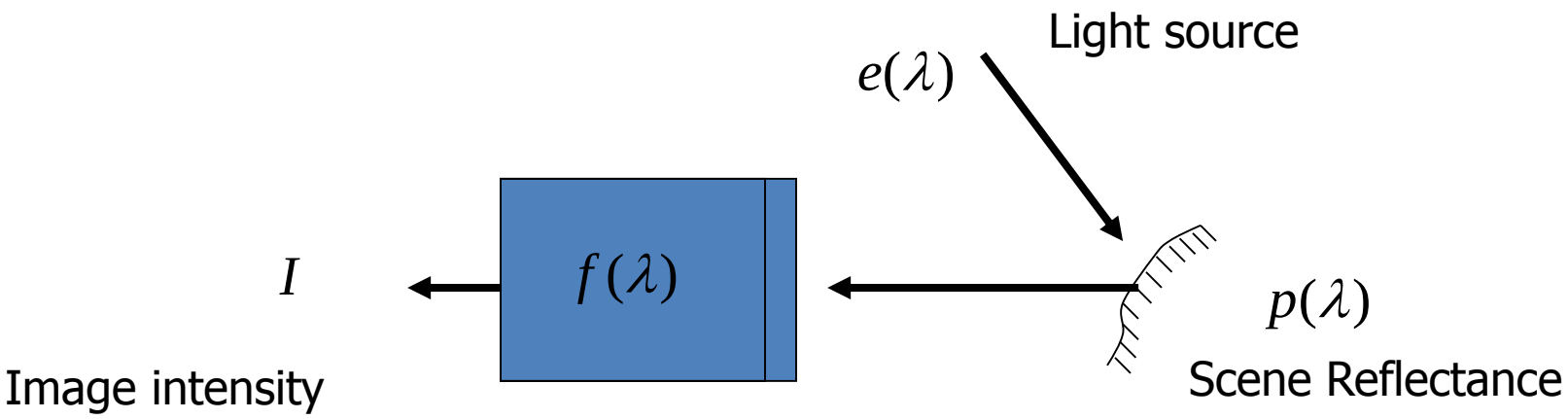
Then: $C = m_b(\mathbf{n},\mathbf{s})e(\lambda_C)\rho(\lambda_C)$ for $C = \{R,G,B\}$

Therefore, we have: $R = m_b(\mathbf{n},\mathbf{s})e(\lambda_R)\rho(\lambda_R)$

$$G = m_b(\mathbf{n},\mathbf{s})e(\lambda_G)\rho(\lambda_B)$$

$$B = m_b(\mathbf{n},\mathbf{s})e(\lambda_B)\rho(\lambda_B)$$

Image Brightness (Intensity)



- Monochromatic Light: $(\lambda = \lambda_i)$ $\quad f(\lambda_i) = 1$

$$I = m_b(\mathbf{n}, \mathbf{s}) e \rho$$

$$I = \mathbf{n} \cdot \mathbf{s} e \rho \quad \text{for Lambertian reflection}$$

Recovering Lightness: Retinex

- Image Intensity: $I = e\rho$
- Take Logarithm: $\log I = \log e + \log \rho$

OR $I = e + \rho$

- Use Laplacian:

$$d = \nabla^2 I = \nabla^2 e + \nabla^2 \rho \qquad \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

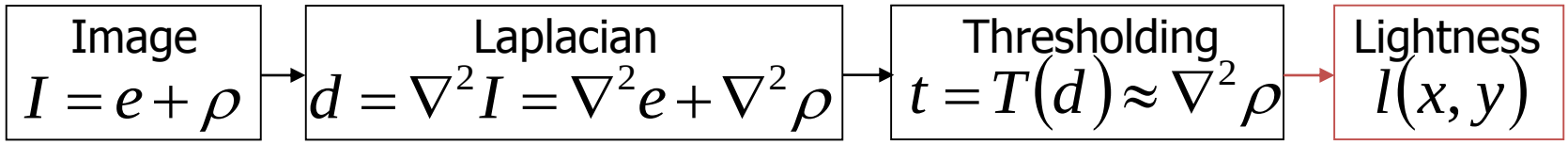
- Sharp changes in reflectance ρ

  $\nabla^2 \rho$ has 2 infinite spikes near edges and $\nabla^2 \rho = 0$ elsewhere

- Smooth changes in illumination e

  $\nabla^2 e \approx 0$ everywhere

Solving the Inverse Problem



Find lightness *l(x,y)* from *t(x,y):*

Poisson's Equation

$$\nabla^2 l = t$$

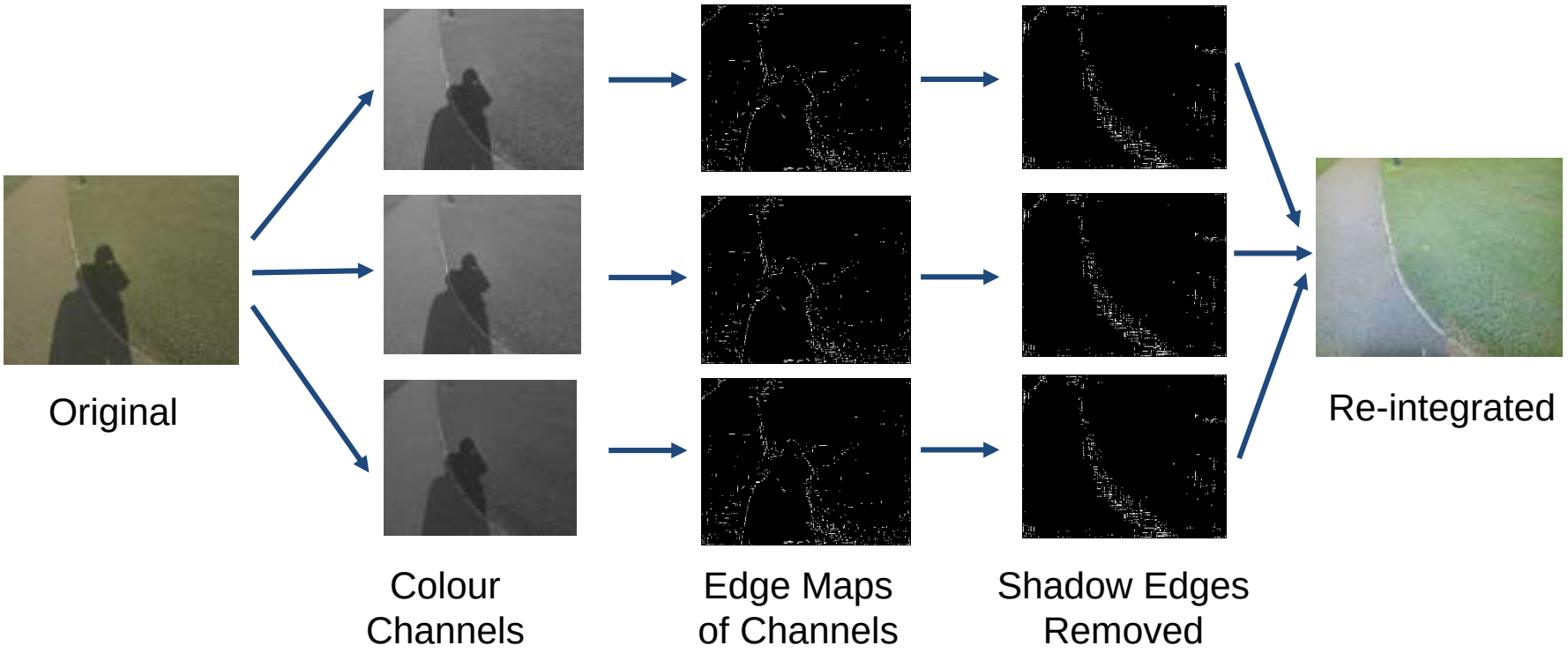
$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) l(x, y) = t(x, y)$$

We have to find *g(x,y)* which satisfies

$$l(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} t(u, v) g(x - u, y - v) du dv$$

$$l(x, y) = t(x, y) * g(x, y)$$

Shadow Removal



An Example

Original Image

Invariant Image

Detected Shadow Edges

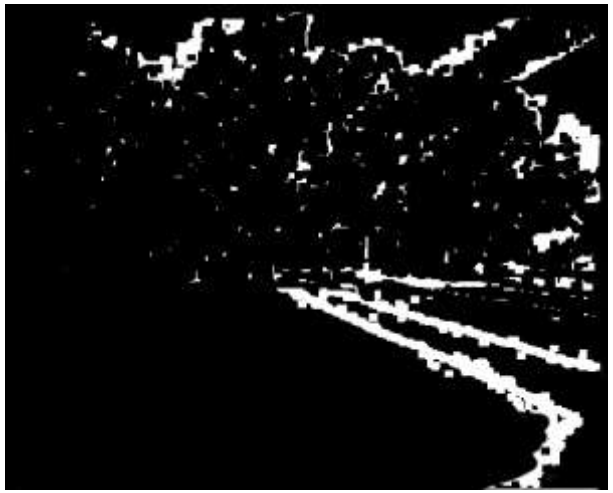
Shadow Removed

A Second Example

Original Image

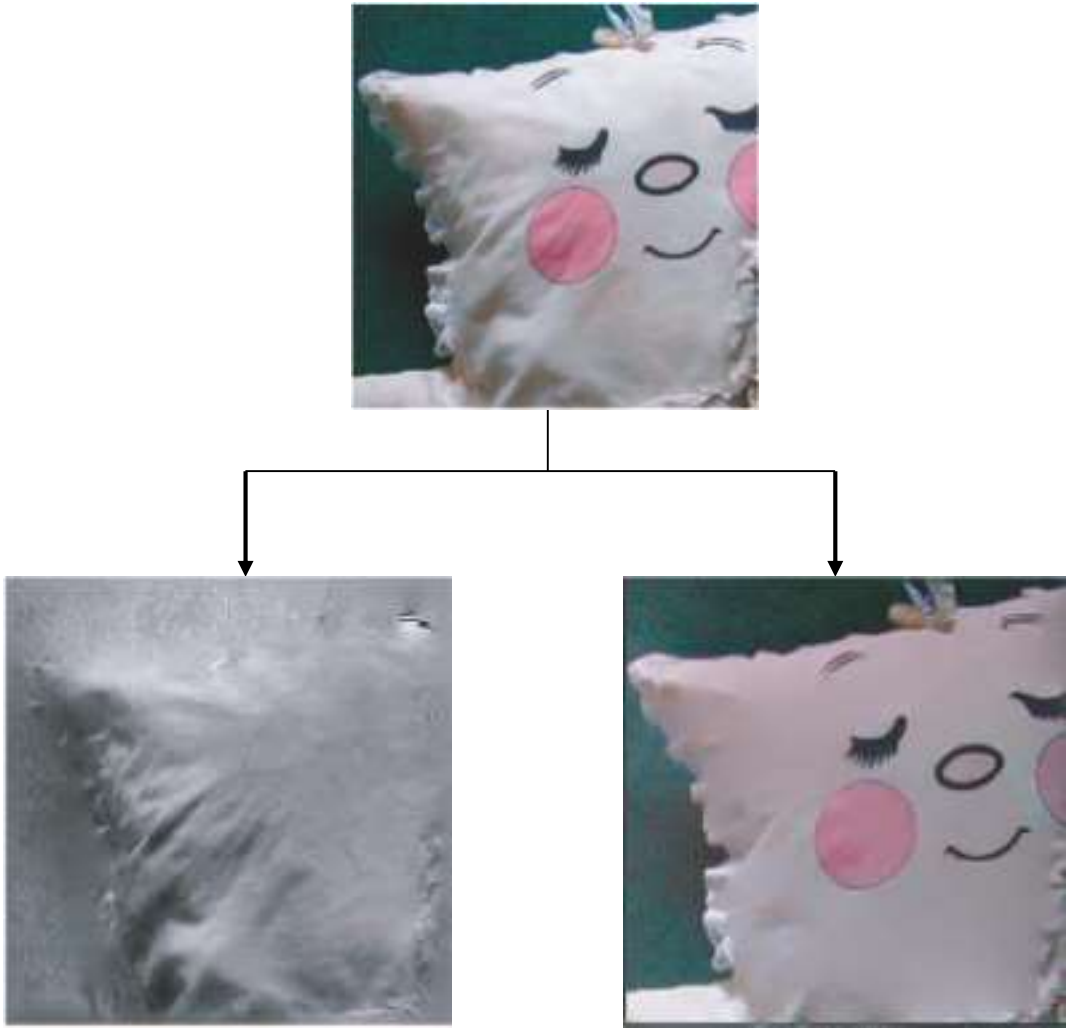
Invariant Image

Detected Shadow Edges

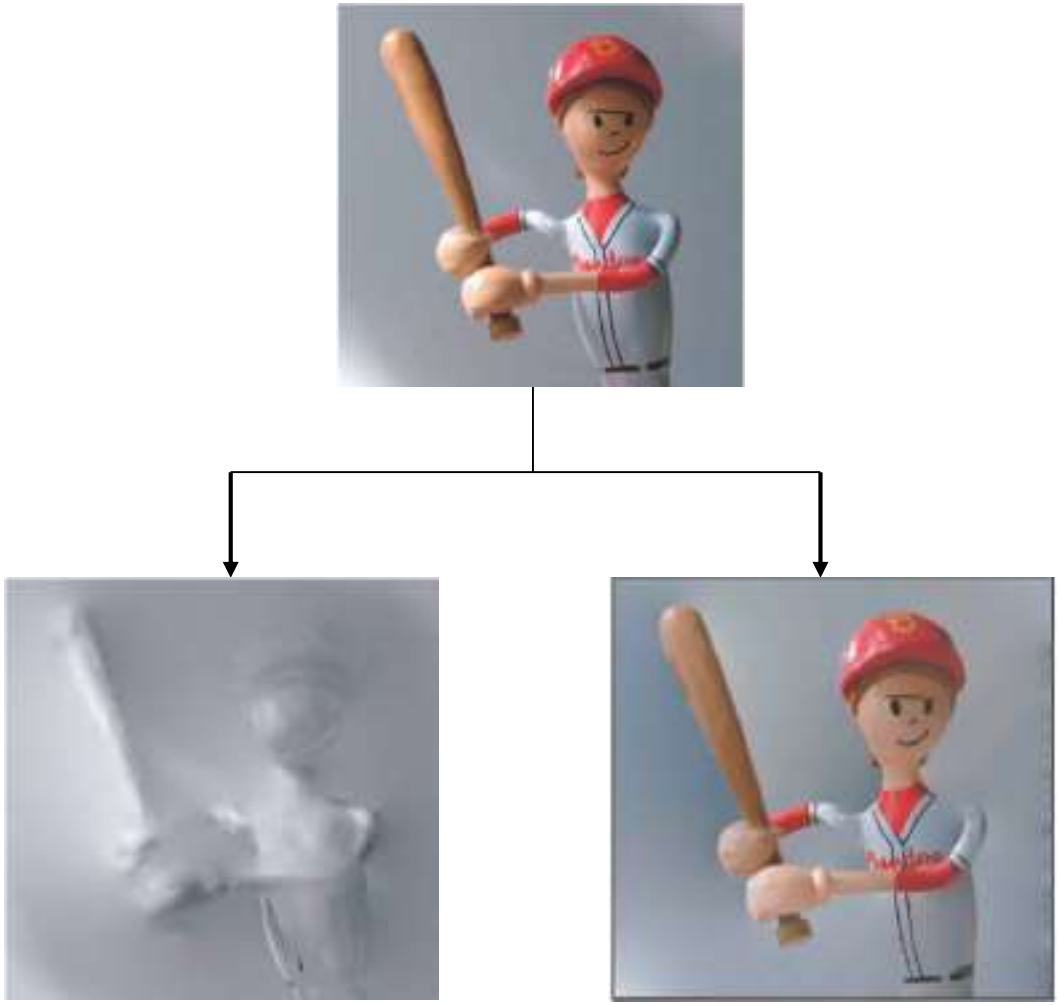


Shadow Removed

Intrinsic Images



Intrinsic Images

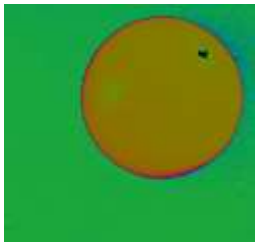


Summary

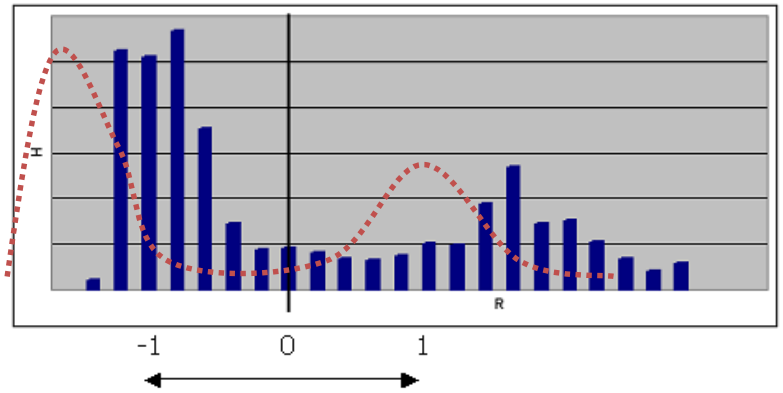
- Estimation of the light source

- Invariant color features

- Be aware of instabilities

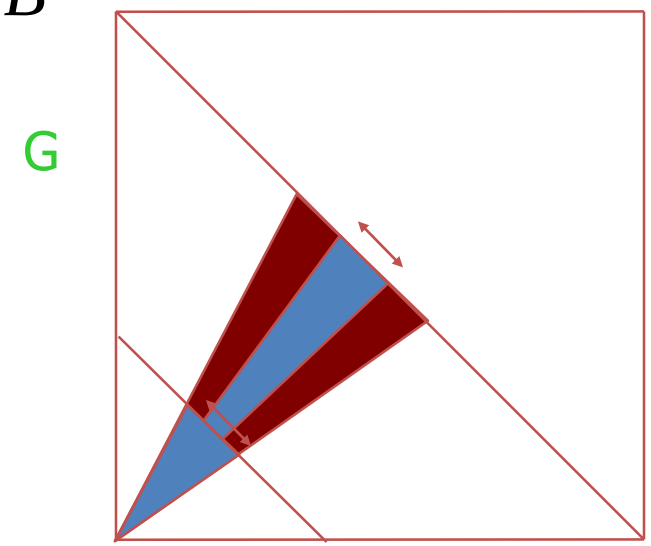


Kernel Density Estimation for Object Recognition



Color Invariants: Instabilities

$$r = \frac{R}{R+G+B}$$



$$I = (R+G+B)/3$$

Color Invariants: Instabilities

Measuring two quantities x and y to calculate their sum $x + y$

Then the highest probable value of $x + y = x_{\text{best}} + y_{\text{best}} + (\delta x + \delta y)$

Then the lowest probable value of $x + y = x_{\text{best}} + y_{\text{best}} - (\delta x + \delta y)$

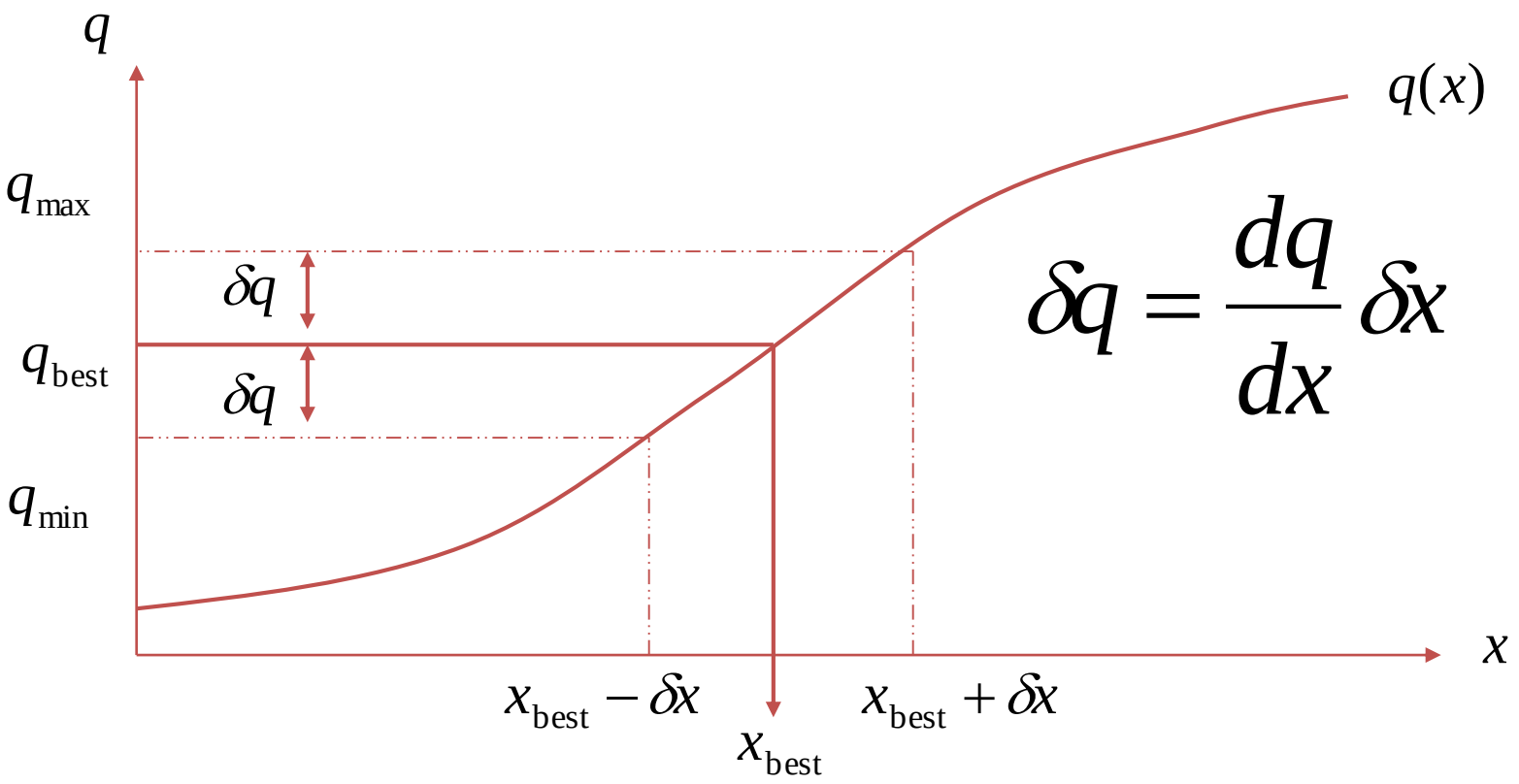
Where x_{best} is the best estimate of x and δx the uncertainty of x

$$q = x + \ldots + z - (u + \ldots + w)$$

$$\delta q \approx \delta x + \ldots + \delta z + \delta u + \ldots + \delta w$$

Color Invariants: Noise Propagation

$$\delta q = q(x_{\text{best}} + \delta x) - q(x_{\text{best}})$$



$$\delta q = \frac{dq}{dx}\delta x$$

Color Invariants: Noise Propagation

Suppose that $u,\ldots,w$ are measured with corresponding uncertainties $\sigma_u,\ldots,\sigma_w$ to compute function $q(u,\ldots,w)$.

