

Computer Vision

Light: Radiometry and Reflectance

Dr. Mrinmoy Ghorai

**Indian Institute of Information Technology
Sri City, Chittoor**



Today's Agenda

- Light
 - Radiometry
 - Reflectance

Why should we care?

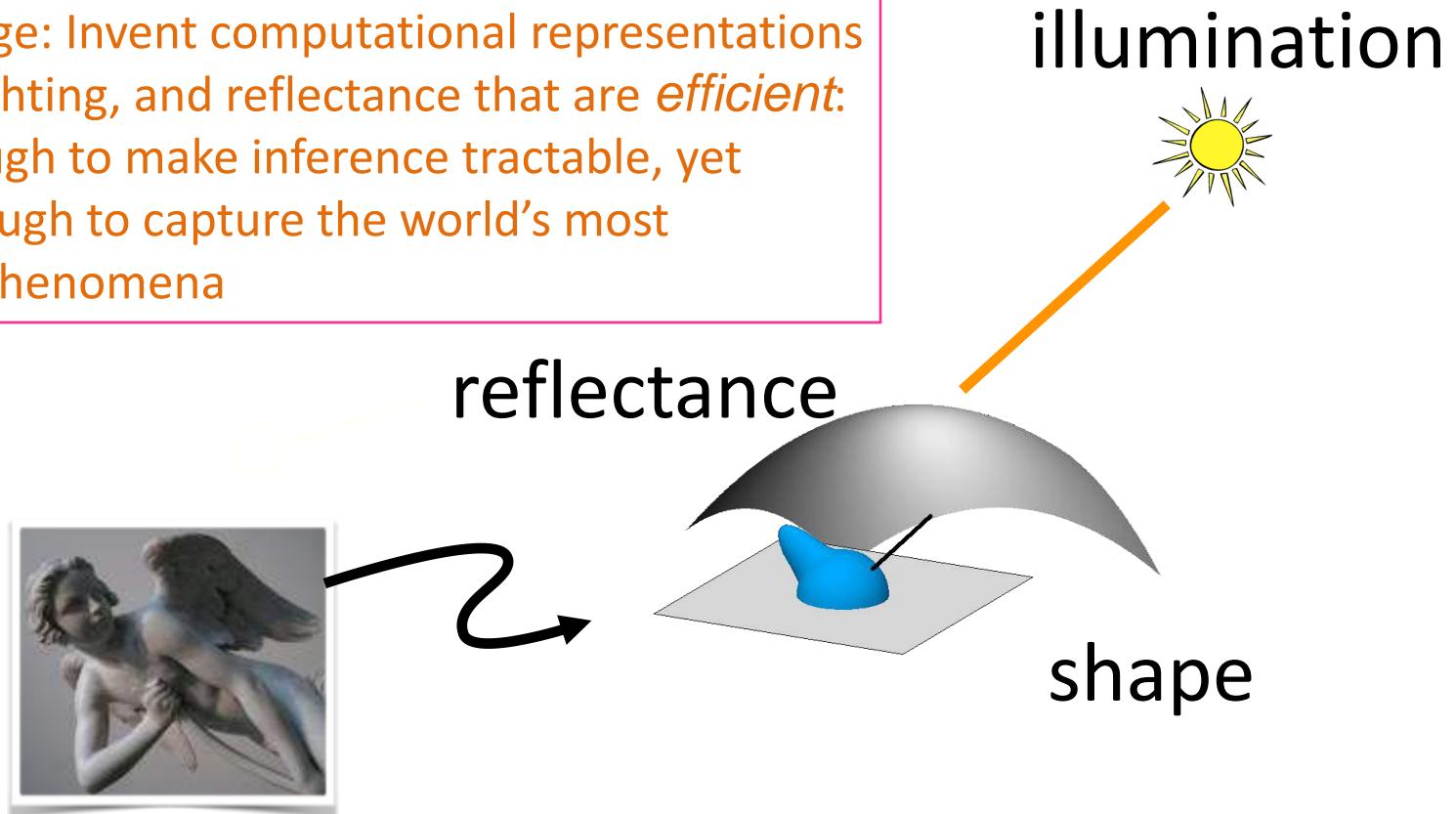
- The **appearance** of objects is given by the way in which they **reflect** and transmit **light**.
- The **color** of objects is determined by the parts of the **spectrum of** (incident white) **light** that are reflected or transmitted without being absorbed.

Appearance



“Physics-based” computer vision (a.k.a “inverse optics”)

Our challenge: Invent computational representations of shape, lighting, and reflectance that are *efficient*: simple enough to make inference tractable, yet general enough to capture the world’s most important phenomena



I → shape, illumination, reflectance

Application: Photometric Stereo



Analysis under different lighting conditions to estimate a normal direction at each pixel.

- Why study the physics (optics) of the world?
- Lets see some pictures!

Light and Shadows



Light and Shadows



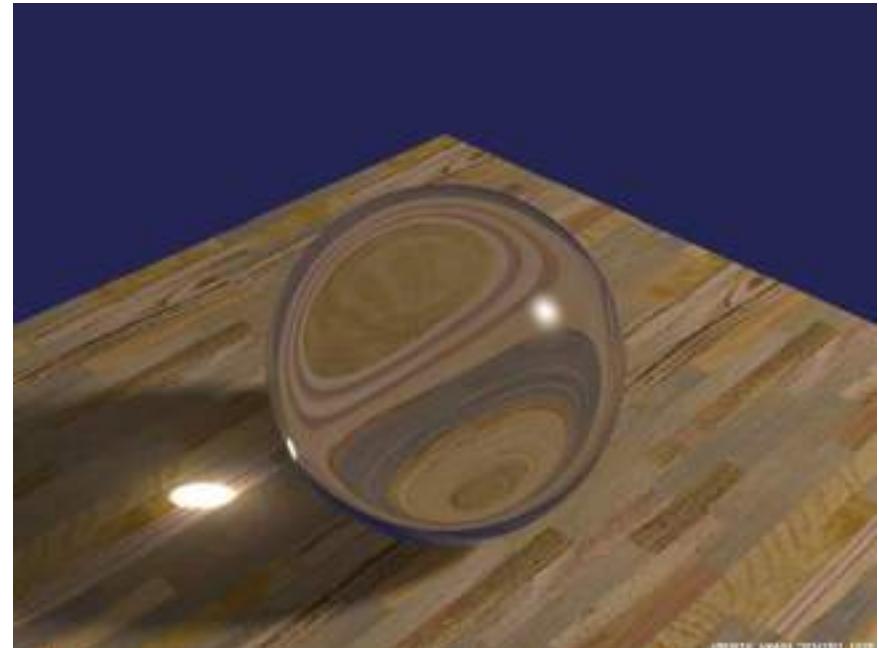
Reflections



Reflections



Refractions



Refractions



Inter-reflections

Mies Courtyard House with Curved Elements



Scattering



Scattering



More Complex Appearances

More Complex Appearances

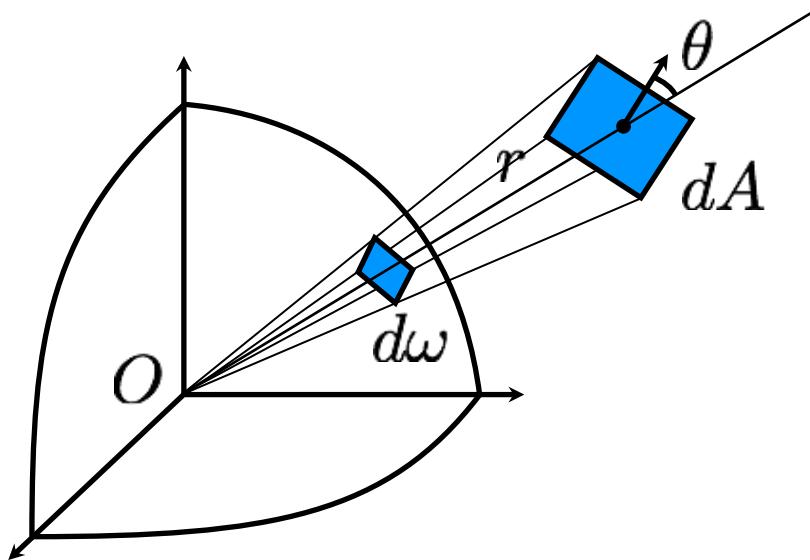


More Complex Appearances



Measuring Light and Radiometry

- **Solid angle:** The *solid angle* subtended by a small surface patch with respect to point O is the area of its central projection onto the unit sphere about O



Depends on:

- orientation of patch
- distance of patch

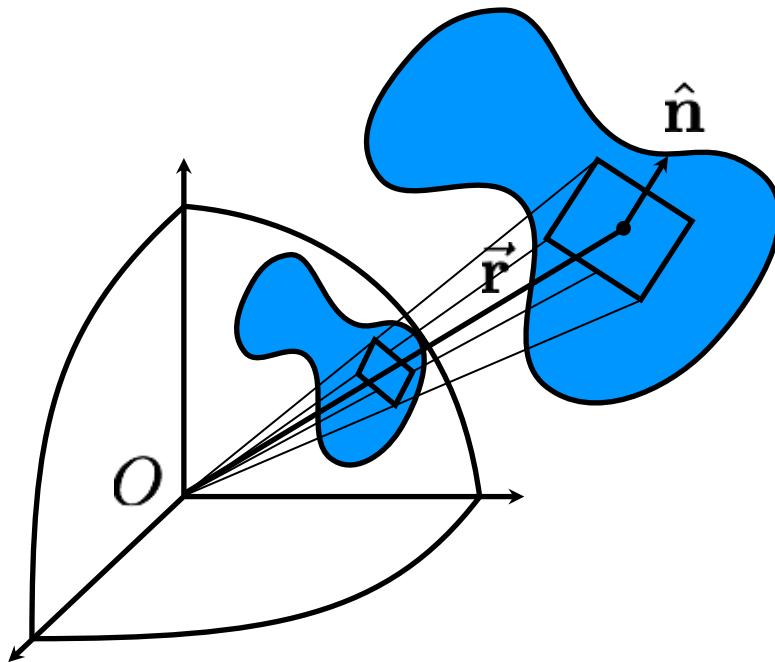
One can show:

$$d\omega = \frac{dA \cos \theta}{r^2}$$

Units: steradians [sr]

Measuring Light and Radiometry

- To calculate solid angle subtended by a surface S relative to O you must add up (integrate) contributions from all tiny patches (nasty integral)



$$\Omega = \iint_S \frac{\vec{r} \cdot \hat{n} \, dS}{|\vec{r}|^3}$$

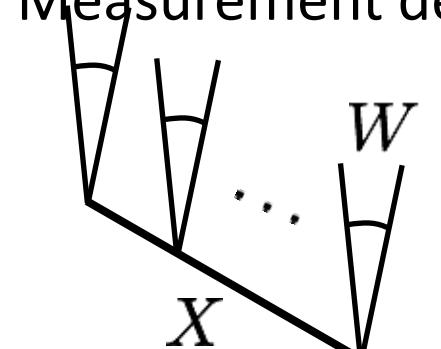
One can show:

$$d\omega = \frac{dA \cos \theta}{r^2}$$

Units: steradians [sr]

Quantifying light: flux, irradiance, and radiance

- Imagine a sensor that counts photons passing through planar patch X in directions within angular wedge W
- It measures *radiant flux* [watts = joules/sec]: rate of photons hitting sensor area
- Measurement depends on sensor area $|X|$

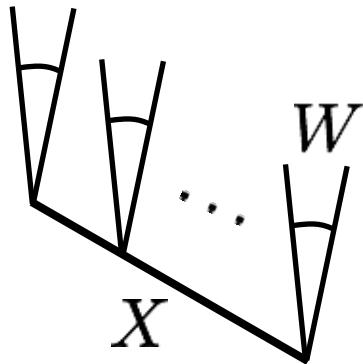


* shown in 2D for clarity; imagine three dimensions

radiant flux $\Phi(W, X)$

Quantifying light: flux, irradiance, and radiance

- *Irradiance:*
A measure of incoming light that is independent of sensor area $|X|$
- Units: watts per square meter [W/m^2]



$$\frac{\Phi(W, X)}{|X|}$$

Quantifying light: flux, irradiance, and radiance

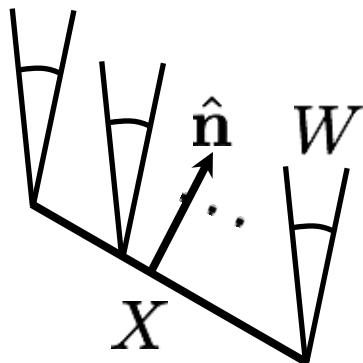
- *Irradiance:*
 - A measure of incoming light that is independent of sensor area $|X|$
- Units: watts per square meter [W/m^2]

The diagram shows a set of parallel vertical lines representing light rays. A shaded triangular region at the bottom left is labeled X . Above it, a single vertical line is labeled W . Ellipses between the lines indicate they continue upwards. To the right, the mathematical expression $\lim_{X \rightarrow x}$ is shown, suggesting that the irradiance is the limit of the flux over the area X as the size of the area approaches zero.

$$\frac{\Phi(W, X)}{|X|}$$

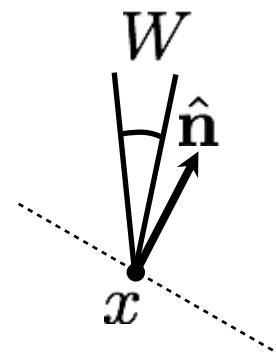
Quantifying light: flux, irradiance, and radiance

- *Irradiance:*
A measure of incoming light that is independent of sensor area $|X|$
- Units: watts per square meter [W/m^2]
- Depends on sensor direction normal.



$$\frac{\Phi(W, X)}{|X|}$$

$$\lim_{X \rightarrow x}$$

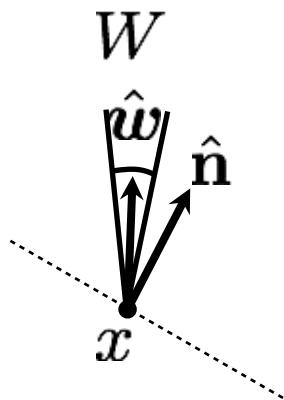


$$E_{\hat{n}}(W, x)$$

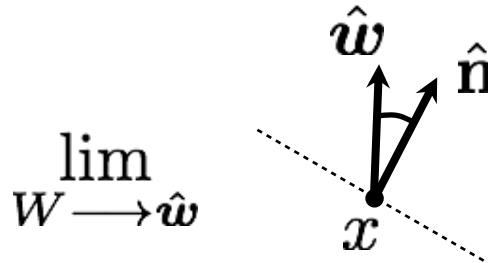
- We keep track of the normal because a planar sensor with distinct orientation would converge to a different limit
- In the literature, notations n and W are often omitted, and values are implied by context

Quantifying light: flux, irradiance, and radiance

- *Radiance:*
 - A measure of incoming light that is independent of sensor area $|X|$, orientation \hat{n} , and wedge size (solid angle) $|W|$
- Units: watts per steradian per square meter [$W/(m^2 \cdot sr)$]



$$\frac{E_{\hat{n}}(W, x)}{|W|}$$

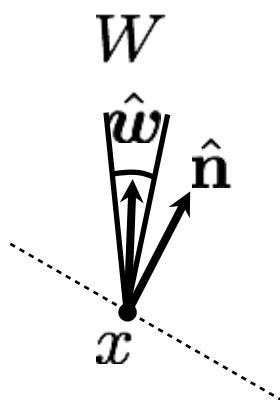


$$L_{\hat{n}}(\hat{\omega}, x)$$

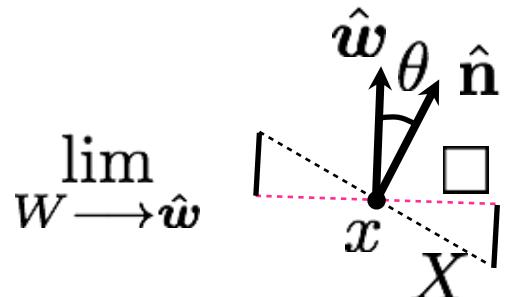
- Has correct units, but still depends on sensor orientation
- To correct this, convert to measurement that would have been made if sensor was perpendicular to direction ω

Quantifying light: flux, irradiance, and radiance

- *Radiance:*
 - A measure of incoming light that is independent of sensor area $|X|$, orientation \hat{n} , and wedge size (solid angle) $|W|$
- Units: watts per steradian per square meter [$W/(m^2 \cdot sr)$]



$$\frac{E_{\hat{n}}(W, x)}{|W|}$$



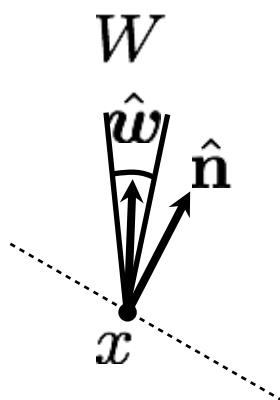
$$L_{\hat{n}}(\hat{\omega}, x)$$

$$\begin{aligned}\cos \theta &= \frac{\square/2}{|X|/2} \\ \rightarrow \square &= |X| \cos \theta \\ &\text{"foreshortened area"}\end{aligned}$$

- Has correct units, but still depends on sensor orientation
- To correct this, convert to measurement that would have been made if sensor was perpendicular to direction ω

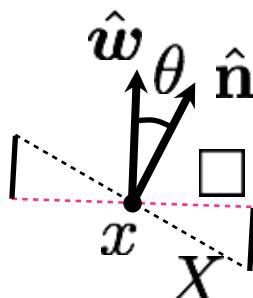
Quantifying light: flux, irradiance, and radiance

- *Radiance:*
 - A measure of incoming light that is independent of sensor area $|X|$, orientation \hat{n} , and wedge size (solid angle) $|W|$
- Units: watts per steradian per square meter [$W/(m^2 \cdot sr)$]



$$\frac{E_{\hat{n}}(W, x)}{|W|}$$

$$\lim_{W \rightarrow \hat{w}}$$



$$L_{\hat{n}}(\hat{w}, x)$$

$$/ \cos \theta$$

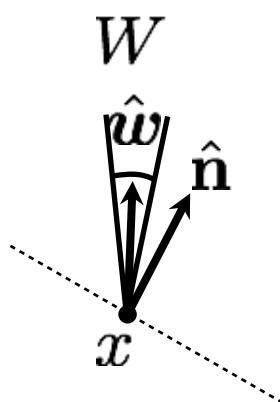


$$L(\hat{w}, x)$$

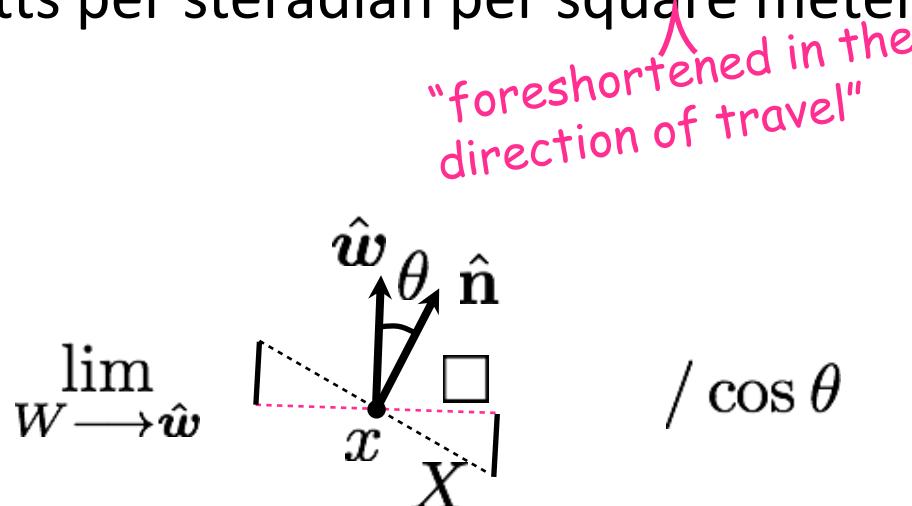
- Has correct units, but still depends on sensor orientation
- To correct this, convert to measurement that would have been made if sensor was

Quantifying light: flux, irradiance, and radiance

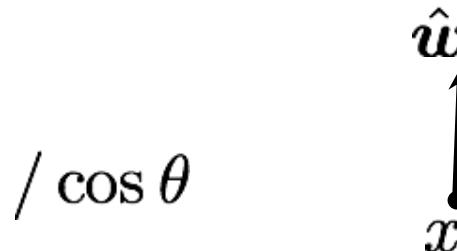
- *Radiance:*
A measure of incoming light that is independent of sensor area $|X|$, orientation \hat{n} , and wedge size (solid angle) $|W|$
- Units: watts per steradian per square meter $[W/(m^2 \cdot sr)]$



$$\frac{E_{\hat{n}}(W, x)}{|W|}$$



$$L_{\hat{n}}(\hat{\omega}, x)$$

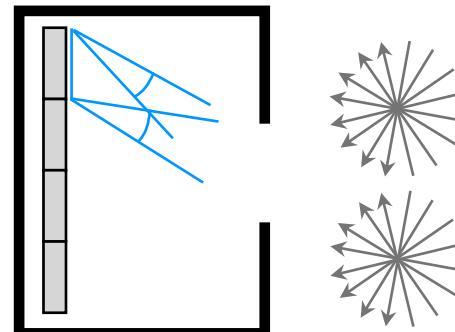


$$L(\hat{\omega}, x)$$

- Has correct units, but still depends on sensor orientation
- To correct this, convert to measurement that would have been made if sensor was perpendicular to direction ω

Quantifying light: flux, irradiance, and radiance

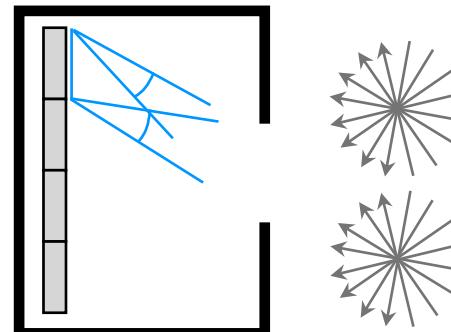
- Attractive properties of radiance:
 - Allows computing the radiant flux measured by *any* finite sensor



Quantifying light: flux, irradiance, and radiance

- Attractive properties of radiance:
 - Allows computing the radiant flux measured by *any* finite sensor

$$\Phi(W, X) = \int_X \int_W L(\hat{\omega}, x) \cos \theta d\omega dA$$



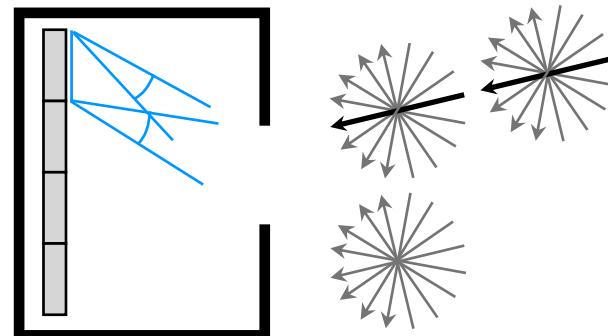
Quantifying light: flux, irradiance, and radiance

- Attractive properties of radiance:
 - Allows computing the radiant flux measured by *any* finite sensor

$$\Phi(W, X) = \int_X \int_W L(\hat{\omega}, x) \cos \theta d\omega dA$$

- Constant along a ray in free space

$$L(\hat{\omega}, x) = L(\hat{\omega}, x + \hat{\omega})$$



Quantifying light: flux, irradiance, and radiance

- Attractive properties of radiance:
 - Allows computing the radiant flux measured by *any* finite sensor

$$\Phi(W, X) = \int_X \int_W L(\hat{\omega}, x) \cos \theta d\omega dA$$

- Constant along a ray in free space

$$L(\hat{\omega}, x) = L(\hat{\omega}, x + \hat{\omega})$$

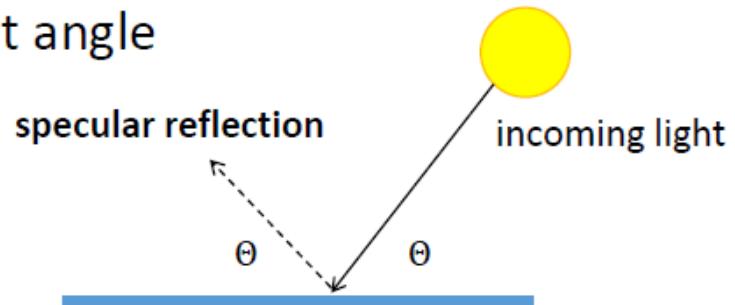
- A camera measures radiance (after a one-time radiometric calibration). So RAW pixel values are proportional to radiance.
 - “Processed” images (like PNG and JPEG) are not linear radiance measurements!!

Reflectance and BRDF

Basic models of reflection

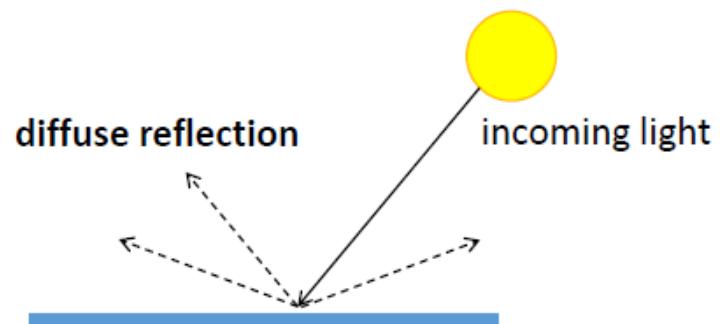
- Specular: light bounces off at the incident angle

- E.g., mirror



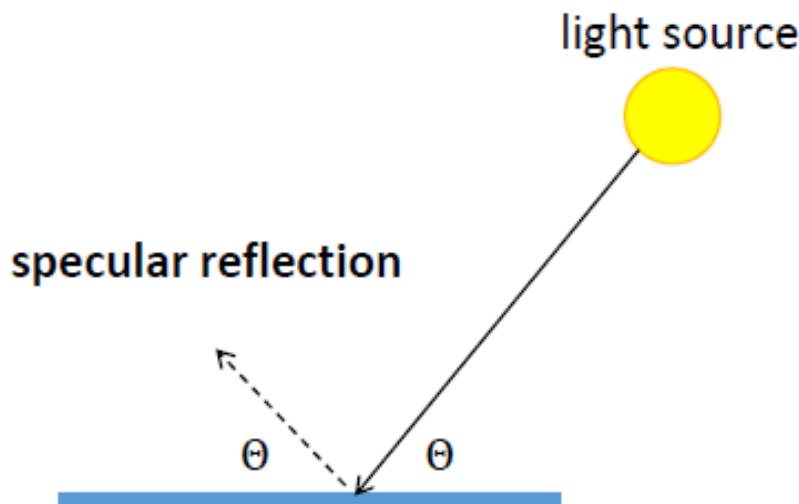
- Diffuse: light scatters in all directions

- E.g., brick, cloth, rough wood



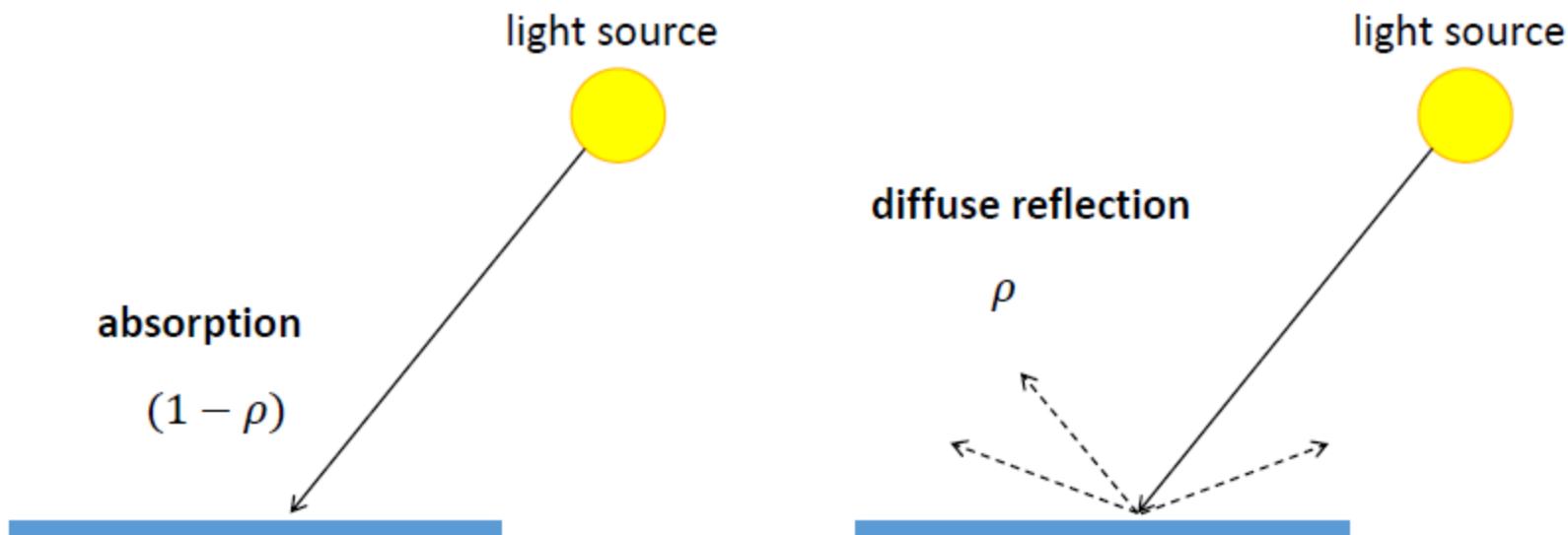
Specular Reflection

- Reflected direction depends on light orientation and surface normal
 - E.g., mirrors are fully specular



Lambertian reflectance model

- Some light is absorbed (function of albedo ρ)
- Remaining light is scattered (diffuse reflection)
- Examples: soft cloth, concrete, matte paints



Most surfaces have both specular and diffuse components

- Specularity = spot where specular reflection dominates (typically reflects light source)



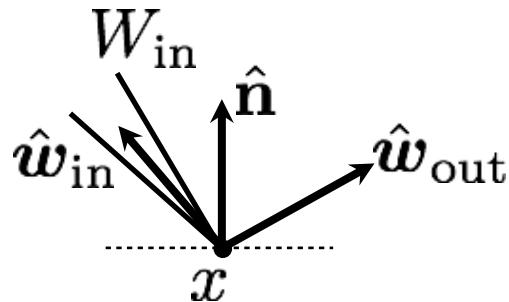
Photo: northcountryhardwoodfloors.com



Typically, specular component is small

Reflectance

- Ratio of outgoing energy to incoming energy at a single point
- Want to define a ratio such that it:
 - converges as we use smaller and smaller incoming and outgoing wedges
 - does not depend on the size of the wedges (i.e. is intrinsic to the material)

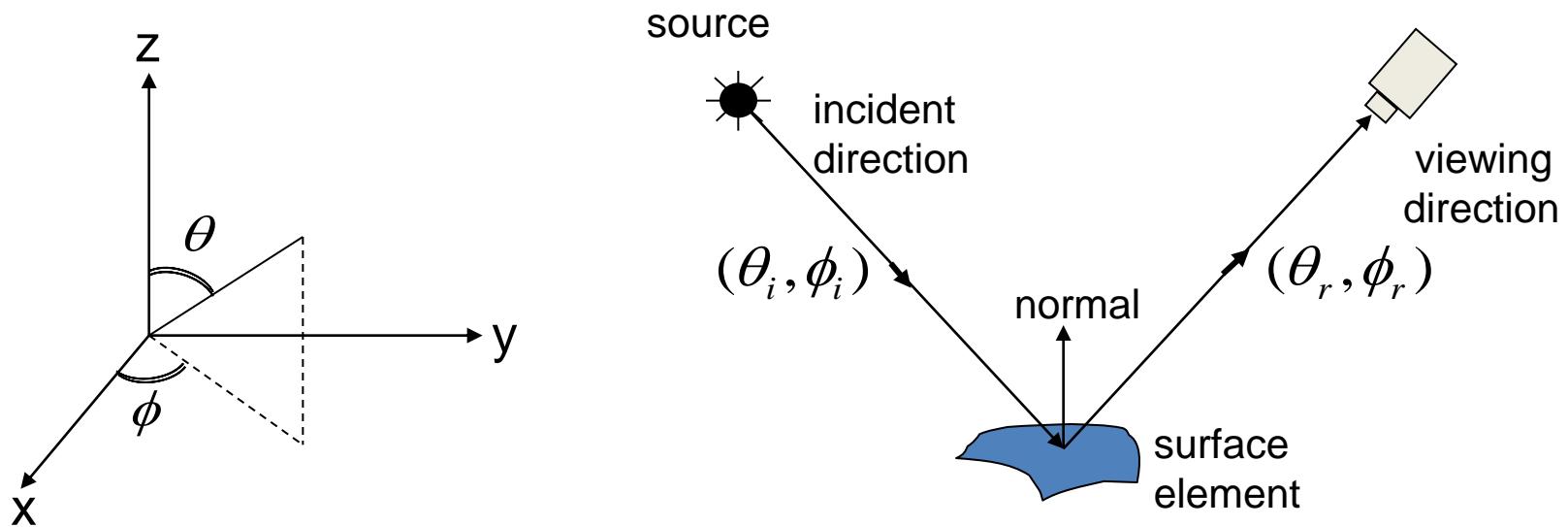


$$\lim_{W_{\text{in}} \rightarrow \hat{\omega}_{\text{in}}} f_{x,\hat{\mathbf{n}}}(\hat{\omega}_{\text{in}}, \hat{\omega}_{\text{out}})$$

$$f_{x,\hat{\mathbf{n}}}(W_{\text{in}}, \hat{\omega}_{\text{out}}) = \frac{L^{\text{out}}(x, \hat{\omega}_{\text{out}})}{E_{\hat{\mathbf{n}}}^{\text{in}}(W_{\text{in}}, x)}$$

- Notations x and n often implied by context and omitted; directions \omega are expressed in local coordinate system defined by normal n (and some chosen tangent vector)
- Units: sr⁻¹
- Called Bidirectional Reflectance Distribution Function (BRDF)

BRDF: Bidirectional Reflectance Distribution Function



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

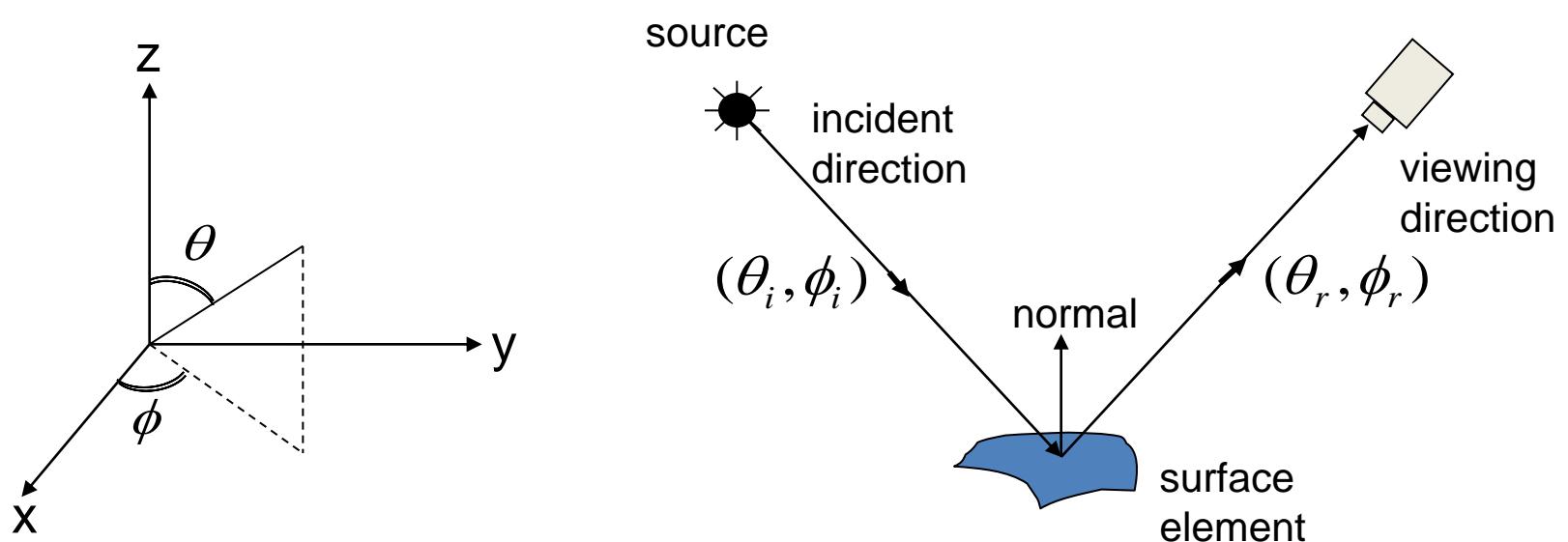
$L^{surface}(\theta_r, \phi_r)$ Radiance of Surface in direction (θ_r, ϕ_r)

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

Reflectance: BRDF

- Units: sr^{-1}
- Real-valued function defined on the double-hemisphere
- Has many useful properties

Important Properties of BRDFs

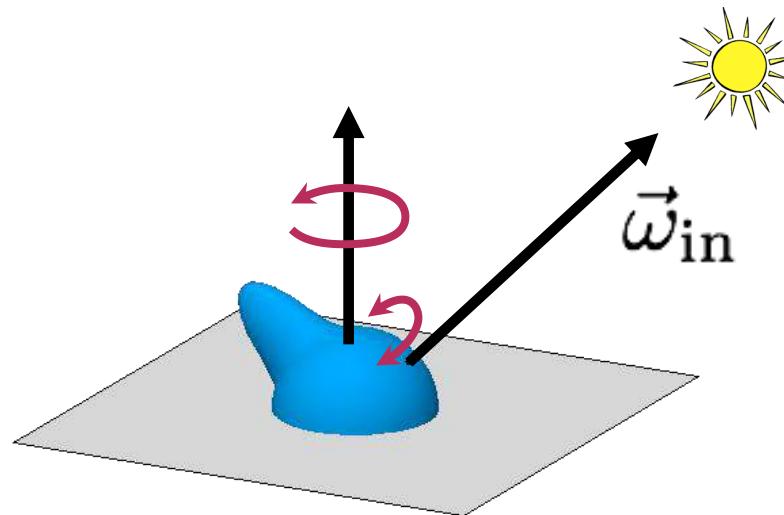
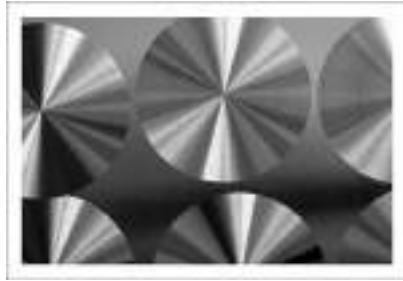


- Conservation of Energy:

$$\forall \hat{\omega}_{\text{in}}, \int_{\Omega_{\text{out}}} f(\hat{\omega}_{\text{in}}, \hat{\omega}_{\text{out}}) \cos \theta_{\text{out}} d\hat{\omega}_{\text{out}} \leq 1$$

Why smaller than or equal?

Common assumption: Isotropy



BRDF does not change
when surface is rotated
about the normal.

$$f_r(\vec{\omega}_{in}, \cdot)$$

4D → 3D

$$f_r(\vec{\omega}_{in}, \vec{\omega}_{out})$$



[Matusik et al., 2003]

Bi-directional Reflectance Distribution Function (BRDF)

Can be written as a function of 3 variables : $f(\theta_i, \theta_r, \phi_i - \phi_r)$

Reflectance: BRDF

- Units: sr^{-1}
- Real-valued function defined on the double-hemisphere
- Has many useful properties
- Allows computing output radiance (and thus pixel value) for *any* configuration of lights and viewpoint

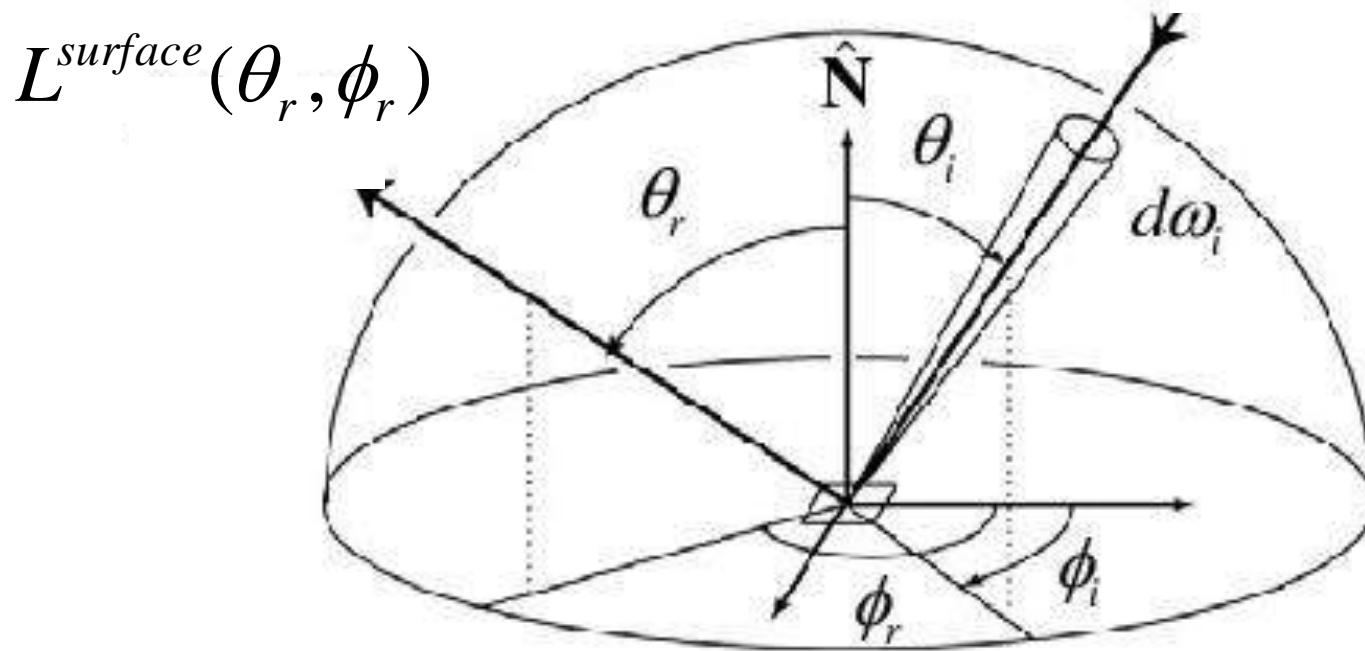
$$L^{\text{out}}(\hat{\omega}) = \int_{\Omega_{\text{in}}} f(\hat{\omega}_{\text{in}}, \hat{\omega}_{\text{out}}) L^{\text{in}}(\hat{\omega}_{\text{in}}) \cos \theta_{\text{in}} d\hat{\omega}_{\text{in}}$$

reflectance equation

Why is there a cosine in the reflectance equation?

Derivation of the Reflectance Equation

$$L^{src}(\theta_i, \phi_i)$$



From the definition of BRDF:

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

Derivation of the Scene Radiance Equation

From the definition of BRDF:

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

Write Surface Irradiance in terms of Source Radiance:

$$L^{surface}(\theta_r, \phi_r) = \frac{L^{src}(\theta_i, \phi_i)f(\theta_i, \phi_i; \theta_r, \phi_r)\cos\theta_i}{\text{solid angle}}$$

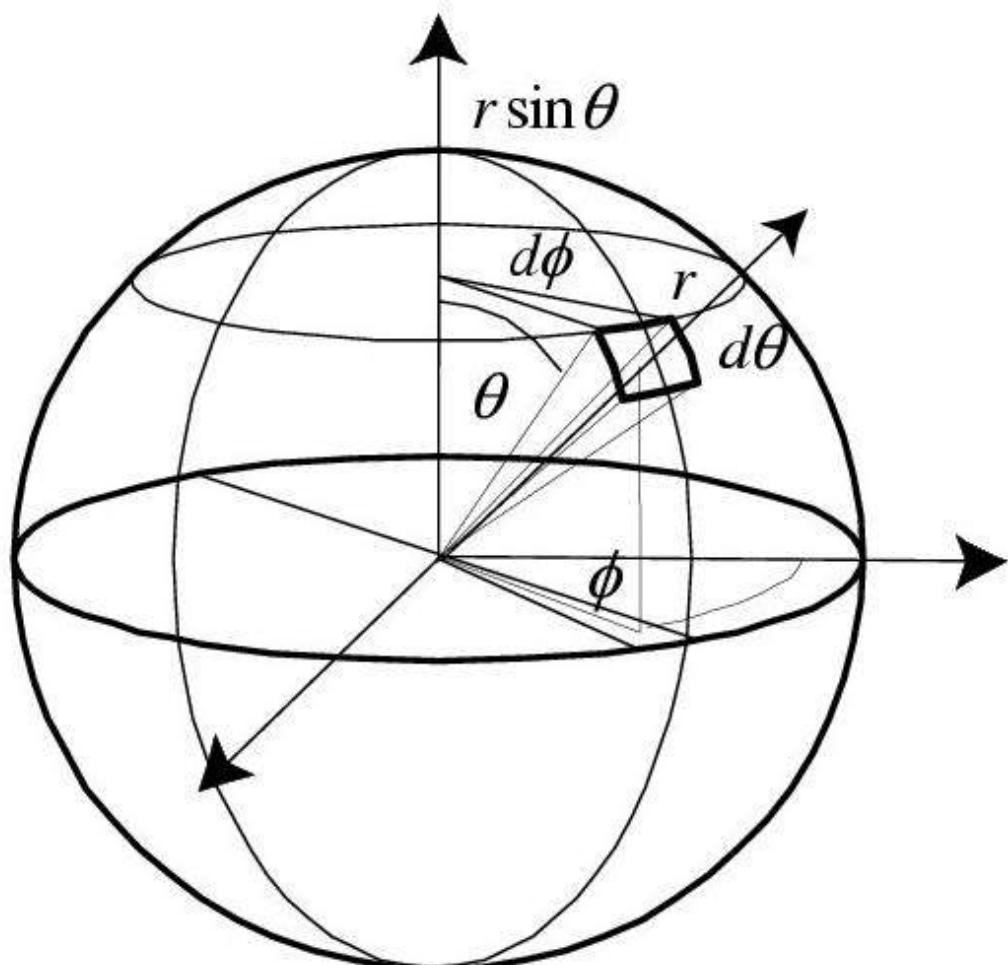
Integrate over entire hemisphere of possible source directions:

$$L^{surface}(\theta_r, \phi_r) = \int_{2\pi}^{\pi} L^{src}(\theta_i, \phi_i)f(\theta_i, \phi_i; \theta_r, \phi_r)\cos\theta_i d\omega_i$$

Convert from solid angle to theta-phi representation:

$$L^{surface}(\theta_r, \phi_r) = \int_{-\pi}^{\pi} \int_0^{\pi/2} L^{src}(\theta_i, \phi_i)f(\theta_i, \phi_i; \theta_r, \phi_r)\cos\theta_i \sin\theta_i d\theta_i d\phi_i$$

Differential Solid Angles

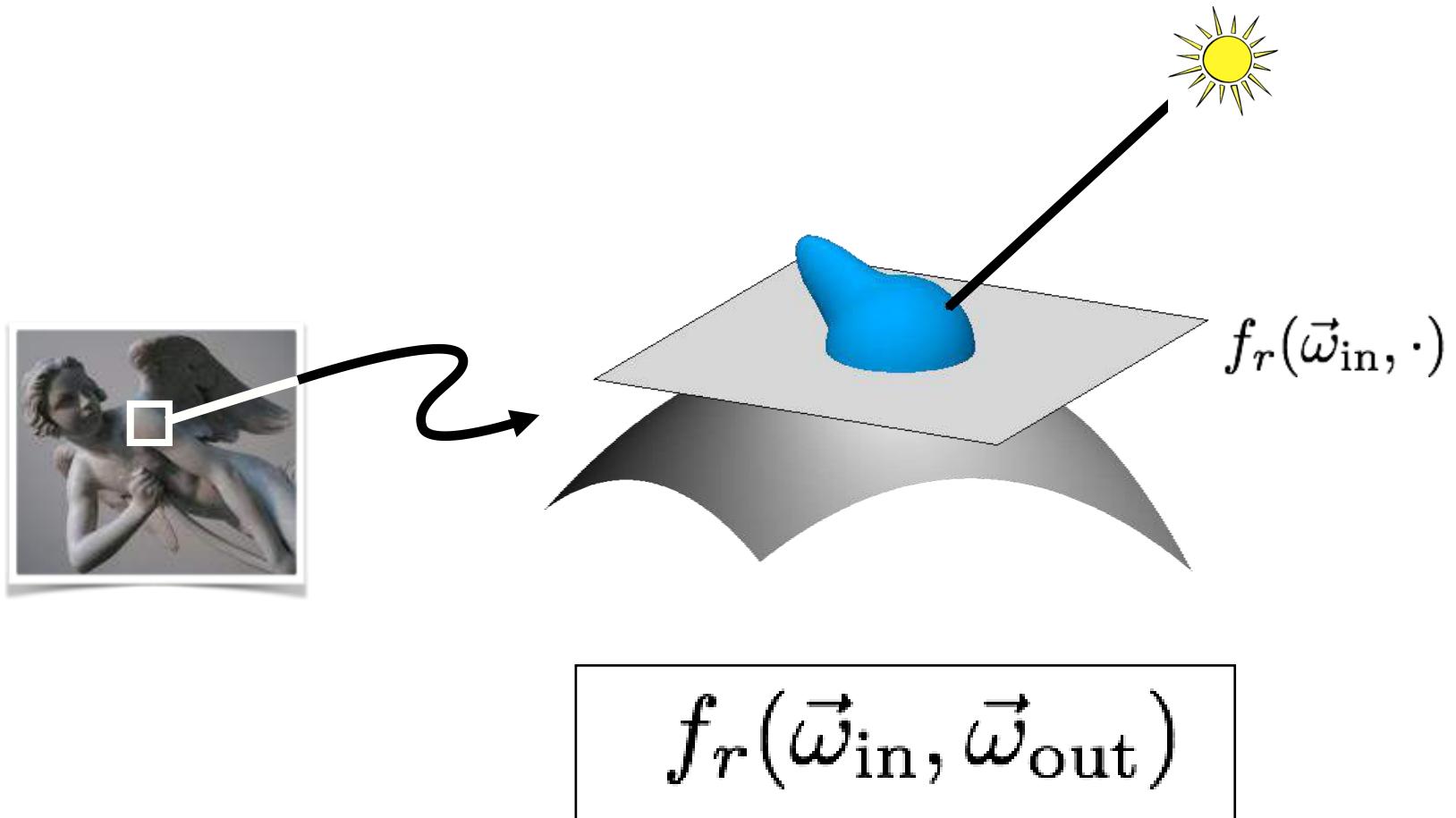


$$\begin{aligned} dA &= (r d\theta)(r \sin \theta d\phi) \\ &= r^2 \sin \theta d\theta d\phi \end{aligned}$$

$$d\omega = \frac{dA}{r^2} = \sin \theta d\theta d\phi$$

$$S = \int_0^{\pi} \int_0^{2\pi} \sin \theta d\theta d\phi = 4\pi$$

BRDF



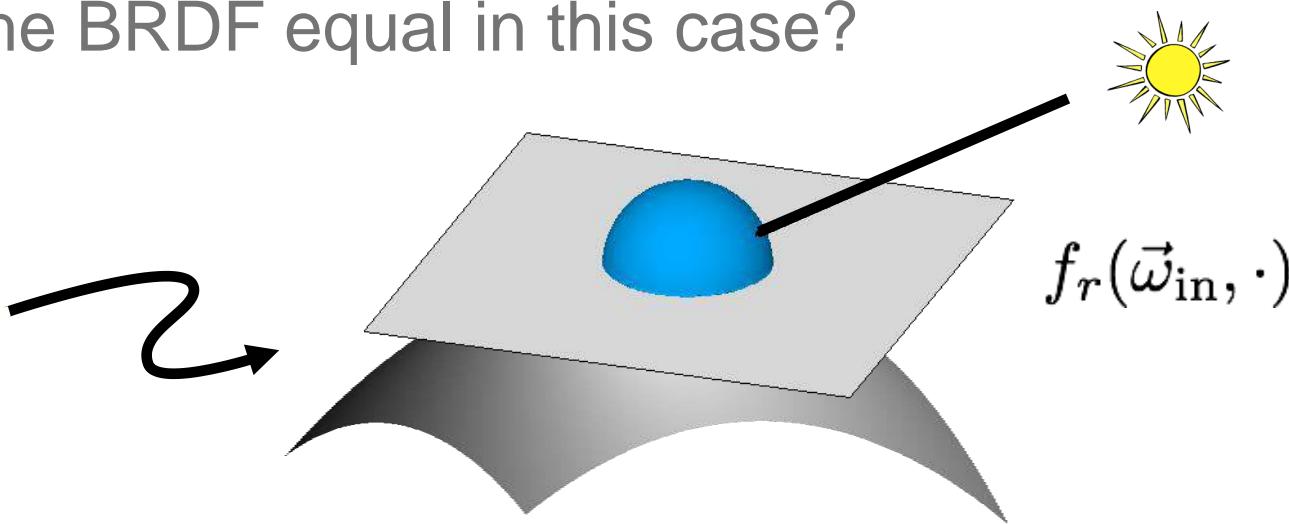
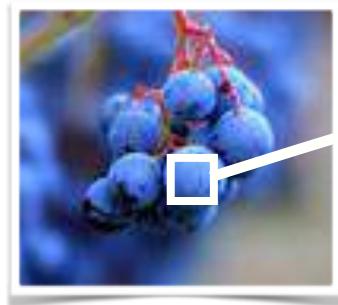
$$f_r(\vec{\omega}_{\text{in}}, \vec{\omega}_{\text{out}})$$

Bi-directional Reflectance Distribution Function (BRDF)

BRDF

Lambertian (diffuse) BRDF: energy equally distributed in all directions

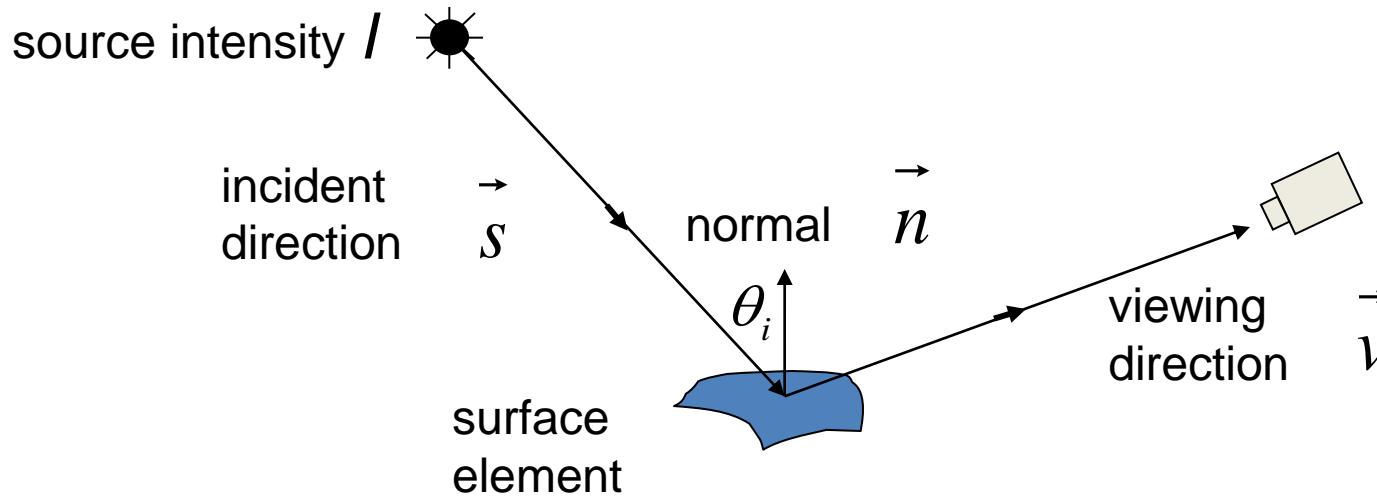
What does the BRDF equal in this case?



$$f_r(\vec{\omega}_{\text{in}}, \vec{\omega}_{\text{out}})$$

Bi-directional Reflectance Distribution Function (BRDF)

Diffuse Reflection and Lambertian BRDF

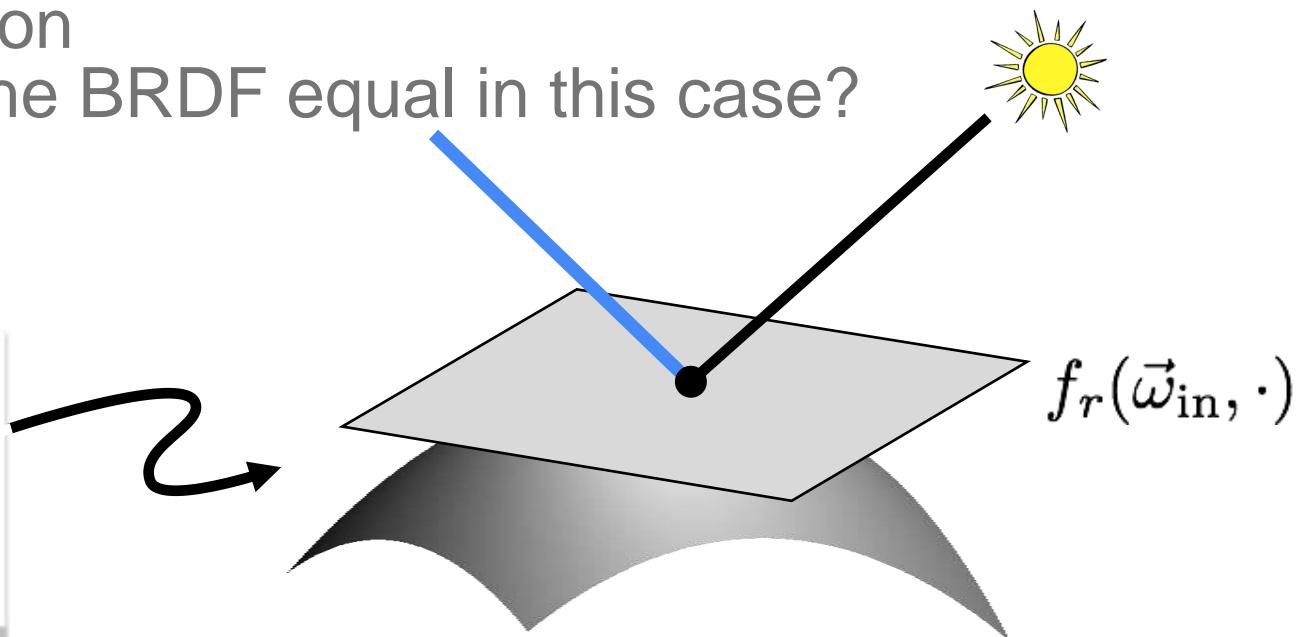


- Surface appears equally bright from ALL directions! (independent of \vec{v})
- Lambertian BRDF is simply a constant : $f(\theta_i, \phi_i; \theta_r, \phi_r) = \frac{\rho_d}{\pi}$ albedo
- Most commonly used BRDF in Vision and Graphics!

BRDF

Specular BRDF: all energy concentrated in mirror direction

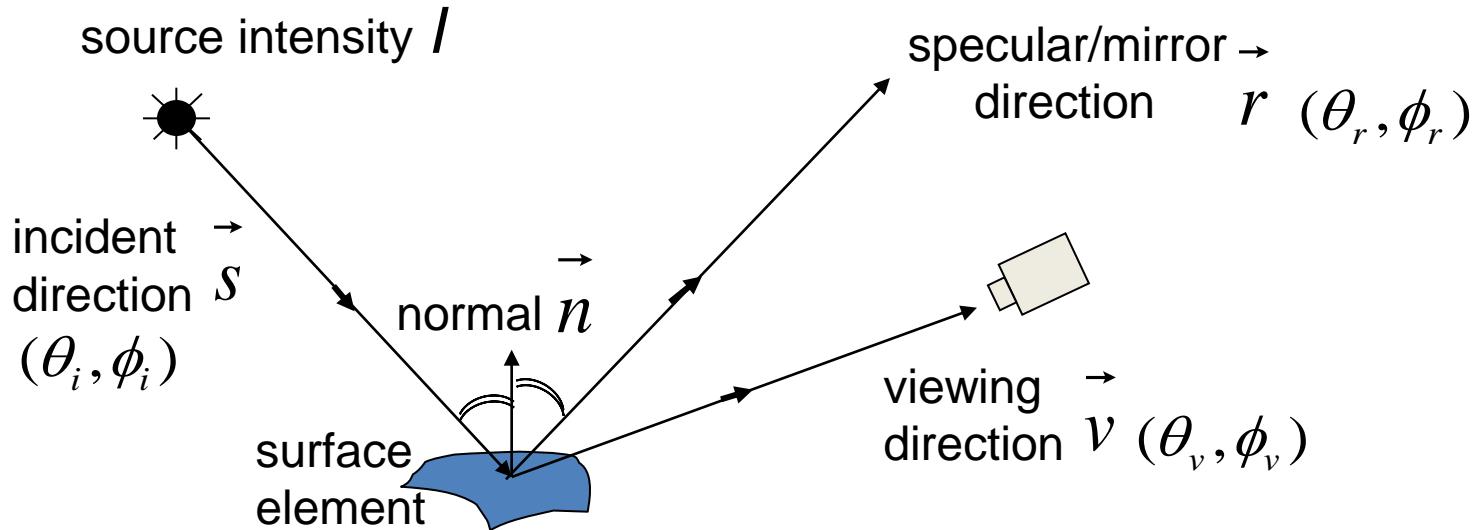
What does the BRDF equal in this case?



$$f_r(\vec{\omega}_{in}, \vec{\omega}_{out})$$

Bi-directional Reflectance Distribution Function (BRDF)

Specular Reflection and Mirror BRDF



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

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

specular albedo

Example Surfaces

Body Reflection:

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



Surface Reflection:

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

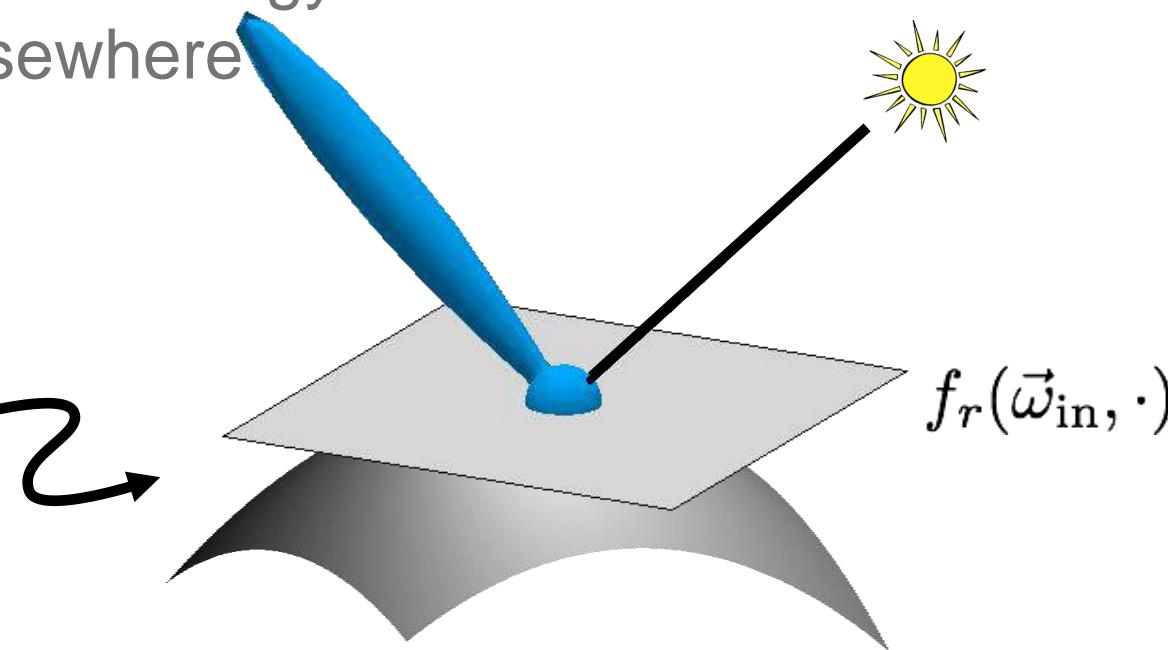
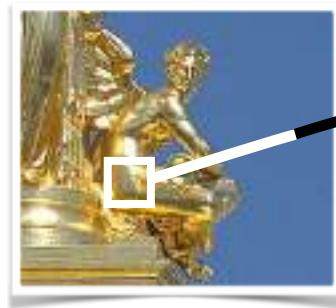


Many materials exhibit both Reflections:



BRDF

Glossy BRDF: more energy concentrated in mirror direction than elsewhere



$$f_r(\vec{\omega}_{in}, \vec{\omega}_{out})$$

Bi-directional Reflectance Distribution Function (BRDF)

Thank you: Question?