The predicted uncertainty is defined by :

$$\sigma_q = \sqrt{\left(\frac{\partial q}{\partial u}\sigma_u\right)^2 + \ldots + \left(\frac{\partial q}{\partial w}\sigma_w\right)^2}$$

The uncertainty in q is never larger than the ordinary sum

$$\sigma_q \leq \left|\frac{\partial q}{\partial u}\right|\sigma_u + \ldots \left|\frac{\partial q}{\partial w}\right|\sigma_w$$

if and only if the uncertainties $\sigma_u,\ldots,\sigma_w$ are relatively small.

Noise Propagation: rgb

Function $q(x,\ldots,z)$ then $\delta_q = \sqrt{\left(\frac{\partial q}{\partial x}\delta x\right)^2 + \ldots + \left(\frac{\partial q}{\partial z}\delta z\right)^2}$

$$r(R,G,B) = \frac{R}{R+G+B}$$

Normalized color value

$$\delta_r = \frac{\sqrt{R^2(\delta_B^2+\delta_G^2)+(B+G)\delta_R^2}}{(B+G+R)^4}$$

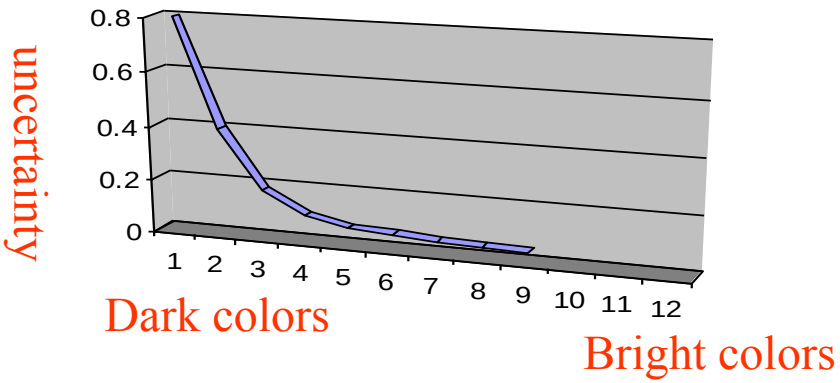
Predicted uncertainty of normalized color value

Measured values and predicted uncertainty

Noise Propagation: Hue

$$H = \arctan\left(\frac{\sqrt{3}(G-B)}{(R-G)+(R-B)}\right)$$

$$\sigma_H = \frac{1}{2}\sqrt{3}\left(\frac{-2BR\sigma_G^2 + R^2\sigma_B^2 + \sigma_G^2 + G^2(\sigma_B^2+\sigma_R^2) + B^2(\sigma_G^2+\sigma_R^2) - 2G(R\sigma_B^2 + B\sigma_R^2)}{B^2 + G^2 - GR + R^2 - B(G+R)^2}\right)$$



Histogram Construction

$$\tilde{f}(x) = \frac{1}{nh}(\text{number of } X_i \text{ in the same bin as } x)$$

n is the number of pixels X_i in the image, h is the bin width and x the range of the data

Kernel density estimator:

$$\tilde{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - X}{h})$$

where K is a function satisfying $\int K(x)dx = 1$

Variable Kernel Density Estimation

The most frequently occuring noise is additive Gaussian noise.

It is widely used to model thermal noise and is the limiting behaviour of photon counting noise and film grain noise

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp^{\frac{-x^2}{2}}$$

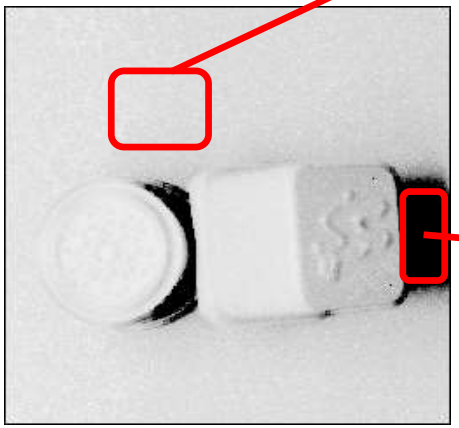
Variable kernel density for *rg* is defined by:

$$\tilde{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \sigma_r^{-1} K(\frac{x-r}{\sigma_r}) \sigma_g^{-1} K(\frac{x-g}{\sigma_g})$$

where σ_r, σ_g denote the uncertainties in normalized color

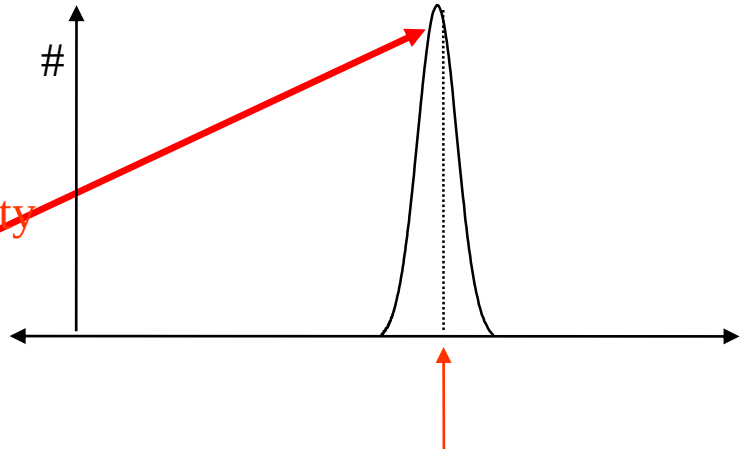
Variable Kernel Density Estimation

image



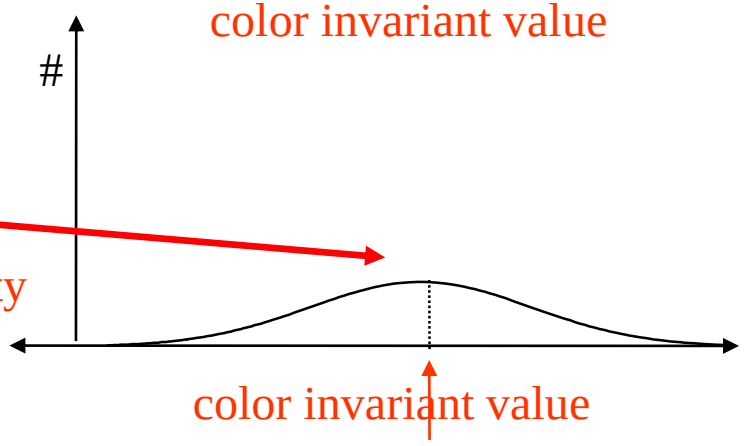
uncertainty

High certainty



color invariant value

Low certainty

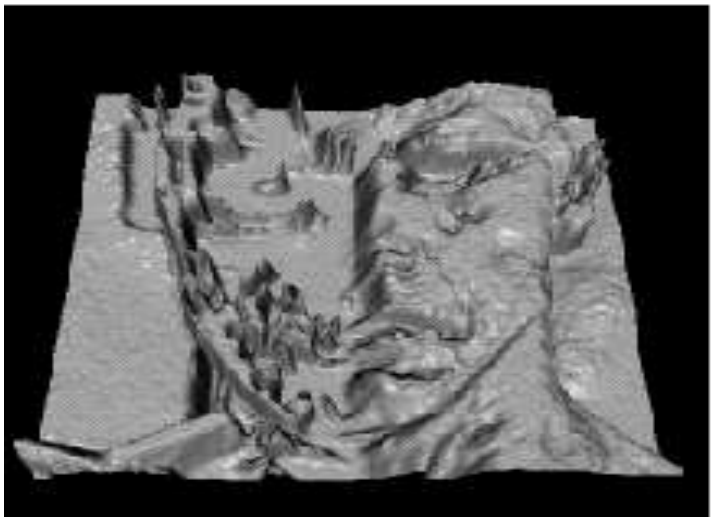


color invariant value

Today's Class

1. Reflection Models
2. Color Invariance
3. **Image Processing**

Images as Functions



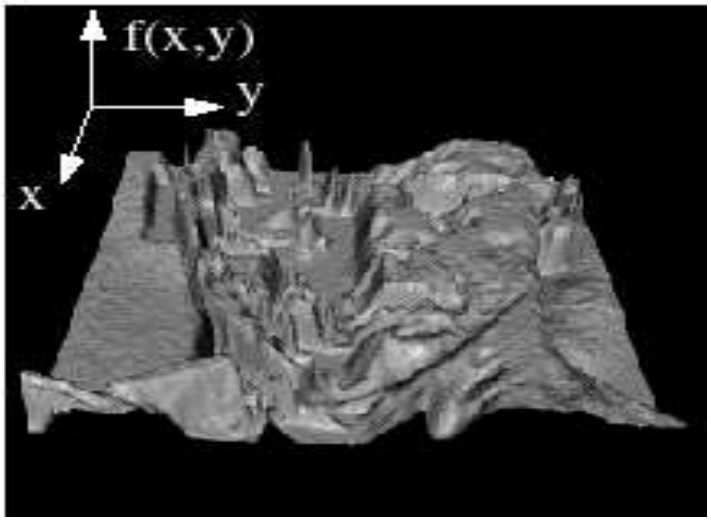


Image Processing

An **image processing** operation typically defines a new image g in terms of an existing image f.

We can transform either the range of f.

$$g(x, y) = t(f(x, y))$$

Or the domain of f:

$$g(x, y) = f(t_x(x, y), t_y(x, y))$$

What kinds of operations can each perform?

Point Processing

The simplest kind of range transformations are these independent of position x,y:

$$g = t(f)$$

This is called point processing.

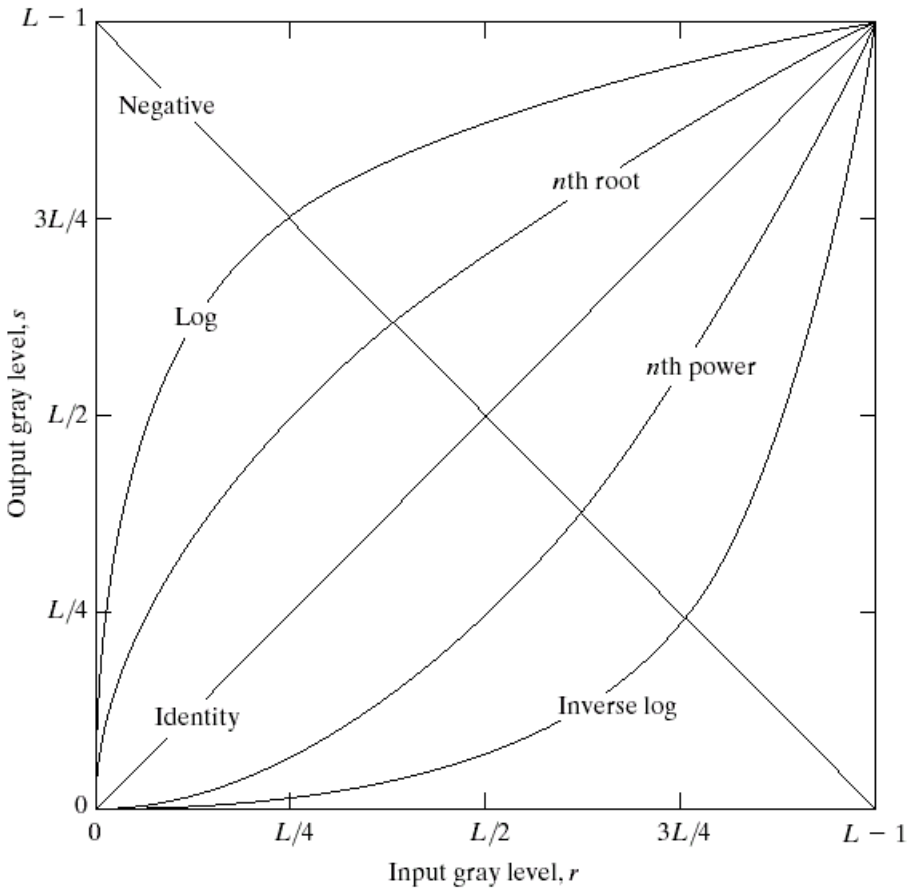
What can they do?

e.g. g = at(f)+b where a = gain (contrast) and b = bias (brightness)

Important: every pixel for himself – spatial information completely lost!

Basic Point Processing

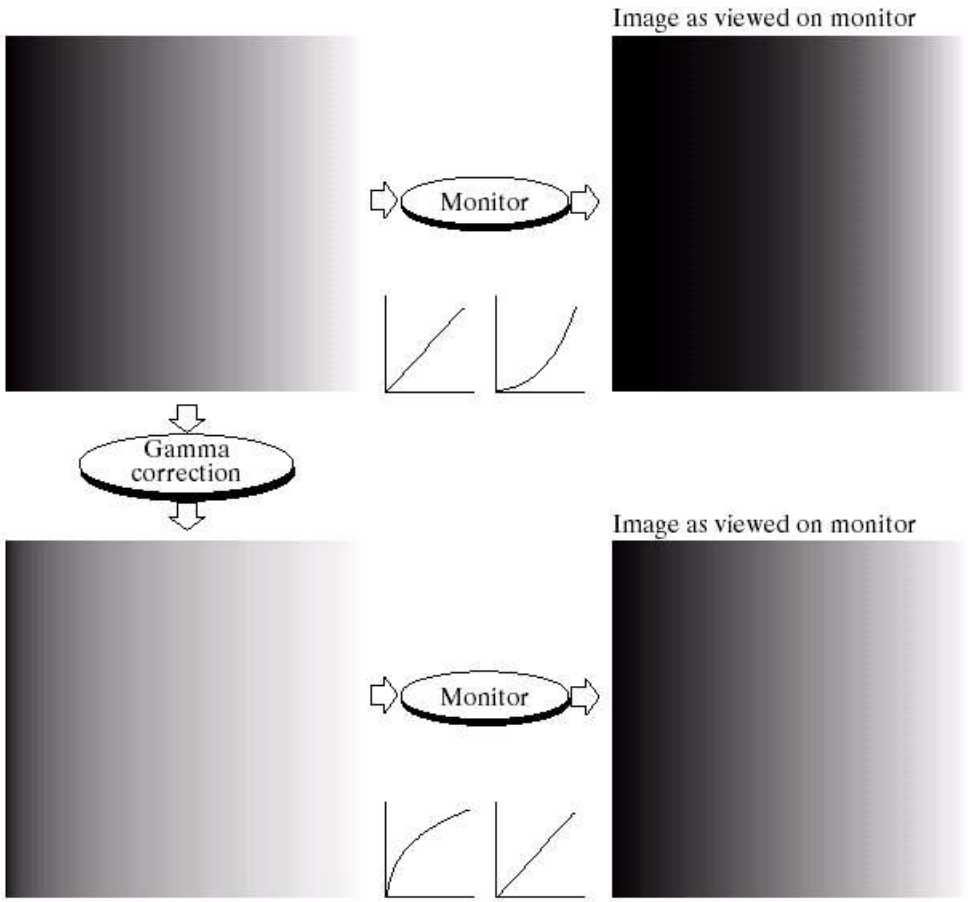
FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.



Gamma Correction

a b
c d

FIGURE 3.7
(a) Linear-wedge gray-scale image. (b) Response of monitor to linear wedge. (c) Gamma-corrected wedge. (d) Output of monitor.



Gamma Measuring Applet:
http://www.cs.berkeley.edu/~efros/java/gamma/gamma.html

Image Enhancement

a b
c d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)

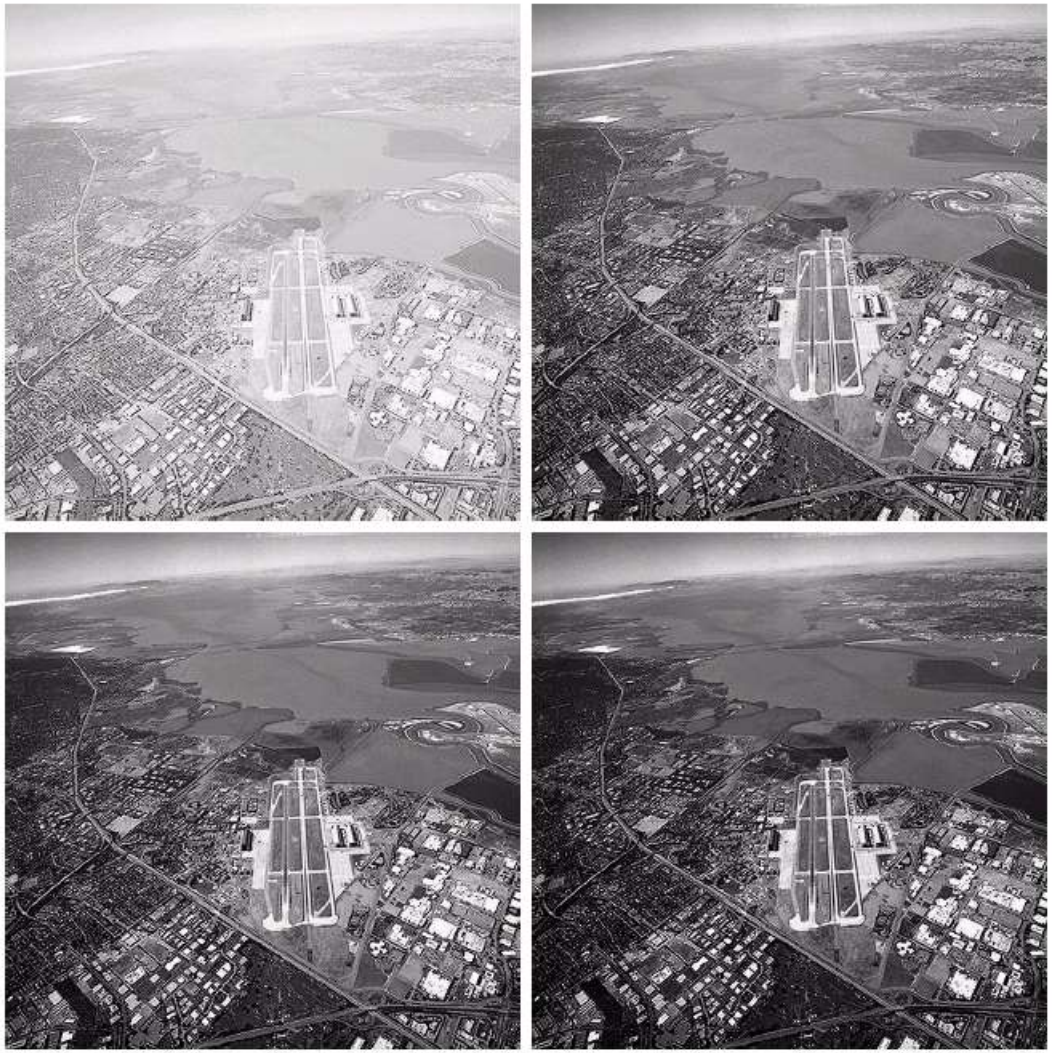


Photo Editing: Matting

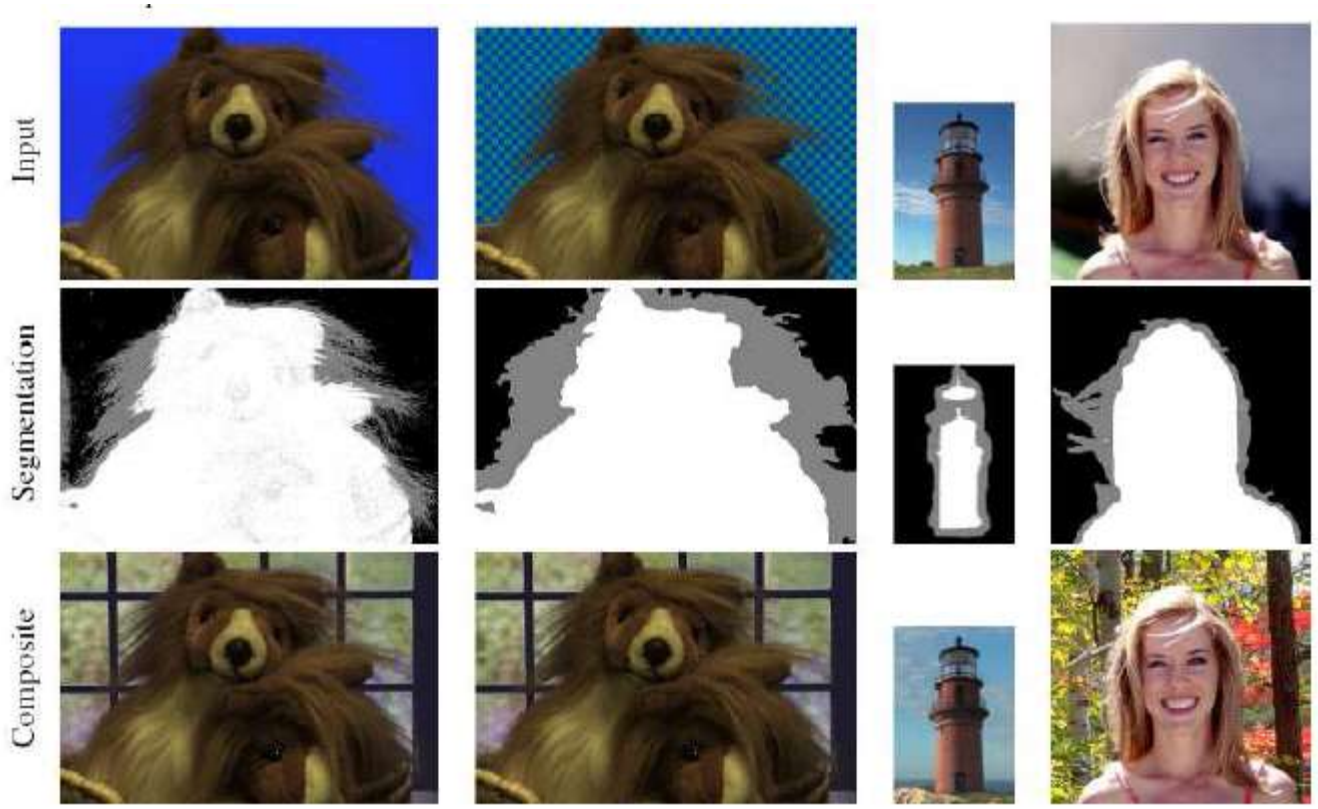
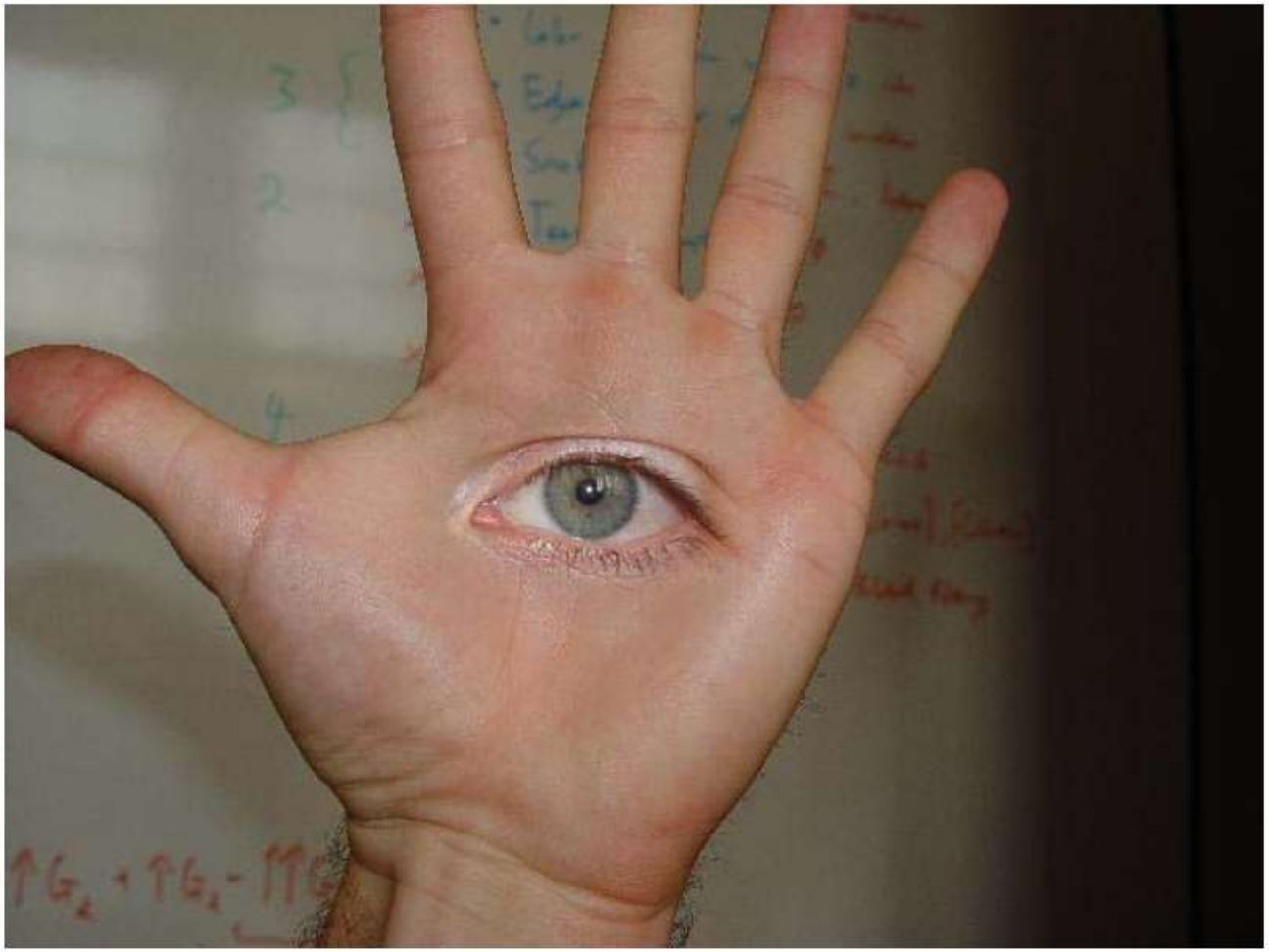


Photo Editing: Matting

Poisson Matting



© prof. dmartin

Neighborhood Processing (filtering)

Q: What happens if I reshuffle all pixels within the image?

A: It's histogram won't change. No point processing will be affected…

Need spatial information to capture this.

Linear Filtering

$$g[\cdot,\cdot]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Slide credit: David Lowe (UBC)

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20						

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30					

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

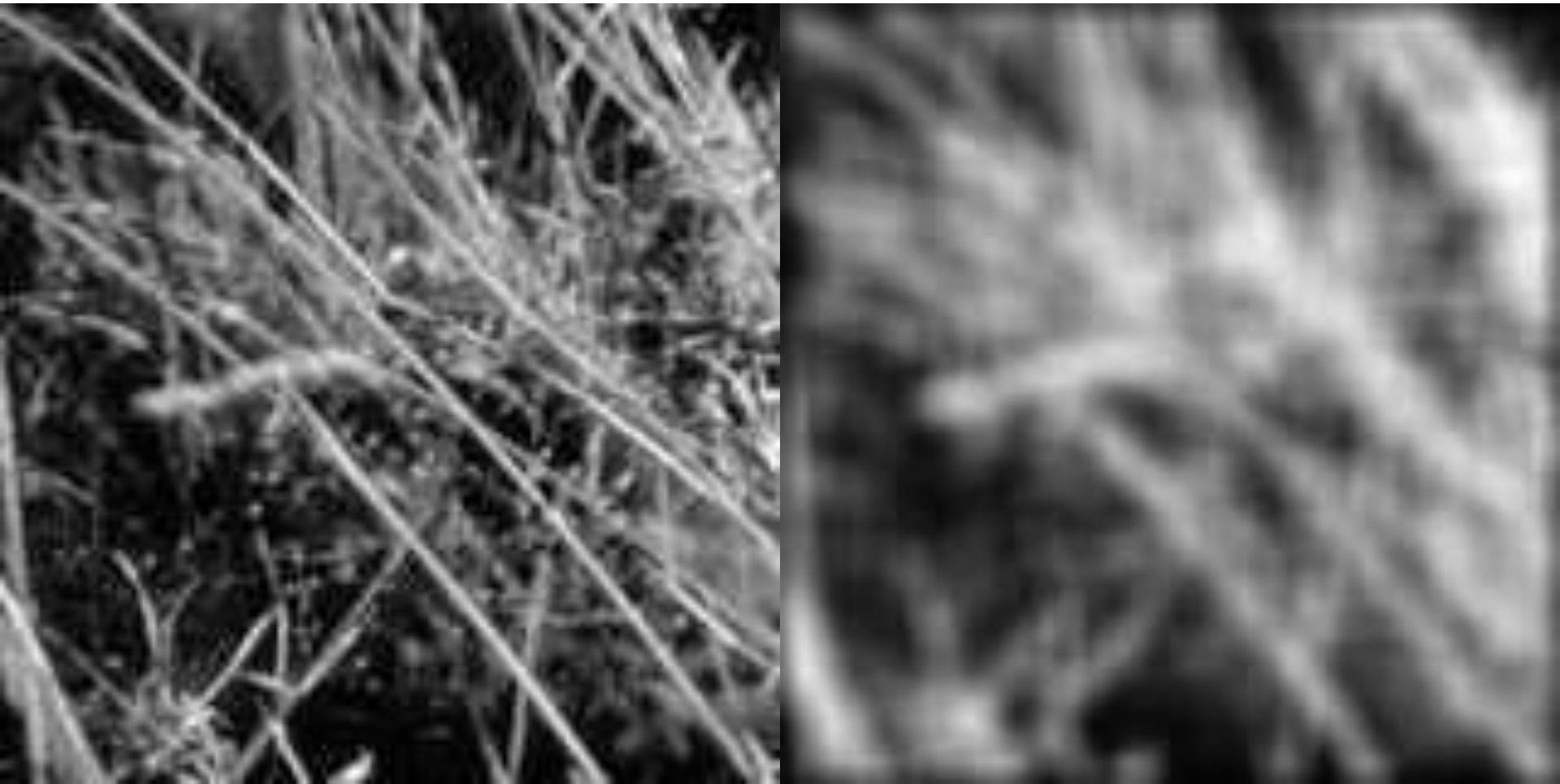
- Achieve smoothing effect (remove sharp features)

$g[\cdot,\cdot]$

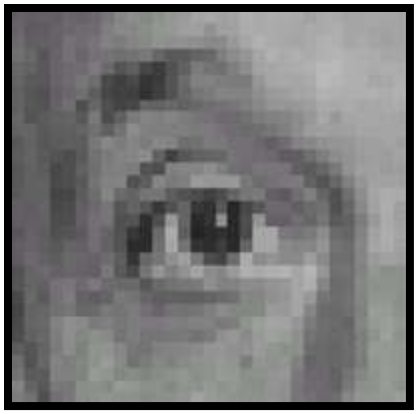
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Slide credit: David Lowe (UBC)

Smoothing with Box Filter



Practice with Linear Filters



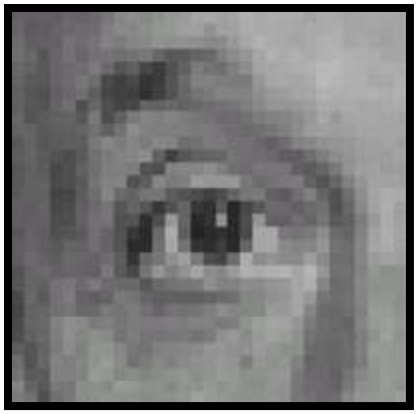
Original

0	0	0
0	1	0
0	0	0

?

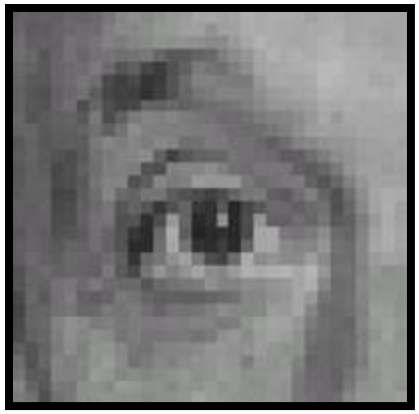
Source: D. Lowe

Practice with Linear Filters



Original

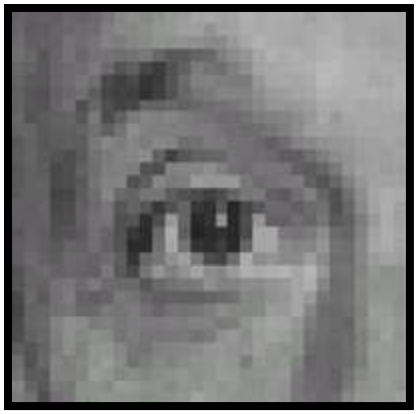
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with Linear Filters



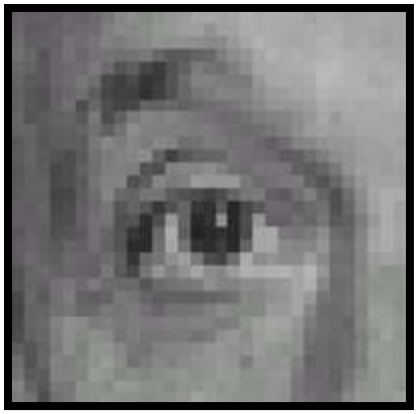
Original

0	0	0
0	0	1
0	0	0

?

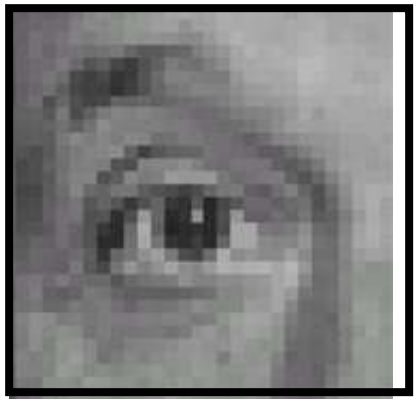
Source: D. Lowe

Practice with Linear Filters



Original

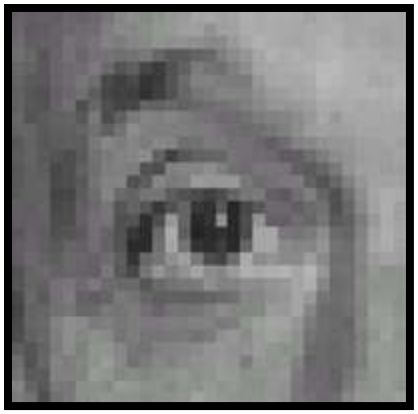
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with Linear Filters

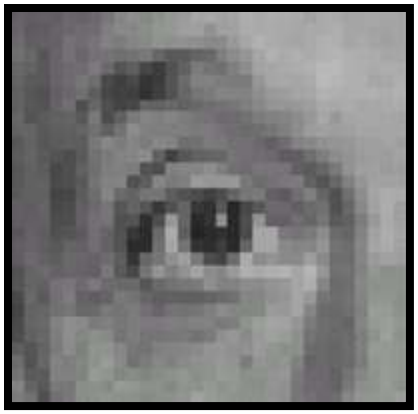


Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

Source: D. Lowe

Practice with Linear Filters



Original

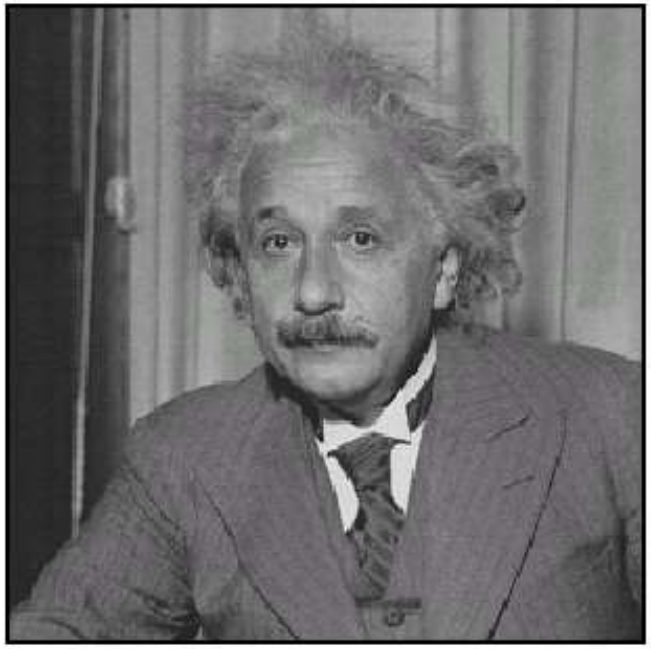
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

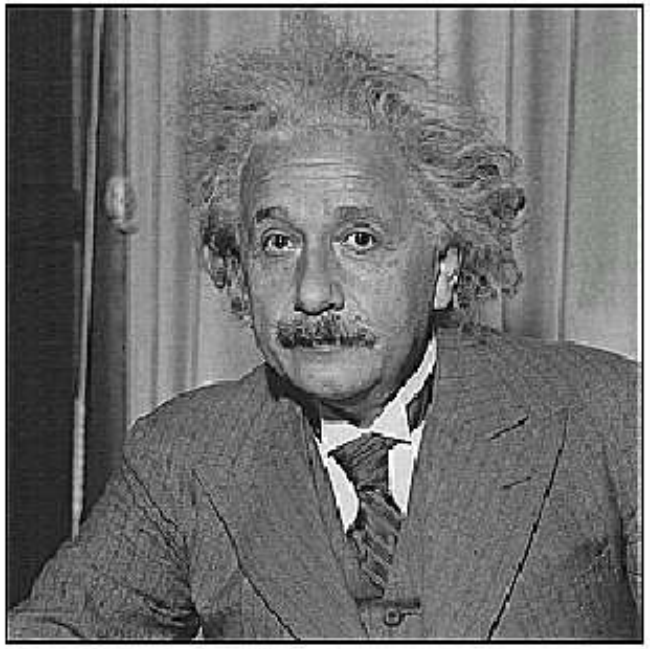
- Accentuates differences with local average

Source: D. Lowe

Sharpening



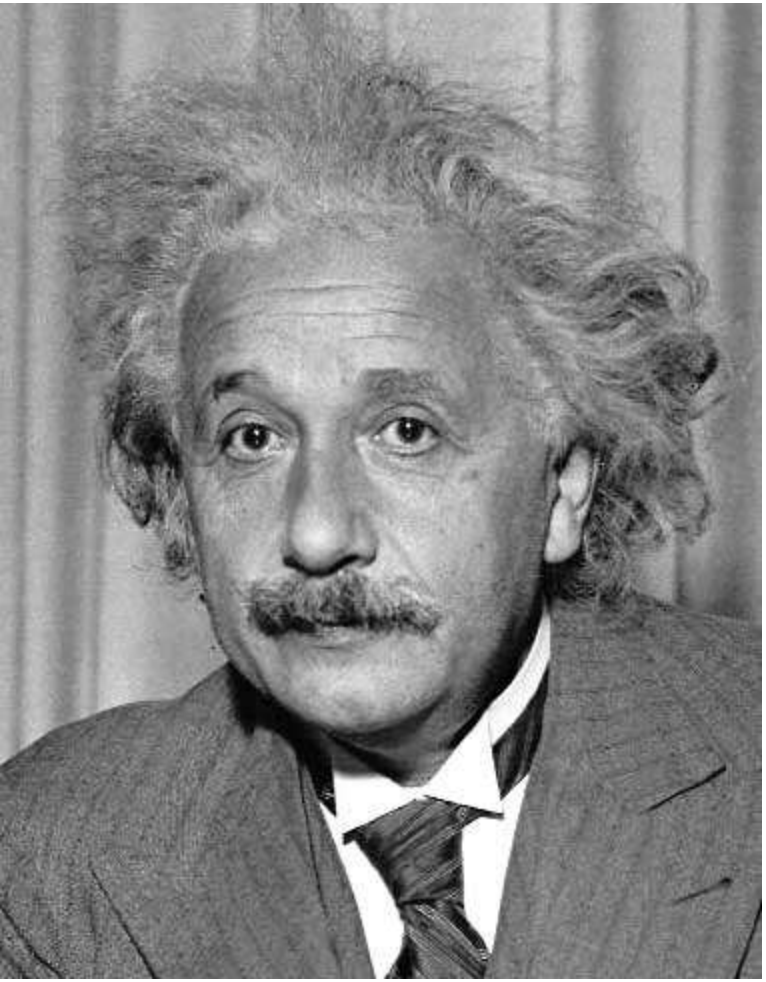
before



after

Source: D. Lowe

Other Filters

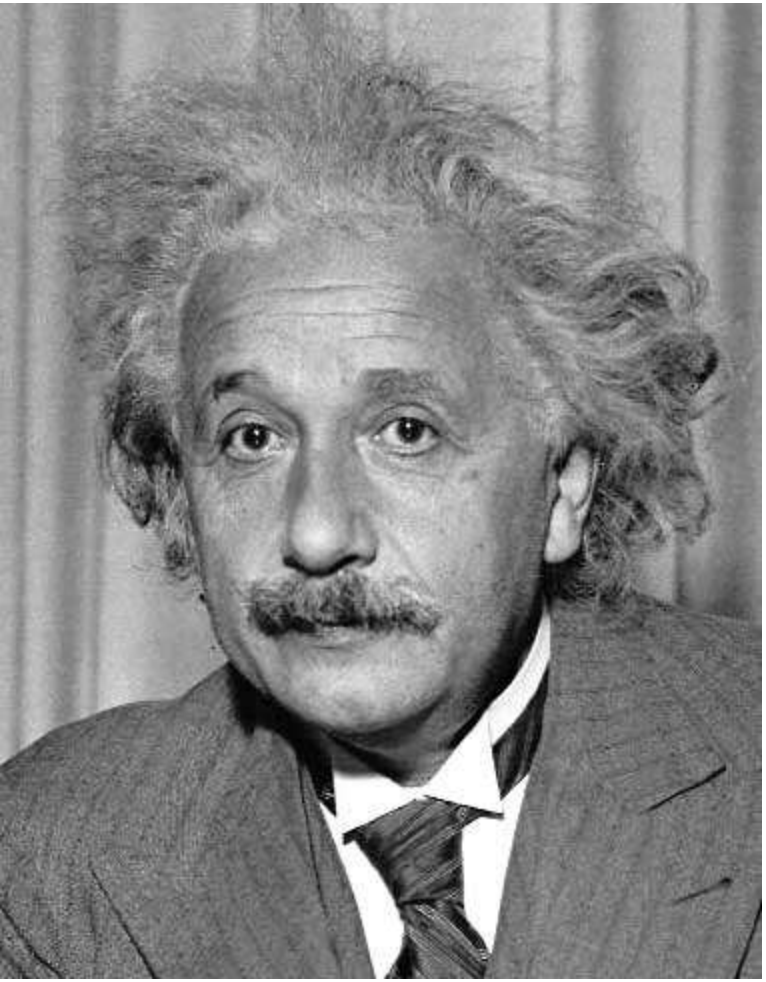


1	0	-1
2	0	-2
1	0	-1

Sobel

Vertical Edge
(absolute value)

Other Filters



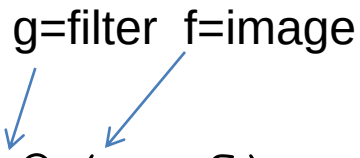
1	2	1
0	0	0
-1	-2	-1

Sobel

Horizontal Edge (absolute value)

Filtering vs. Convolution

- 2d filtering



  - `h=filter2(g,f);` or `h=imfilter(f,g);`

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

- 2d convolution
  - `h=conv2(g,f);`

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m-k,n-l]$$

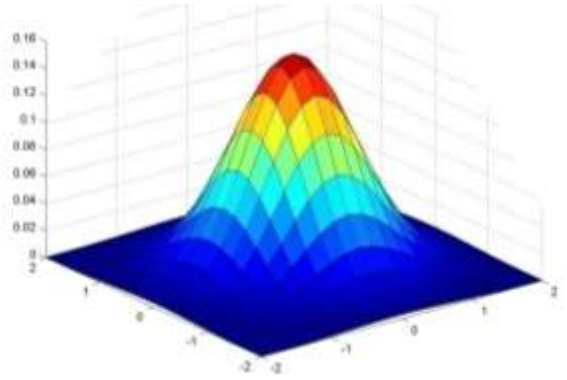
Properties

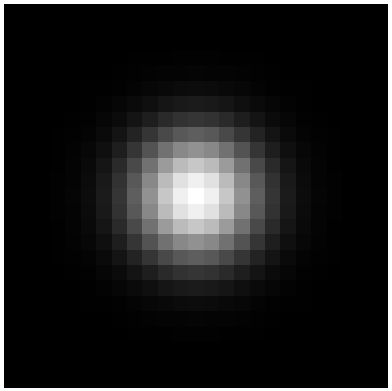
- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$,
  $a * e = a$