Computer Vision

Image Formation

Dr. Mrinmoy Ghorai

**Indian Institute of Information Technology
Sri City, Chittoor**

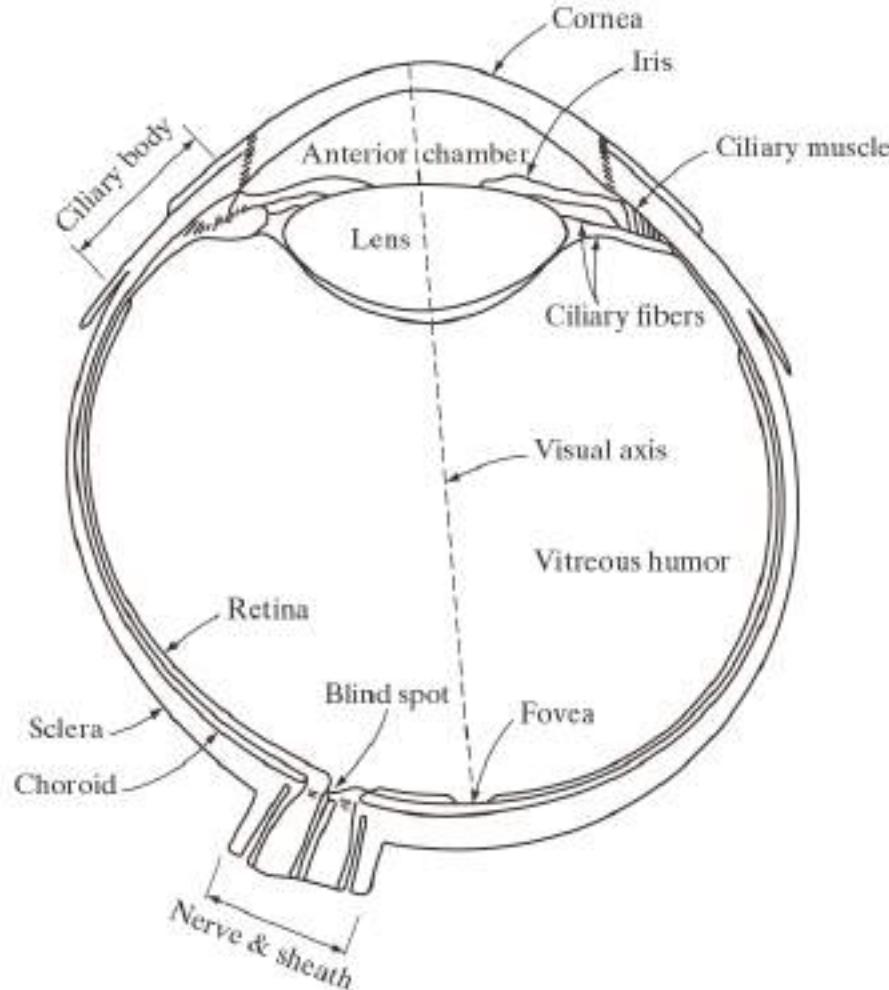


Today's Agenda

- Image Formation
 - Elements of Visual Perception
 - Image Sensing and Acquisition
 - Image Sampling and Quantization
 - Representing Digital Images
 - Fundamental Radiometric Relation

Elements of Visual Perception

Structure of the Human Eye



Elements of Visual Perception

Image Formation in the Eye

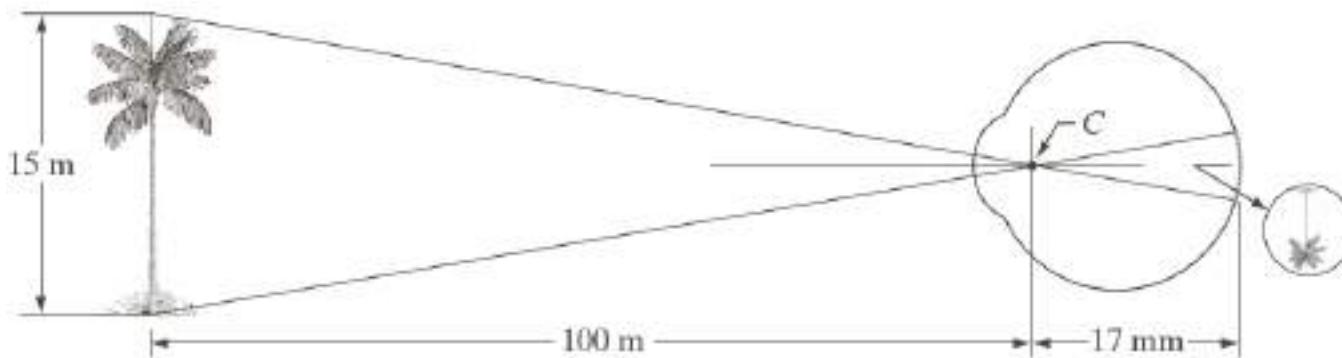


FIGURE 2.3
Graphical representation of the eye looking at a palm tree. Point C is the optical center of the lens.

- If h is the height in mm of the object in the retinal image then the figure yields $\frac{15}{100} = \frac{h}{17} \Rightarrow h = 2.55 \text{ mm}$.
- The retinal image is reflected primarily in the area of the fovea.

Elements of Visual Perception

Brightness Adaption and Discrimination

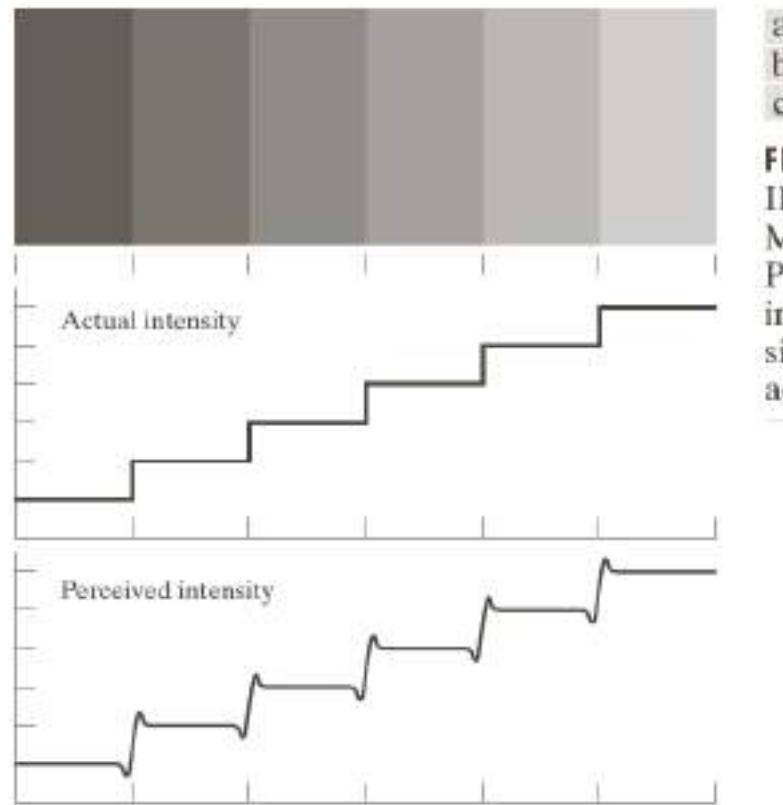
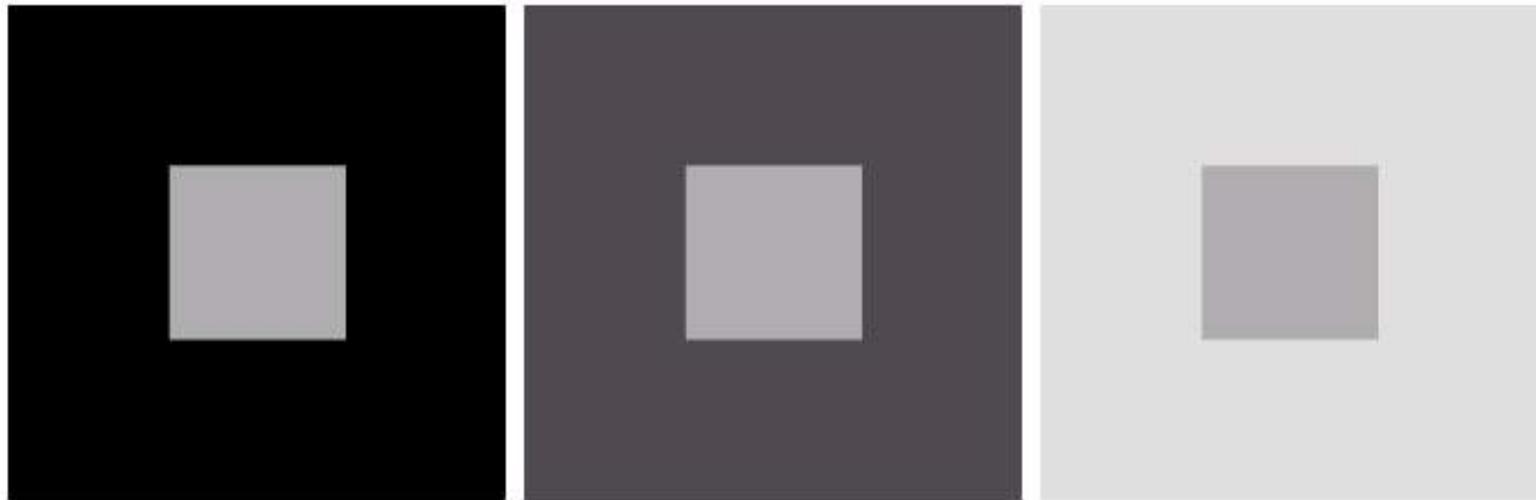


FIGURE 2.7
Illustration of the
Mach band effect.
Perceived
intensity is not a
simple function of
actual intensity.

Elements of Visual Perception

Brightness Adaption and Discrimination



a | b | c

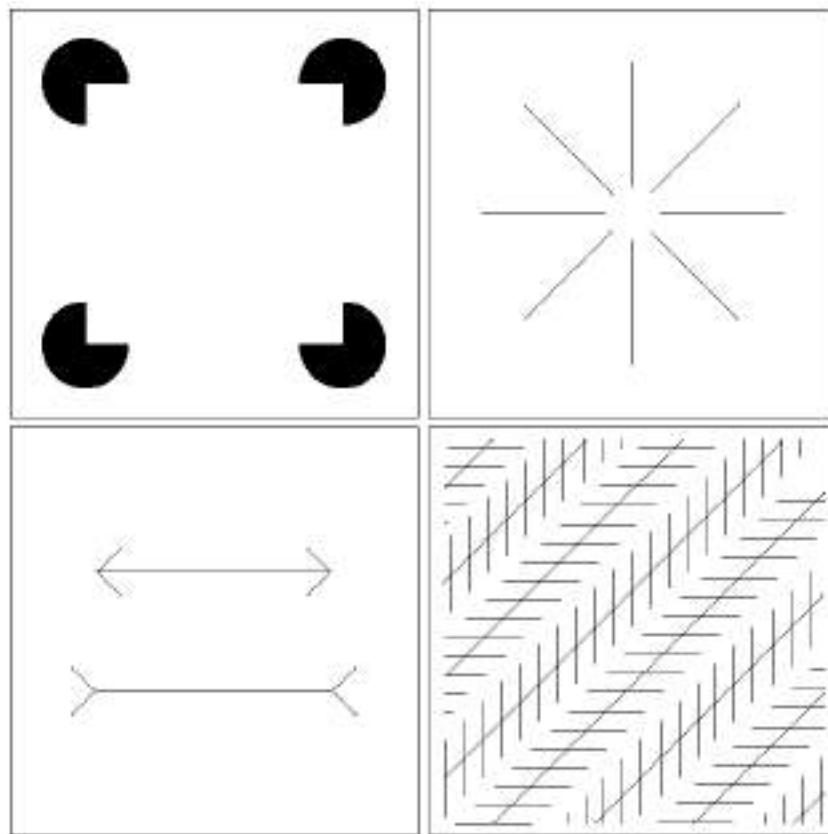
FIGURE 2.8 Examples of simultaneous contrast. All the inner squares have the same intensity, but they appear progressively darker as the background becomes lighter.

Elements of Visual Perception

Brightness Adaption and Discrimination

a b
c d

FIGURE 2.9 Some well-known optical illusions.



Elements of Visual Perception

Light and the Electromagnetic Spectrum

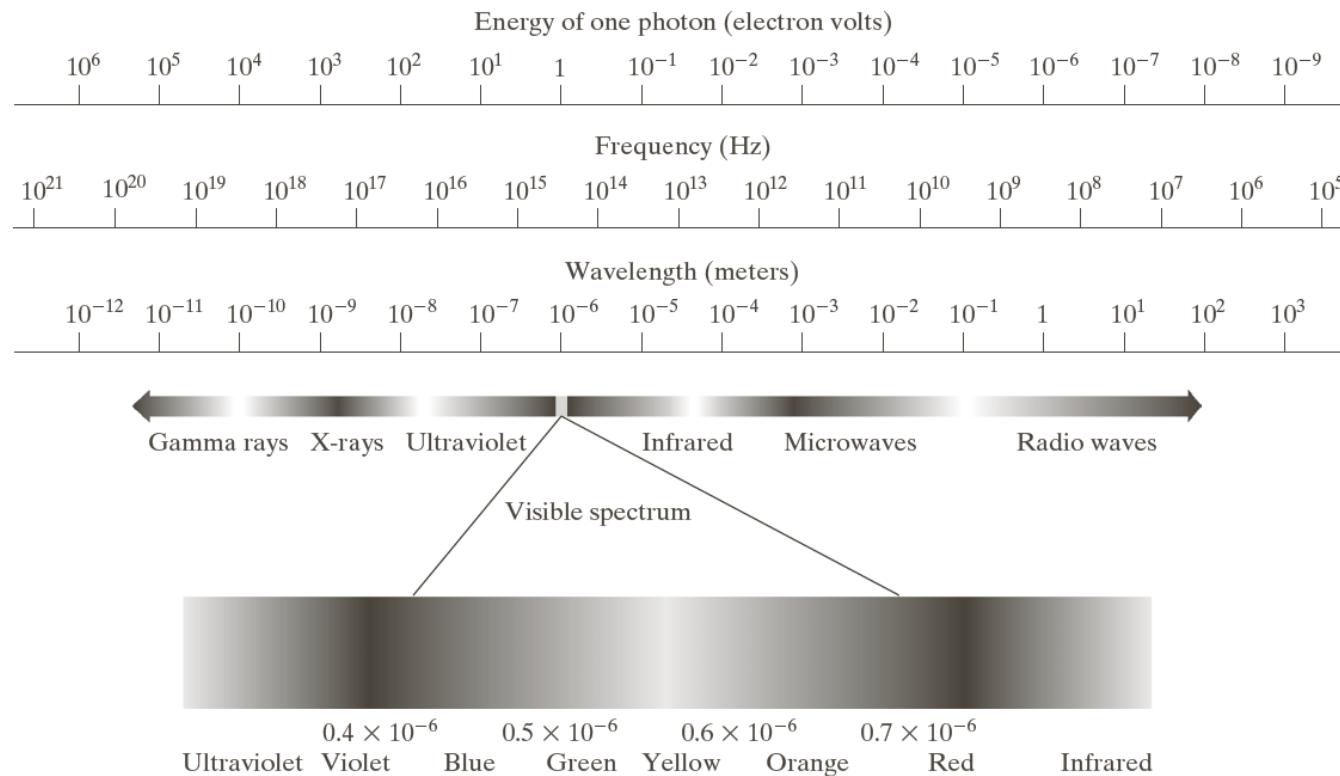
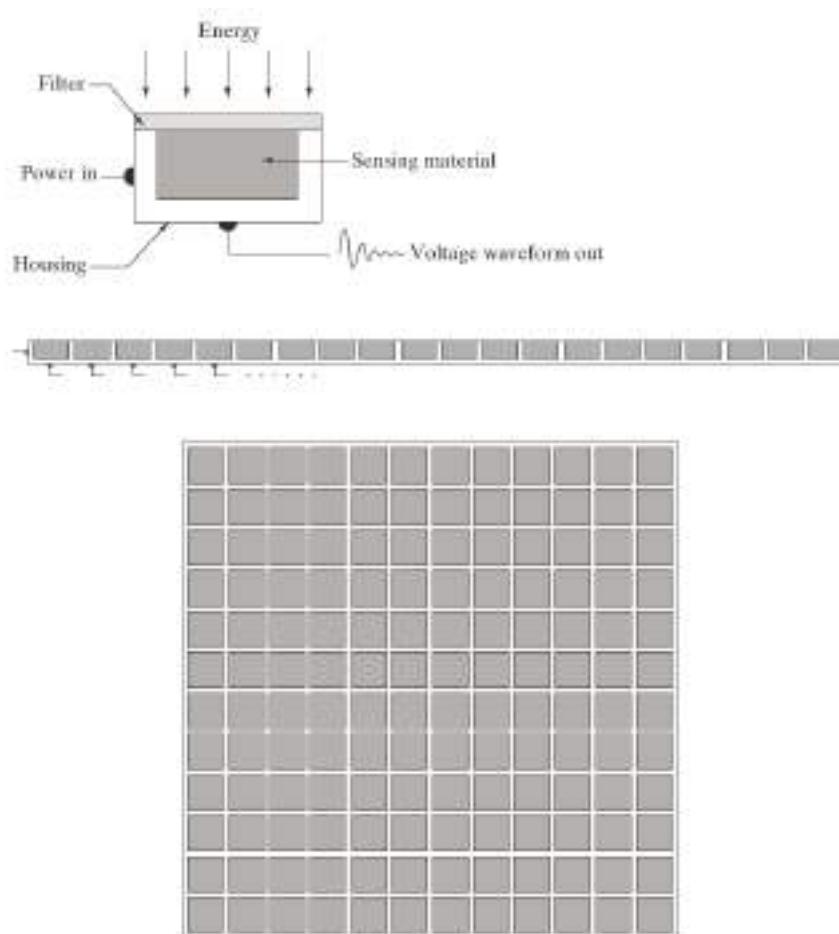


FIGURE 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanation, but note that the visible spectrum is a rather narrow portion of the EM spectrum.

Image Sensing and Acquisition

Image Acquisition using Sensor Strips and Sensor Arrays

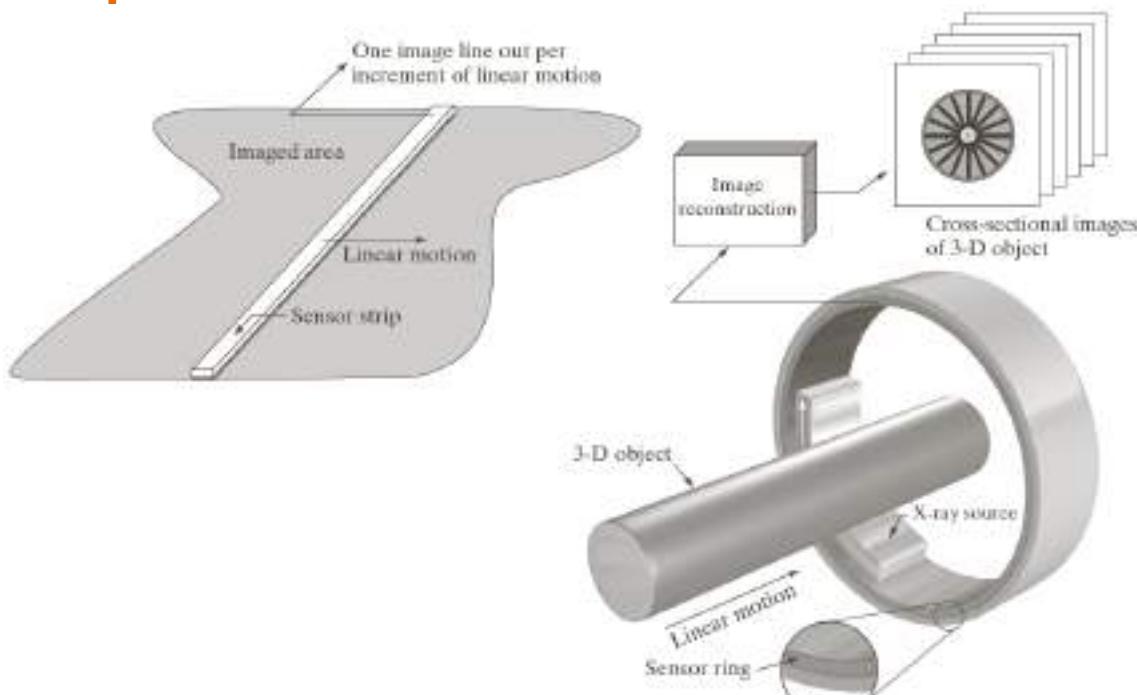


a
b
c

FIGURE 2.12
(a) Single imaging sensor.
(b) Line sensor.
(c) Array sensor.

Image Sensing and Acquisition

Image Acquisition using Linear Sensor Strip and Circular Sensor Strip



a b

FIGURE 2.14 (a) Image acquisition using a linear sensor strip. (b) Image acquisition using a circular sensor strip.

Image Sensing and Acquisition

Image Acquisition Process

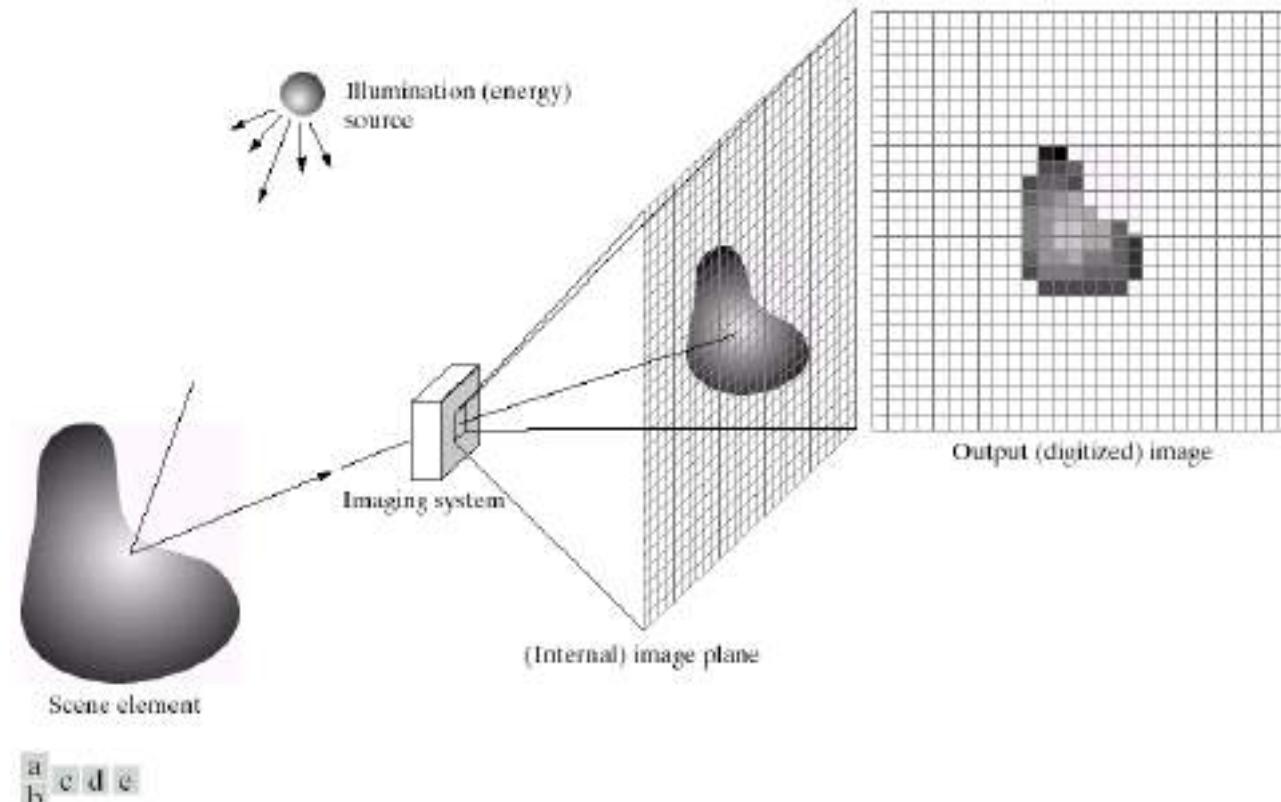


FIGURE 2.15 An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Image Sensing and Acquisition

A Simple Image Formation Model

$$f(x, y) = i(x, y) * r(x, y)$$

$f(x, y)$: intensity at the point (x, y)

$i(x, y)$: illumination at the point (x, y)

(the amount of source illumination incident on the scene)

$r(x, y)$: reflectance at the point (x, y)

(the amount of illumination reflected by the object)

where $0 < i(x, y) < \infty$ and $0 < r(x, y) < 1$

Image Sensing and Acquisition

Some Typical Ranges of Illumination

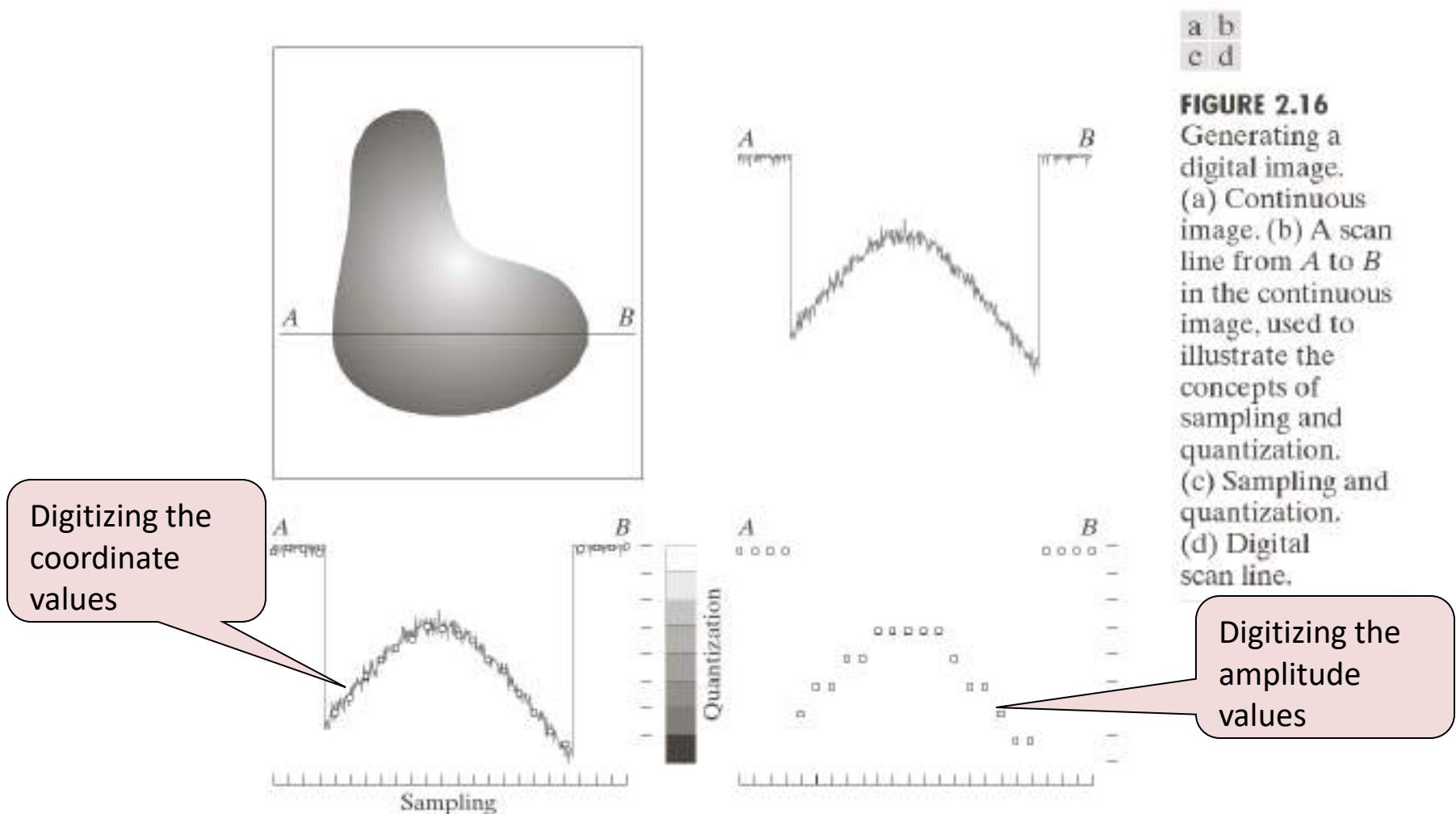
- **Lumen** - a unit of light flow or luminous flux
- **Lumen per square meter (lm/m^2)** - the metric unit of measure for illuminance of a surface
- On a clear day, the sun may produce in excess of 90,000 lm/m^2 of illumination on the surface of the Earth
- On a cloudy day, the sun may produce less than 10,000 lm/m^2 of illumination on the surface of the Earth
- On a clear evening, the moon yields about 0.1 lm/m^2 of illumination

Image Sensing and Acquisition

Some Typical Ranges of Reflectance

- 0.01 for black velvet
- 0.65 for stainless steel
- 0.80 for flat-white wall paint
- 0.90 for silver-plated metal
- 0.93 for snow

Image Sampling and Quantization



a b
c d

FIGURE 2.16
Generating a digital image.
(a) Continuous image.
(b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization.
(c) Sampling and quantization.
(d) Digital scan line.