Source: S. Lazebnik

Important Filter: Gaussian

Weight contributions of neighboring pixels by nearness





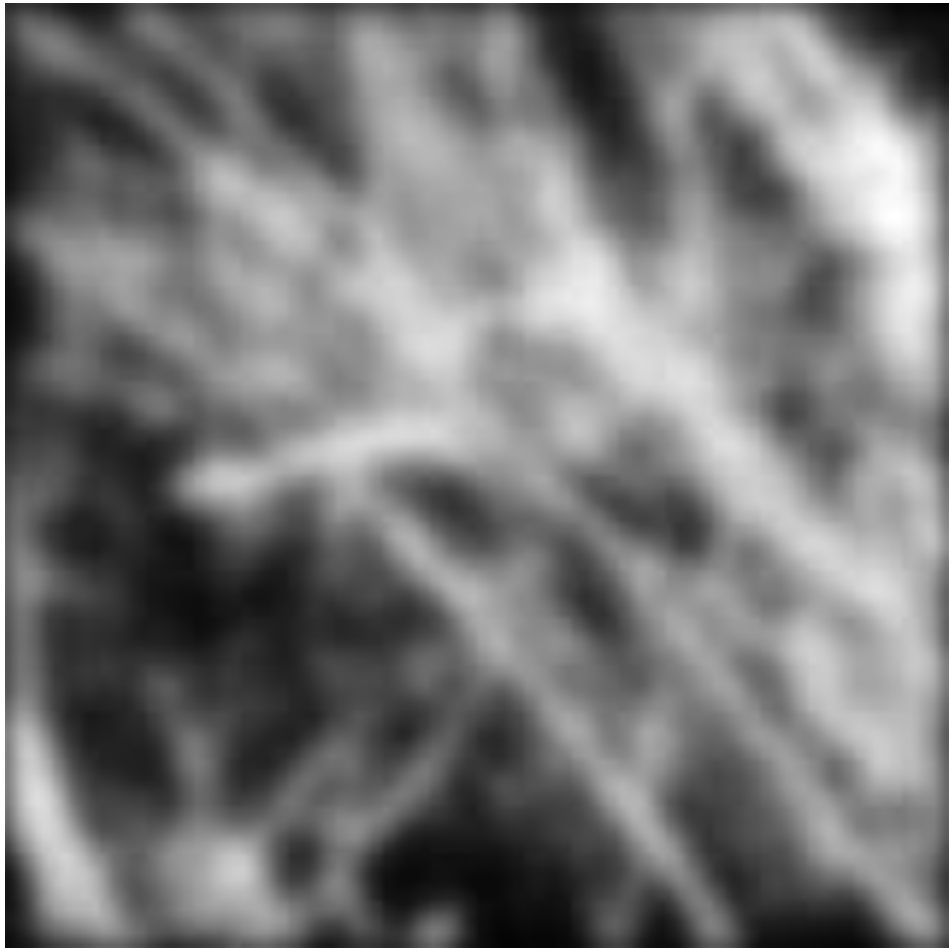
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

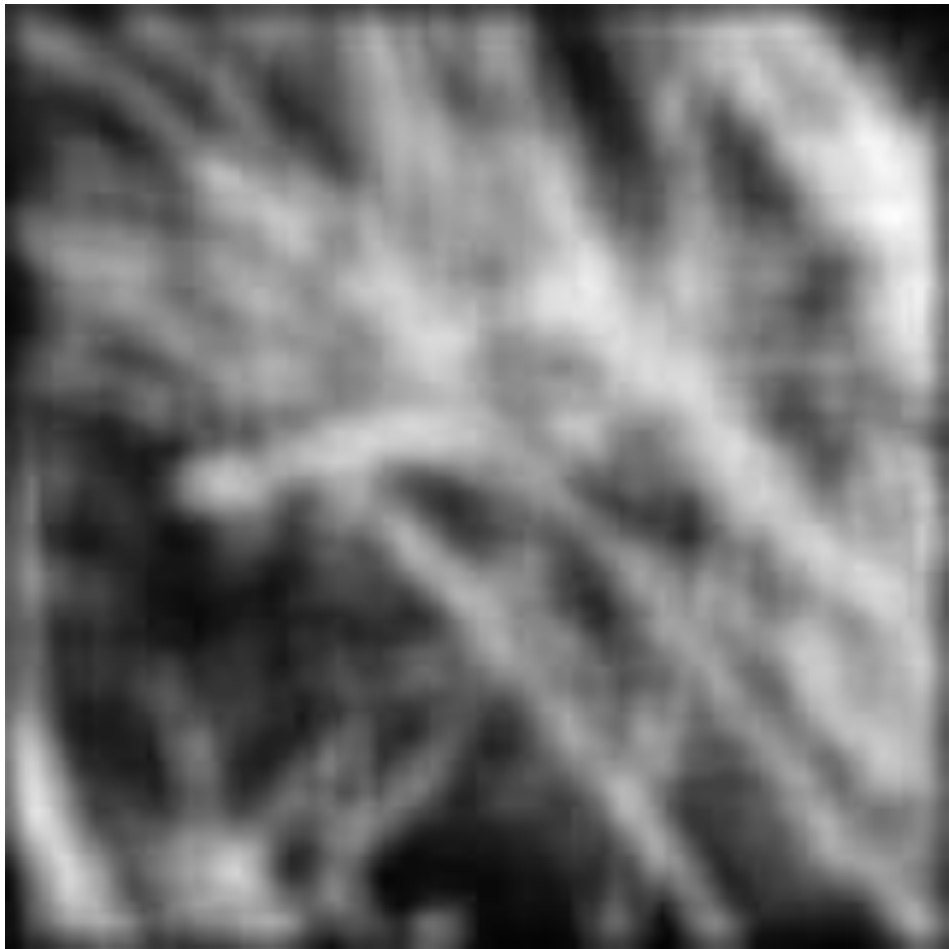
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

Smoothing with Gaussian filter



Smoothing with Box Filter



Gaussian Filters

- Remove "high-frequency" components from the image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian Filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

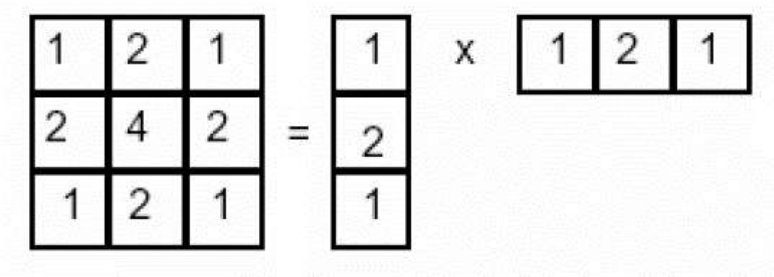
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Separability Example

The filter factors into a product of 1D filters:



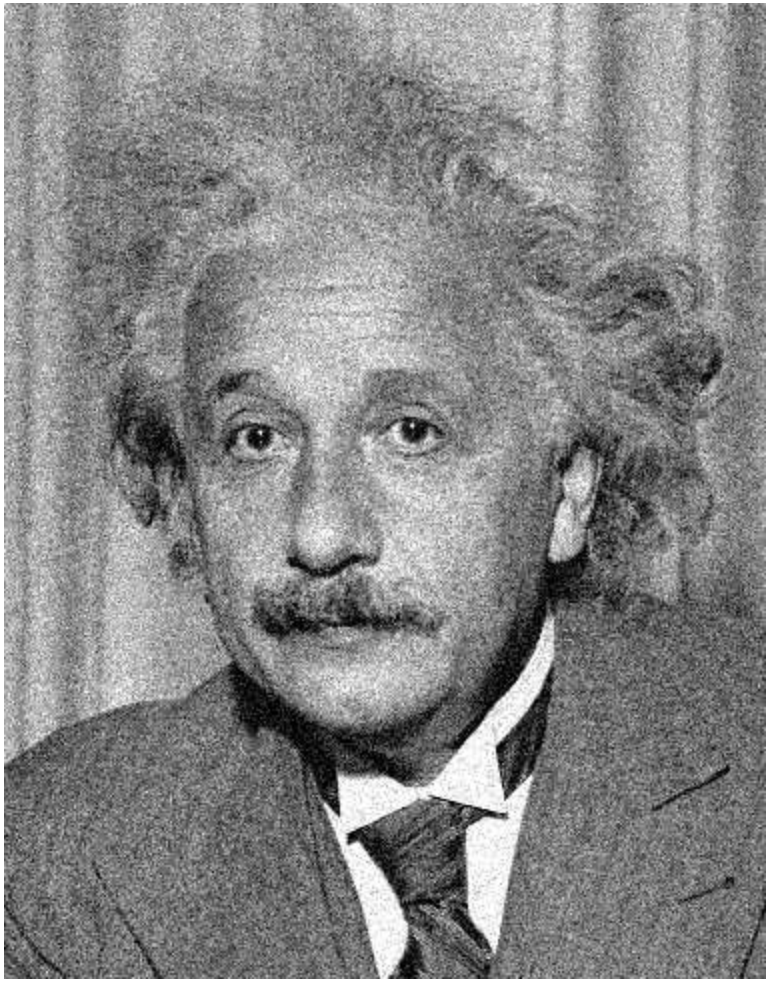
Source: K. Grauman

Practical Matters

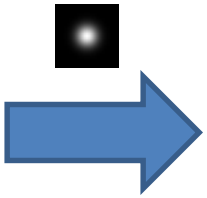
- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

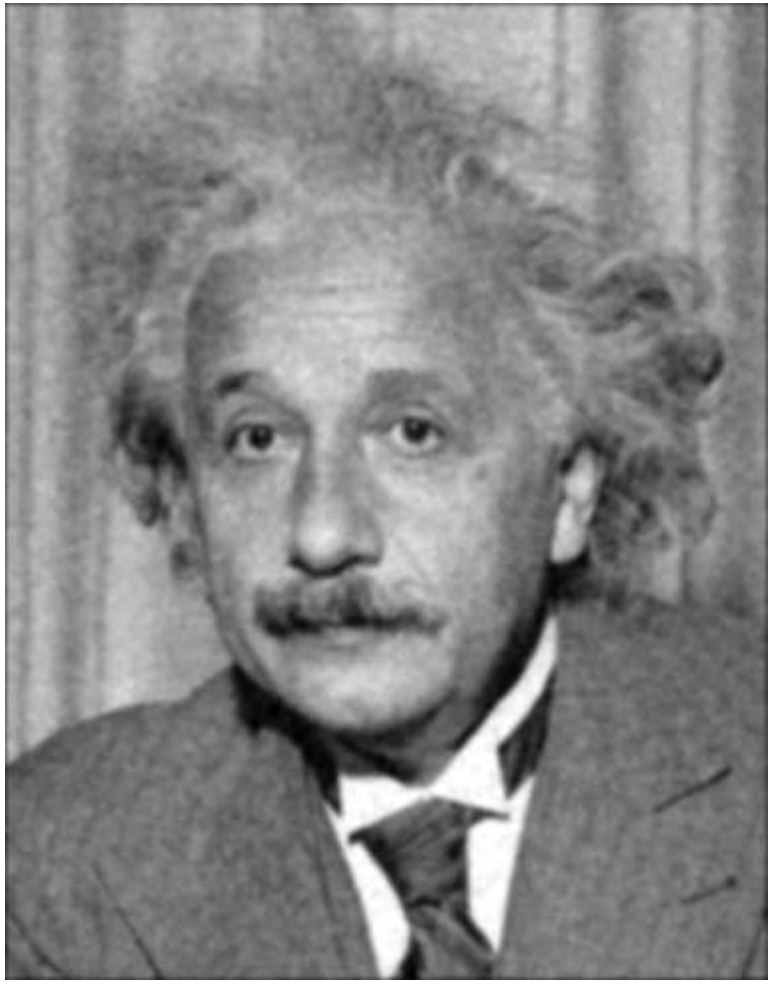
Denoising



Additive Gaussian Noise



Gaussian Filter



Slide: Hoiem

Salt-and-Pepper Noise: Gaussian

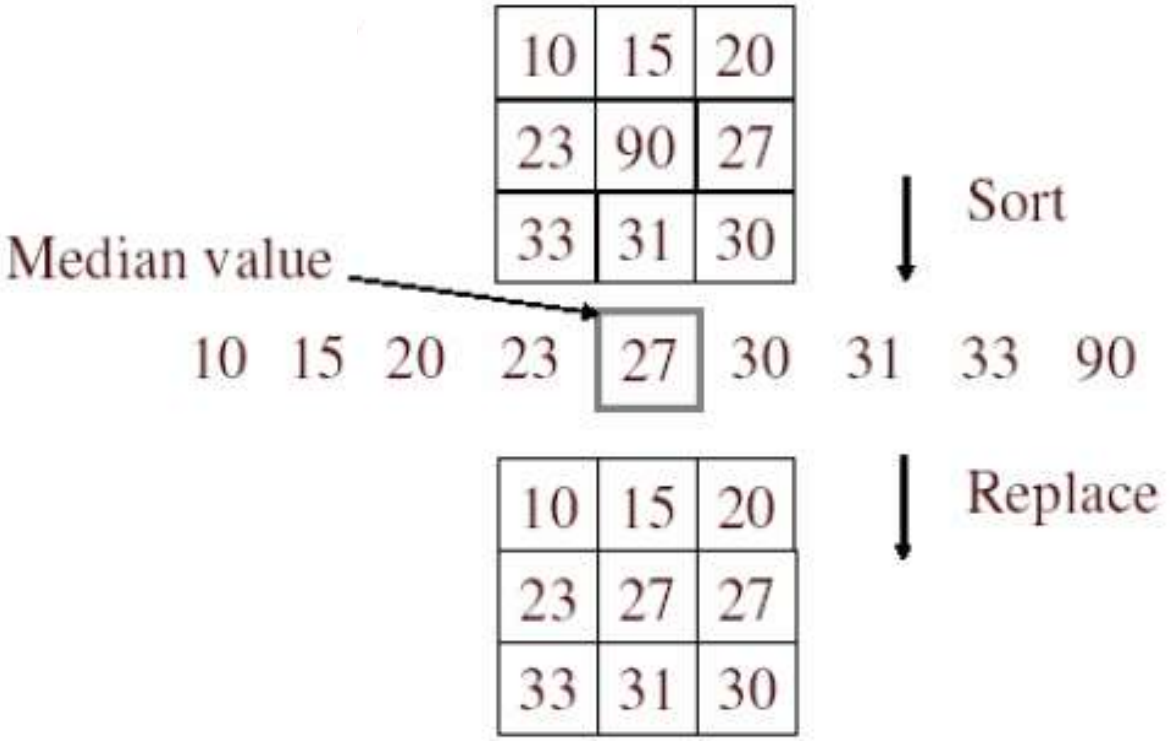
3x3

5x5

7x7

Alternative Idea: Median Filtering

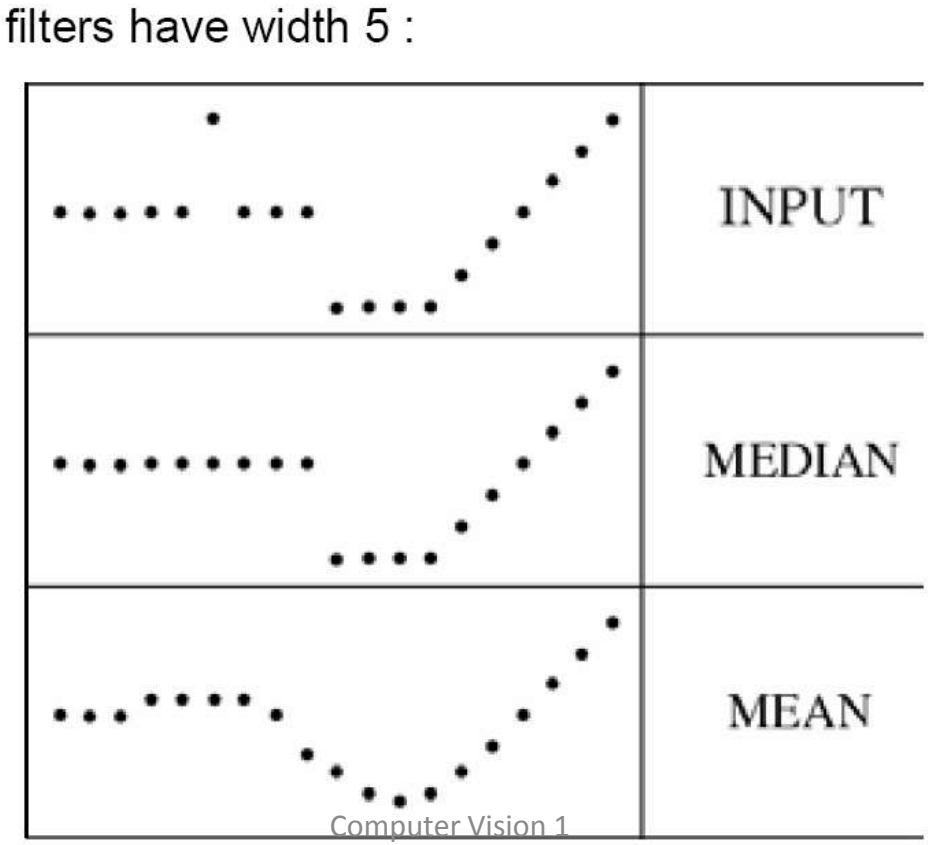
- A **median filter** operates over a window by selecting the median intensity in the window



Source: K. Grauman

Median Filter

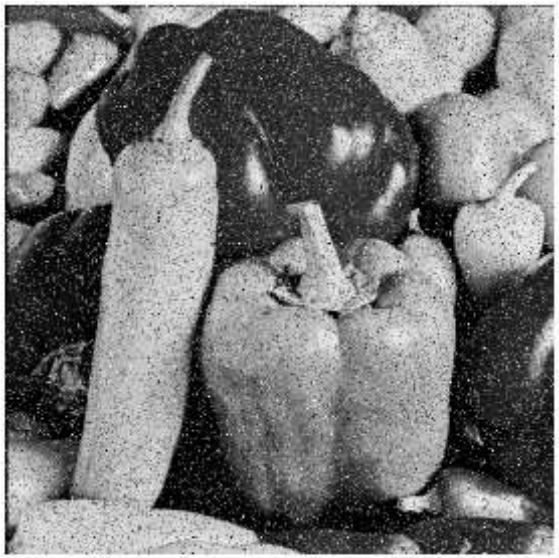
- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers

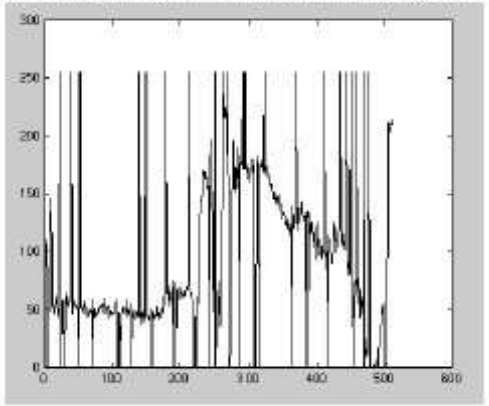


Source: K. Grauman

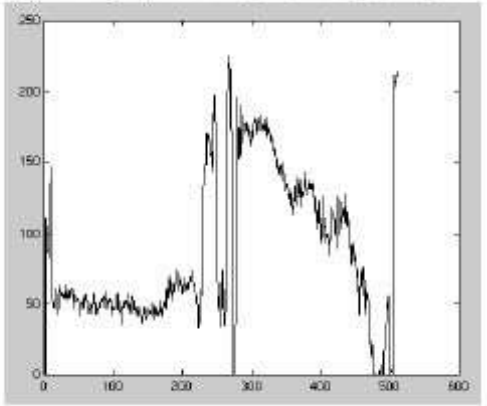
Median Filter

Salt-and-pepper noise



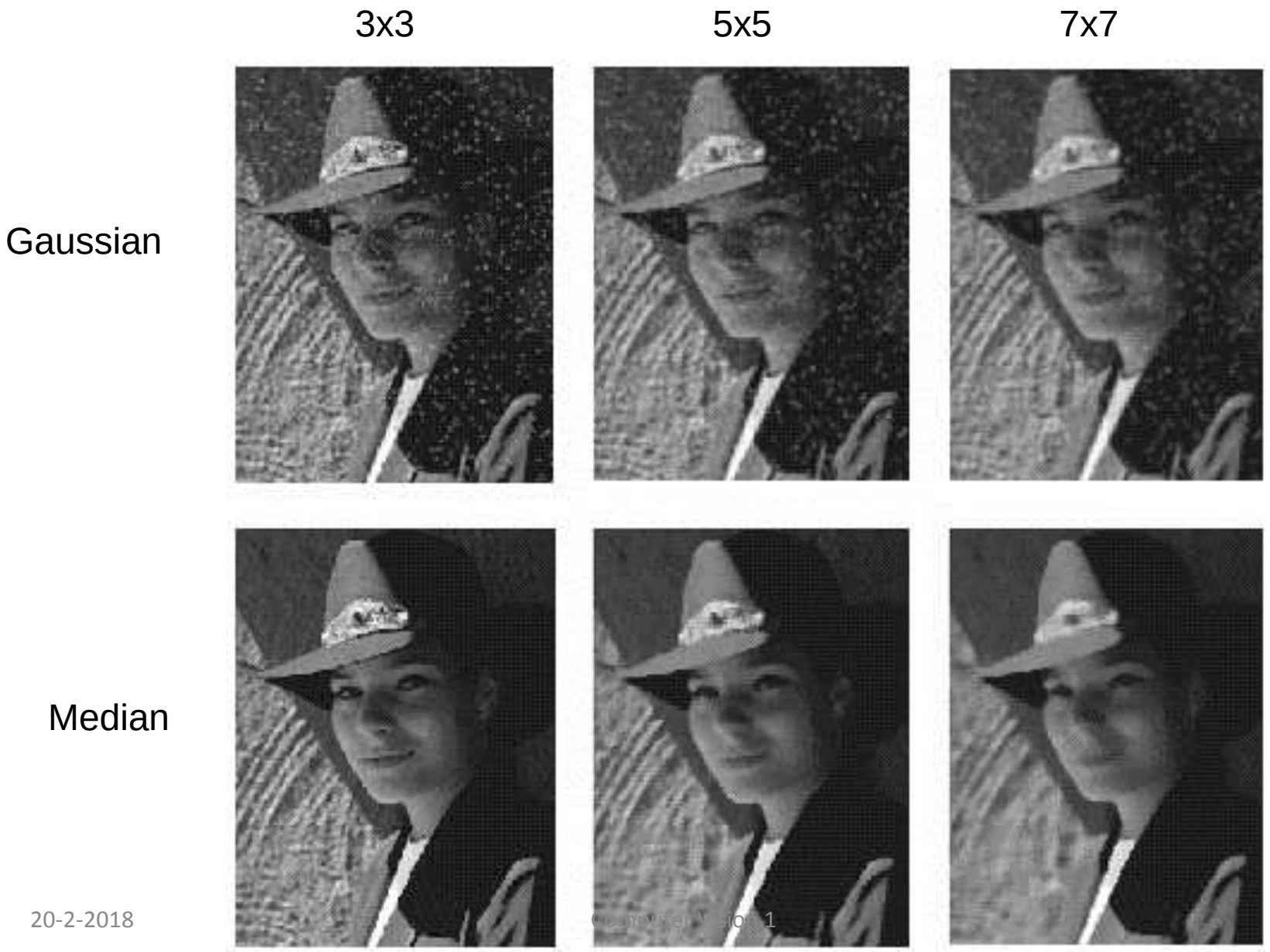


Median filtered



Source: M. Hebert

Median vs. Gaussian Filtering



Other Non-linear Filters

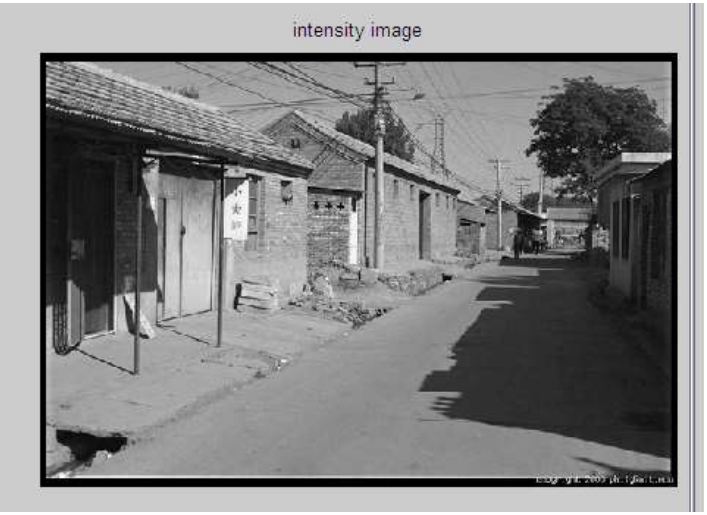
- Weighted median (pixels further from center count less)
- Clipped mean (average, ignoring few brightest and darkest pixels)
- Bilateral filtering (weight by spatial distance *and* intensity difference)

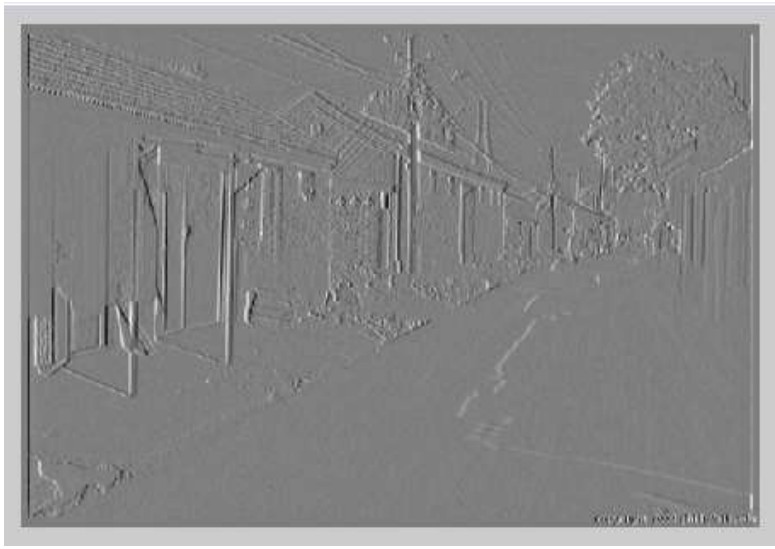
Bilateral filtering

Image: http://vision.ai.uiuc.edu/?p=1455

Review: Edge Detection

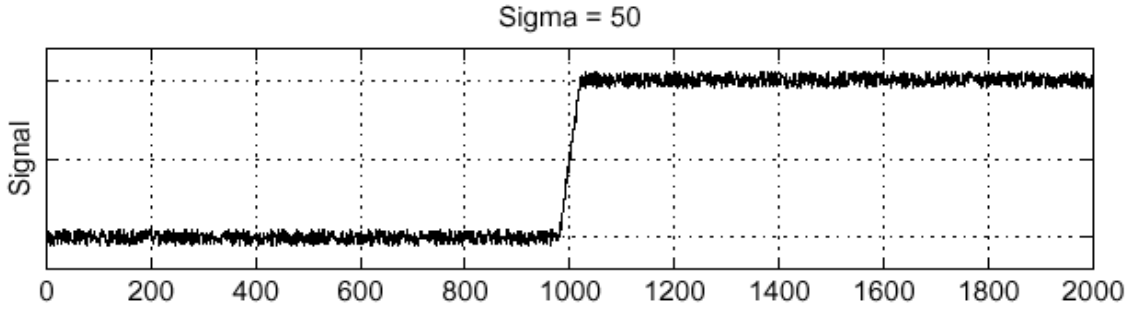
1	0	-1
2	0	-2
1	0	-1





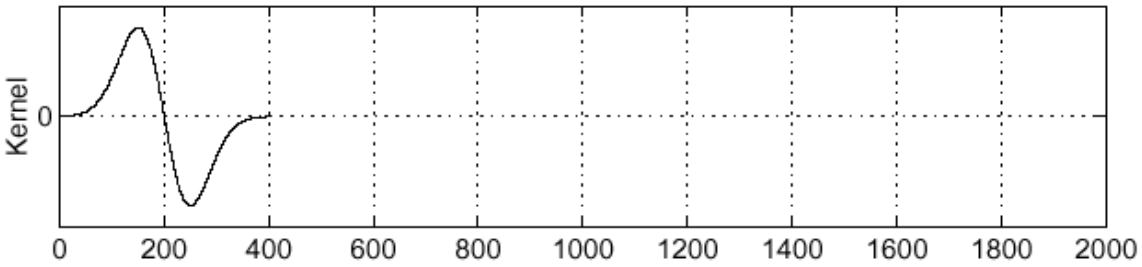
Review: Edge Detection

f



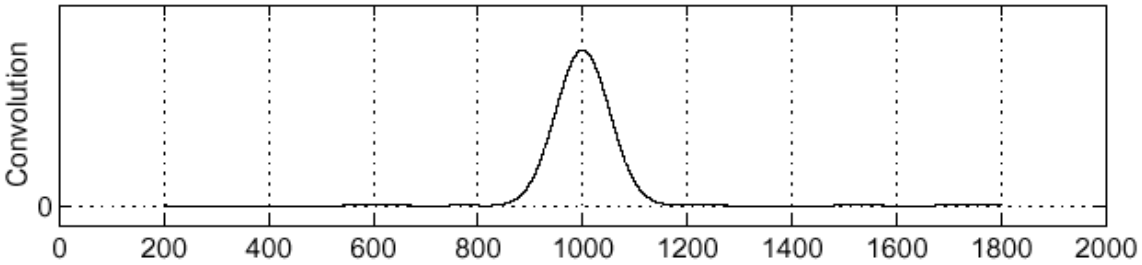
Edge

$\frac{d}{dx} g$



Derivative of Gaussian

$f * \frac{d}{dx} g$

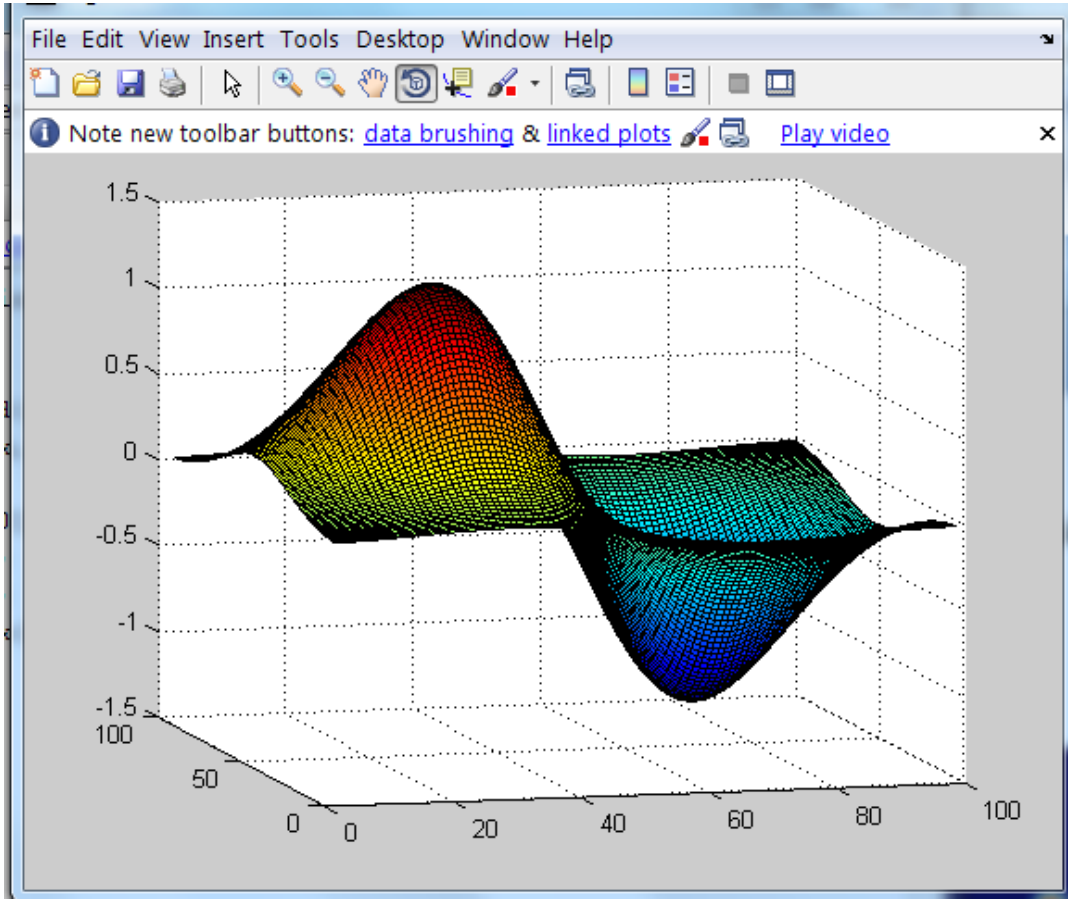


Edge = maximum of derivative

Source: S. Seitz

Review

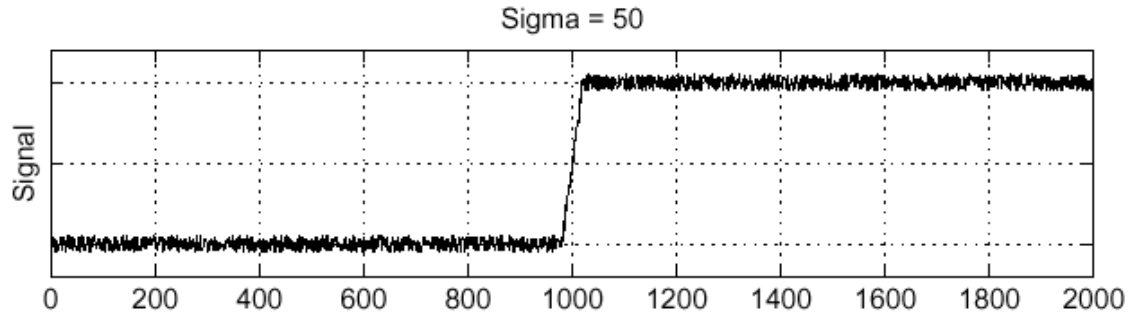
Derivative of Gaussian



Slide: Hoiem

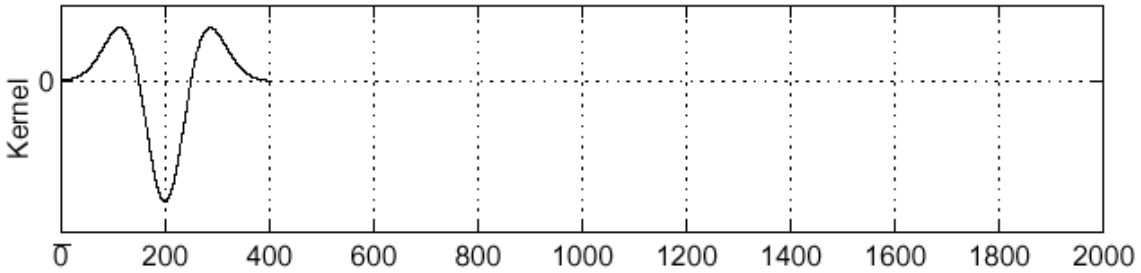
Review: Edge Detection

f



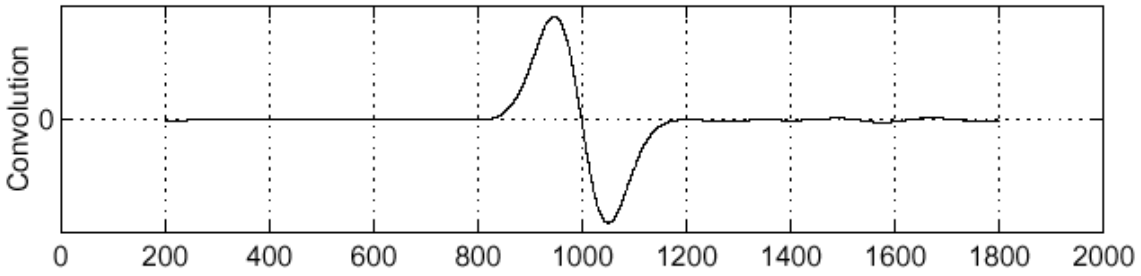
Edge

$\frac{d^2}{dx^2} g$



Second derivative of Gaussian (Laplacian)

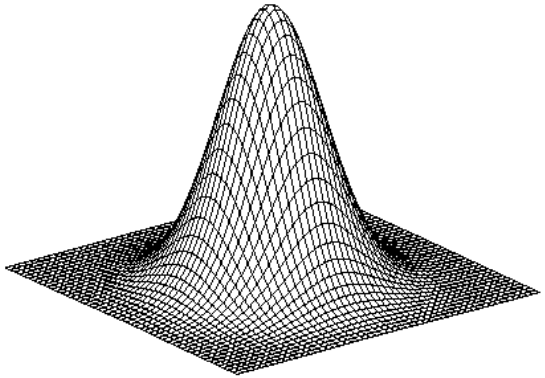
$f * \frac{d^2}{dx^2} g$



Edge = zero crossing of second derivative

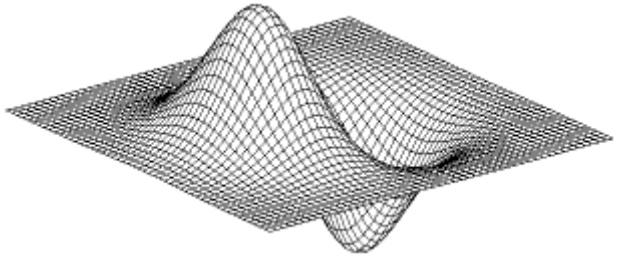
Source: S. Seitz

2D Edge Detection Filters



Gaussian

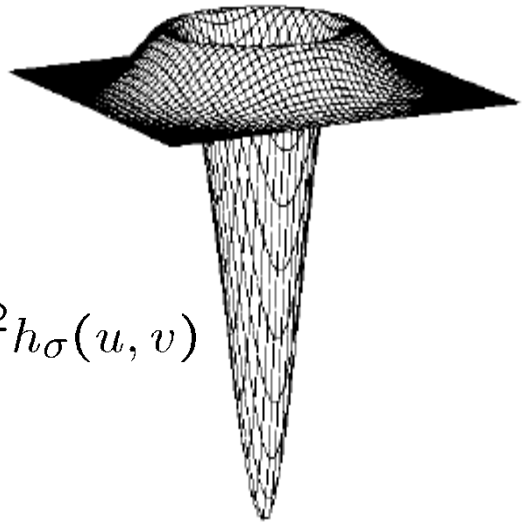
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

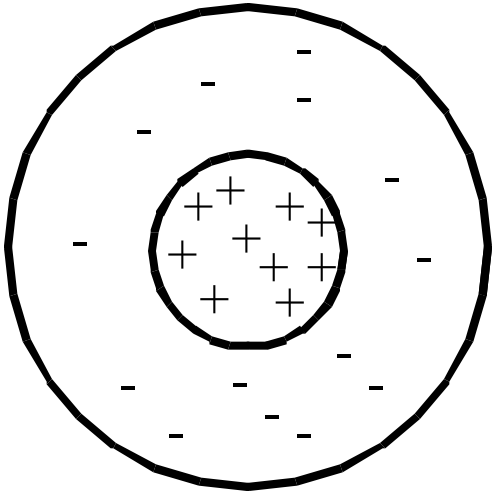


$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

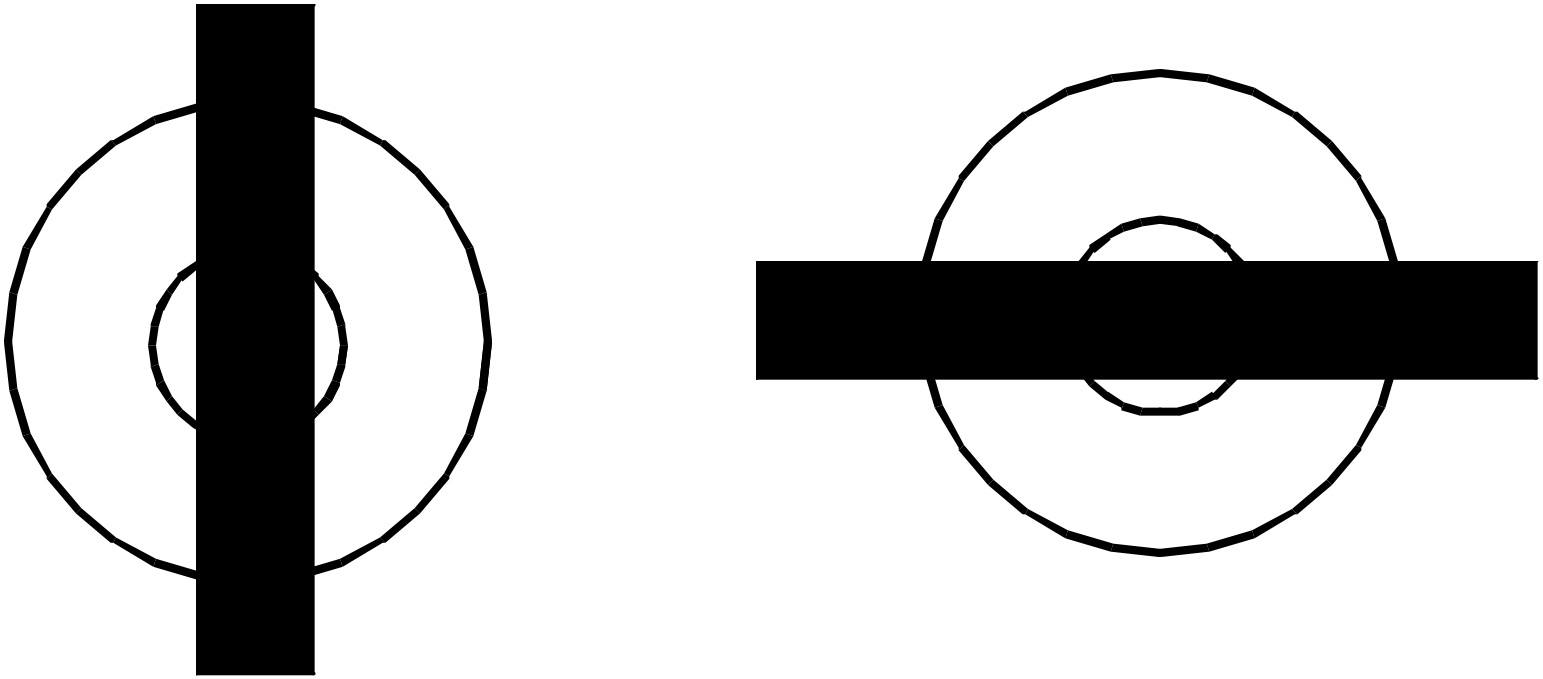
“On-Center” Ganglion Cell



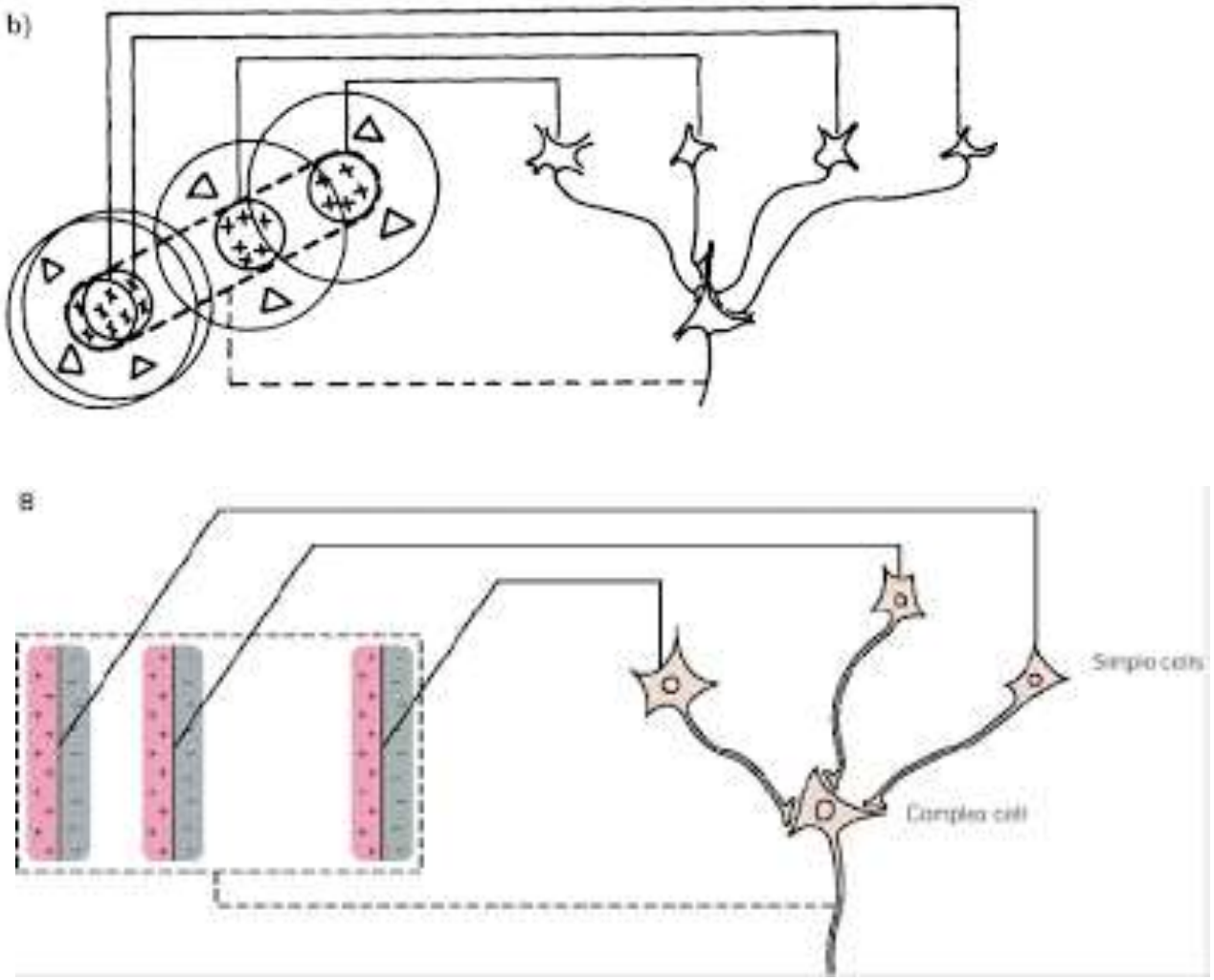
Responds maximally to light increments in the center,

and light decrements in the surround.