Digitizing the amplitude values

Image Sampling and Quantization

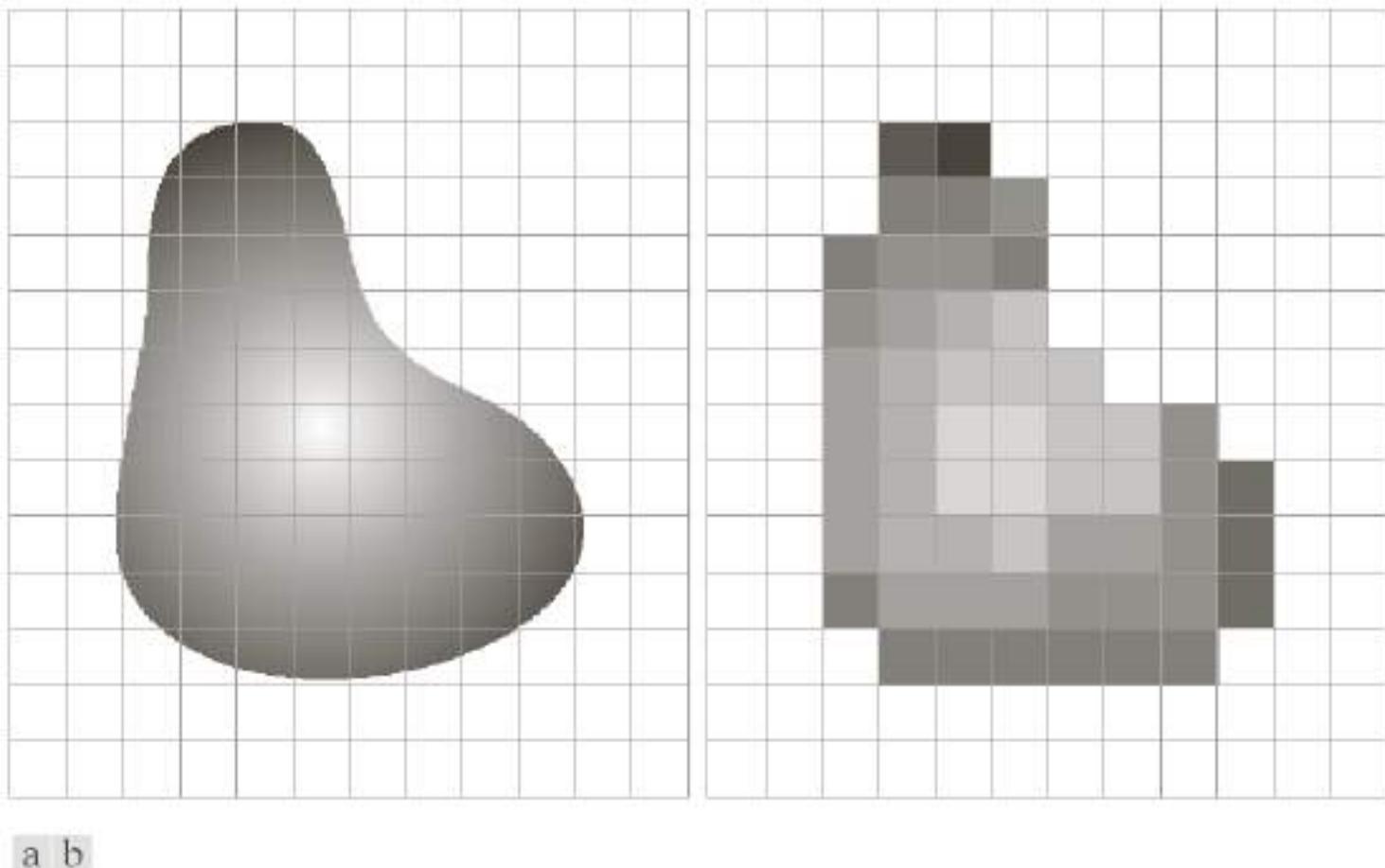


FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Representing Digital Images

The representation of a $M \times N$ numerical array as

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \dots & \dots & \dots & \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix}$$

Representing Digital Images

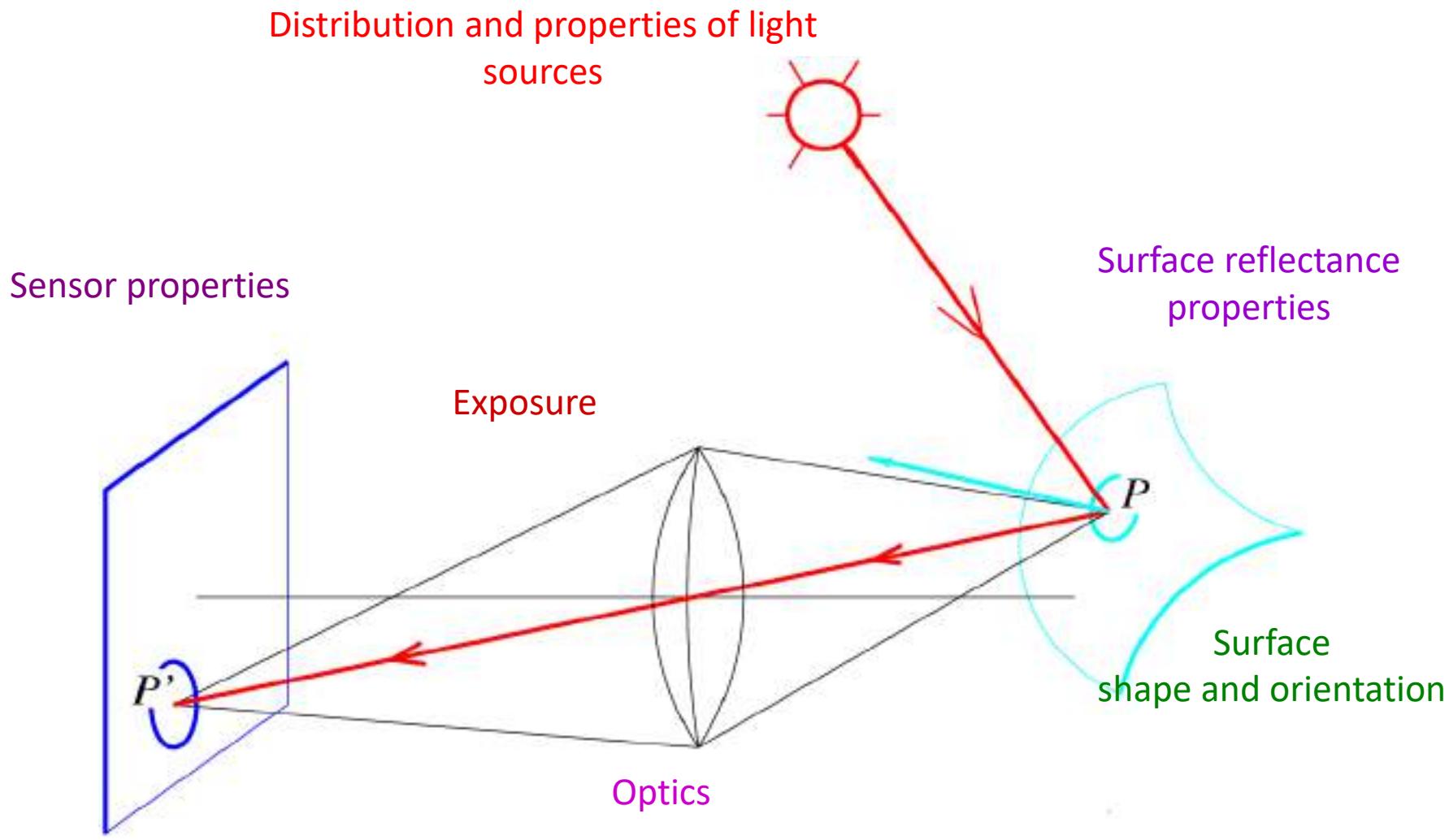
- Discrete intensity interval $[0, L - 1]$, $L = 2^k$
- The number b bits required to store a $M \times N$ digitized Image

$$b = M \times N \times k$$

Spatial and Intensity Resolution

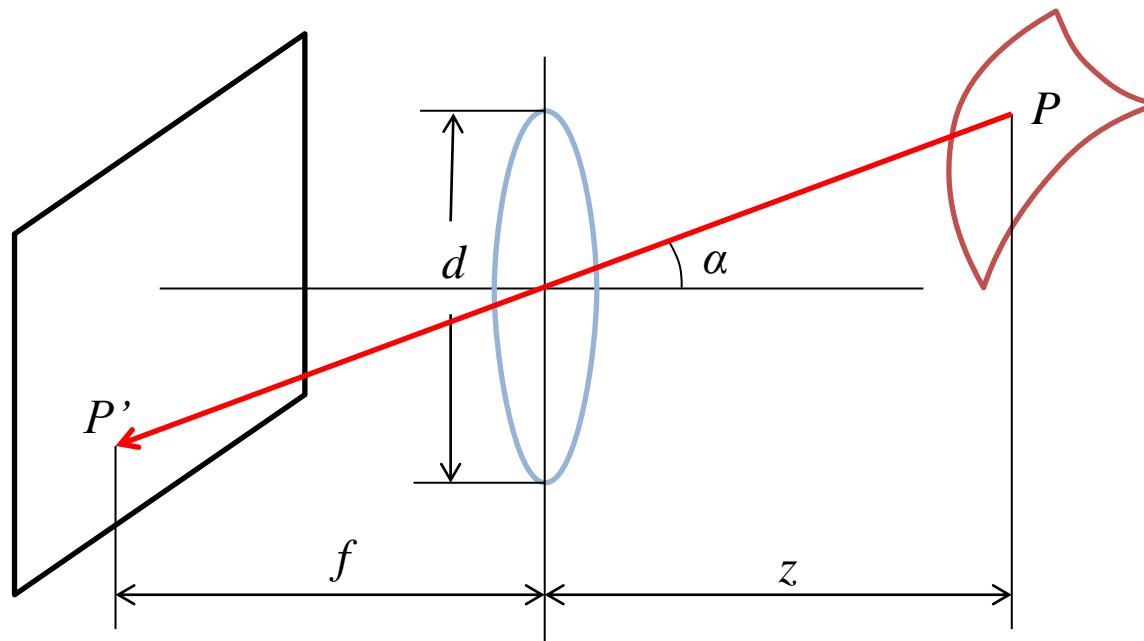
- **Spatial Resolution**
 - a measure of smallest discernible detail in an image
 - stated with ***dots (pixels) per unit distance, dots per inch (dpi)***
- **Intensity Resolution**
 - the smallest discernible change in intensity level
 - stated with ***8 bits, 16 bits, 24 bits***, etc.

What determines the brightness of an image pixel?



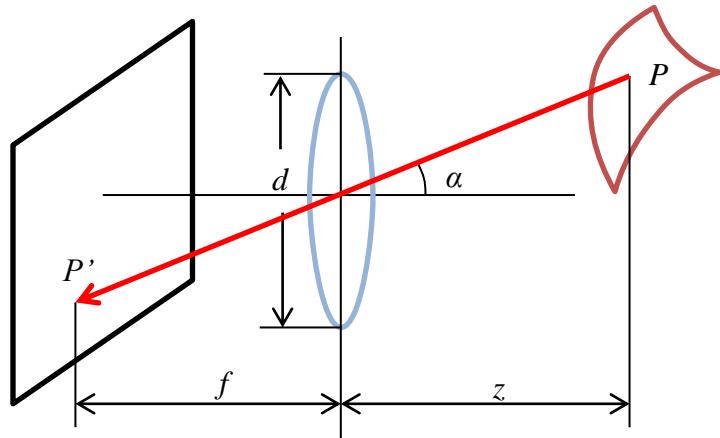
Fundamental Radiometric Relation

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



What is the relationship between E and L ?

Fundamental Radiometric Relation

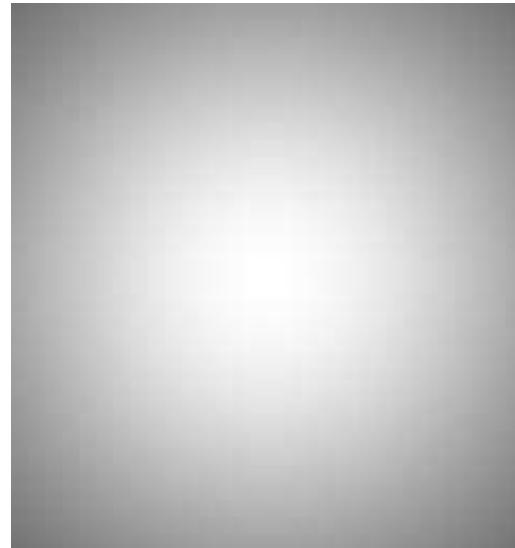


$$E = \left[\frac{\pi}{4} \left(\frac{d}{f} \right)^2 \cos^4 \alpha \right] L$$

- Image irradiance is linearly related to scene radiance
- Irradiance is proportional to the area of the lens and inversely proportional to the squared distance between the lens and the image plane
- The irradiance falls off as the angle between the viewing ray and the optical axis increases

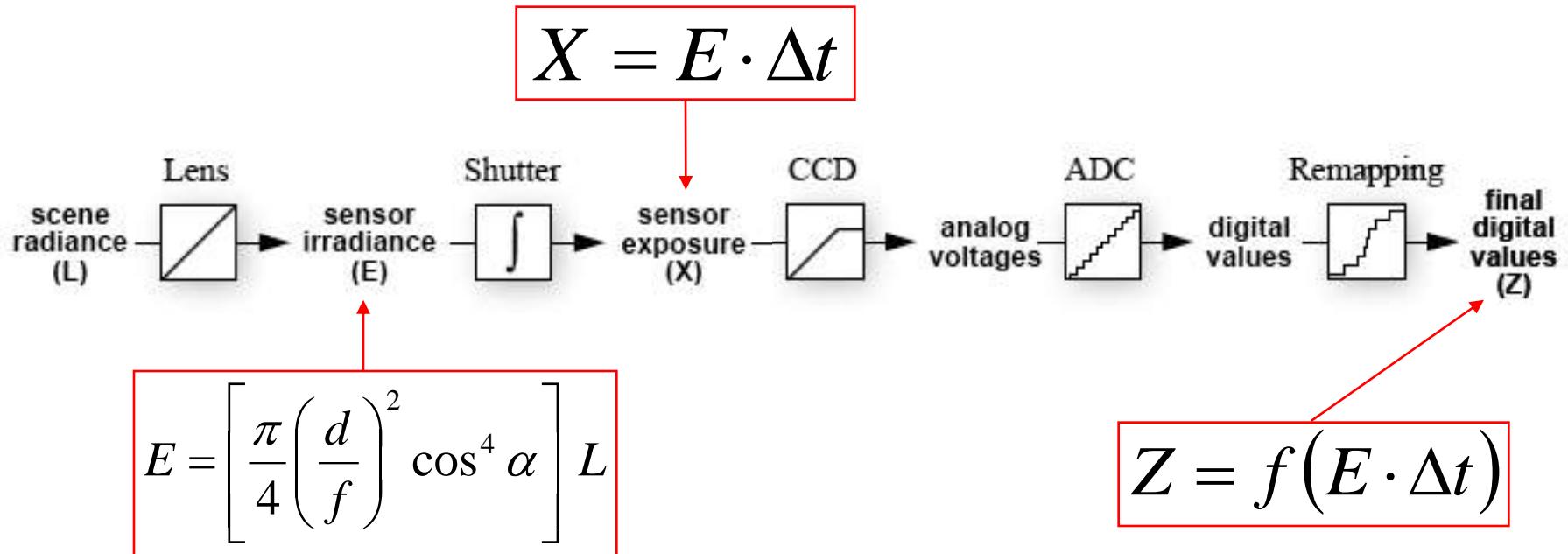
Fundamental Radiometric Relation

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



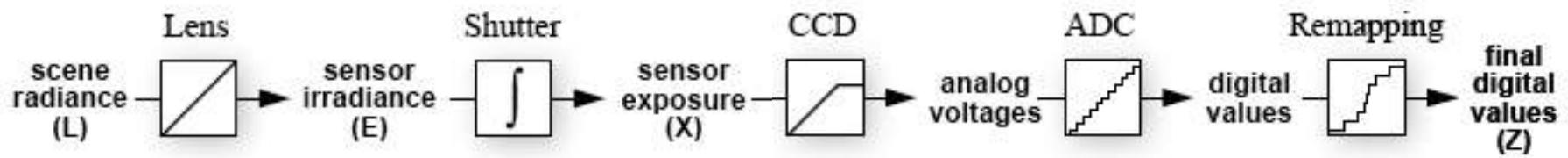
S. B. Kang and R. Weiss. [Can we calibrate a camera using an image of a flat, textureless Lambertian surface?](#) ECCV 2000

From light rays to pixel values



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

From light rays to pixel values



- For more information:
 - M. Brown, [Understanding the In-Camera Image Processing Pipeline for Computer Vision](#), CVPR 2016 Tutorial

Thank you: Question?

Computer Vision

Image Filtering

Dr. Mrinmoy Ghorai

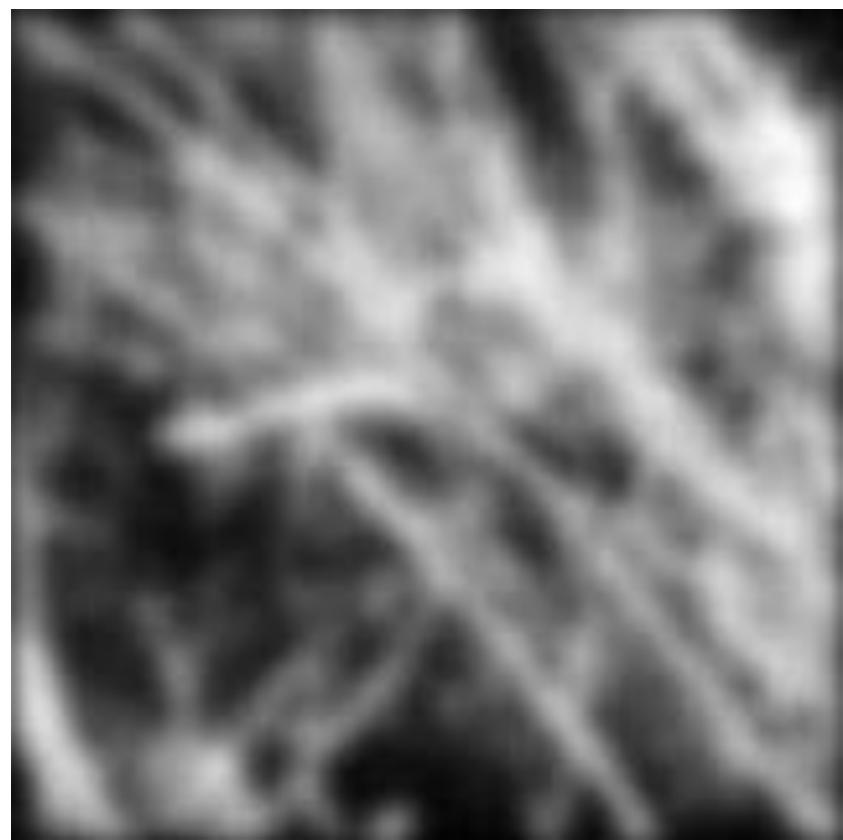
**Indian Institute of Information Technology
Sri City, Chittoor**



Today's Agenda

- Image Filtering
 - Smoothing
 - Sharpening

Linear Filtering



Motivation: Image denoising

- How can we reduce noise in a photograph?



Moving average

- Let's replace each pixel with a *weighted average* of its neighborhood
- The weights are called the *filter kernel*
- What are the weights for the average of a 3x3 neighborhood?

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

“box filter”

Image filtering

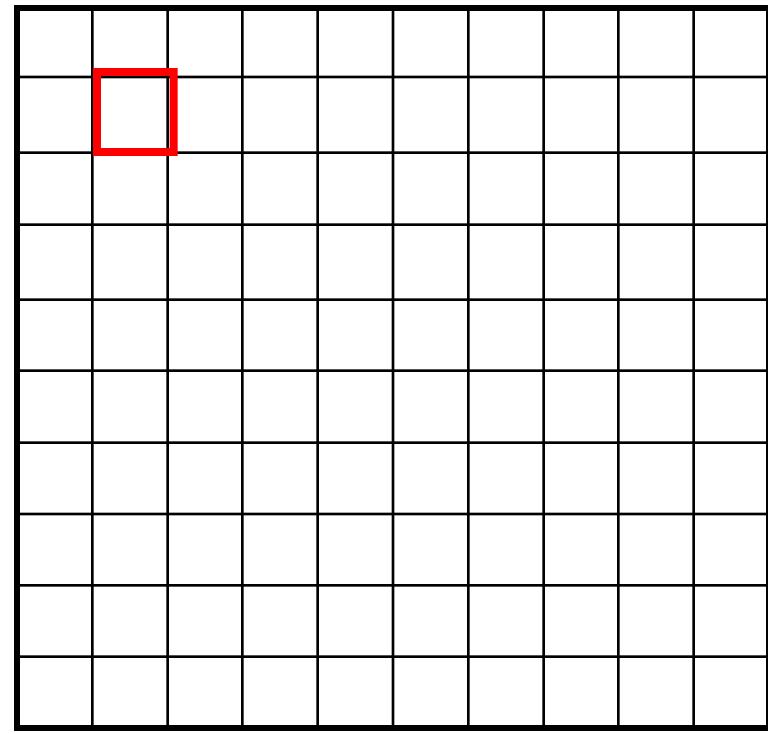
$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[.,.]$

$h[.,.]$

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



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

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

$h[.,.]$

0	10									

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

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

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

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

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

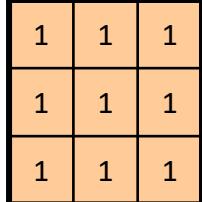
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

0 10 20 30 30

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

Image filtering

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


A 3x3 matrix with all entries equal to 1/9, representing a uniform averaging filter.

$$f[.,.]$$

$$h[.,.]$$

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

	0	10	20	30	30				

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

Image filtering

 $f[.,.]$

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

 $h[.,.]$

	0	10	20	30	30					

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

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

1	1	1
1	1	1
1	1	1

Image filtering

$$g[\cdot, \cdot] \quad \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$$f[., .]$$

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

$$h[., .]$$

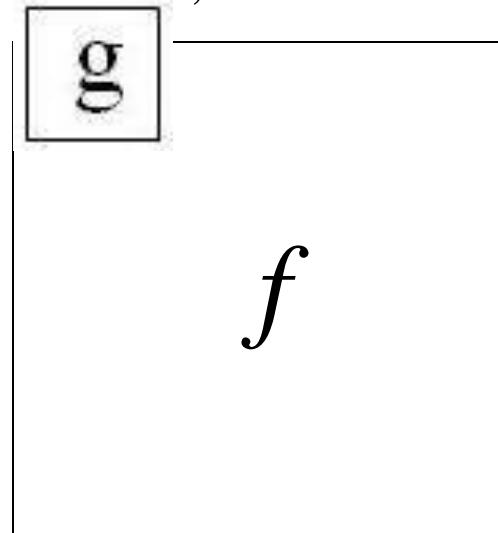
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

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

Defining convolution

- Let f be the image and g be the kernel. The output of convolving f with g is denoted $f * g$.

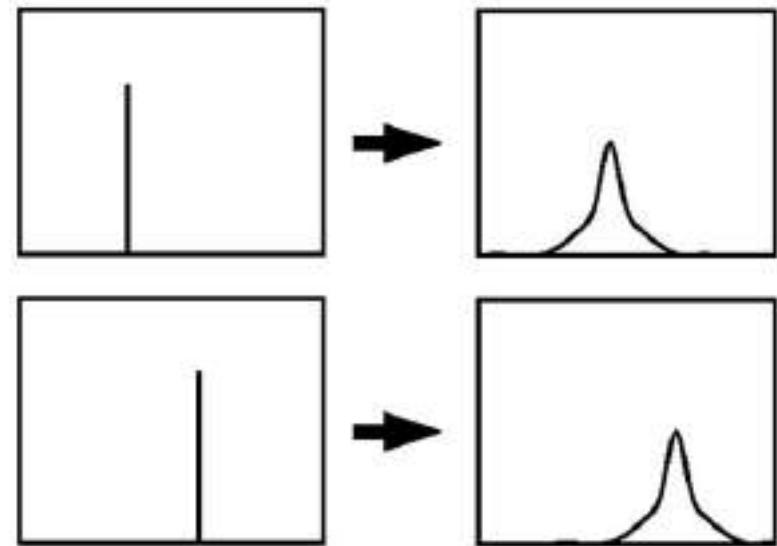
$$(f * g)[m, n] = \sum_{k, l} f[m - k, n - l] g[k, l]$$



Key properties

- **Shift invariance:** same behavior regardless of pixel location:

$$\text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f))$$



- **Linearity:**

$$\begin{aligned}\text{filter}(f_1 + f_2) &= \\ \text{filter}(f_1) + \text{filter}(f_2)\end{aligned}$$

- Theoretical result: any linear shift-invariant operator can be represented as a convolution

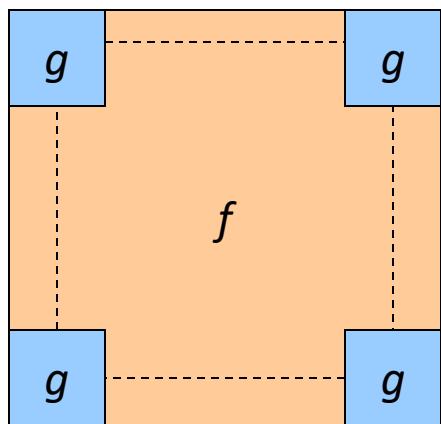
Properties in more detail

- 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 $e = [..., 0, 0, 1, 0, 0, ...]$,
 $a * e = a$

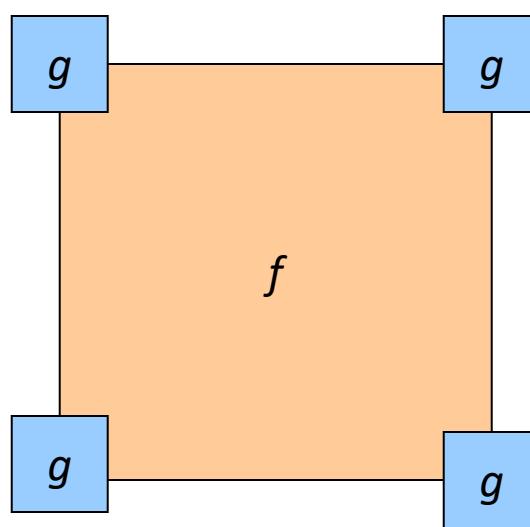
Dealing with edges

- If we convolve image f with filter g , what is the size of the output?

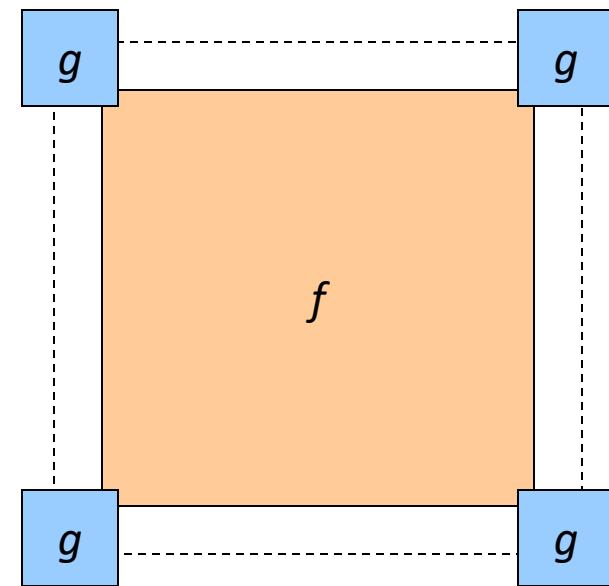
Output is smaller
than input



Output is same size
as input

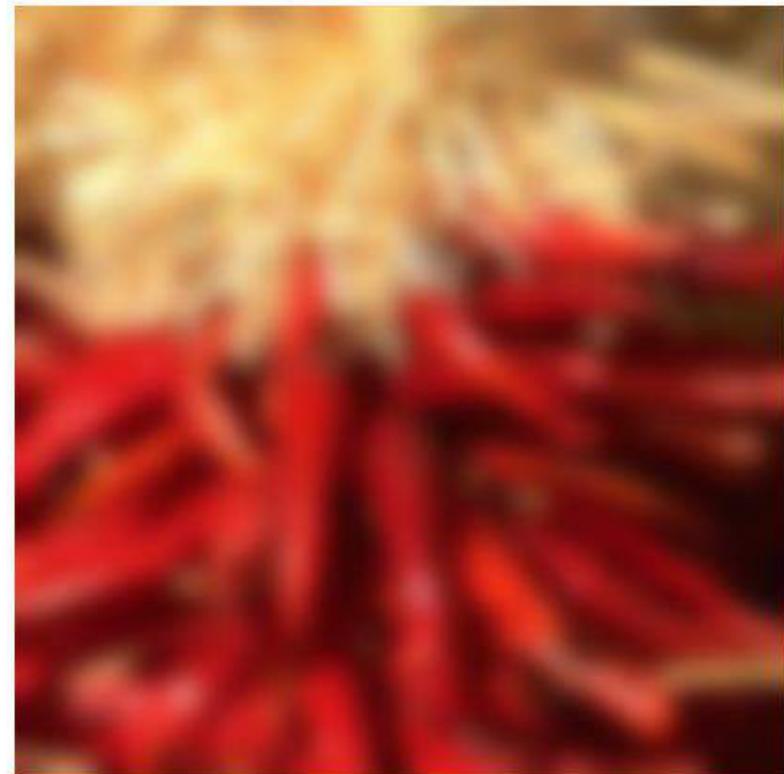


Output is larger than
input



Dealing with edges

- If the filter window falls off the edge of the image, we need to pad the image
 - Zero pad (or clip filter)
 - Wrap around
 - Copy edge
 - Reflect across edge



Practice with linear filters



Original

0	0	0
0	1	0
0	0	0

?

Practice with linear filters



Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters



Original

0	0	0
0	0	1
0	0	0

?

Practice with linear filters



Original

0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Practice with linear filters



Original

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

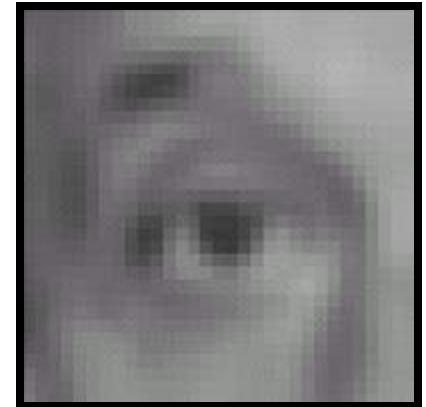
?

Practice with linear filters



Original

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



Blur (with a
box filter)

Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

?

(Note that filter sums to 1)

Practice with linear filters

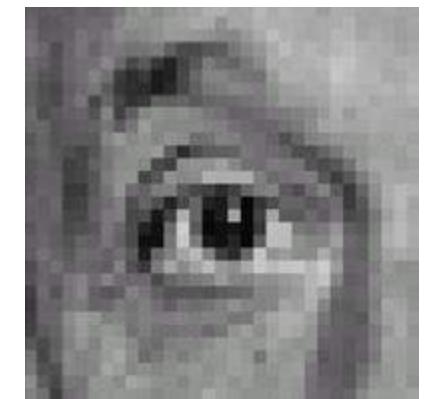


Original

0	0	0
0	2	0
0	0	0

-

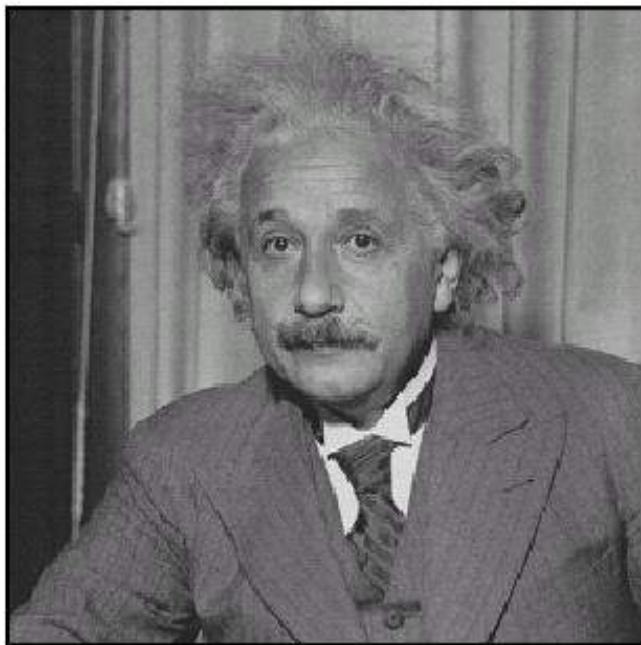
$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1



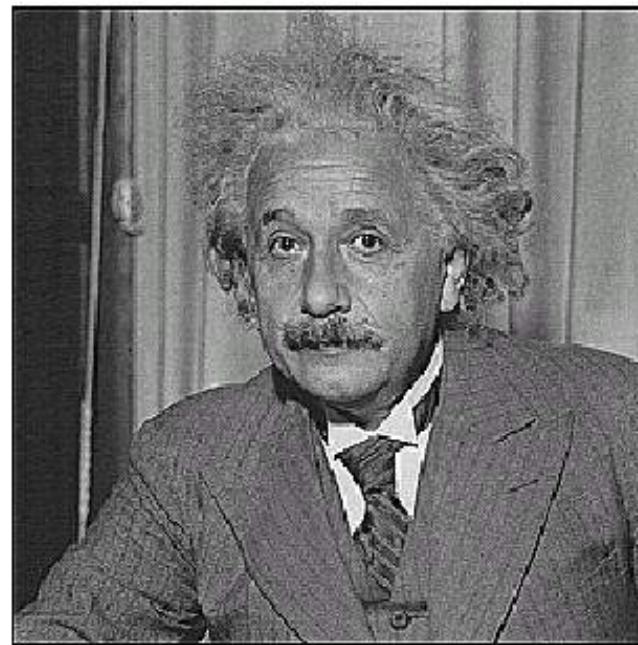
Sharpening filter

- Accentuates differences with local average

Sharpening



before



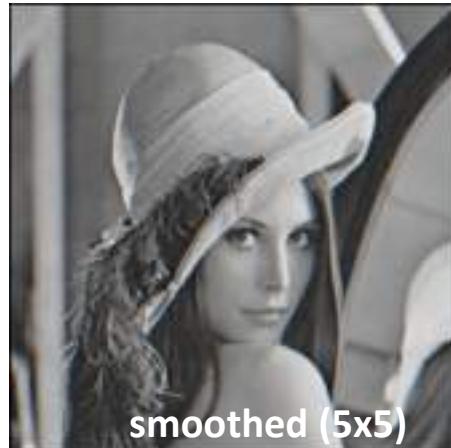
after

Sharpening

- What does blurring take away?



-



=



Let's add it back:



+

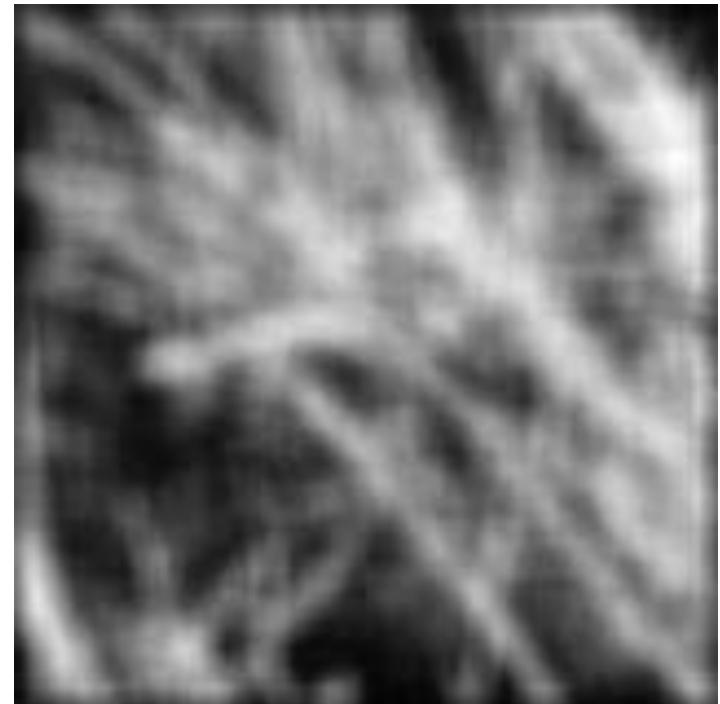
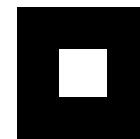


=



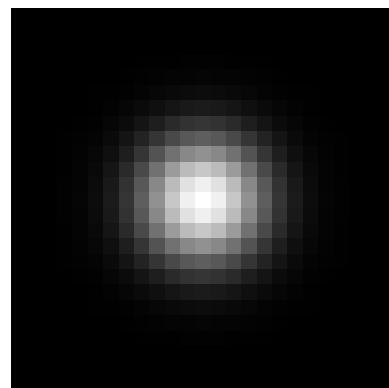
Smoothing with box filter revisited

- What's wrong with this picture?
- What's the solution?