Ganglion cells have no orientation preference.

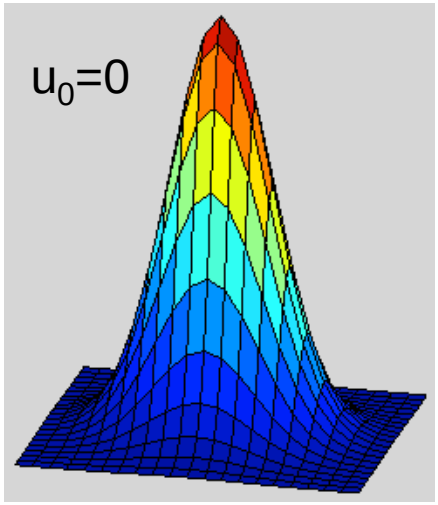


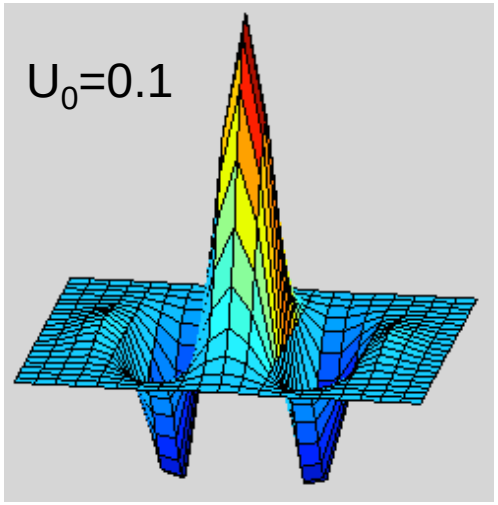
Model of how center-surround cells can be building blocks for simple cells.

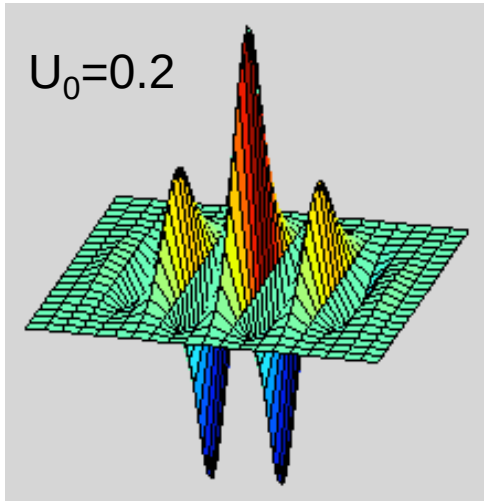


Gabor wavelets

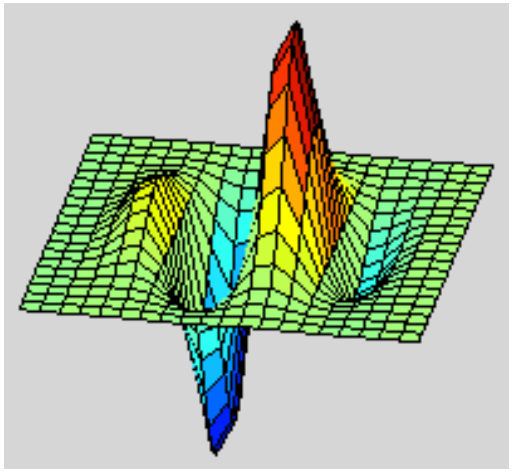
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

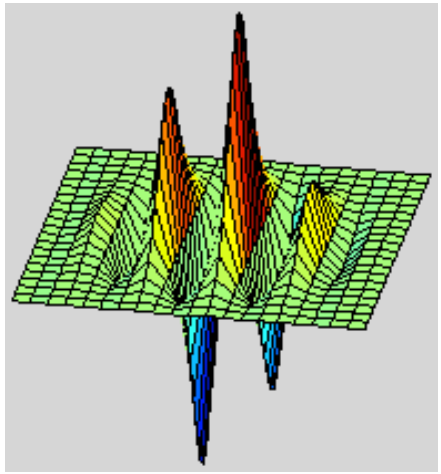


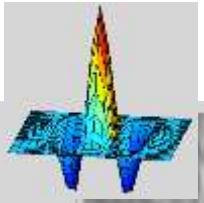


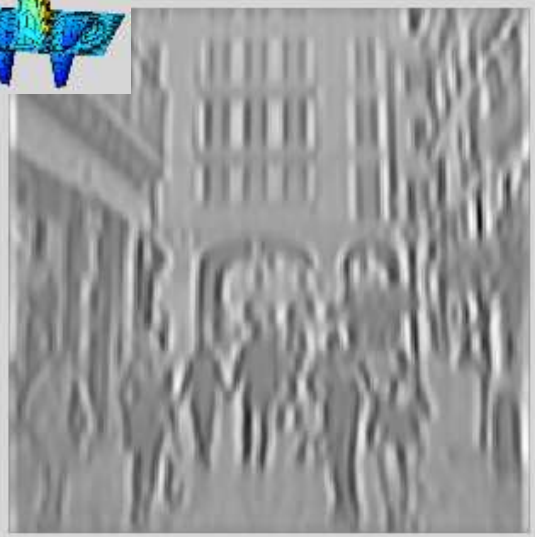


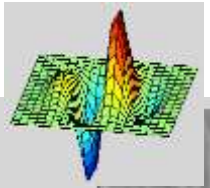
$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$

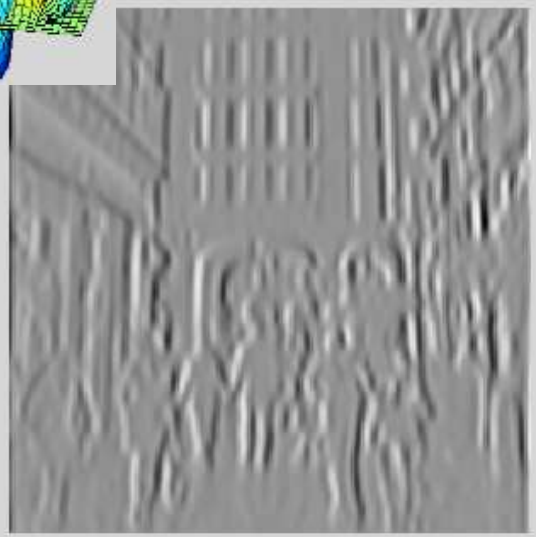


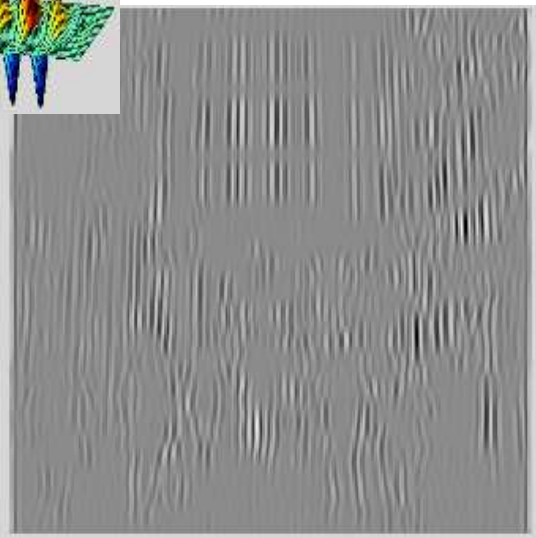


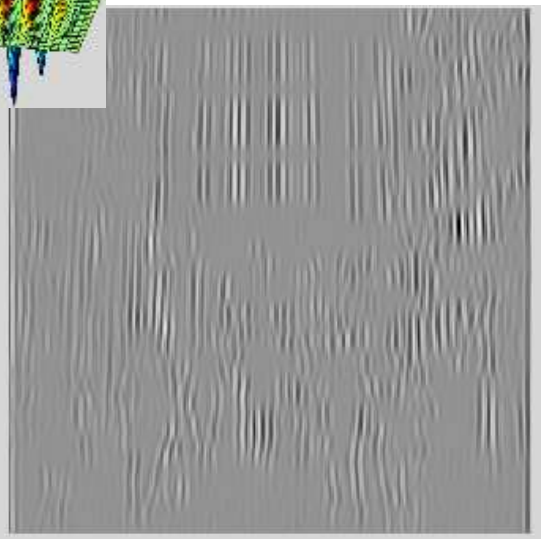












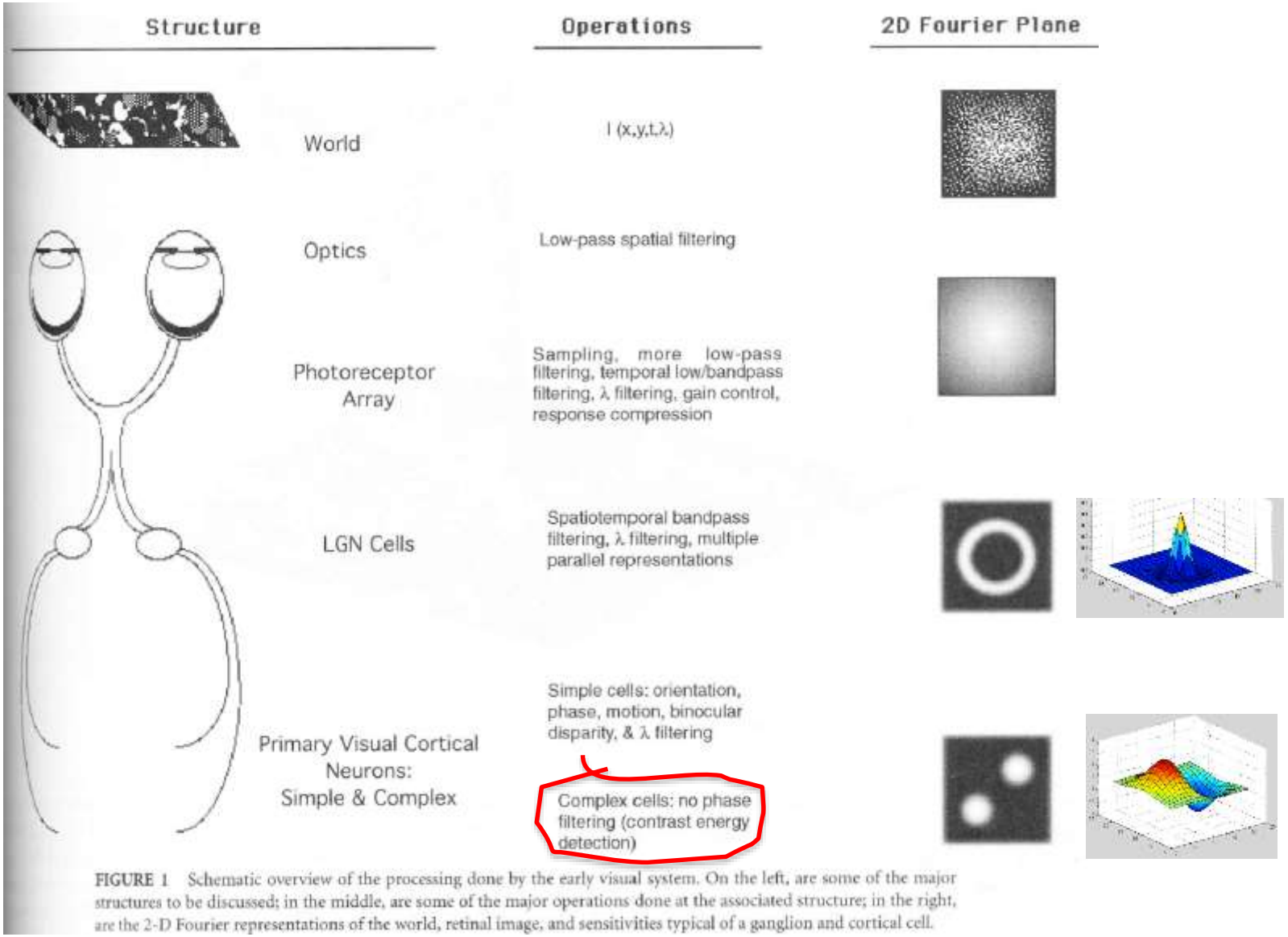
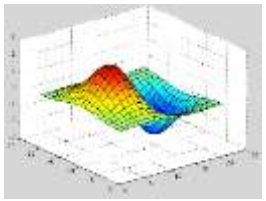


FIGURE 1 Schematic overview of the processing done by the early visual system. On the left, are some of the major structures to be discussed; in the middle, are some of the major operations done at the associated structure; in the right, are the 2-D Fourier representations of the world, retinal image, and sensitivities typical of a ganglion and cortical cell.

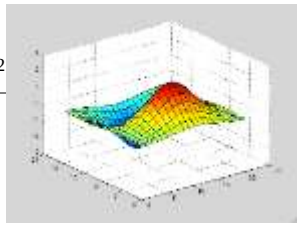
Steerable filters

Derivatives of a Gaussian:

$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$h_y(x,y) = \frac{\partial h(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



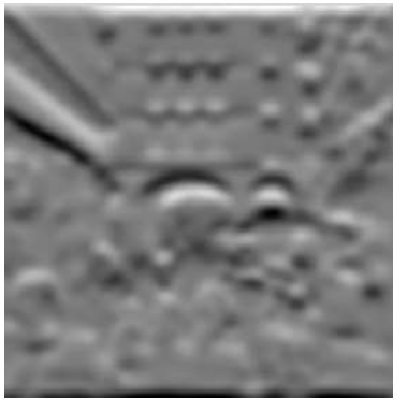
An arbitrary orientation can be computed as a linear combination of those two basis functions:

$$h_\alpha(x,y) = \cos(\alpha) h_x(x,y) + \sin(\alpha) h_y(x,y)$$

The representation is “shiftable” on orientation: We can interpolate any other orientation from a finite set of basis functions.

cos(〈)

+sin(〈)

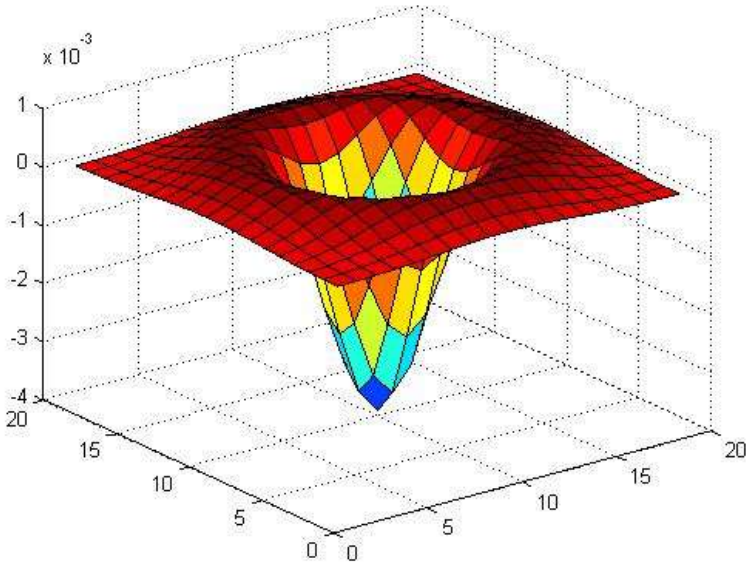


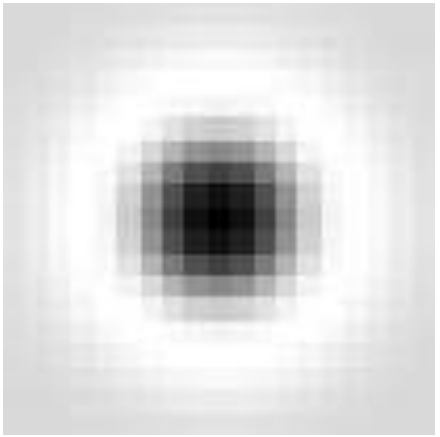
=

Freeman & Adelson 92

Laplacian of Gaussian

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

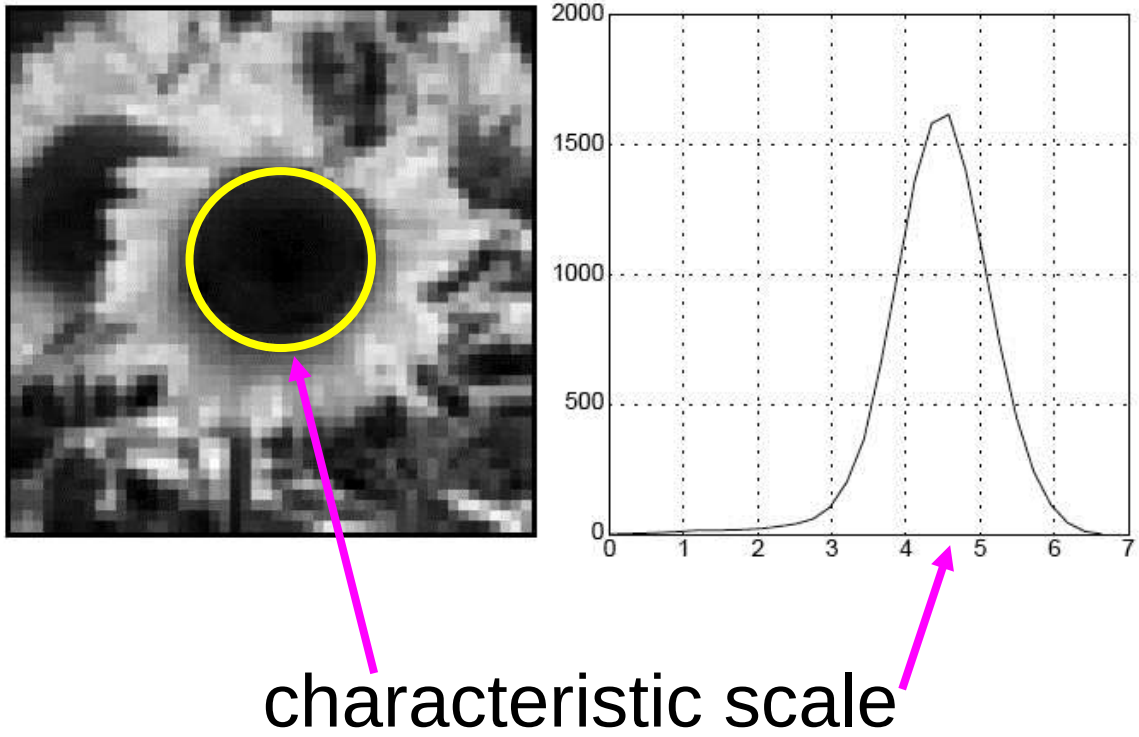




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Blob Detection in 2D

- We define the *characteristic scale* as the scale that produces peak of Laplacian response