Smoothing with box filter revisited

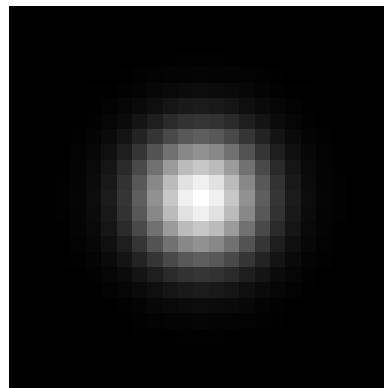
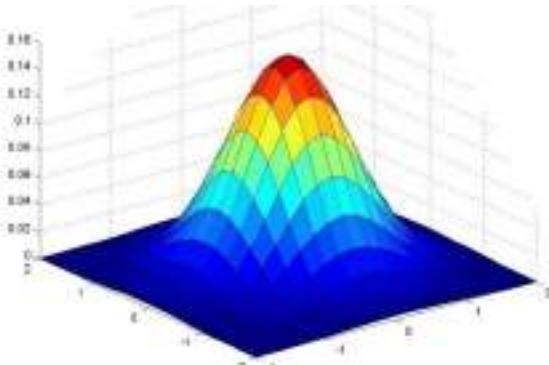
- What's wrong with this picture?
- What's the solution?
 - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center



“fuzzy blob”

Gaussian Kernel

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



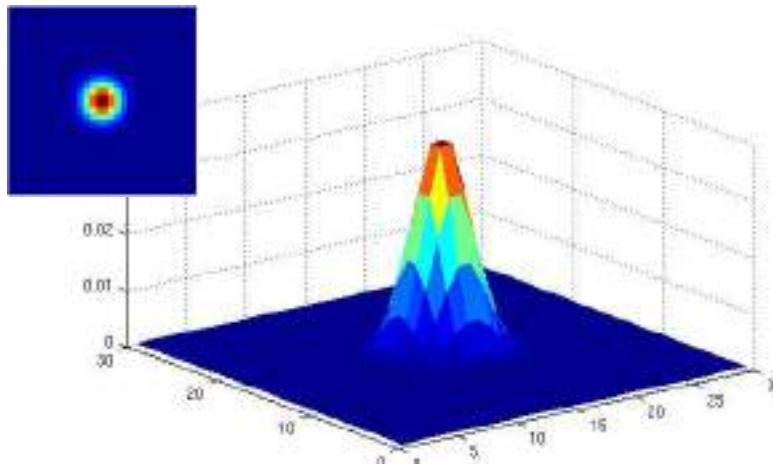
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

$5 \times 5, \sigma = 1$

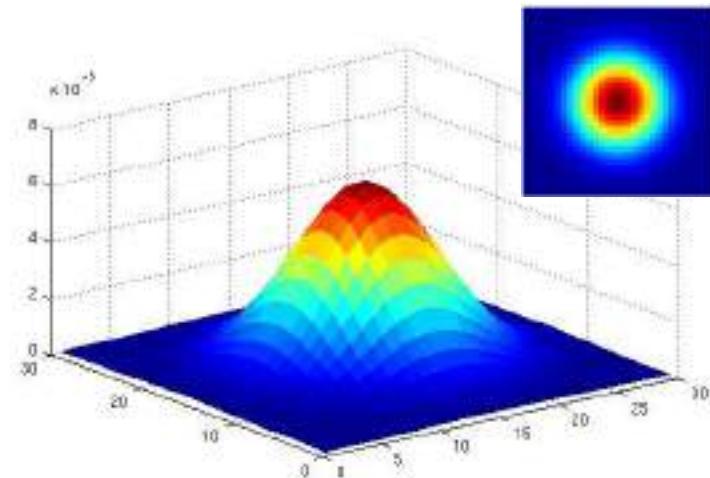
- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

Gaussian Kernel

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



$\sigma = 2$ with 30×30 kernel

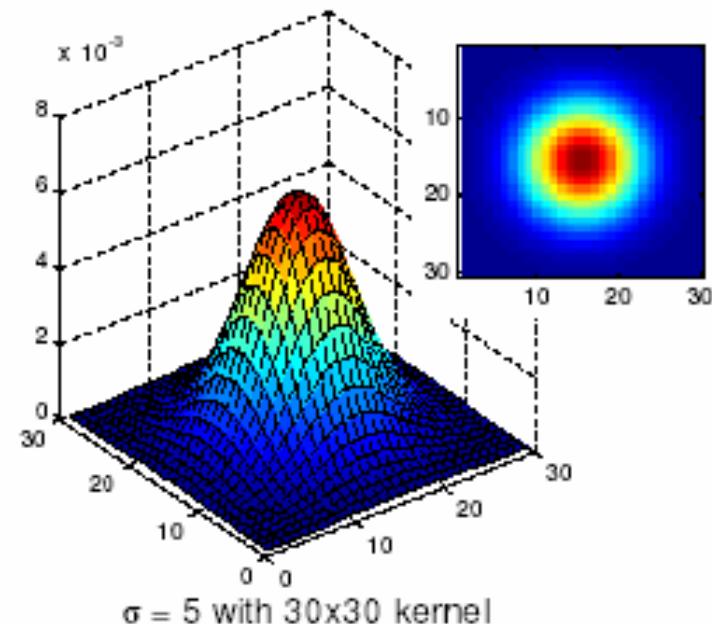
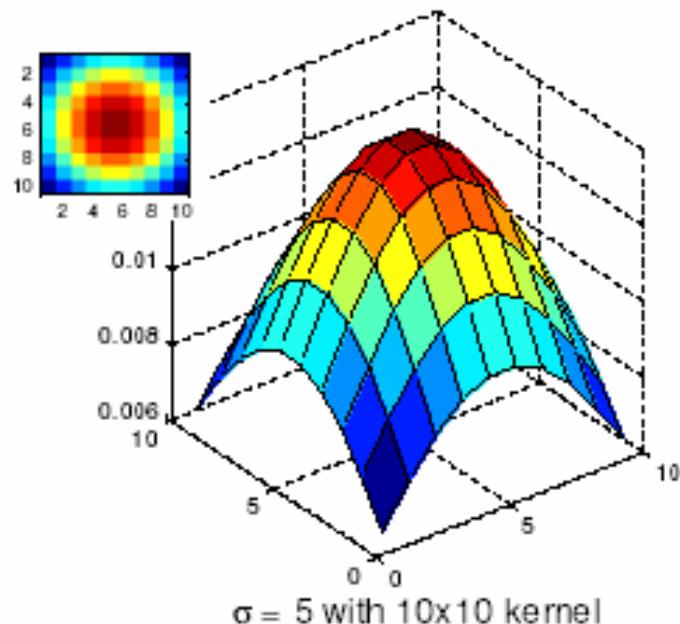


$\sigma = 5$ with 30×30 kernel

- Standard deviation σ : determines extent of smoothing

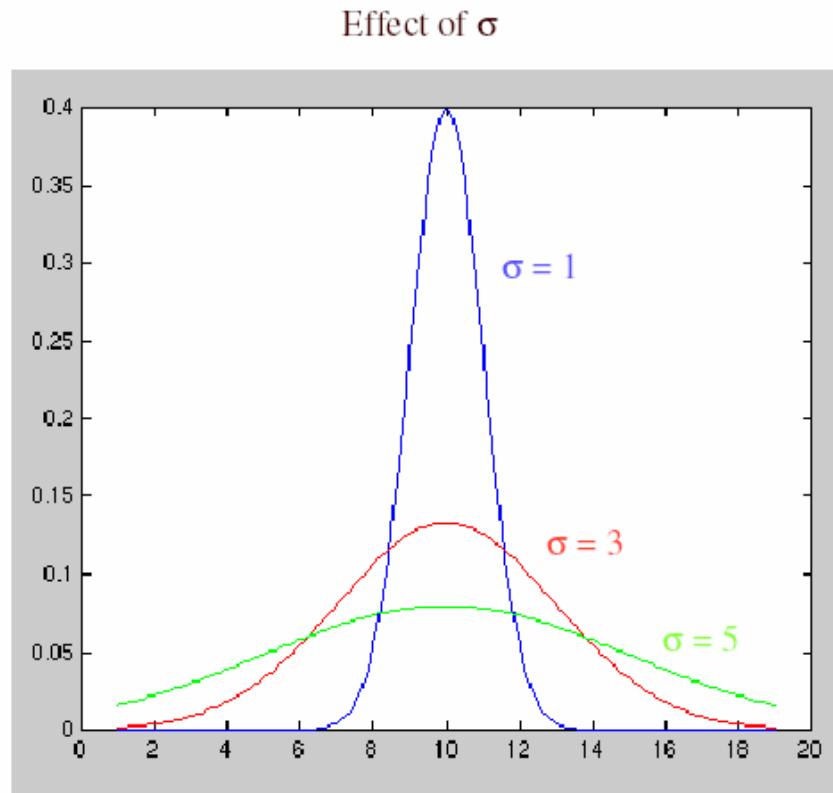
Choosing kernel width

- The Gaussian function has infinite support, but discrete filters use finite kernels

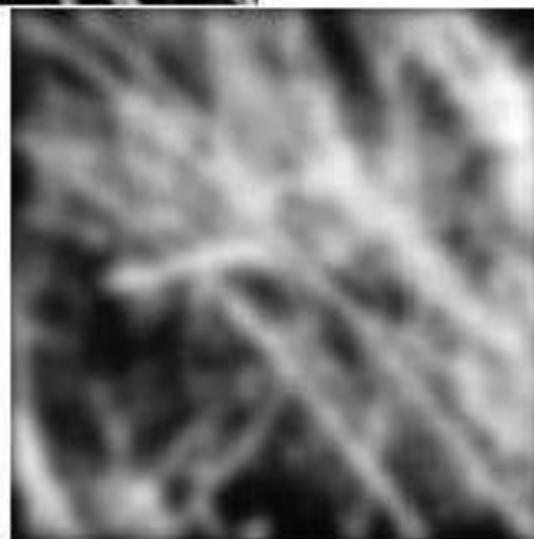
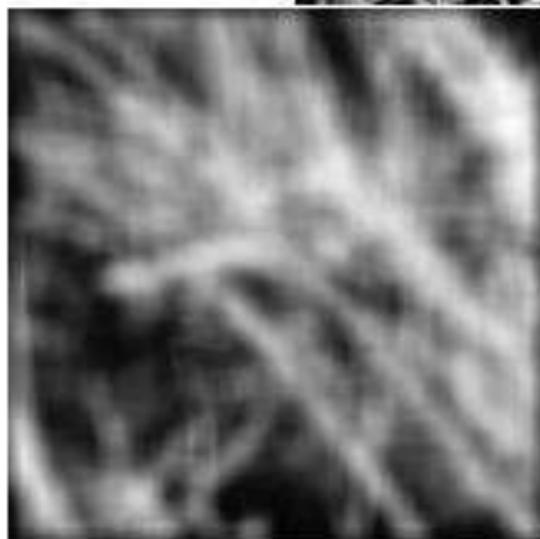


Choosing kernel width

- Rule of thumb: set filter half-width to about 3σ



Gaussian vs. box filtering



Gaussian filters

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

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Separability of the Gaussian filter

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 + y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

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

Why is separability useful?

- Separability means that a 2D convolution can be reduced to two 1D convolutions (one along rows and one along columns)
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
 - $O(n^2 m^2)$
- What if the kernel is separable?
 - $O(n^2 m)$

Noise



Original



Salt and pepper noise



Impulse noise

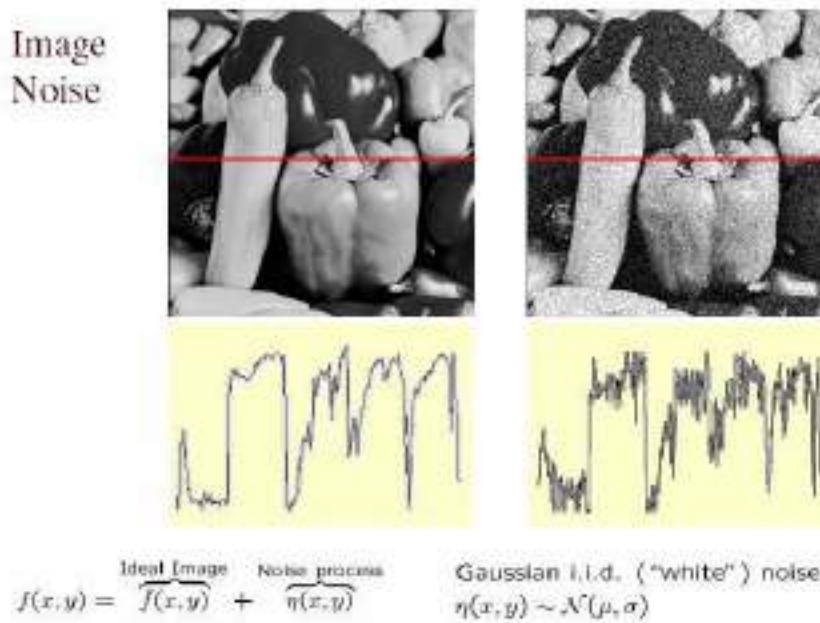


Gaussian noise

- **Salt and pepper noise:** contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution

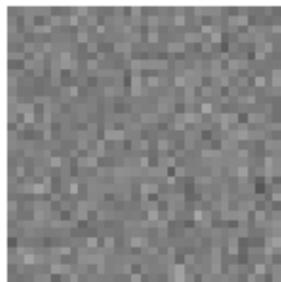
Gaussian noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise

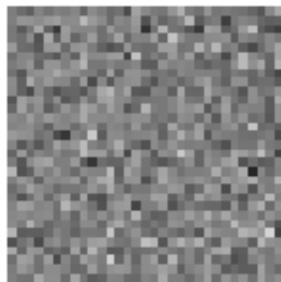


Reducing Gaussian noise

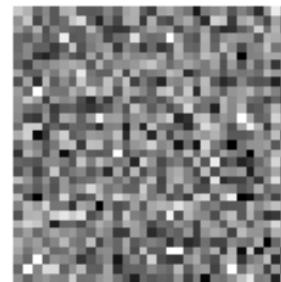
$\sigma=0.05$



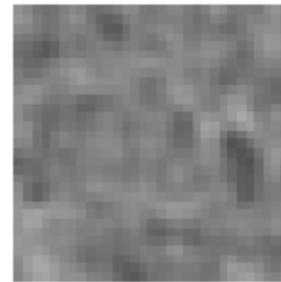
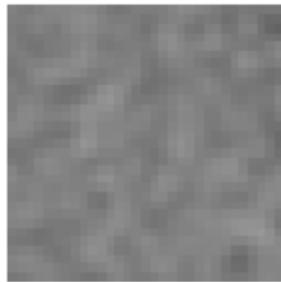
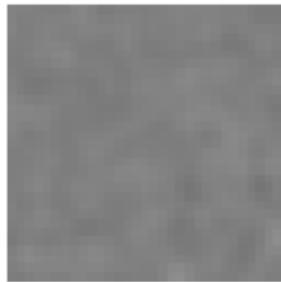
$\sigma=0.1$



$\sigma=0.2$



no
smoothing



$\sigma=1$ pixel



$\sigma=2$ pixels



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing salt-and-pepper noise

3x3



5x5



7x7

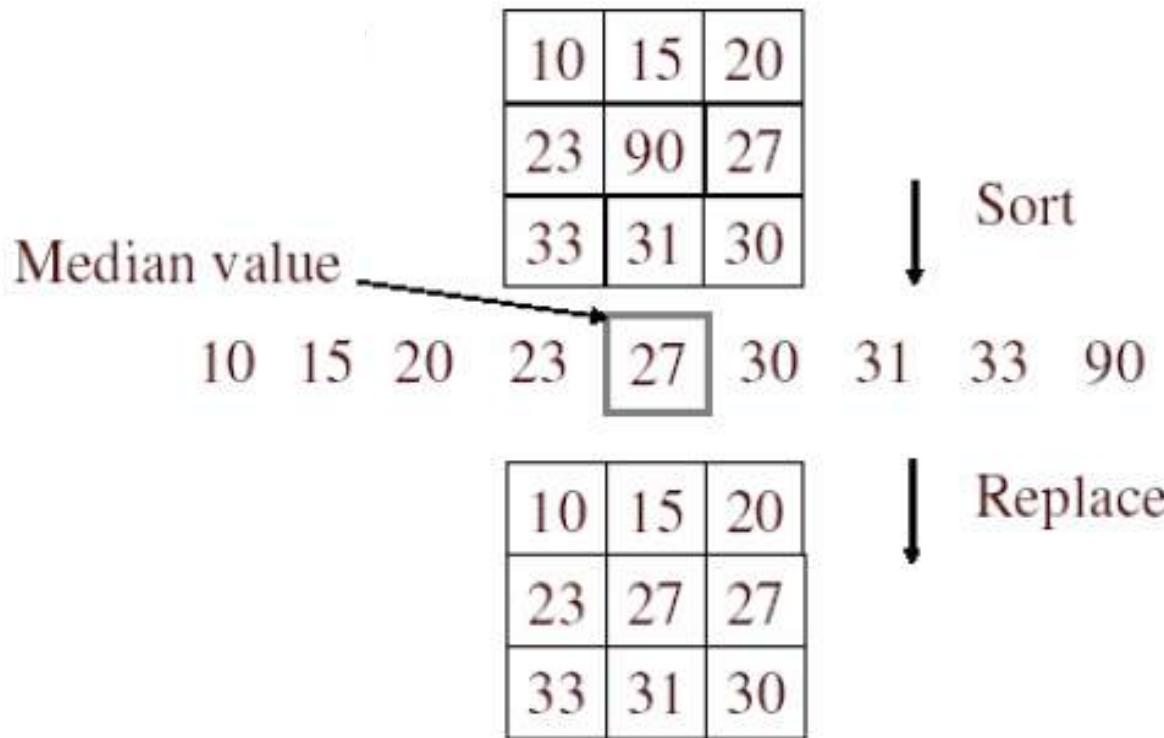


- What's wrong with the results?

Alternative idea: Median filtering

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

v



- Is median filtering linear?

Median filter

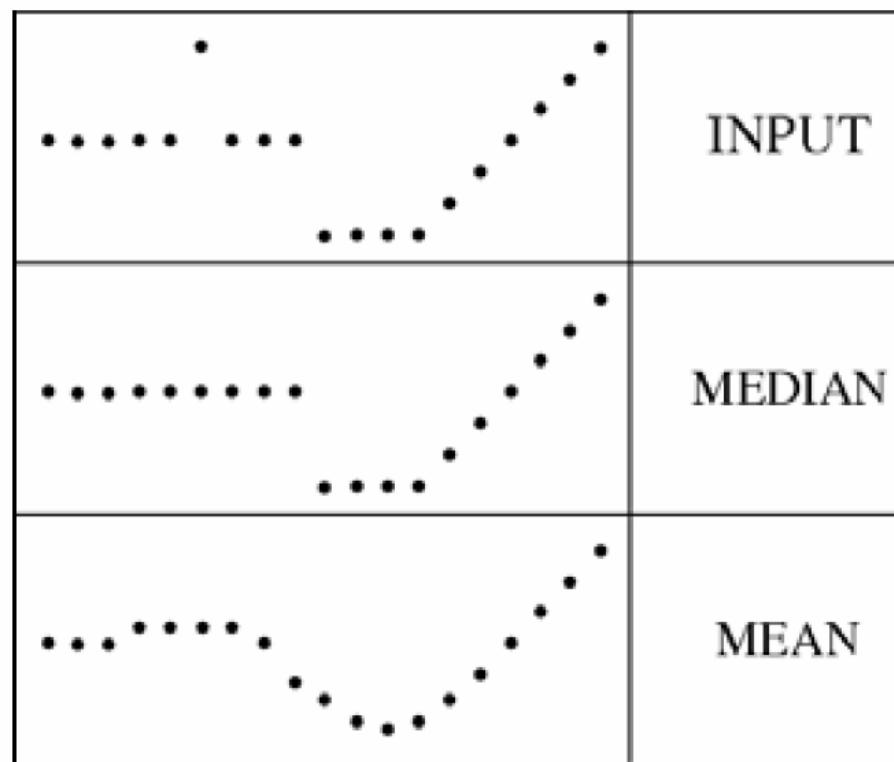
- Is median filtering linear?
- Let's try filtering

é	1	1	1	ù	é	0	0	0	ù
ê				ú	ê				ú
ê	1	1	2	ú	+ ê	0	1	0	ú
ê	2	2	2	ú	ê	0	0	0	ú

Median filter

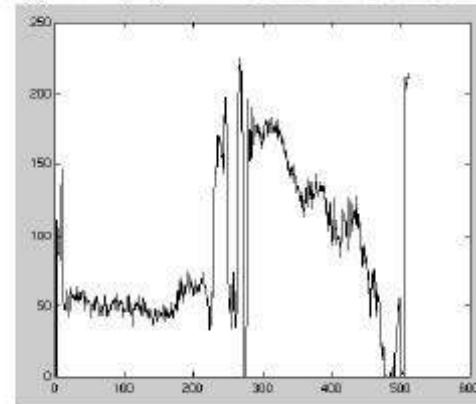
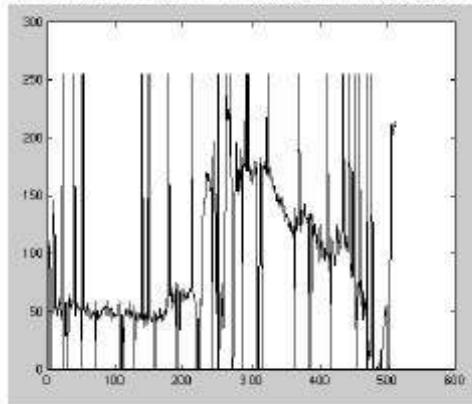
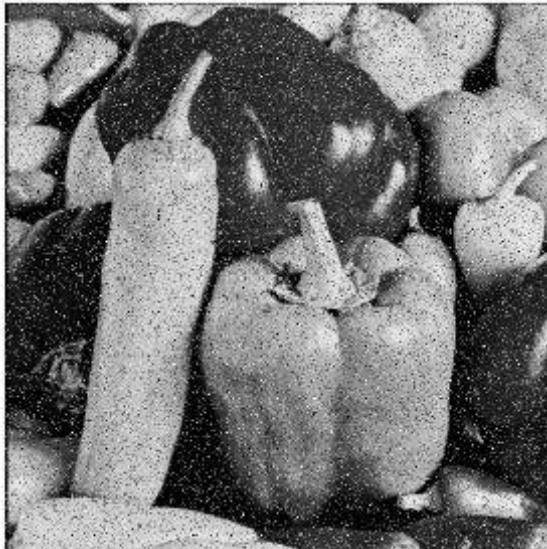
- What advantage does median filtering have over Gaussian filtering?
 - Robustness to outliers

filters have width 5 :



Median filter

Salt-and-pepper noise Median filtered



Gaussian vs. median filtering

3x3



Gaussian

5x5



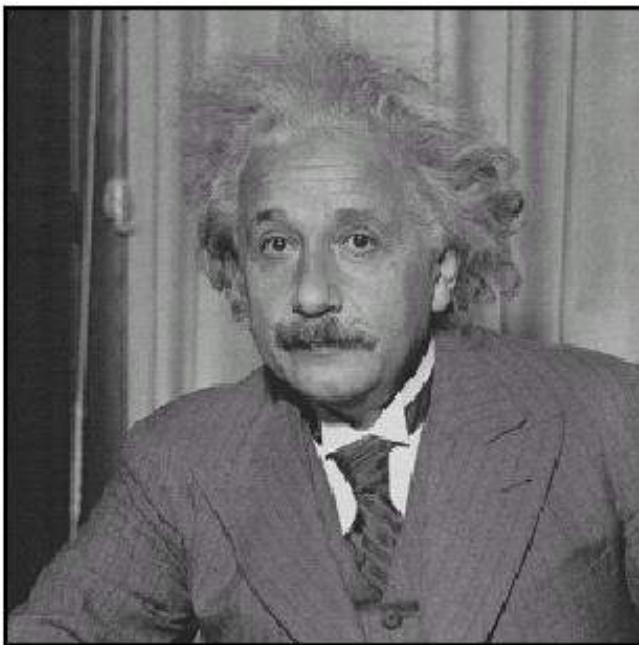
7x7



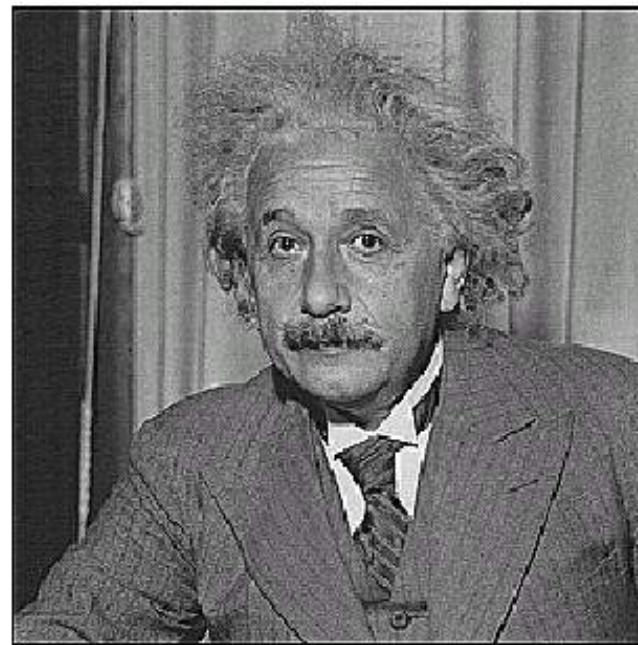
Median



Sharpening revisited



before



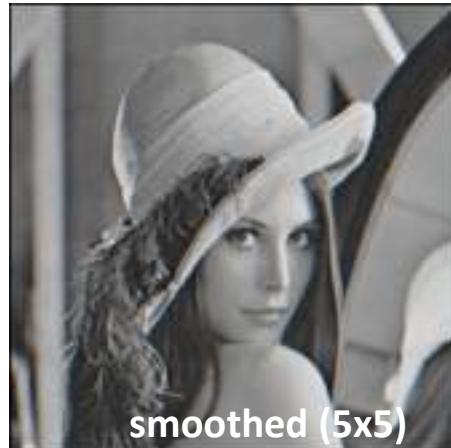
after

Sharpening revisited

- What does blurring take away?



-



=



Let's add it back:



$+ \alpha$

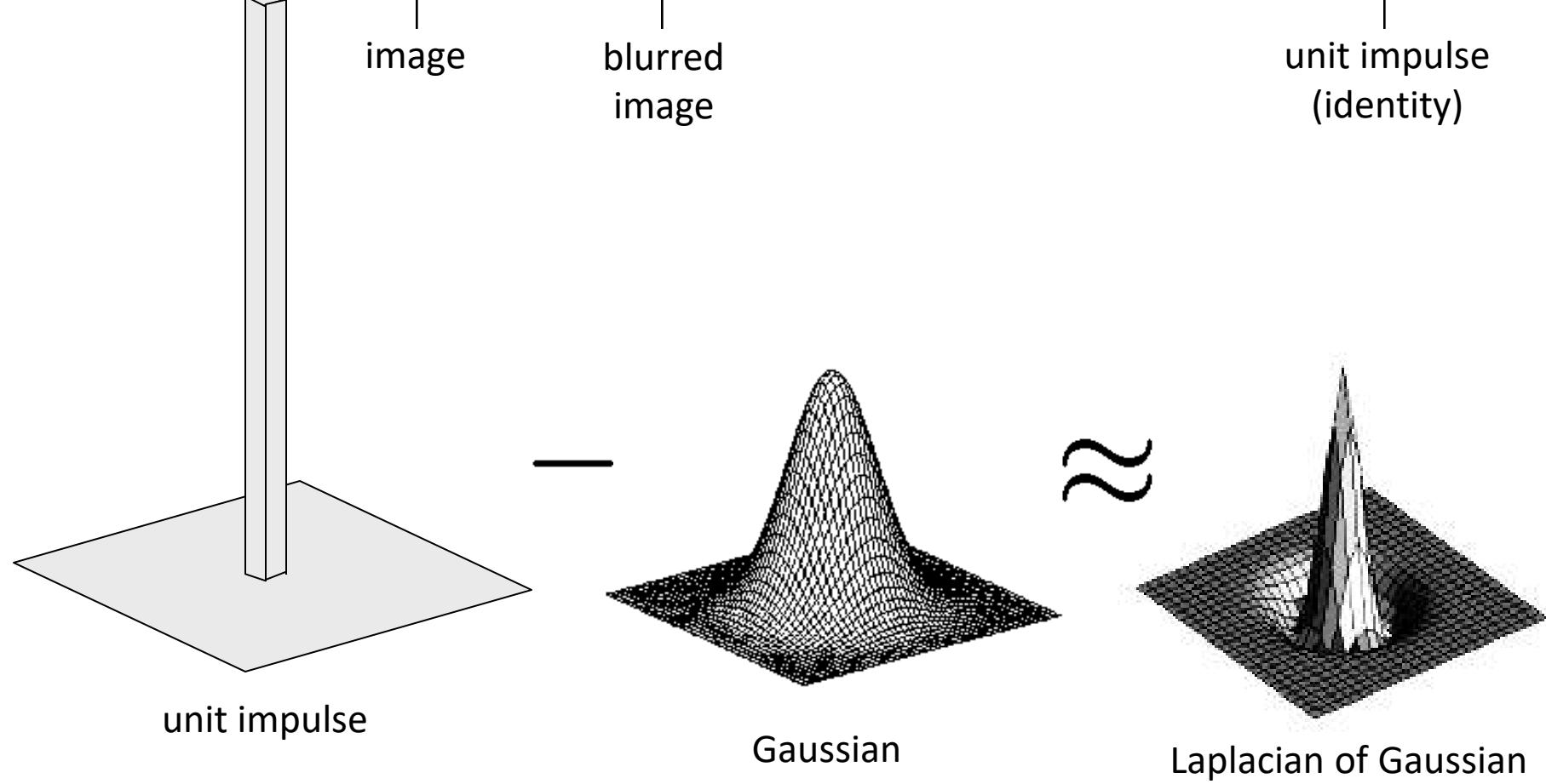


=



Unsharp mask filter

$$f + \alpha(f - f * g) = (1 + \alpha)f - \alpha f * g = f * ((1 + \alpha)e - g)$$



Application: Hybrid Images



A. Oliva, A. Torralba, P.G. Schyns,
Hybrid Images, SIGGRAPH 2006

Changing expression



SIGGRAPH2006

Sad

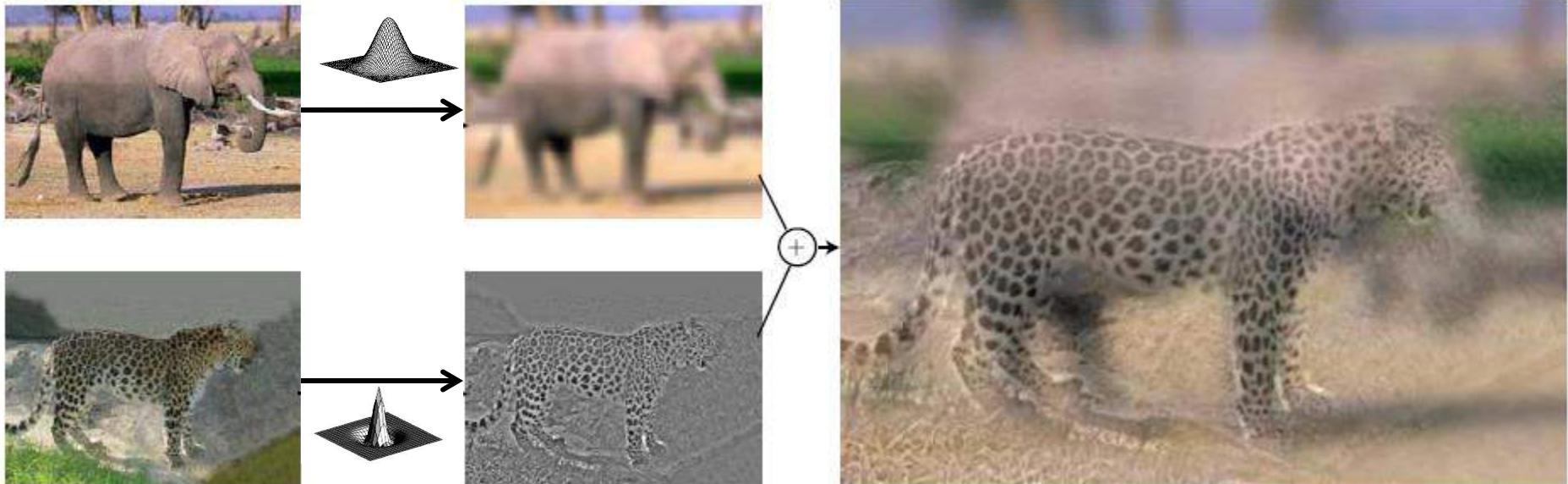


Surprised



Application: Hybrid Images

Gaussian Filter



Laplacian Filter

- A. Oliva, A. Torralba, P.G. Schyns,
[Hybrid Images](#), SIGGRAPH 2006

Thank you: Question?