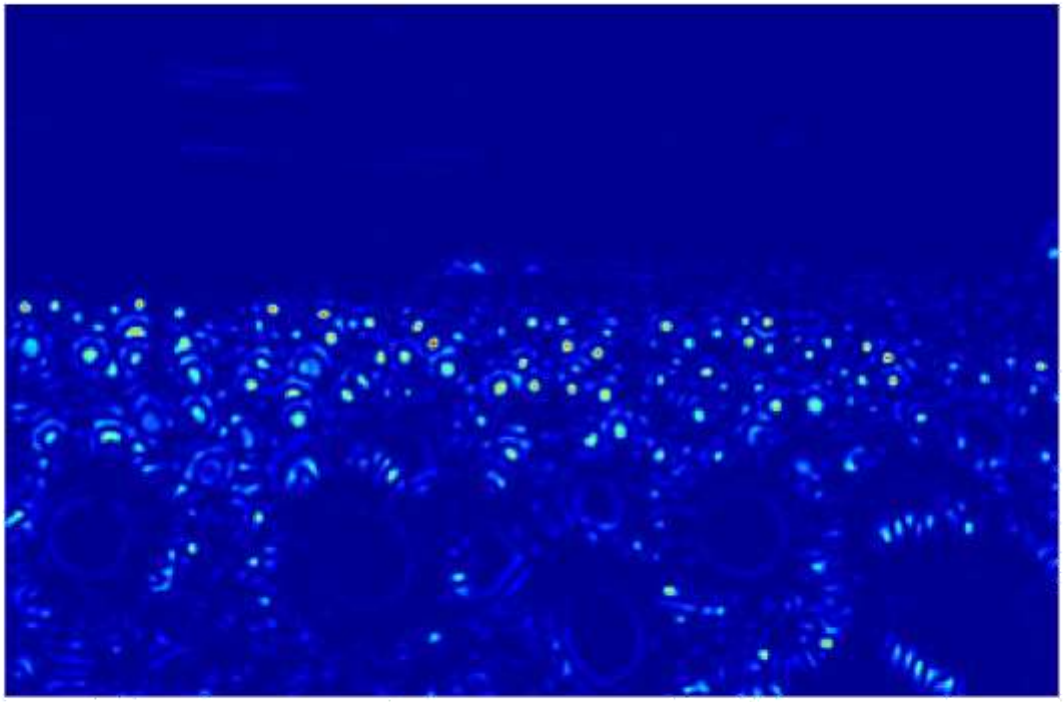


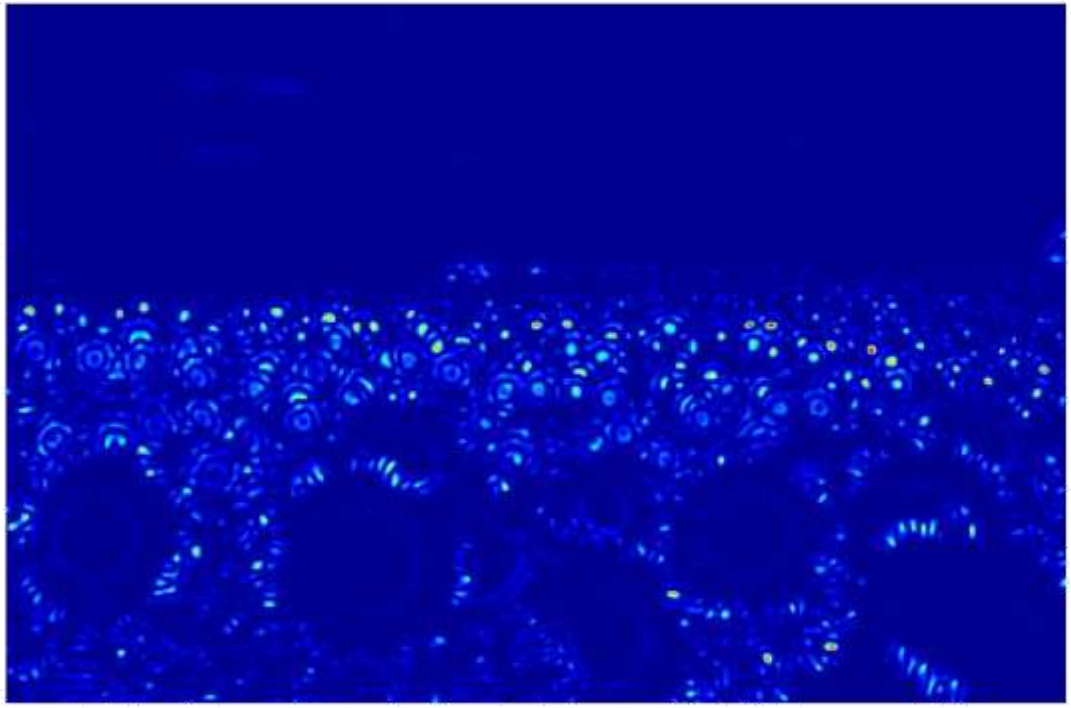
Slide credit: Lana Lazebnik

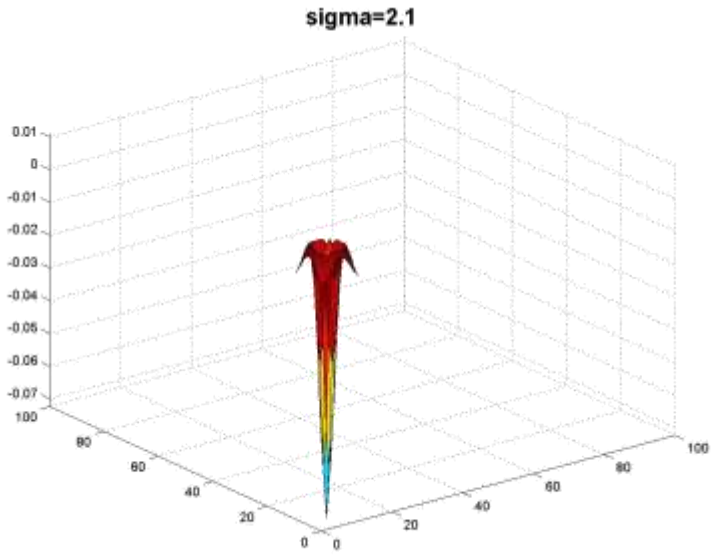
Example

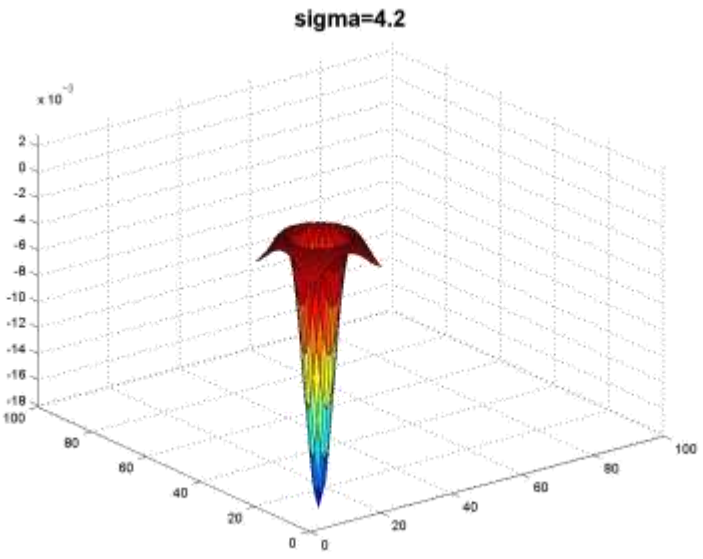
Original image
at ¾ the size

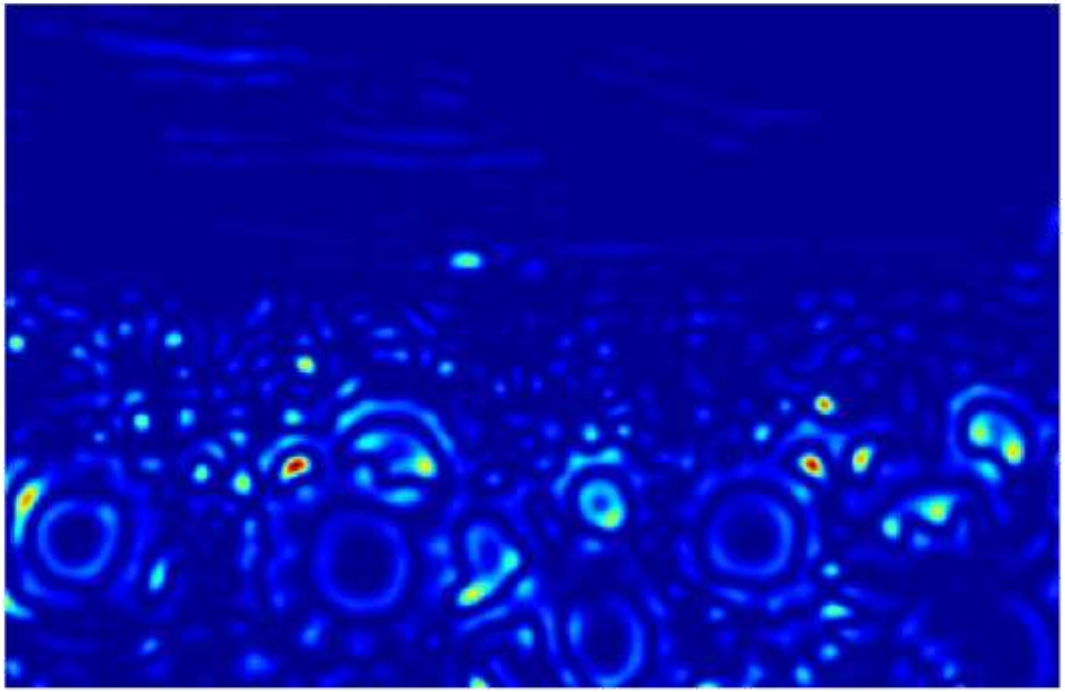
Original image
at ¾ the size

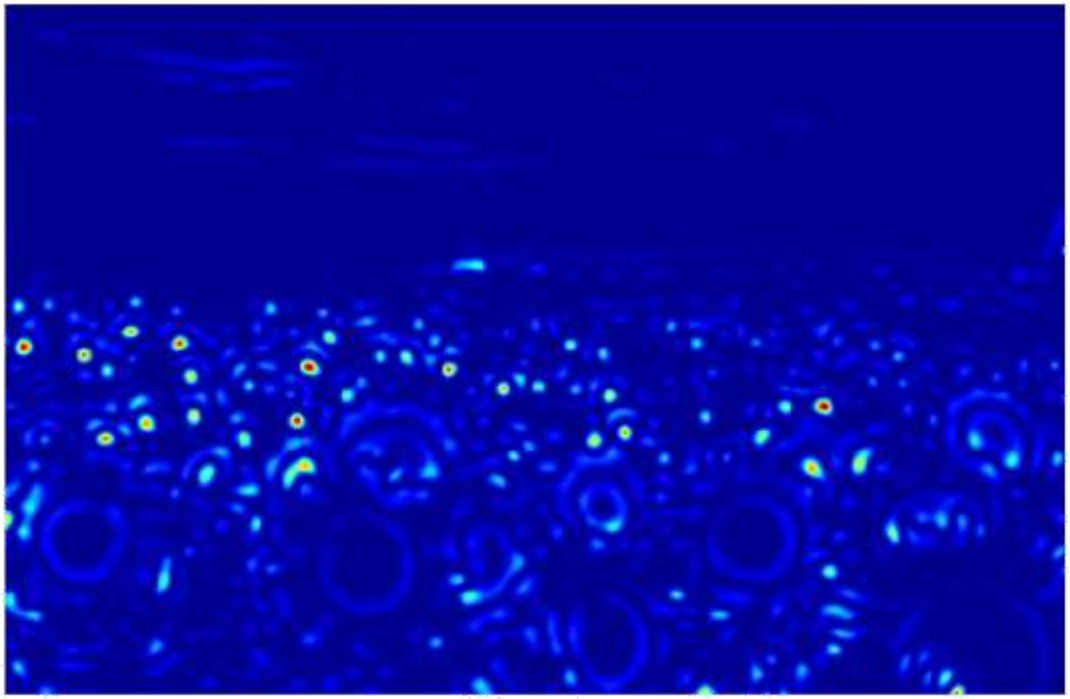


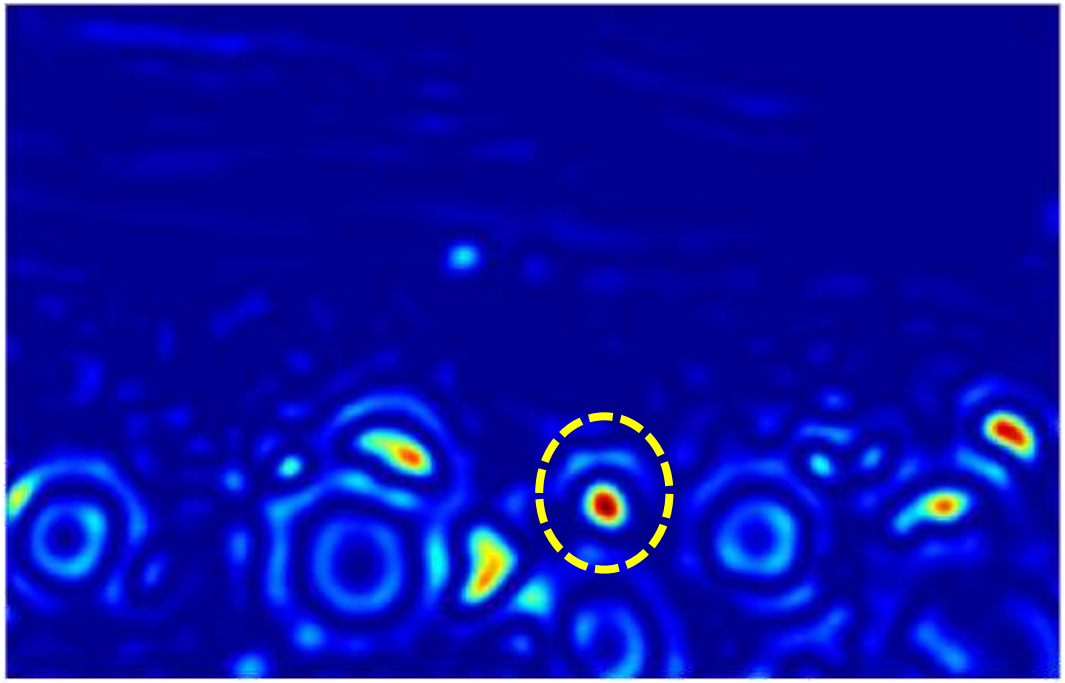


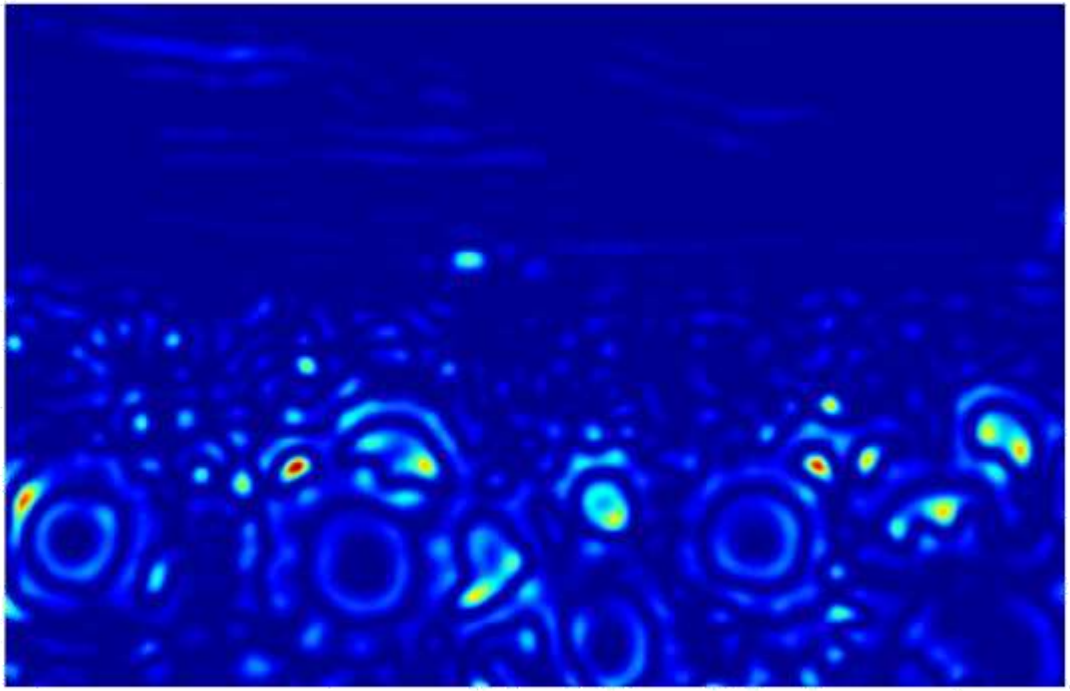


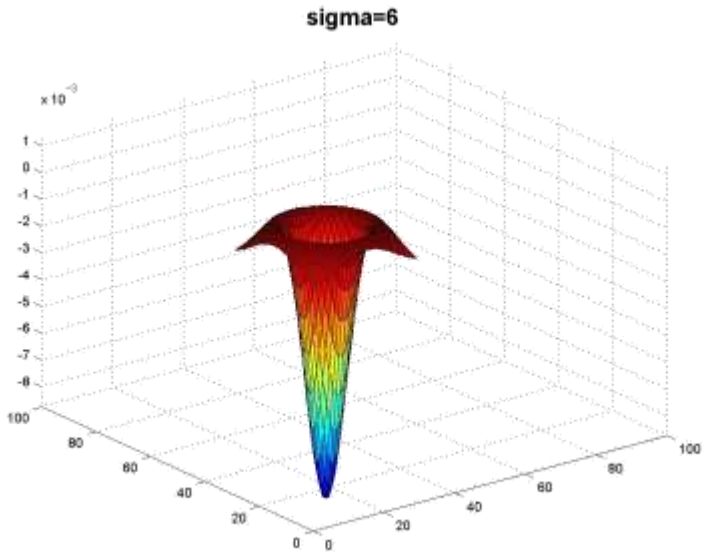

sigma=4.2
x 10^-3
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
100
80
60
40
20
0
0
20
40
60
80
100