Computer Vision

Image Filtering in Spatial Domain

Dr. Mrinmoy Ghorai

**Indian Institute of Information Technology
Sri City, Chittoor**



Today's Agenda

- Image Filtering in Spatial Domain

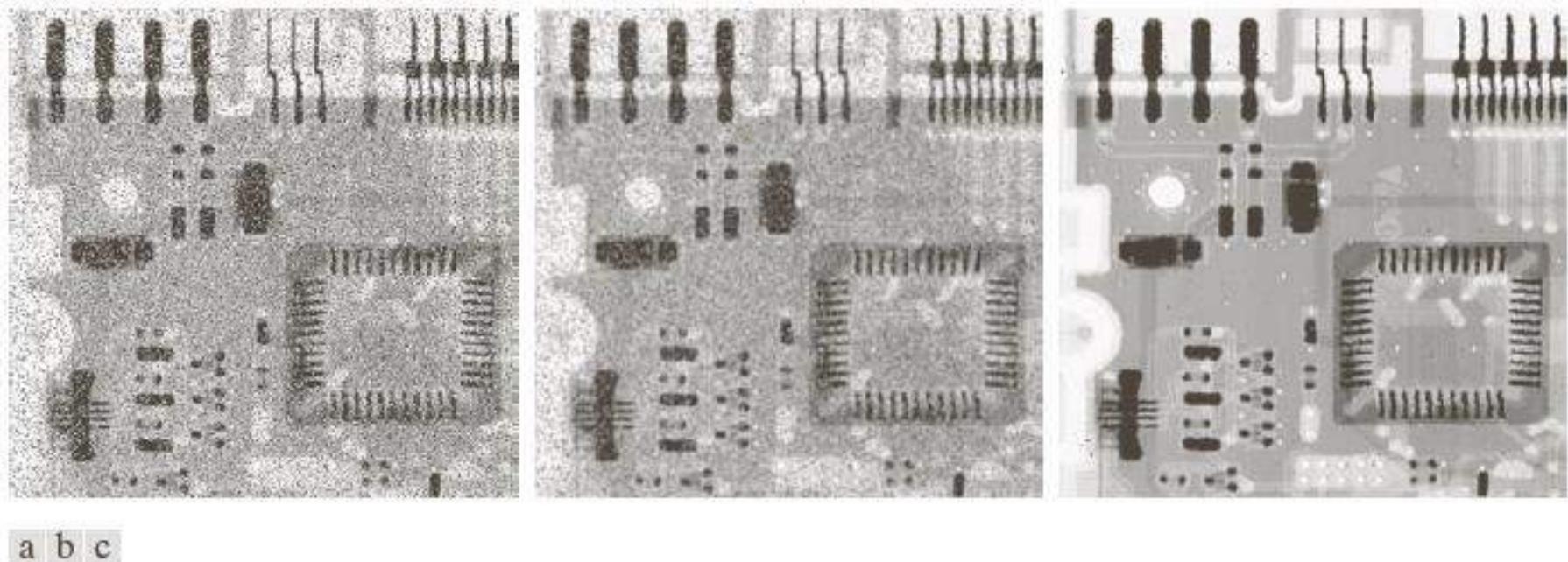
Image Filtering in Spatial Domain

Order-Statistic (Nonlinear) Filters

- ❑ Nonlinear
- ❑ Based on ordering (ranking) the pixels contained in the filter mask
- ❑ Replacing the value of the center pixel with the value determined by the ranking result
- ❑ E.g., median filter, max filter, min filter

Image Filtering in Spatial Domain

Example: Use of Median Filtering for Noise Reduction



a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Image Filtering in Spatial Domain

Sharpening Spatial Filters

- Foundation
- Laplacian Operator
- Unsharp Masking and Highboost Filtering
- Using First-Order Derivatives for Nonlinear Image Sharpening
 - The Gradient

Image Filtering in Spatial Domain

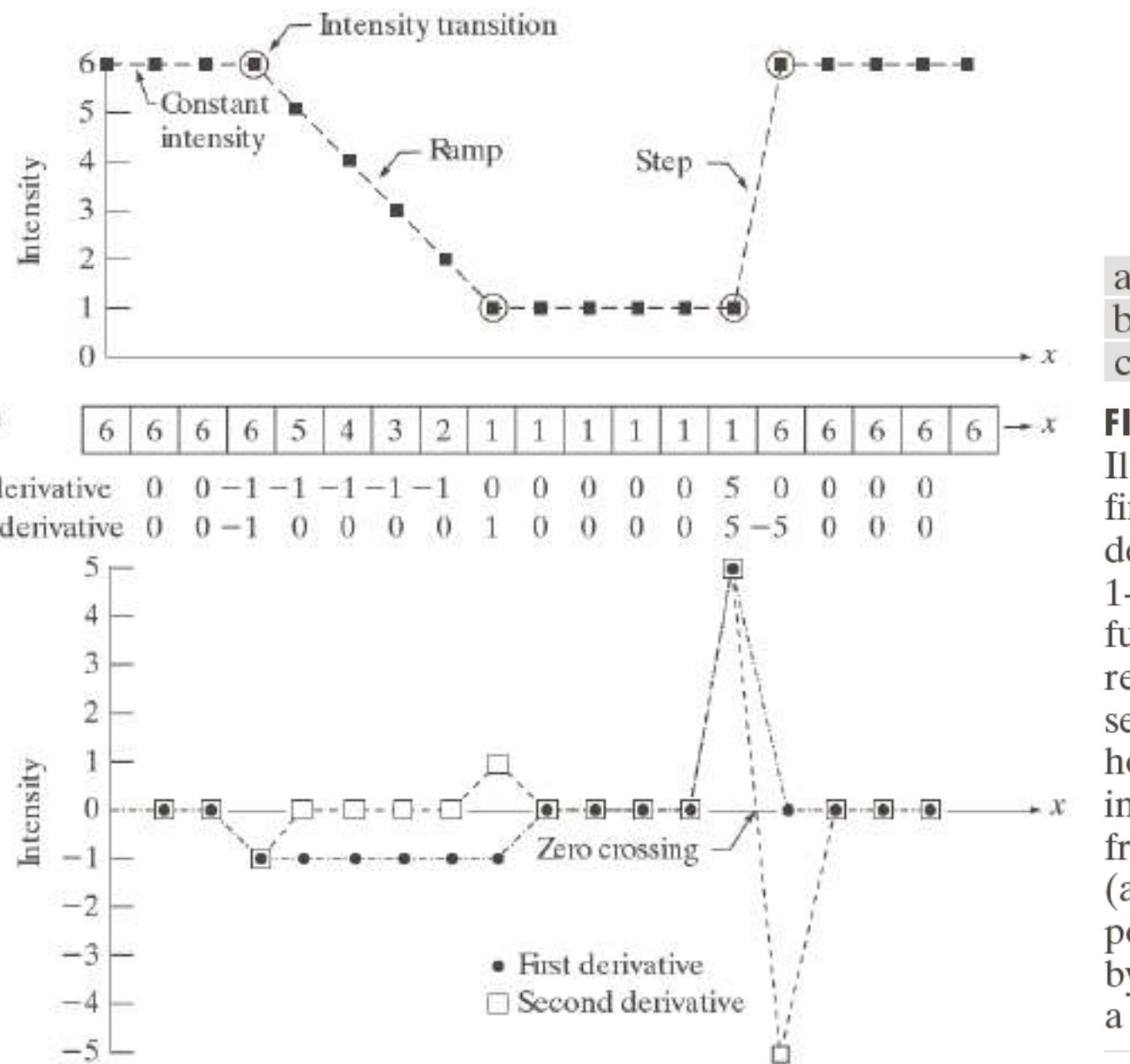
Sharpening Spatial Filters: Foundation

- The first-order derivative of a one-dimensional function $f(x)$ is the difference

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- The second-order derivative of $f(x)$ as the difference

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$



a
b
c

FIGURE 3.36
Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

Image Filtering in Spatial Domain

Sharpening Spatial Filters: Laplacian Operator

The second-order isotropic derivative operator is the Laplacian for a function (image) $f(x, y)$

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

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\begin{aligned} \nabla^2 f = & f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) \\ & - 4f(x, y) \end{aligned}$$

Image Filtering in Spatial Domain

Sharpening Spatial Filters: Laplacian Operator

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

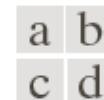


FIGURE 3.37

(a) Filter mask used to implement Eq. (3.6-6).

(b) Mask used to implement an extension of this equation that includes the diagonal terms.

(c) and (d) Two other implementations of the Laplacian found frequently in practice.

Image Filtering in Spatial Domain

Sharpening Spatial Filters: Laplacian Operator

Image sharpening in the way of using the Laplacian:

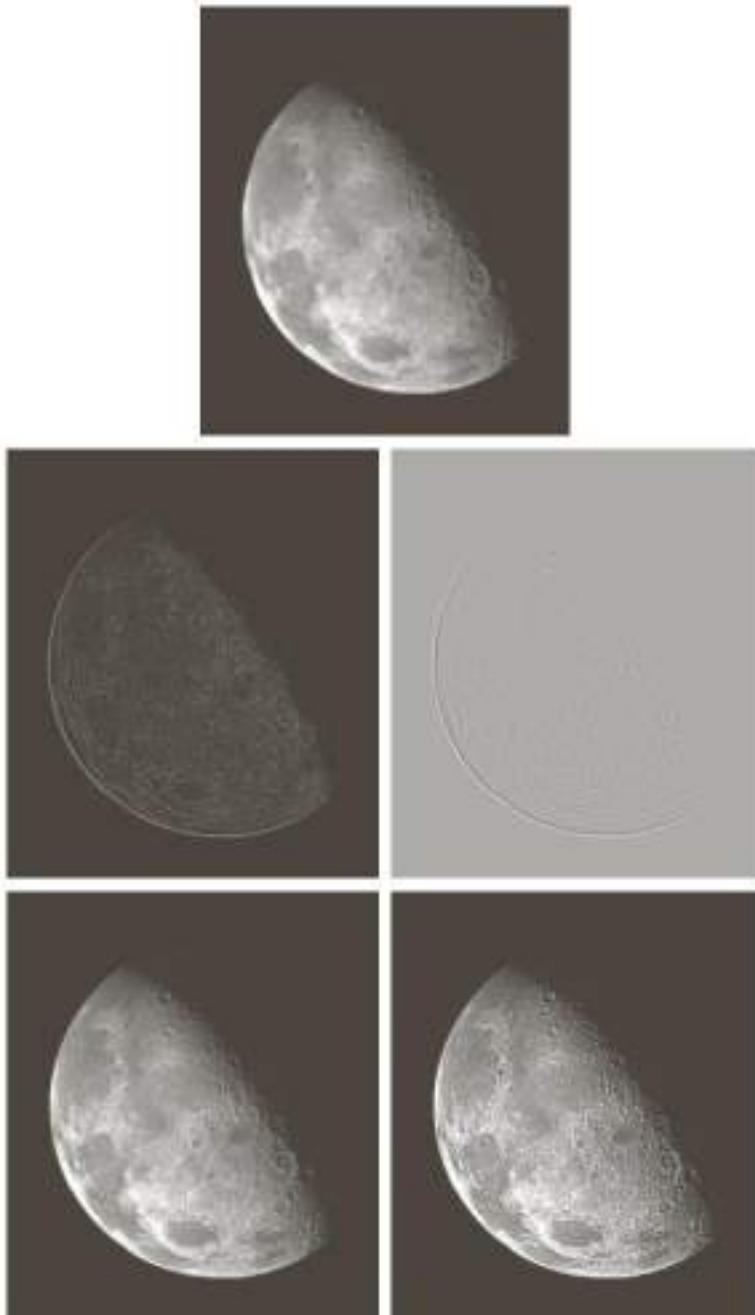
$$g(x, y) = f(x, y) + c \left[\nabla^2 f(x, y) \right]$$

where,

$f(x, y)$ is input image,

$g(x, y)$ is sharpened images,

$c = -1$ if $\nabla^2 f(x, y)$ corresponding to Fig. 3.37(a) or (b)
and $c = 1$ if either of the other two filters is used.



a
b c
d e

FIGURE 3.38

- (a) Blurred image of the North Pole of the moon.
(b) Laplacian without scaling.
(c) Laplacian with scaling. (d) Image sharpened using the mask in Fig. 3.37(a). (e) Result of using the mask in Fig. 3.37(b).
(Original image courtesy of NASA.)

Image Filtering in Spatial Domain

Unsharp Masking and Highboost Filtering

Unsharp masking

Sharpen images consists of subtracting an unsharp (smoothed) version of an image from the original image
e.g., printing and publishing industry

Steps

- I. Blur the original image
- II. Subtract the blurred image from the original
- III. Add the mask to the original

Image Filtering in Spatial Domain

Unsharp Masking and Highboost Filtering

Let $\bar{f}(x, y)$ denote the blurred image, unsharp masking is

$$g_{mask}(x, y) = f(x, y) - \bar{f}(x, y)$$

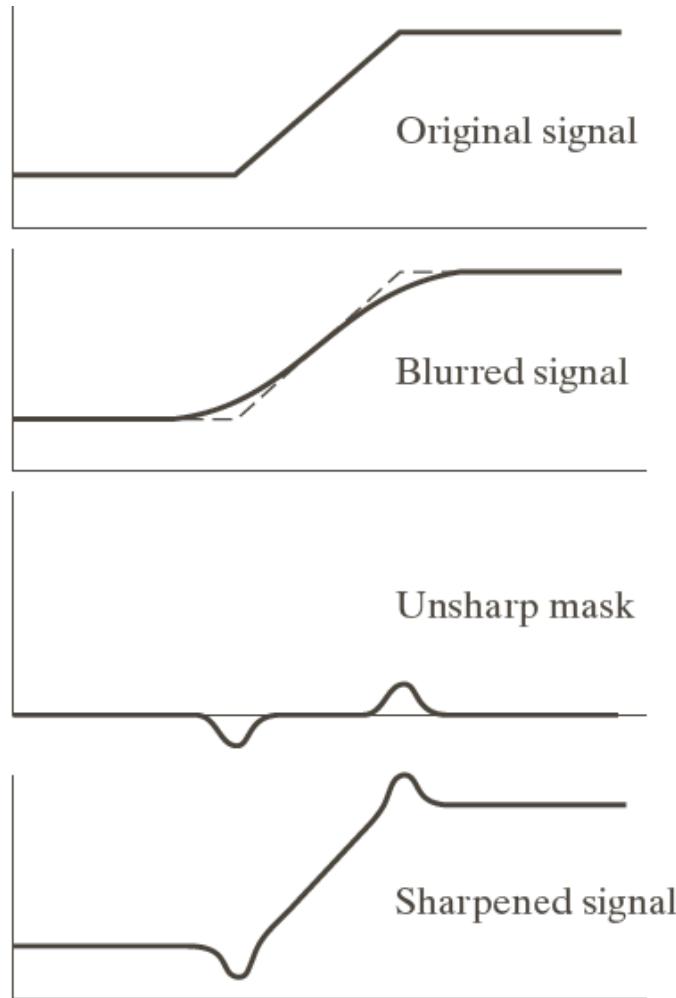
Then add a weighted portion of the mask back to the original

$$g(x, y) = f(x, y) + k * g_{mask}(x, y) \quad k \geq 0$$

when $k > 1$, the process is referred to as highboost filtering.

Image Filtering in Spatial Domain

Unsharp Masking and Highboost Filtering



a
b
c
d

FIGURE 3.39 1-D illustration of the mechanics of unsharp masking.
(a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).

Image Filtering in Spatial Domain

Unsharp Masking and Highboost Filtering: Example



a
b
c
d
e

FIGURE 3.40

- (a) Original image.
- (b) Result of blurring with a Gaussian filter.
- (c) Unsharp mask.
- (d) Result of using unsharp masking.
- (e) Result of using highboost filtering.

Image Filtering in Spatial Domain

Image Sharpening based on First-Order Derivatives

For function $f(x, y)$, the gradient of f at coordinates (x, y) is defined as

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The *magnitude* of vector ∇f , denoted as $M(x, y)$

Gradient Image $M(x, y) = \text{mag}(\nabla f) = \sqrt{{g_x}^2 + {g_y}^2}$

Image Filtering in Spatial Domain

Image Sharpening based on First-Order Derivatives

The *magnitude* of vector ∇f , denoted as $M(x, y)$

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{{g_x}^2 + {g_y}^2}$$

$$M(x, y) \approx |g_x| + |g_y|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$M(x, y) = |z_8 - z_5| + |z_6 - z_5|$$

Image Sharpening based on First-Order Derivatives

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-1	0	0	-1
0	1	1	0

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

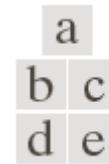


FIGURE 3.41

A 3×3 region of an image (the z s are intensity values).

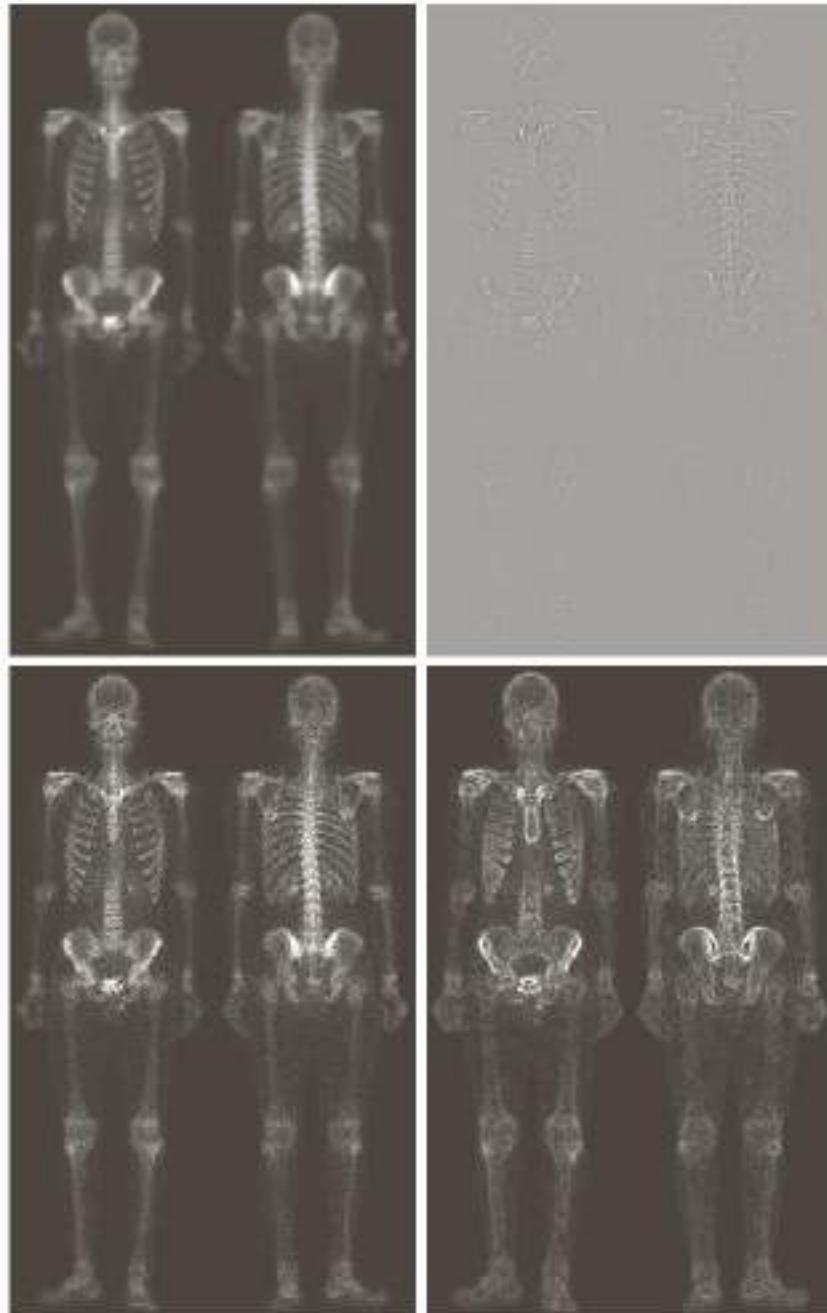
(b)–(c) Roberts cross gradient operators.

(d)–(e) Sobel operators. All the mask coefficients sum to zero, as expected of a derivative operator.

Example

Combining
Spatial
Enhancement
Methods

Goal:
Enhance the
image by
sharpening it
and by bringing
out more of
the skeletal detail



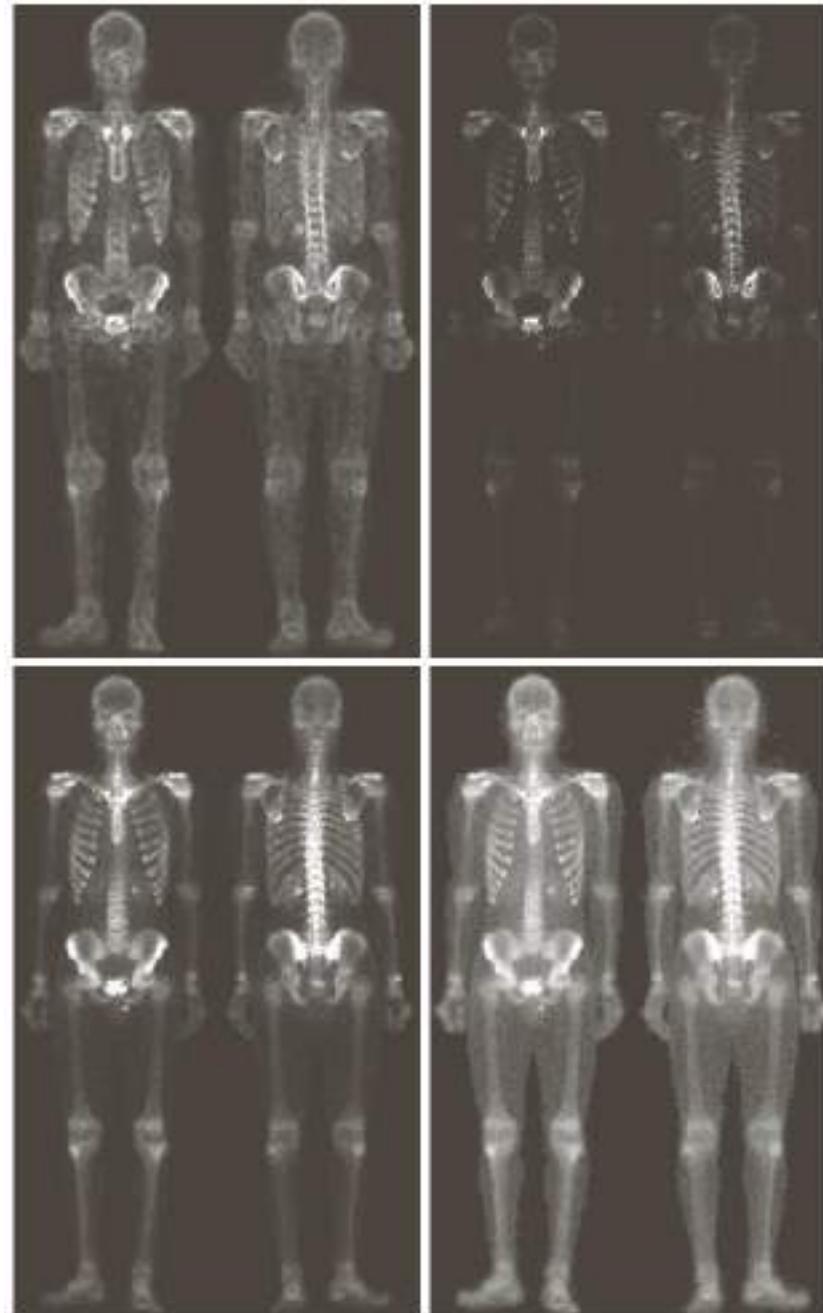
a b
c d

FIGURE 3.43
(a) Image of whole body bone scan.
(b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b).
(d) Sobel gradient of (a).

Example

Combining Spatial Enhancement Methods

Goal:
Enhance the
image by
sharpening it
and by bringing
out more of
the skeletal detail



e f
g h

FIGURE 3.43
(Continued)
(e) Sobel image smoothed with a 5×5 averaging filter. (f) Mask image formed by the product of (c) and (e).
(g) Sharpened image obtained by the sum of (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)

Thank you: Question?

Computer Vision

Image Filtering in Frequency Domain

Dr. Mrinmoy Ghorai

Indian Institute of Information Technology Sri City



Introduction

□ Fourier Series

Any periodic function can be expressed as the sum of sines and /or cosines of different frequencies, each multiplied by a different coefficients.

□ Fourier Transform

Any function that is not periodic can be expressed as the integral of sines and /or cosines multiplied by a weighing function.

Jean Baptiste Joseph Fourier, French mathematician and physicist (03/21/1768-05/16/1830)

Example of Fourier Series

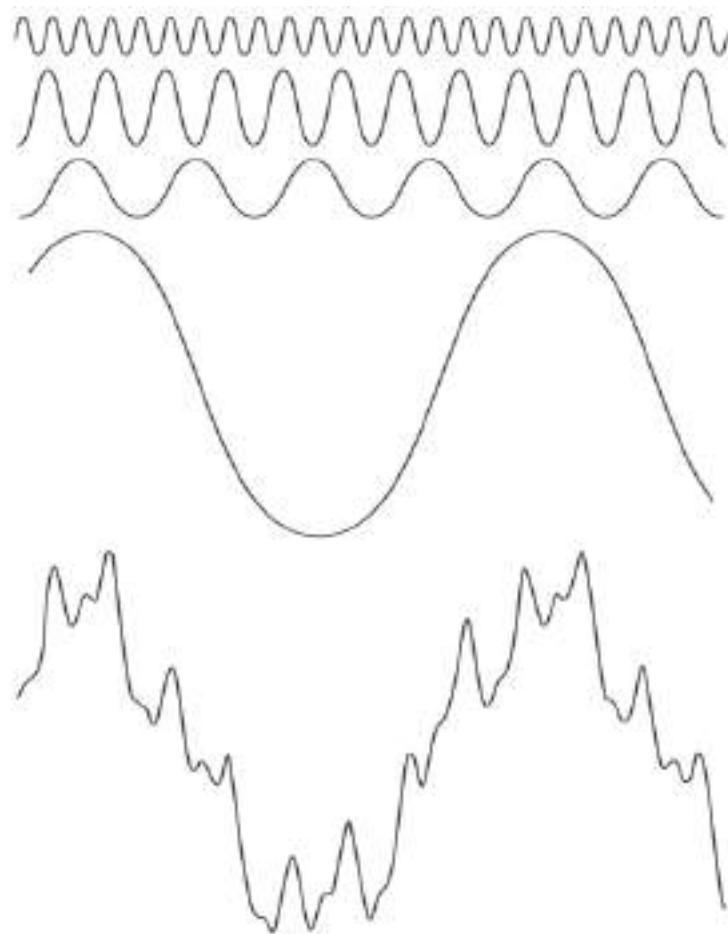


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

Preliminary Concepts

$j = \sqrt{-1}$, a complex number

$$C = R + jI$$

the conjugate

$$C^* = R - jI$$

$$|C| = \sqrt{R^2 + I^2} \text{ and } \theta = \arctan(I / R)$$

$$C = |C|(\cos \theta + j \sin \theta)$$

Using Euler's formula,

$$C = |C| e^{j\theta}$$

Fourier Series

A function $f(t)$ of a continuous variable t that is periodic with period, T , can be expressed as the sum of sines and cosines multiplied by appropriate coefficients

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j \frac{2\pi n}{T} t}$$

where

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-j \frac{2\pi n}{T} t} dt \quad \text{for } n = 0, \pm 1, \pm 2, \dots$$

1-D Fourier Transform: Continuous Variable

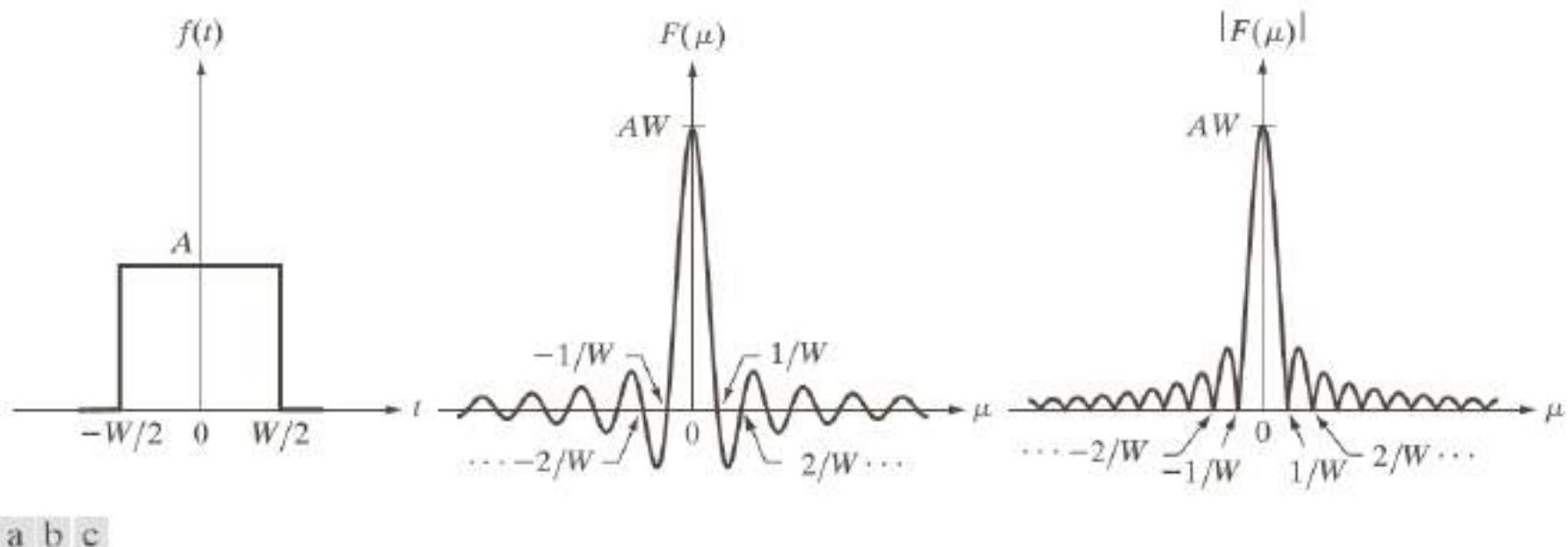
The *Fourier Transform* of a continuous function $f(t)$

$$F(\mu) = \mathfrak{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t) e^{-j2\pi\mu t} dt$$

The *Inverse Fourier Transform* of $F(\mu)$

$$f(t) = \mathfrak{F}^{-1}\{F(\mu)\} = \int_{-\infty}^{\infty} F(\mu) e^{j2\pi\mu t} d\mu$$

1-D Fourier Transform: Continuous Variable



a b c

FIGURE 4.4 (a) A simple function; (b) its Fourier transform; and (c) the spectrum. All functions extend to infinity in both directions.

1-D Discrete Fourier Transform

$$F(\mu) = \sum_{x=0}^{M-1} f(x) e^{-j2\pi\mu x/M}, \quad \mu = 0, 1, \dots, M-1$$

$$f(x) = \frac{1}{M} \sum_{\mu=0}^{M-1} F(\mu) e^{j2\pi\mu x/M}, \quad x = 0, 1, 2, \dots, M-1$$

1. The domain (values of μ) over which the value of $F(\mu)$ range is called the **Frequency Domain**.
2. Each of the M terms of $F(\mu)$ is called a **Frequency Component** of the transform.

2-D Fourier Transform: Continuous Variable

$$F(\mu, \nu) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) e^{-j2\pi(\mu t + \nu z)} dt dz$$

and

$$f(t, z) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\mu, \nu) e^{j2\pi(\mu t + \nu z)} d\mu d\nu$$

2-D Discrete Fourier Transform and Its Inverse

DFT:

$$F(\mu, \nu) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(\mu x/M + \nu y/N)}$$

$\mu = 0, 1, 2, \dots, M-1; \nu = 0, 1, 2, \dots, N-1;$

$f(x, y)$ is a digital image of size $M \times N$.

IDFT:

$$f(x, y) = \frac{1}{MN} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} F(\mu, \nu) e^{j2\pi(\mu x/M + \nu y/N)}$$

Properties of 2-D DFT

Relationships between Samples in the Frequency and Spatial Domains

Let ΔT and ΔZ denote the separations between samples, then the separations between the corresponding discrete, frequency domain variables are given by

$$\Delta\mu = \frac{1}{M\Delta T}$$

and $\Delta\nu = \frac{1}{N\Delta Z}$

Properties of 2-D DFT

Translation and Rotation

$$f(x, y)e^{j2\pi(\mu_0x/M + \nu_0y/N)} \Leftrightarrow F(\mu - \mu_0, \nu - \nu_0)$$

and

$$f(x - x_0, y - y_0) \Leftrightarrow F(\mu, \nu)e^{-j2\pi(\mu x_0/M + \nu y_0/N)}$$

Using the polar coordinates

$$x = r \cos \theta \quad y = r \sin \theta \quad \mu = \omega \cos \varphi \quad \nu = \omega \sin \varphi$$

results in the following transform pair:

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$$

Properties of 2-D DFT

Periodicity

2-D Fourier transform and its inverse are infinitely periodic

$$F(\mu, \nu) = F(\mu + k_1 M, \nu) = F(\mu, \nu + k_2 N) = F(\mu + k_1 M, \nu + k_2 N)$$

$$f(x, y) = f(x + k_1 M, y) = f(x, y + k_2 N) = f(x + k_1 M, y + k_2 N)$$

$$f(x)e^{j2\pi(\mu_0 x/M)} \Leftrightarrow F(\mu - \mu_0)$$

$$\mu_0 = M/2, \quad f(x)(-1)^x \Leftrightarrow F(\mu - M/2)$$

$$f(x, y)(-1)^{x+y} \Leftrightarrow F(\mu - M/2, \nu - N/2)$$

Properties of 2-D DFT

Symmetry

	Spatial Domain [†]		Frequency Domain [†]
1)	$f(x, y)$ real	\Leftrightarrow	$F^*(u, v) = F(-u, -v)$
2)	$f(x, y)$ imaginary	\Leftrightarrow	$F^*(-u, -v) = -F(u, v)$
3)	$f(x, y)$ real	\Leftrightarrow	$R(u, v)$ even; $I(u, v)$ odd
4)	$f(x, y)$ imaginary	\Leftrightarrow	$R(u, v)$ odd; $I(u, v)$ even
5)	$f(-x, -y)$ real	\Leftrightarrow	$F^*(u, v)$ complex
6)	$f(-x, -y)$ complex	\Leftrightarrow	$F(-u, -v)$ complex
7)	$f^*(x, y)$ complex	\Leftrightarrow	$F^*(-u - v)$ complex
8)	$f(x, y)$ real and even	\Leftrightarrow	$F(u, v)$ real and even
9)	$f(x, y)$ real and odd	\Leftrightarrow	$F(u, v)$ imaginary and odd
10)	$f(x, y)$ imaginary and even	\Leftrightarrow	$F(u, v)$ imaginary and even
11)	$f(x, y)$ imaginary and odd	\Leftrightarrow	$F(u, v)$ real and odd
12)	$f(x, y)$ complex and even	\Leftrightarrow	$F(u, v)$ complex and even
13)	$f(x, y)$ complex and odd	\Leftrightarrow	$F(u, v)$ complex and odd

TABLE 4.1 Some symmetry properties of the 2-D DFT and its inverse. $R(u, v)$ and $I(u, v)$ are the real and imaginary parts of $F(u, v)$, respectively. The term *complex* indicates that a function has nonzero real and imaginary parts.

[†]Recall that x, y, u , and v are *discrete* (integer) variables, with x and u in the range $[0, M - 1]$, and y , and v in the range $[0, N - 1]$. To say that a complex function is *even* means that its real *and* imaginary parts are even, and similarly for an odd complex function.

Properties of 2-D DFT

Fourier Spectrum and Phase Angle

2-D DFT in polar form

$$F(u, v) = |F(u, v)| e^{j\phi(u, v)}$$

Fourier spectrum

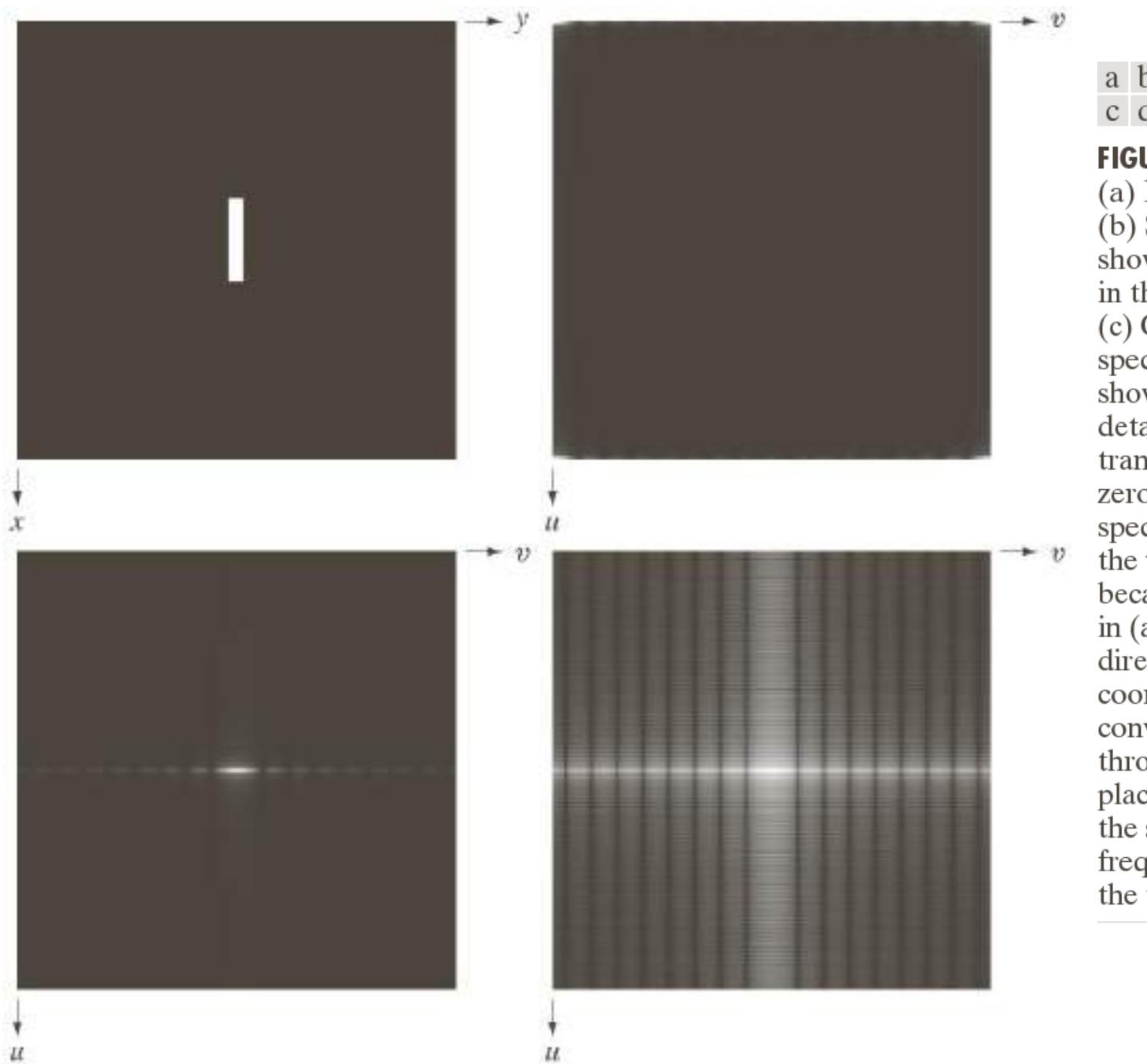
$$|F(u, v)| = \left[R^2(u, v) + I^2(u, v) \right]^{1/2}$$

Power spectrum

$$P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v)$$

Phase angle

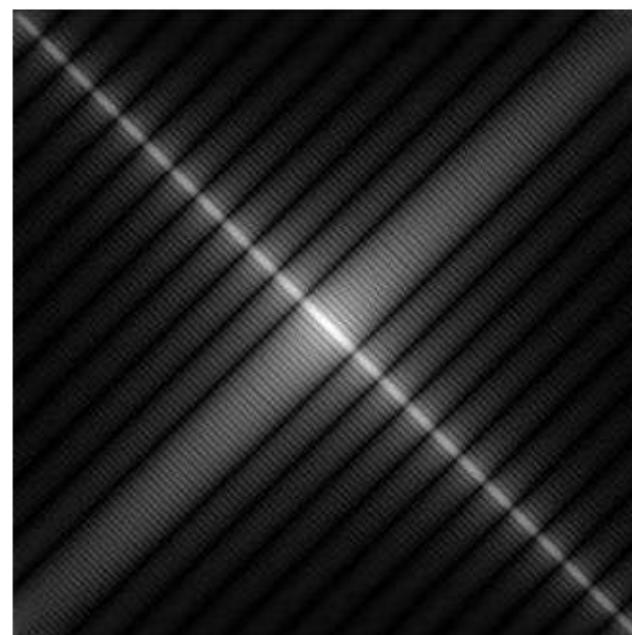
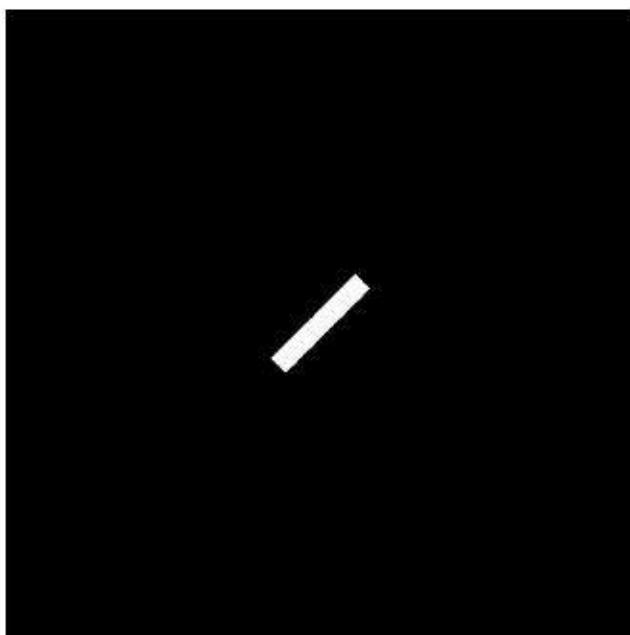
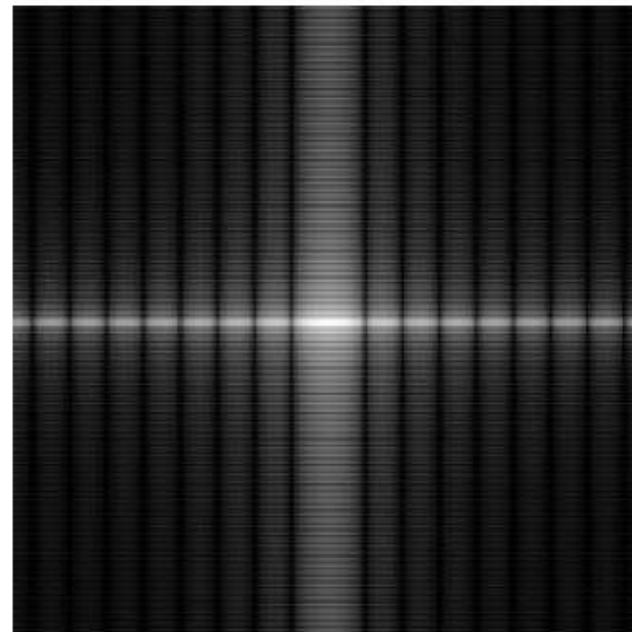
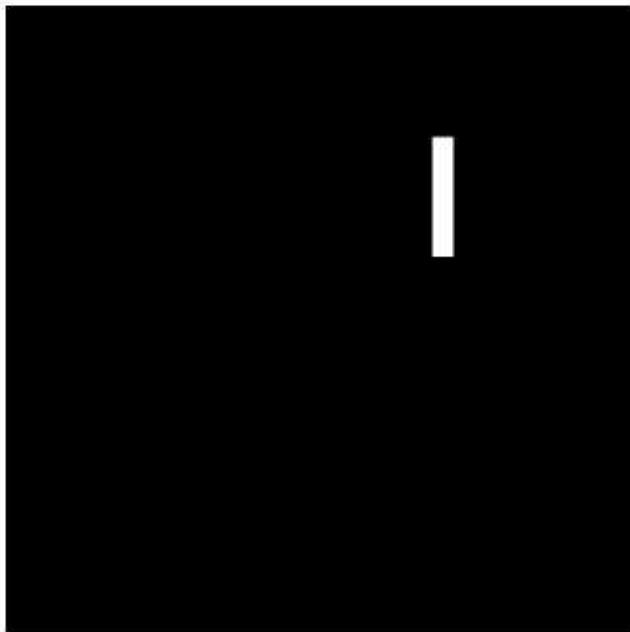
$$\phi(u, v) = \arctan \left[\frac{I(u, v)}{R(u, v)} \right]$$



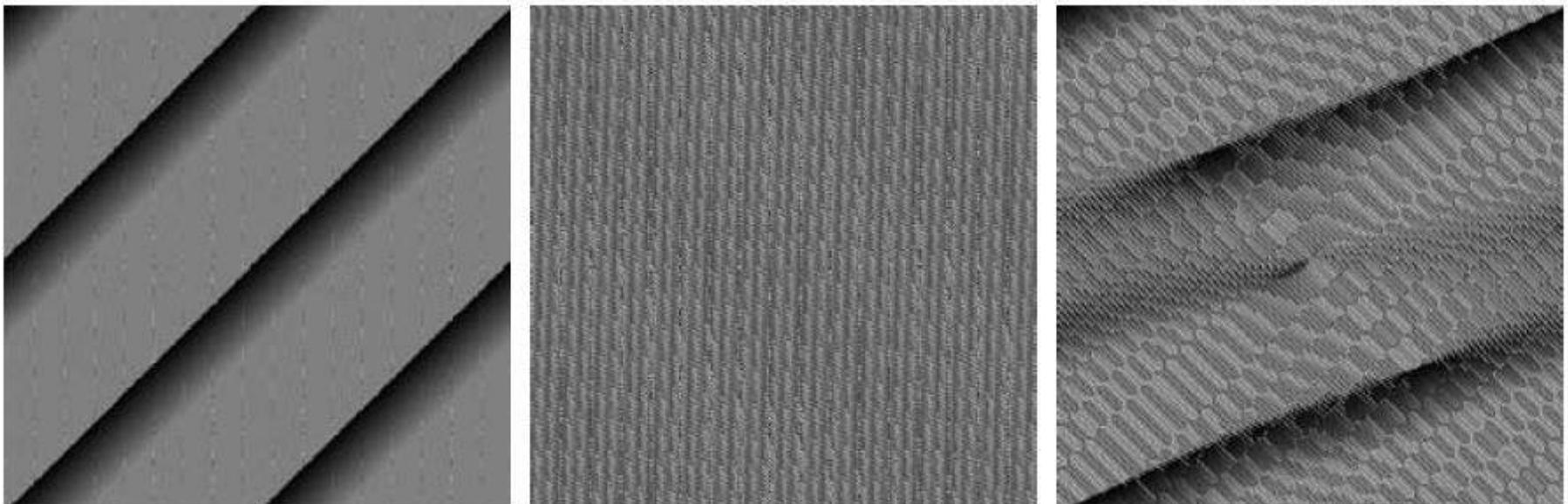
a	b
c	d

FIGURE 4.24

- (a) Image.
- (b) Spectrum showing bright spots in the four corners.
- (c) Centered spectrum.
- (d) Result showing increased detail after a log transformation. The zero crossings of the spectrum are closer in the vertical direction because the rectangle in (a) is longer in that direction. The coordinate convention used throughout the book places the origin of the spatial and frequency domains at the top left.



Phase Angle: Example



a b c

FIGURE 4.26 Phase angle array corresponding (a) to the image of the centered rectangle in Fig. 4.24(a), (b) to the translated image in Fig. 4.25(a), and (c) to the rotated image in Fig. 4.25(c).

Phase Angle and The Reconstructed: Example

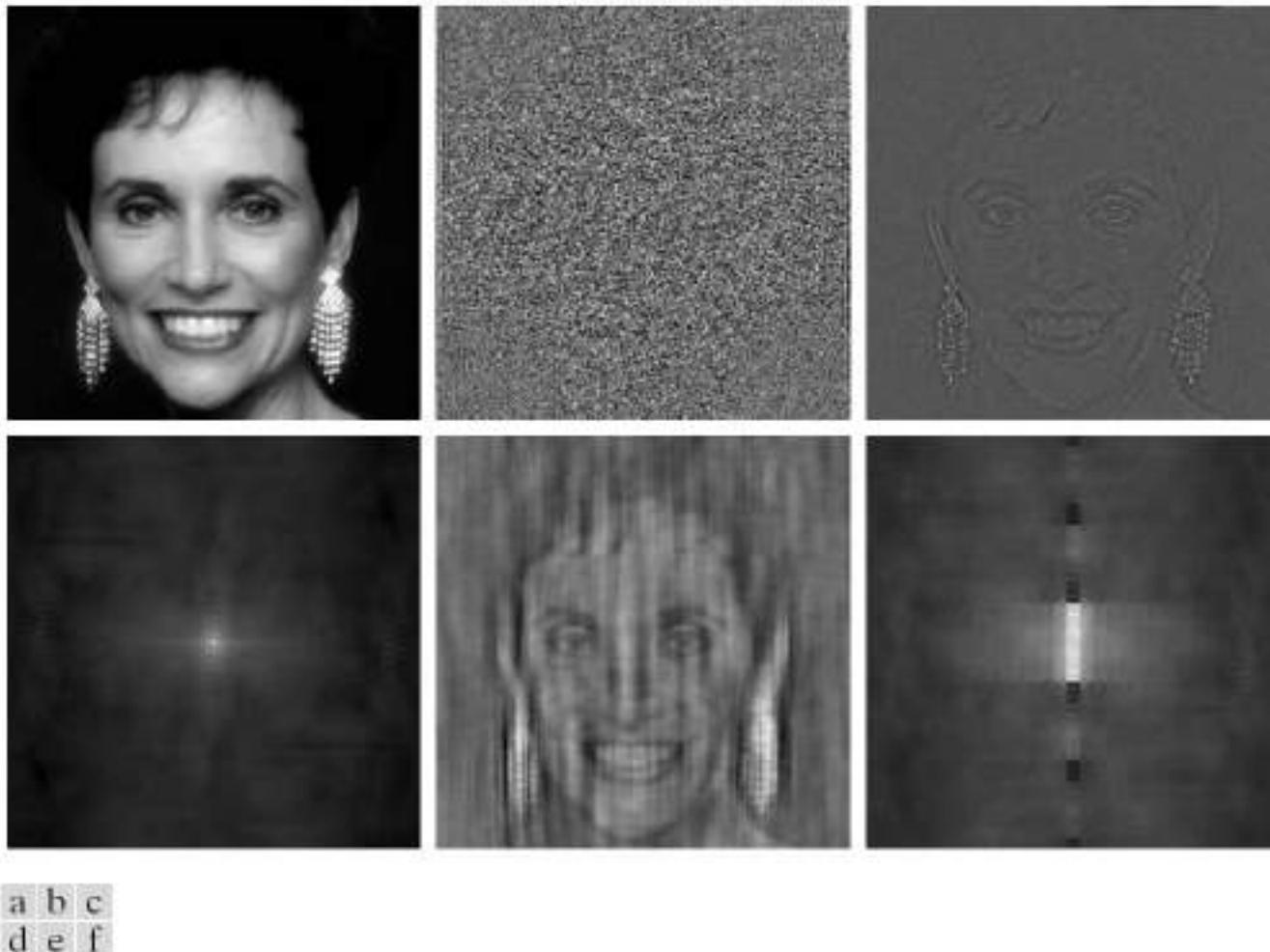


FIGURE 4.27 (a) Woman. (b) Phase angle. (c) Woman reconstructed using only the phase angle. (d) Woman reconstructed using only the spectrum. (e) Reconstruction using the phase angle corresponding to the woman and the spectrum corresponding to the rectangle in Fig. 4.24(a). (f) Reconstruction using the phase of the rectangle and the spectrum of the woman.

Properties of 2-D DFT: Fourier Spectrum and Phase Angle

2-D DFT in polar form

$$F(u, v) = |F(u, v)| e^{j\phi(u, v)}$$

Fourier spectrum

$$|F(u, v)| = \left[R^2(u, v) + I^2(u, v) \right]^{1/2}$$

Power spectrum

$$P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v)$$

Phase angle

$$\phi(u, v) = \arctan \left[\frac{I(u, v)}{R(u, v)} \right]$$

Filtering in Frequency Domain: Steps

1. Multiply the input image by $-1^{(x+y)}$ to center the transform
2. Compute $F(u, v)$, the DFT of the image
3. Multiply $F(u, v)$ by a filter function $H(u, v)$
4. Compute the inverse DFT of the result in (3)
5. Obtain the real part of the result in (4)
6. Multiply the result in (5) by $-1^{(x+y)}$

Filtering in Frequency Domain

- Let $f(x, y)$ is the input image and $F(u, v)$ its Fourier transform. Then the Fourier transform the output image is given by

$$G(u, v) = H(u, v)F(u, v)$$

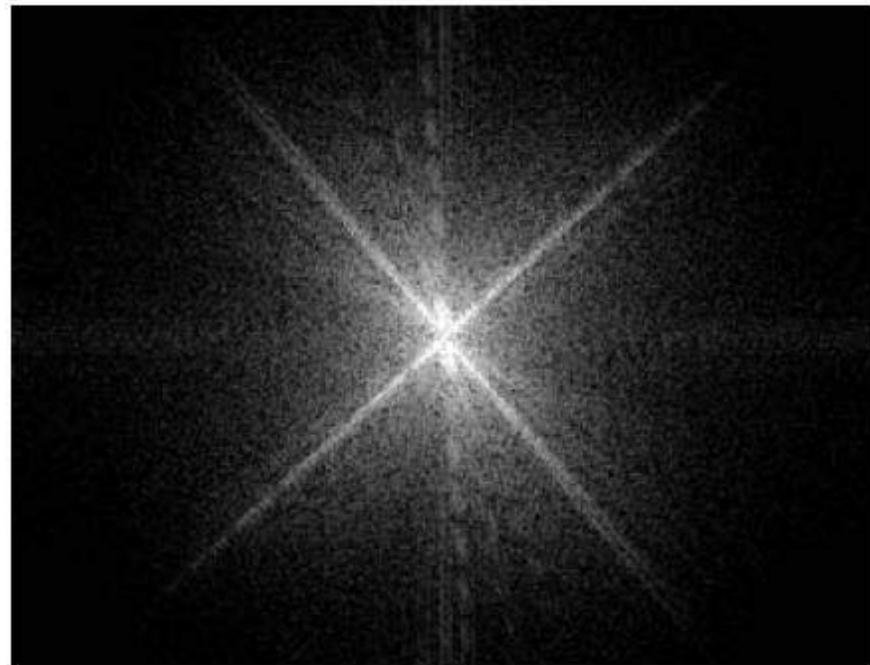
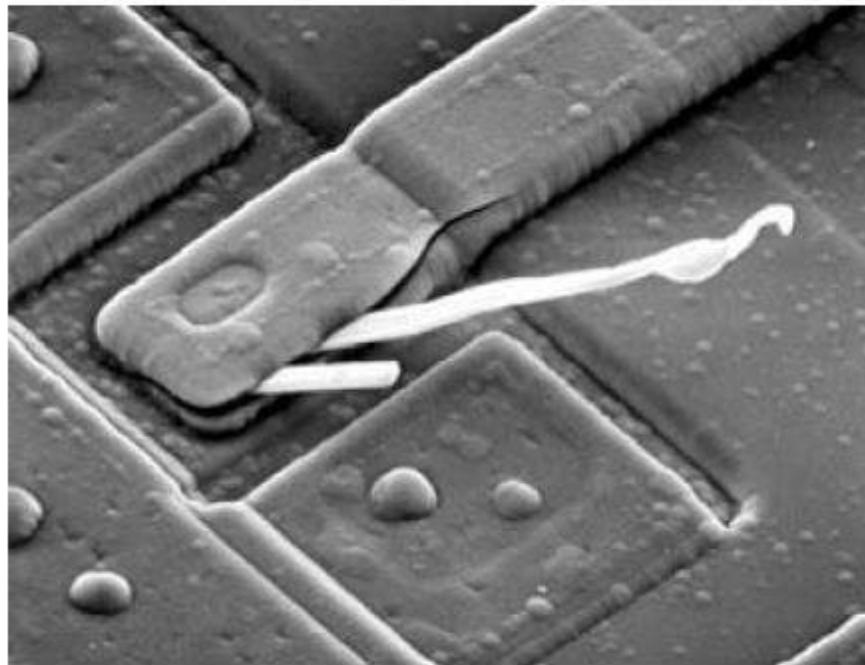
- Computing the inverse transform to obtain the processed result

$$g(x, y) = \mathcal{I}^{-1}\{H(u, v)F(u, v)\}$$

$F(u, v)$ is the DFT of the input image

$H(u, v)$ is a filter function.

Filtering in Frequency Domain

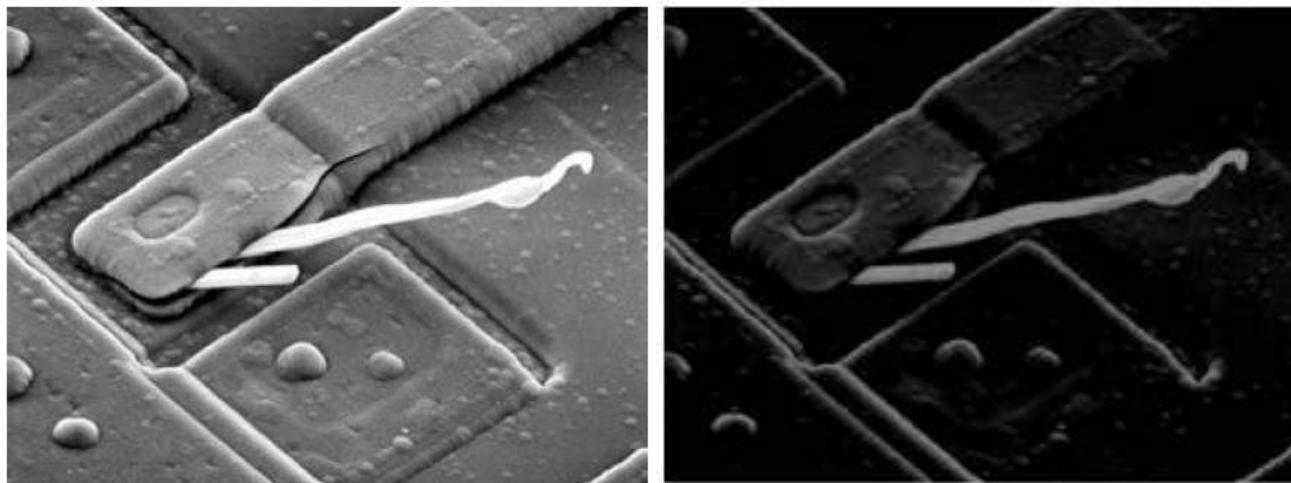


a | b

FIGURE 4.29 (a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

Filtering in Frequency Domain: Notch Filter

In a filter $H(u,v)$ that is 0 at the center of the transform and 1 elsewhere, what's the output image?



Filtering in Frequency Domain: Properties

- Low frequencies in the Fourier transform are responsible for the general gray-level appearance of an image over smooth regions.
- High frequencies are responsible for detail, such as edges and noise.
- A filter that attenuates **high frequencies** while “passing” the **low frequencies** is called *lowpass filter*.
- A filter that attenuates **low frequencies** while “passing” the **high frequencies** is called *highpass filter*.

Filtering in Frequency Domain

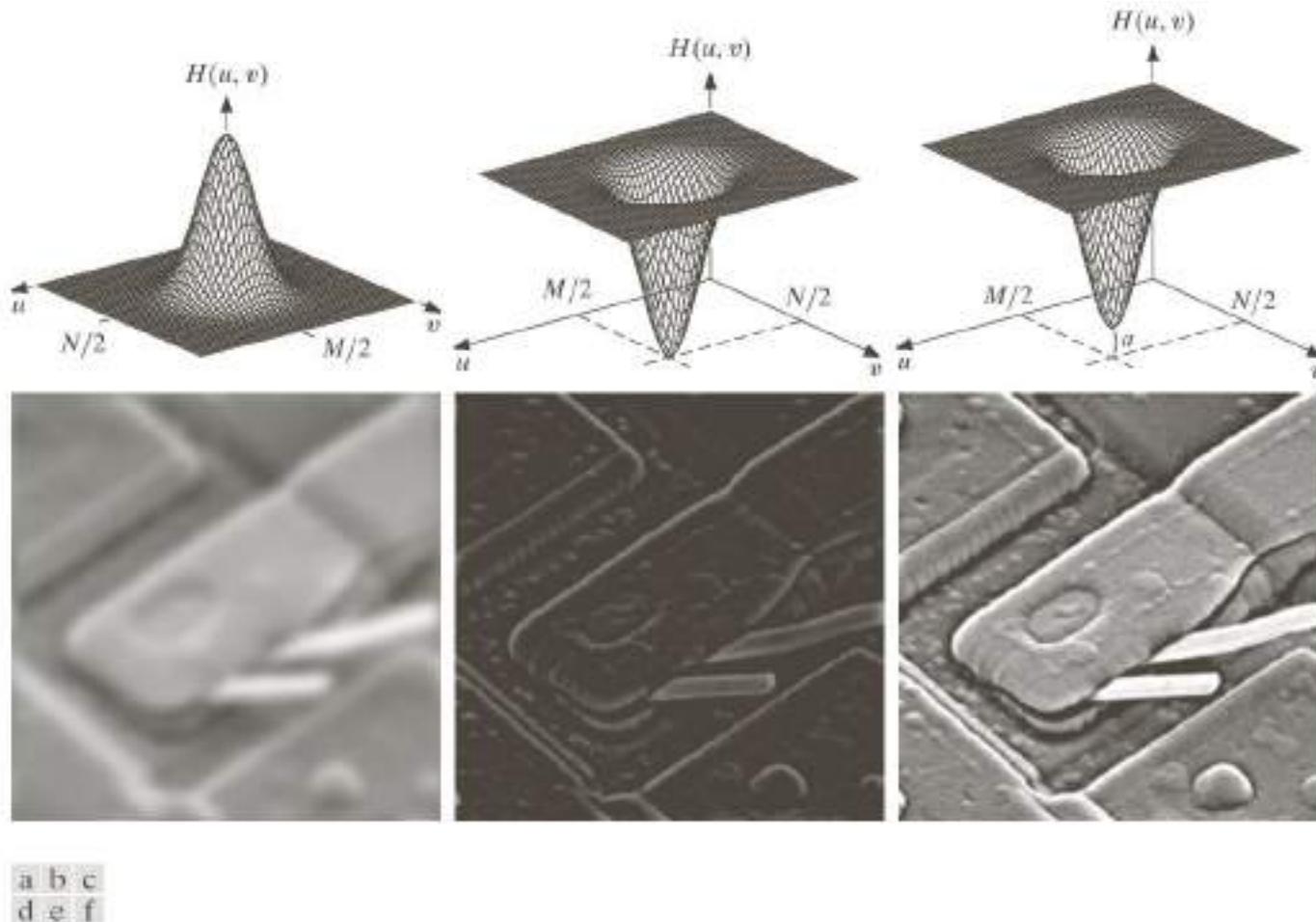


FIGURE 4.31 Top row: frequency domain filters. Bottom row: corresponding filtered images obtained using Eq. (4.7-1). We used $a = 0.85$ in (c) to obtain (f) (the height of the filter itself is 1). Compare (f) with Fig. 4.29(a).

2-D Convolution Theorem

1-D convolution

$$f(x) \star h(x) = \sum_{m=0}^{M-1} f(m)h(x-m)$$

2-D convolution

$$f(x, y) \star h(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x-m, y-n)$$

$$x = 0, 1, 2, \dots, M-1; y = 0, 1, 2, \dots, N-1.$$

$$f(x, y) \star h(x, y) \Leftrightarrow F(u, v)H(u, v)$$

$$f(x, y)h(x, y) \Leftrightarrow F(u, v) \star H(u, v)$$

1-D Impulses and the Shifting Property: **Continuous**

A *unit impulse* of a continuous variable t located at $t=0$, denoted $\delta(t)$, defined as

$$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases}$$

and is constrained also to satisfy the identity

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

The *sifting property* $\int_{-\infty}^{\infty} f(t) \delta(t - t_0) dt = f(t_0)$

$$\int_{-\infty}^{\infty} f(t) \delta(t) dt = f(0)$$

1- D Impulses and the Shifting Property: **Discrete**

A *unit impulse* of a discrete variable x located at $x=0$, denoted $\delta(x)$, defined as

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}$$

and is constrained also to satisfy the identity

$$\sum_{x=-\infty}^{\infty} \delta(x) = 1$$

The *sifting property*

$$\sum_{x=-\infty}^{\infty} f(x)\delta(x - x_0) = f(x_0)$$

$$\sum_{x=-\infty}^{\infty} f(x)\delta(x) = f(0)$$

2-D Impulses and the Shifting Property: Continuous

The impulse $\delta(t, z)$,

$$\delta(t, z) = \begin{cases} \infty & \text{if } t = z = 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t, z) dt dz = 1$$

The sifting property

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) \delta(t, z) dt dz = f(0, 0)$$

and

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) \delta(t - t_0, z - z_0) dt dz = f(t_0, z_0)$$

2-D Impulses and the Shifting Property: Discrete

The impulse $\delta(x, y)$,

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y = 0 \\ 0 & \text{otherwise} \end{cases}$$

The sifting property

$$\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y) \delta(x, y) = f(0, 0)$$

and

$$\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y) \delta(x - x_0, y - y_0) = f(x_0, y_0)$$

2-D Impulses and the Shifting Property: Discrete

For a unit impulse located at origin (0,0),

$$\sum_{x=0}^{m-1} \sum_{y=0}^{n-1} f(x, y) \delta(x, y) = f(0, 0)$$

Fourier transform of a unit impulse located at origin

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \delta(x, y) e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

$$= \frac{1}{MN}$$

Relation between Spatial and Fourier domain

Let $f(x, y) = \delta(x, y)$.