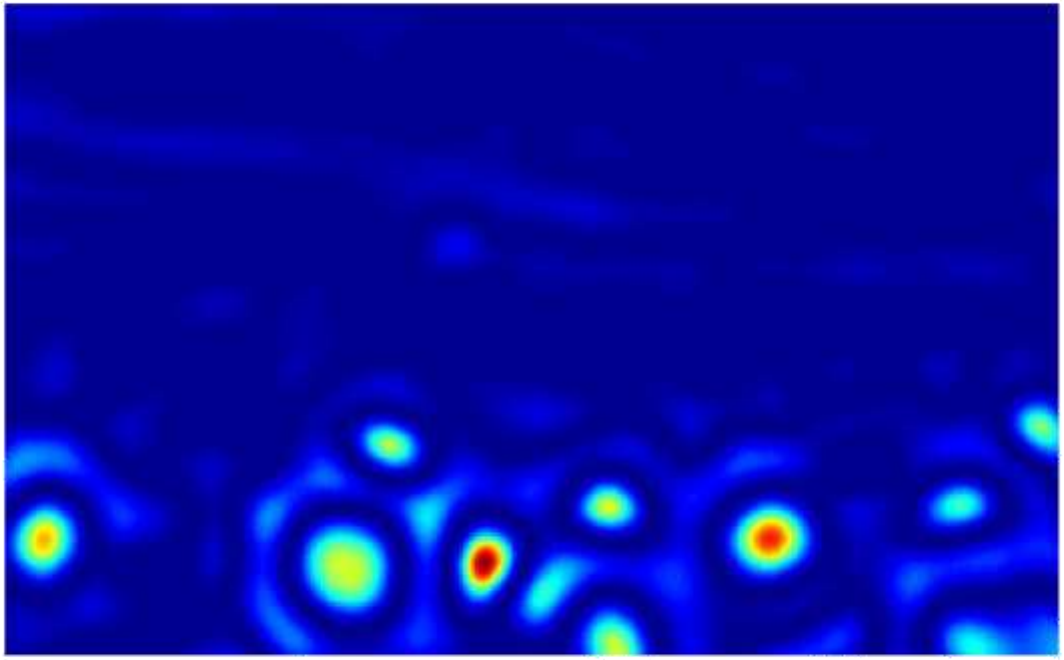


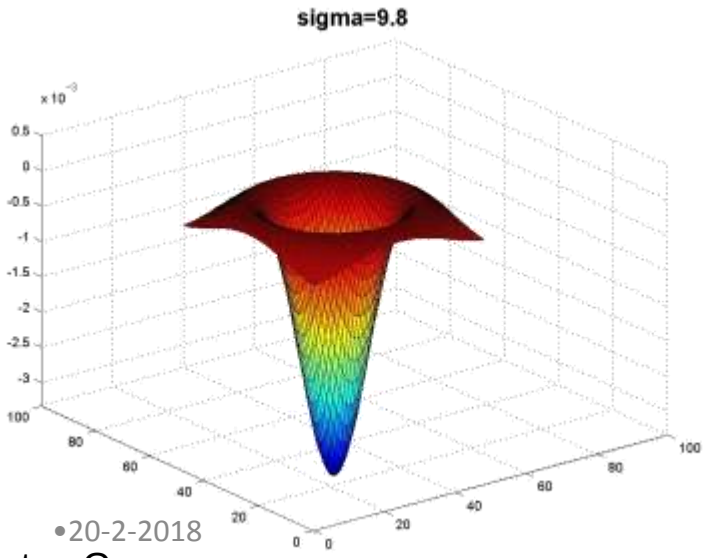


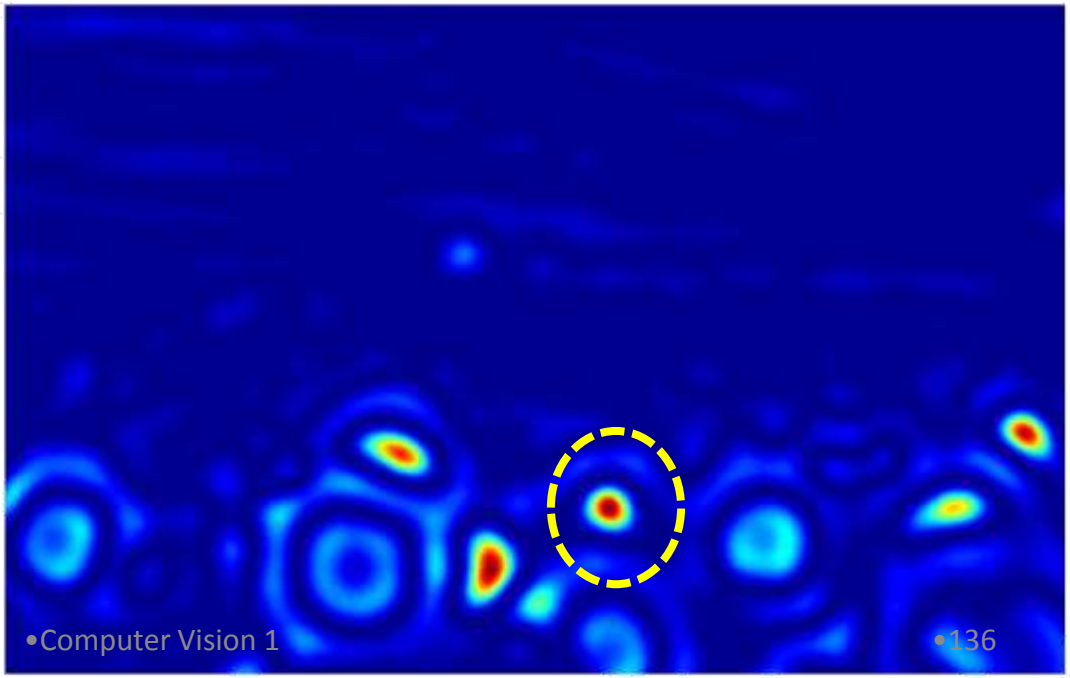


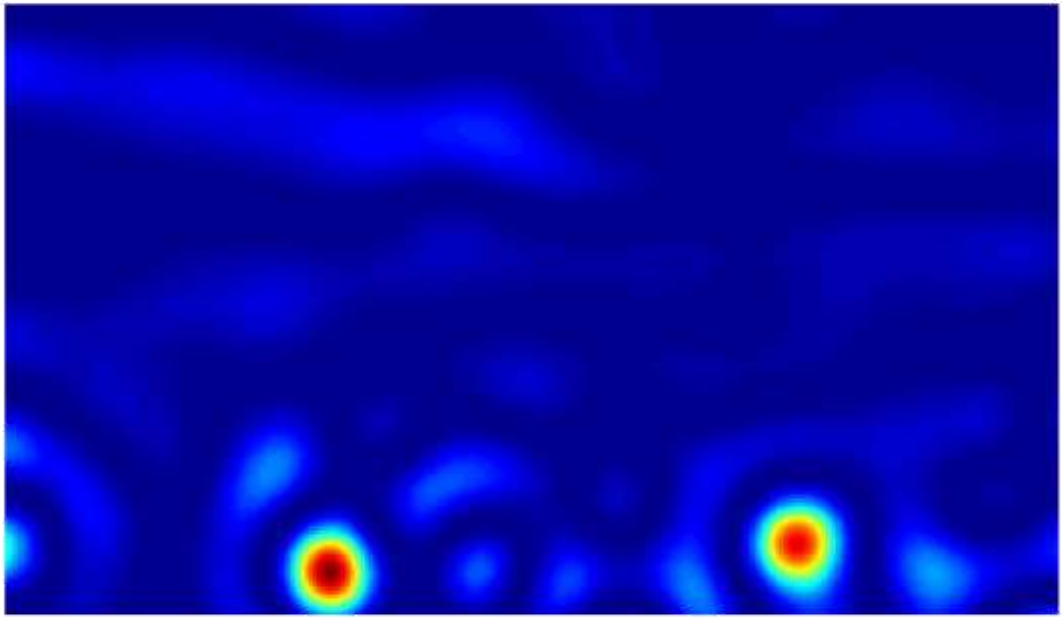


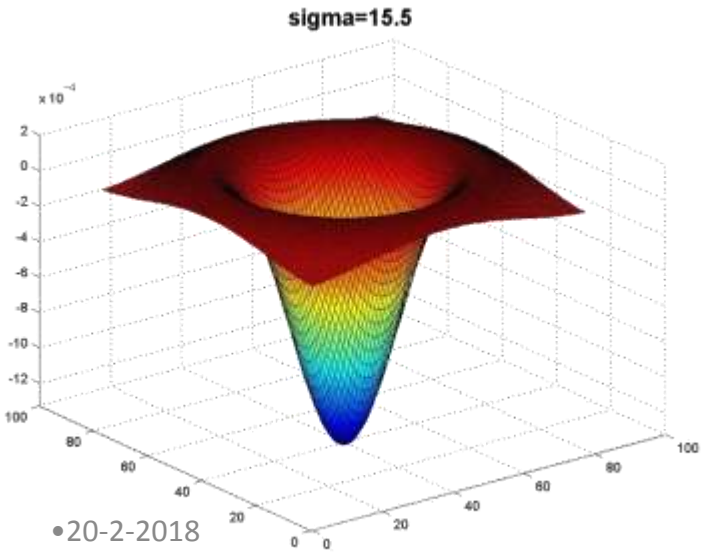

sigma=6
x 10^-3
1
0
-1
-2
-3
-4
-5
-6
-7
-8
100
80
60
40
20
0
0
20
40
60
80
100

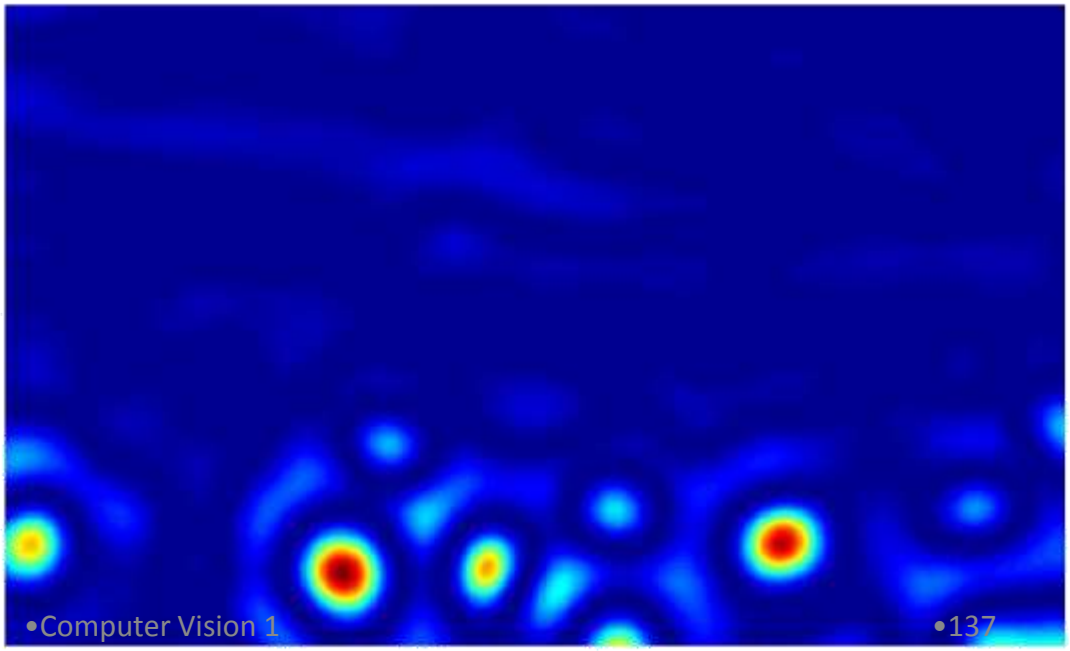



sigma=9.8
x 10^-3
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
100
80
60
40
20
0
0
20
40
60
80
100







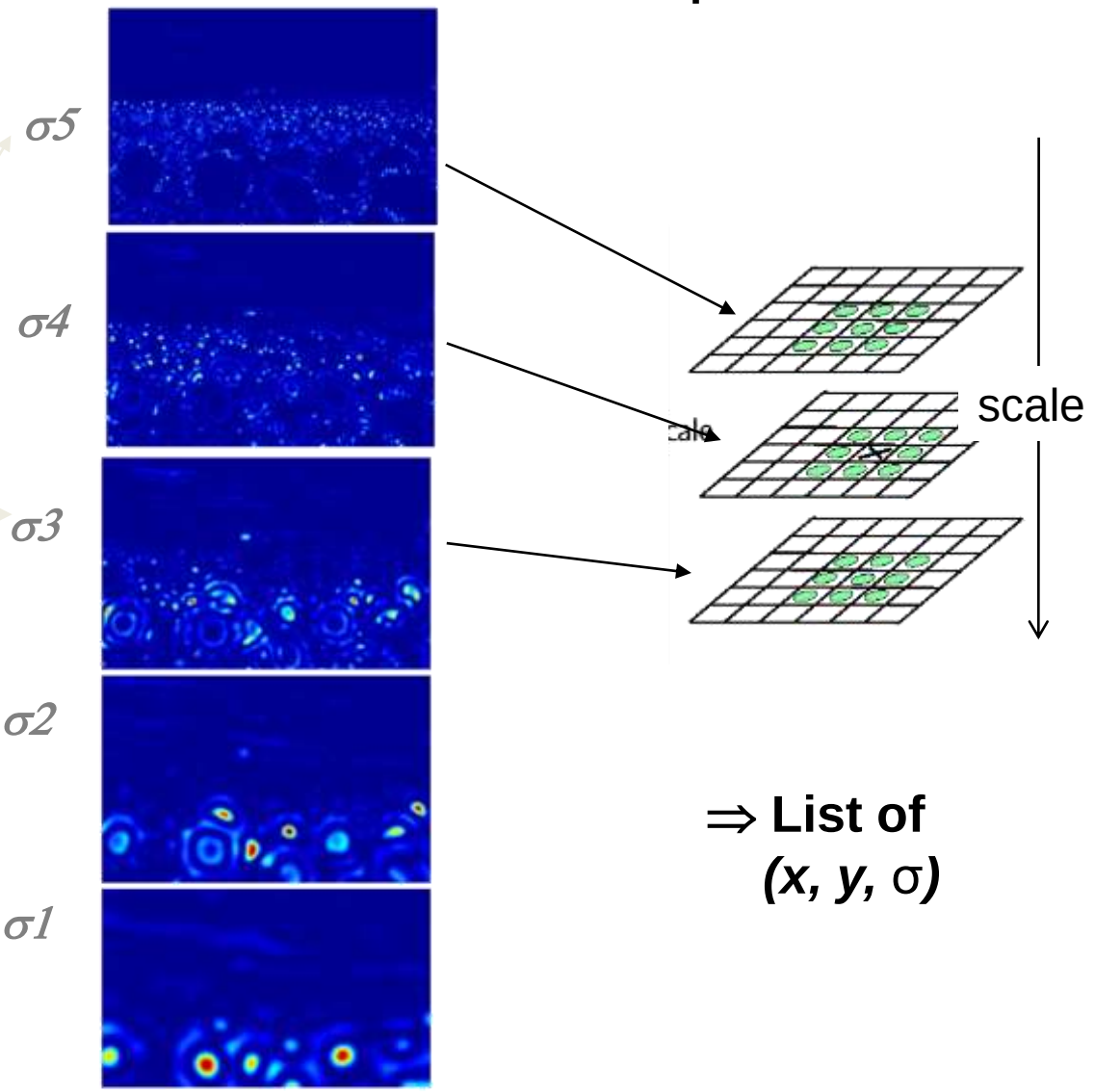


Kristen Grauman

Scale Invariant Interest Points

Interest points are local maxima in both position and scale.

$$L_{xx}(\sigma) + L_{yy}(\sigma)$$



Squared filter response maps

⇒ **List of** ***(x, y,*** **σ*)***

Scale-space Blob Detector: Example

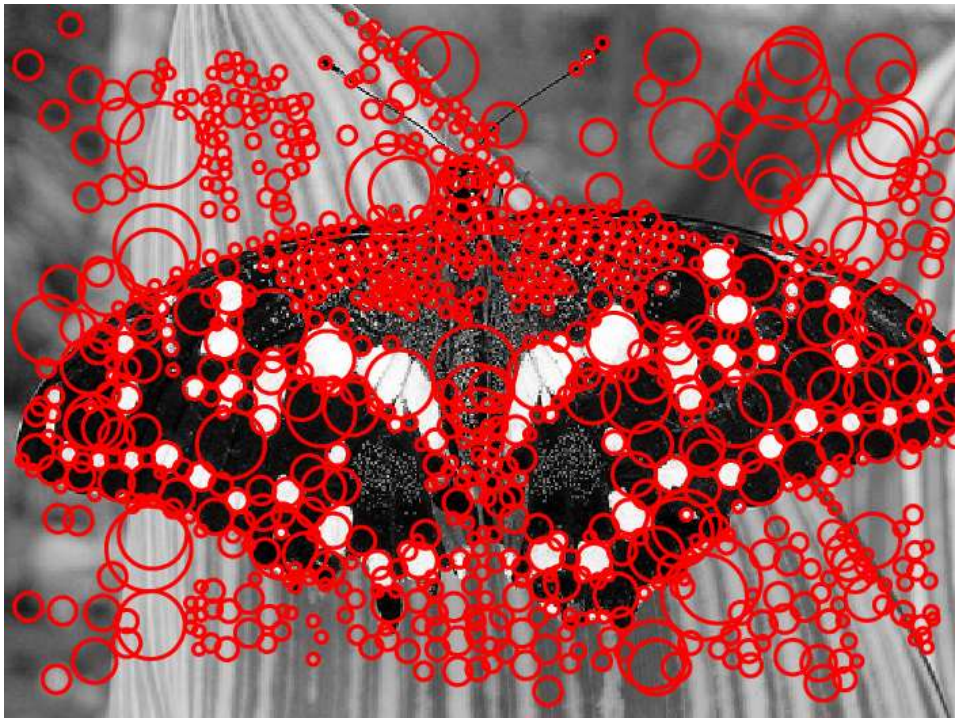
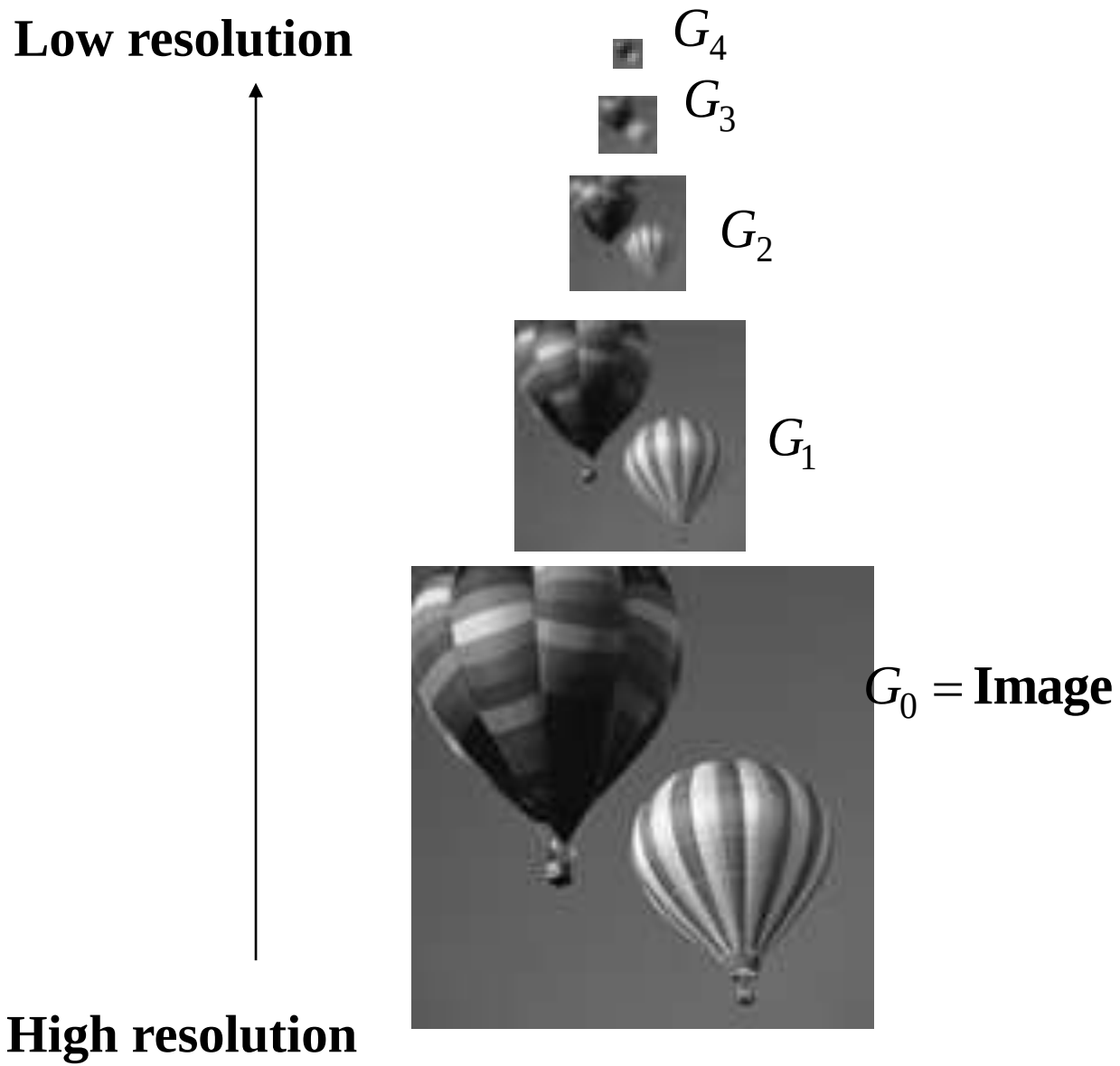


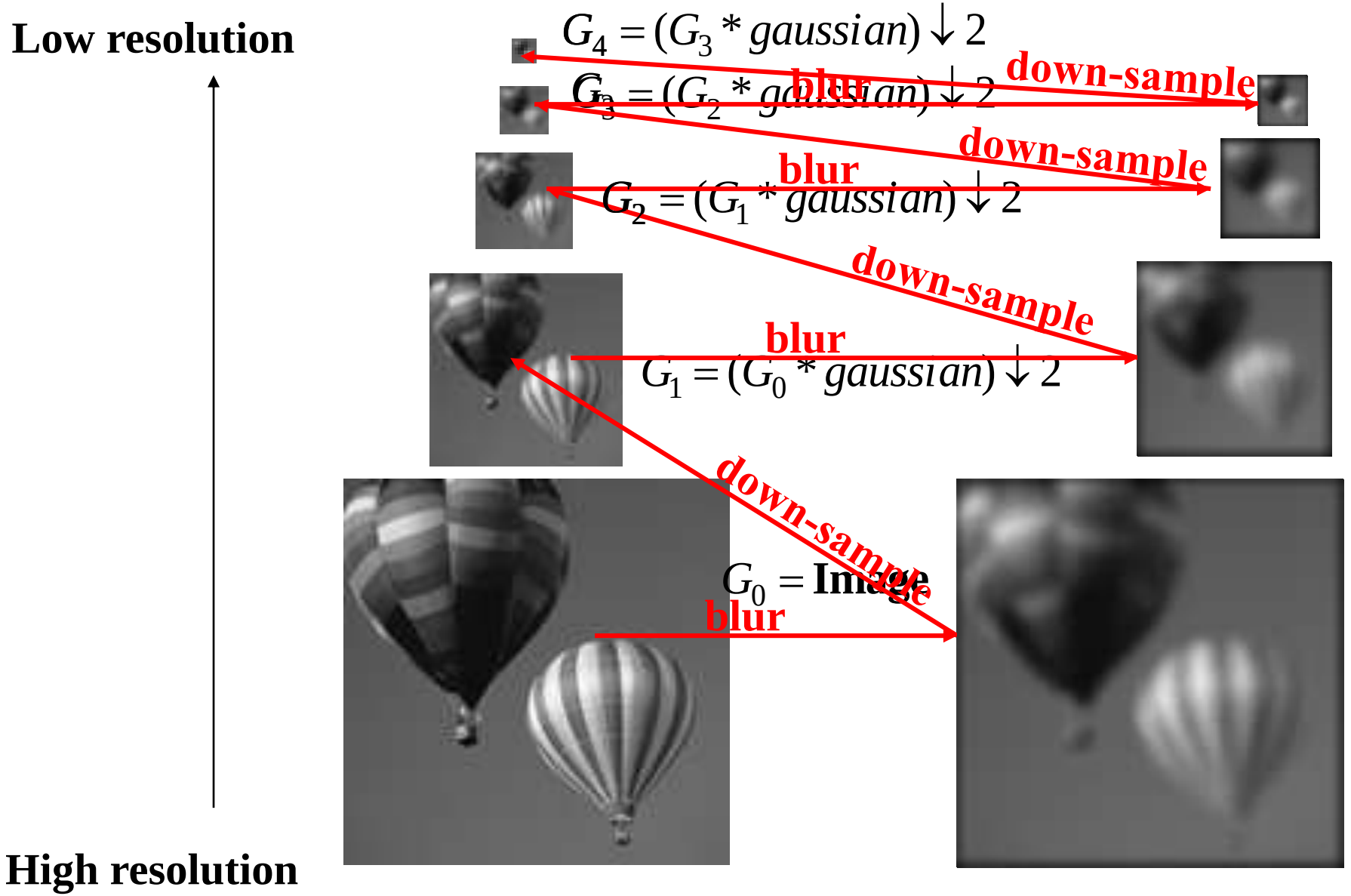
Image credit: Lana Lazebnik

The Gaussian Pyramid



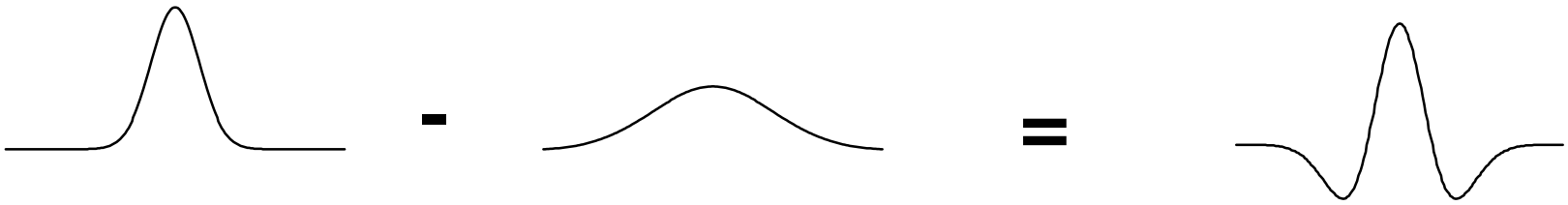
[www.wisdom.weizmann.ac.il/~mblank/CVfall04/handouts/lec4.ppt]

The Gaussian Pyramid



[www.wisdom.weizmann.ac.il/~mblank/CVfall04/handouts/lec4.ppt]

Laplacian ~ Difference of Gaussians



DOG = Difference Of Gaussians

More details on Gaussian and Laplacian pyramids can be found in the paper by Burt and Adelson

[www.wisdom.weizmann.ac.il/~mblank/CVfall04/handouts/lec4.ppt]

Template Matching with Image Pyramids

Input: Image, Template

1. Match template at current scale

2. Downsample image

3. Repeat 1-2 until image is very small

4. Take responses above some threshold, perhaps with non-maxima suppression

Slide: Hoiem

Summary

- Edge Detection
  - Non-maximum suppression on gradient.
  - Zero-crossings for Laplacian.
  - Scale invariance
    - Gaussian pyramids. Coarse-to-fine search, multi-scale detection.
    - Laplacian pyramid. More compact image representation

Computer Vision 1
(total #slides 145 | Lecture 3)

Summary

1. Color invariants
2. Image processing
3. Filtering and convolutions