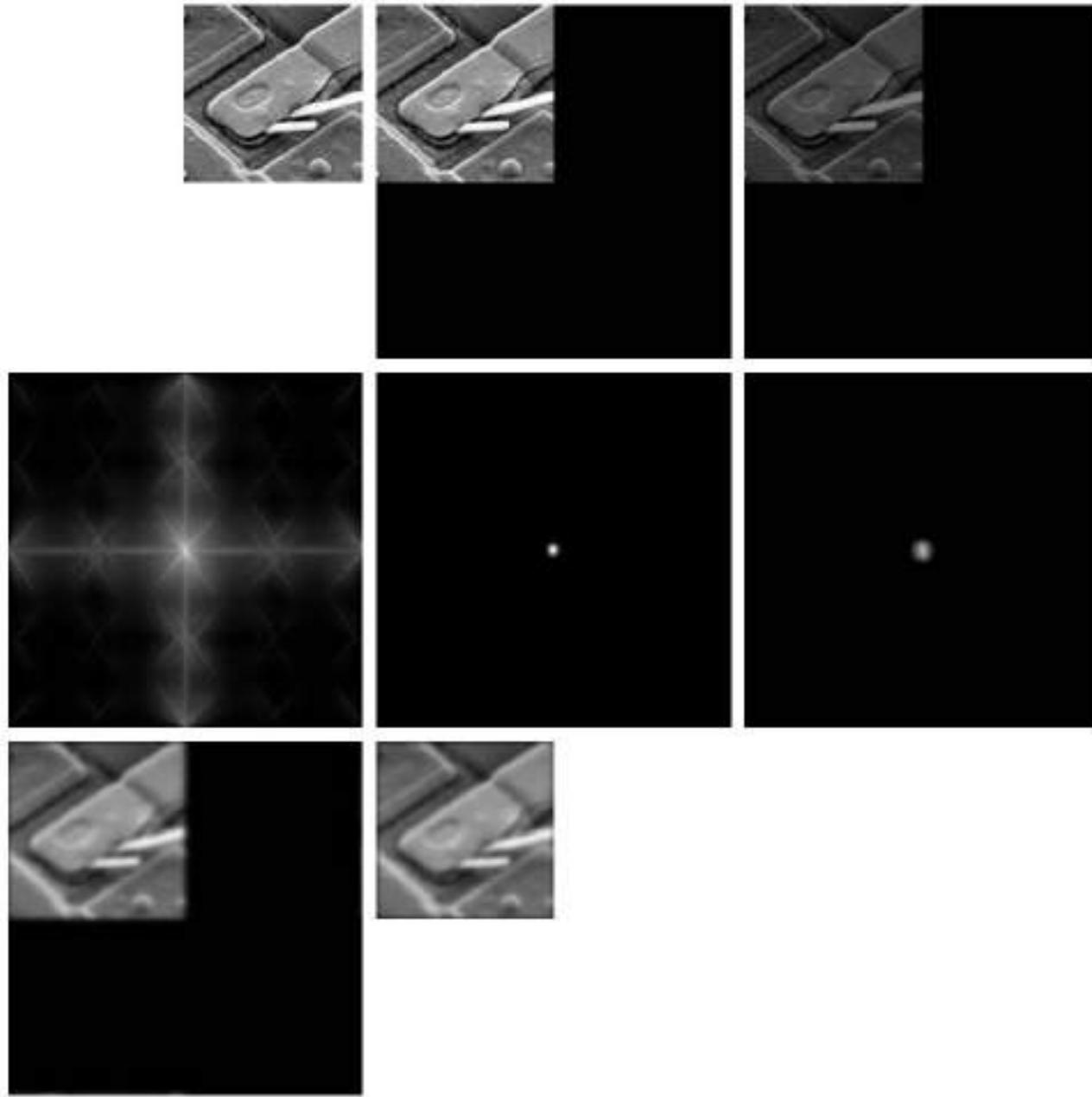
$$\begin{aligned} f(x, y) * h(x, y) &= \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(m, n) h(x - m, y - n) \\ &= \frac{1}{MN} h(x, y) \end{aligned}$$

$$f(x, y) * h(x, y) \iff F(u, v)H(u, v)$$

$$\delta(x, y) * h(x, y) \iff \mathfrak{F}[\delta(x, y)]H(u, v)$$

$$h(x, y) \iff H(u, v)$$

Given a filter in the frequency domain, we can obtain the corresponding filter in the spatial domain by taking the inverse Fourier transform of the former. The reverse is also true.



a	b	c
d	e	f
g	h	

FIGURE 4.36

- (a) An $M \times N$ image, f .
- (b) Padded image, f_p of size $P \times Q$.
- (c) Result of multiplying f_p by $(-1)^{x+y}$.
- (d) Spectrum of F_p .
- (e) Centered Gaussian lowpass filter, H , of size $P \times Q$.
- (f) Spectrum of the product HF_p .
- (g) g_p , the product of $(-1)^{x+y}$ and the real part of the IDFT of HF_p .
- (h) Final result, g , obtained by cropping the first M rows and N columns of g_p .

Spatial Domain vs. Frequency Domain Filtering

Let $H(u)$ denote the 1-D frequency domain Gaussian filter

$$H(u) = Ae^{-u^2/2\sigma^2}$$

The corresponding filter in the spatial domain

$$h(x) = \sqrt{2\pi}\sigma A e^{-2\pi^2\sigma^2x^2}$$

1. Both components are Gaussian and real
2. The functions behave reciprocally

Spatial Domain vs. Frequency Domain Filtering

Let $H(u)$ denote the difference of Gaussian filter

$$H(u) = Ae^{-u^2/2\sigma_1^2} - Be^{-u^2/2\sigma_2^2}$$

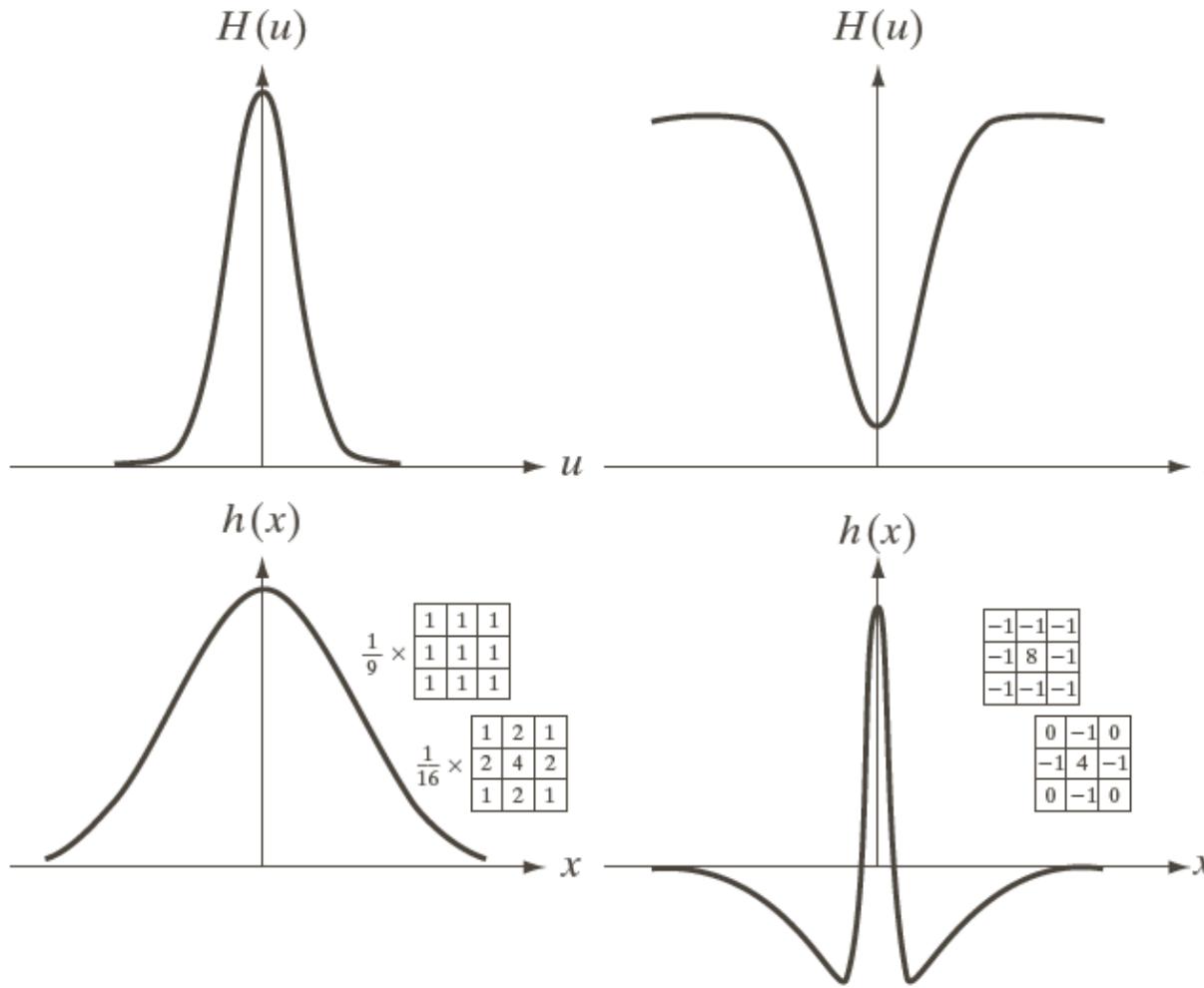
with $A \geq B$ and $\sigma_1 \geq \sigma_2$

The corresponding filter in the spatial domain

$$h(x) = \sqrt{2\pi}\sigma_1 Ae^{-2\pi^2\sigma_1^2x^2} - \sqrt{2\pi}\sigma_2 Ae^{-2\pi^2\sigma_2^2x^2}$$

High-pass filter or low-pass filter ?

Spatial Domain vs. Frequency Domain Filtering

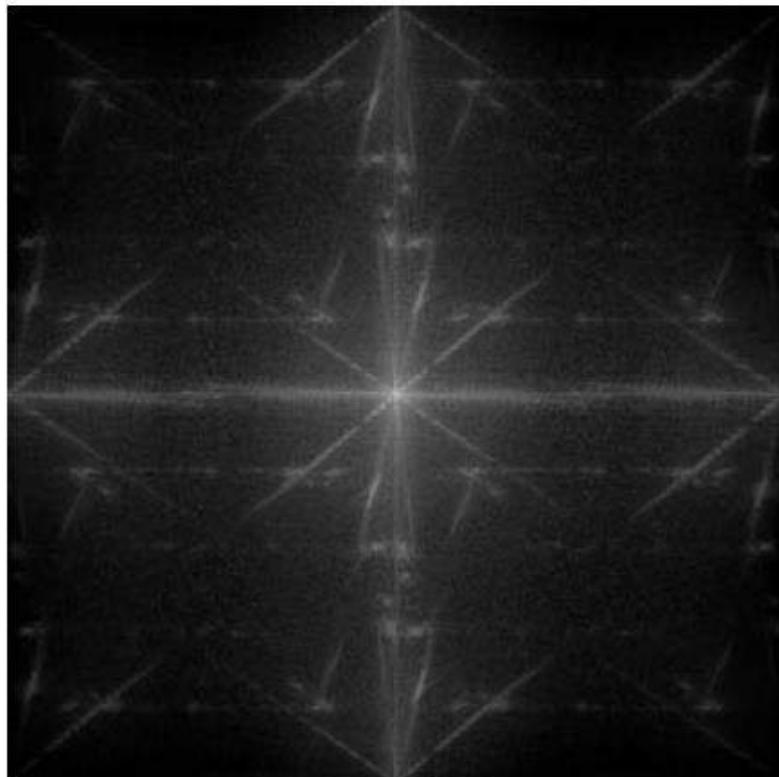


a	c
b	d

FIGURE 4.37

(a) A 1-D Gaussian lowpass filter in the frequency domain.
(b) Spatial lowpass filter corresponding to (a).
(c) Gaussian highpass filter in the frequency domain.
(d) Spatial highpass filter corresponding to (c). The small 2-D masks shown are spatial filters we used in Chapter 3.

Spatial Domain vs. Frequency Domain Filtering

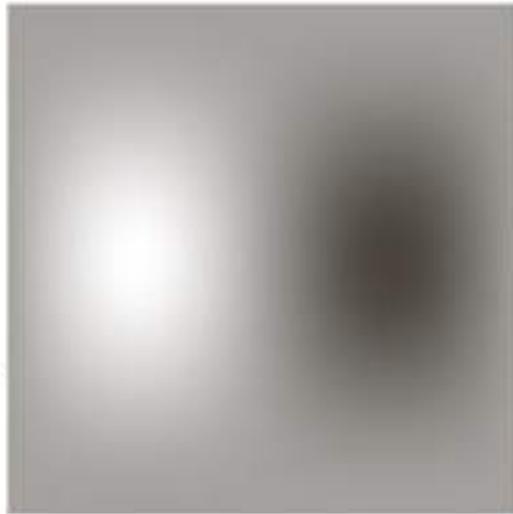
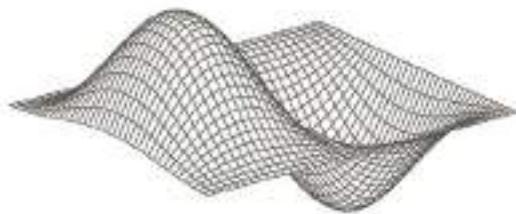


a b

FIGURE 4.38
(a) Image of a
building, and
(b) its spectrum.

Spatial Domain vs. Frequency Domain Filtering

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



a	b
c	d

FIGURE 4.39
(a) A spatial mask and perspective plot of its corresponding frequency domain filter. (b) Filter shown as an image. (c) Result of filtering Fig. 4.38(a) in the frequency domain with the filter in (b). (d) Result of filtering the same image with the spatial filter in (a). The results are identical.

Smoothing Filters

- Edges and other sharp transitions (such as noise) contribute significantly to the high frequency content of Fourier transform.
- Smoothing (blurring) is achieved in frequency domain by attenuating a specified range of high frequency components.
- Three types of lowpass filters: Ideal, Butterworth and Gaussian filters
- These filters cover the range from very sharp (ideal) to very smooth (Gaussian) filter functions.

Smoothing Filters: Ideal Lowpass Filters (ILPF)

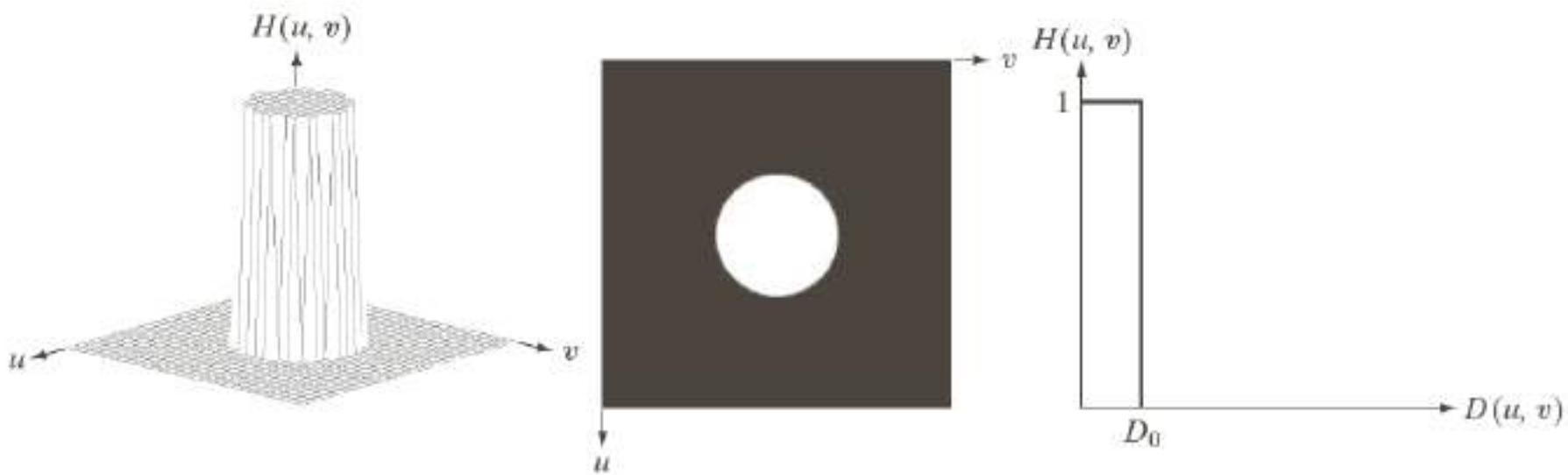
Ideal Lowpass Filters (ILPF)

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$$

D_0 is a positive constant and $D(u, v)$ is the distance between a point (u, v) in the frequency domain and the center of the frequency rectangle

$$D(u, v) = \left[(u - P/2)^2 + (v - Q/2)^2 \right]^{1/2}$$

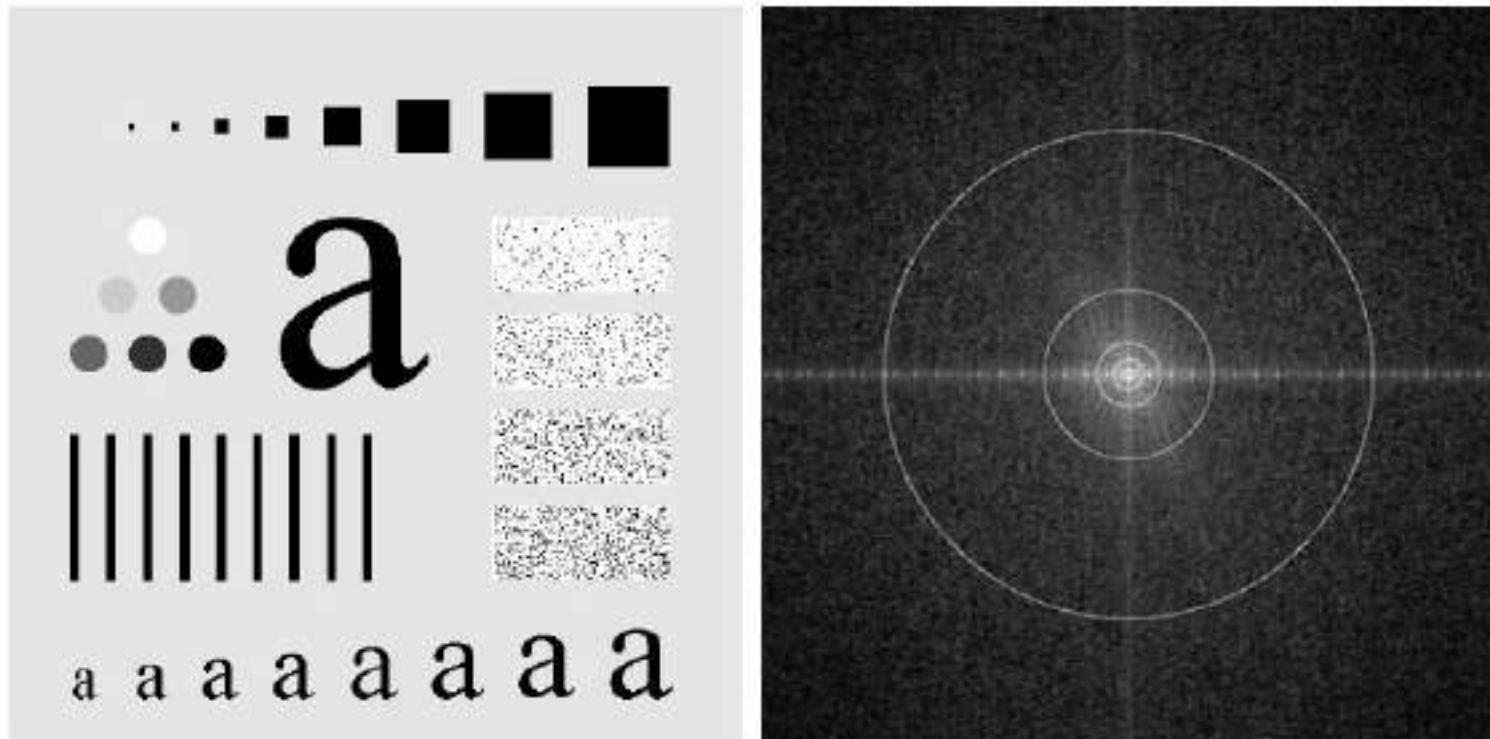
Smoothing Filters: ILPF



a b c

FIGURE 4.40 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Smoothing Filters: ILPF



a b

FIGURE 4.41 (a) Test pattern of size 688×688 pixels, and (b) its Fourier spectrum. The spectrum is double the image size due to padding but is shown in half size so that it fits in the page. The superimposed circles have radii equal to 10, 30, 60, 160, and 460 with respect to the full-size spectrum image. These radii enclose 87.0, 93.1, 95.7, 97.8, and 99.2% of the padded image power, respectively.

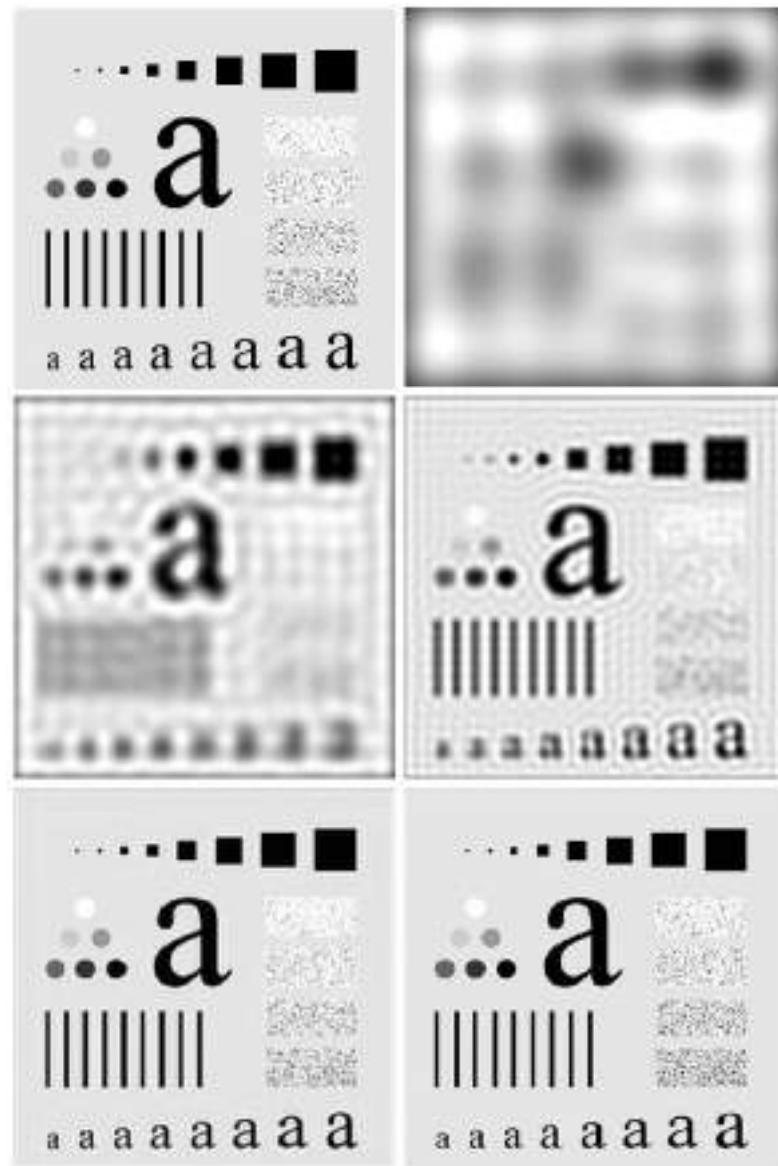
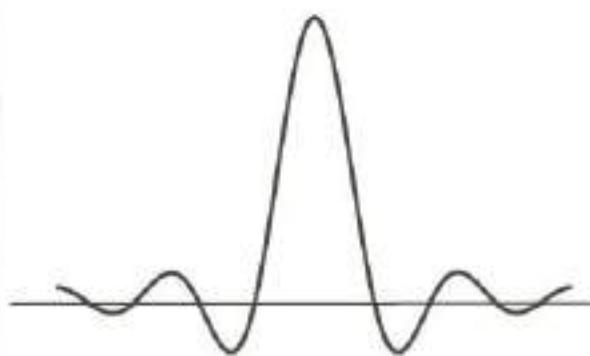
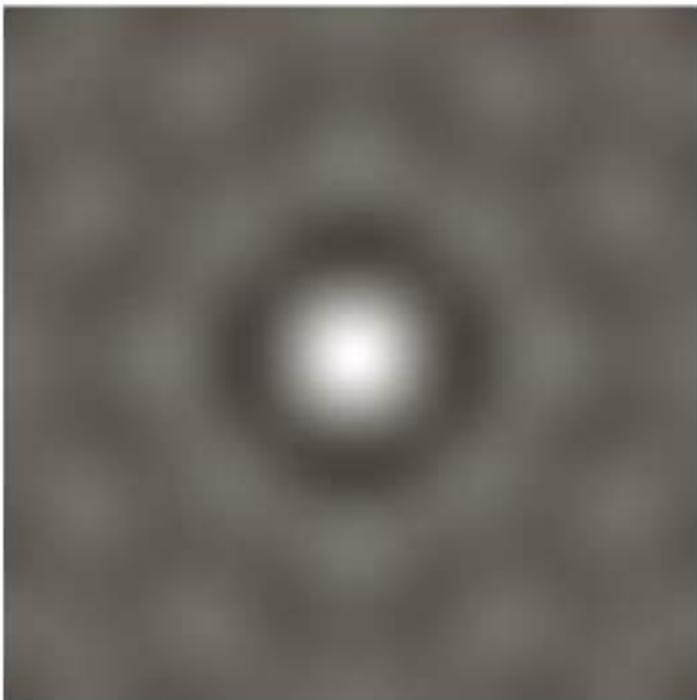


FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using HLPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

Smoothing Filters: Spatial Representation of ILPF



a b

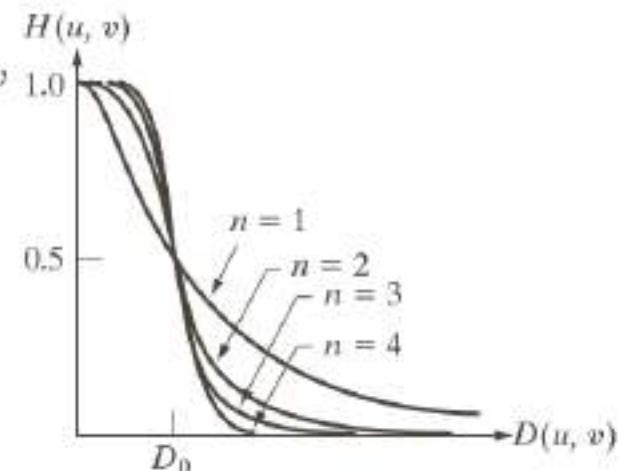
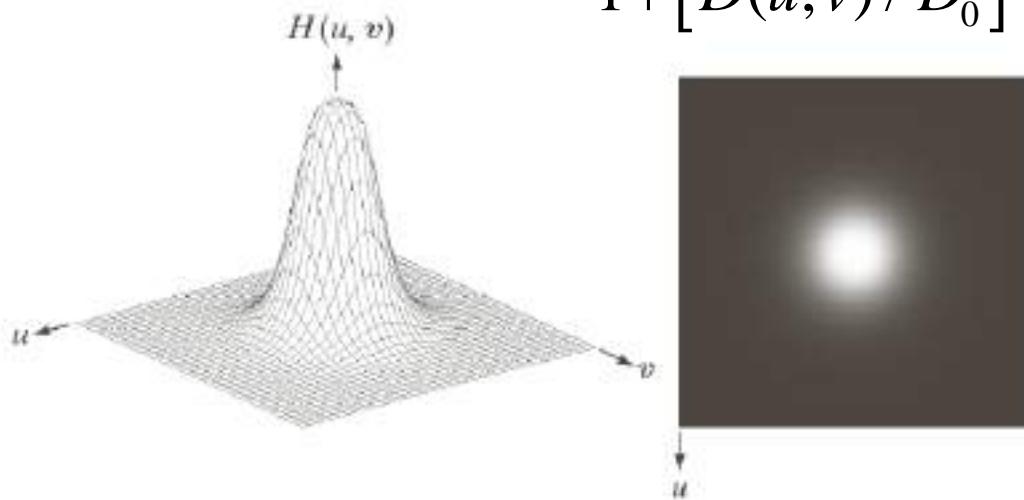
FIGURE 4.43

(a) Representation in the spatial domain of an ILPF of radius 5 and size 1000×1000 .
(b) Intensity profile of a horizontal line passing through the center of the image.

Smoothing Filters: Butterworth Lowpass Filters (BLPF)

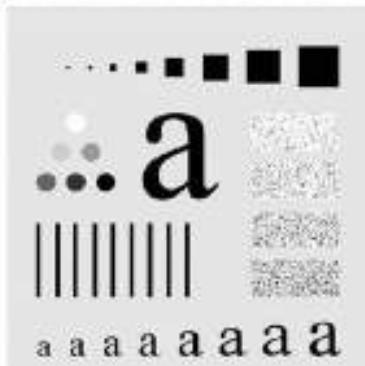
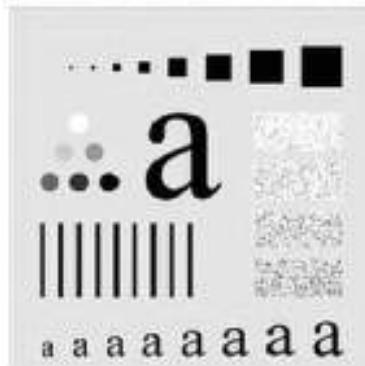
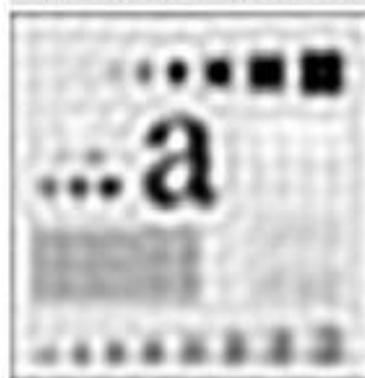
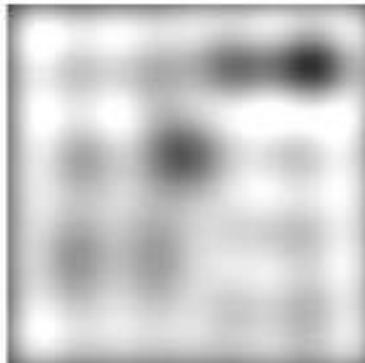
Butterworth Lowpass Filters (BLPF) of order n and with cutoff frequency D_0

$$H(u, v) = \frac{1}{1 + [D(u, v) / D_0]^{2n}}$$



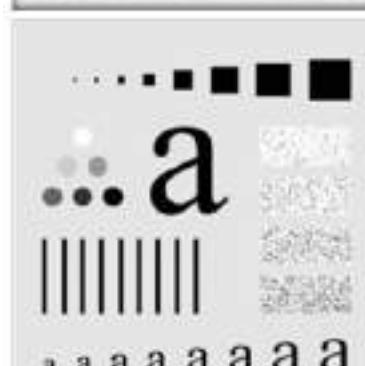
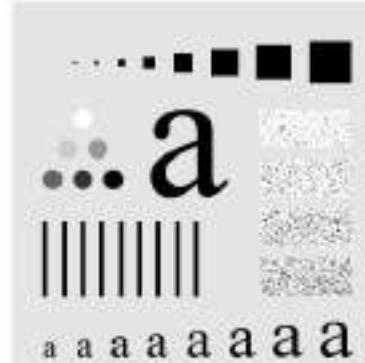
a b c

FIGURE 4.44 (a) Perspective plot of a Butterworth lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.



a b
c d
e f

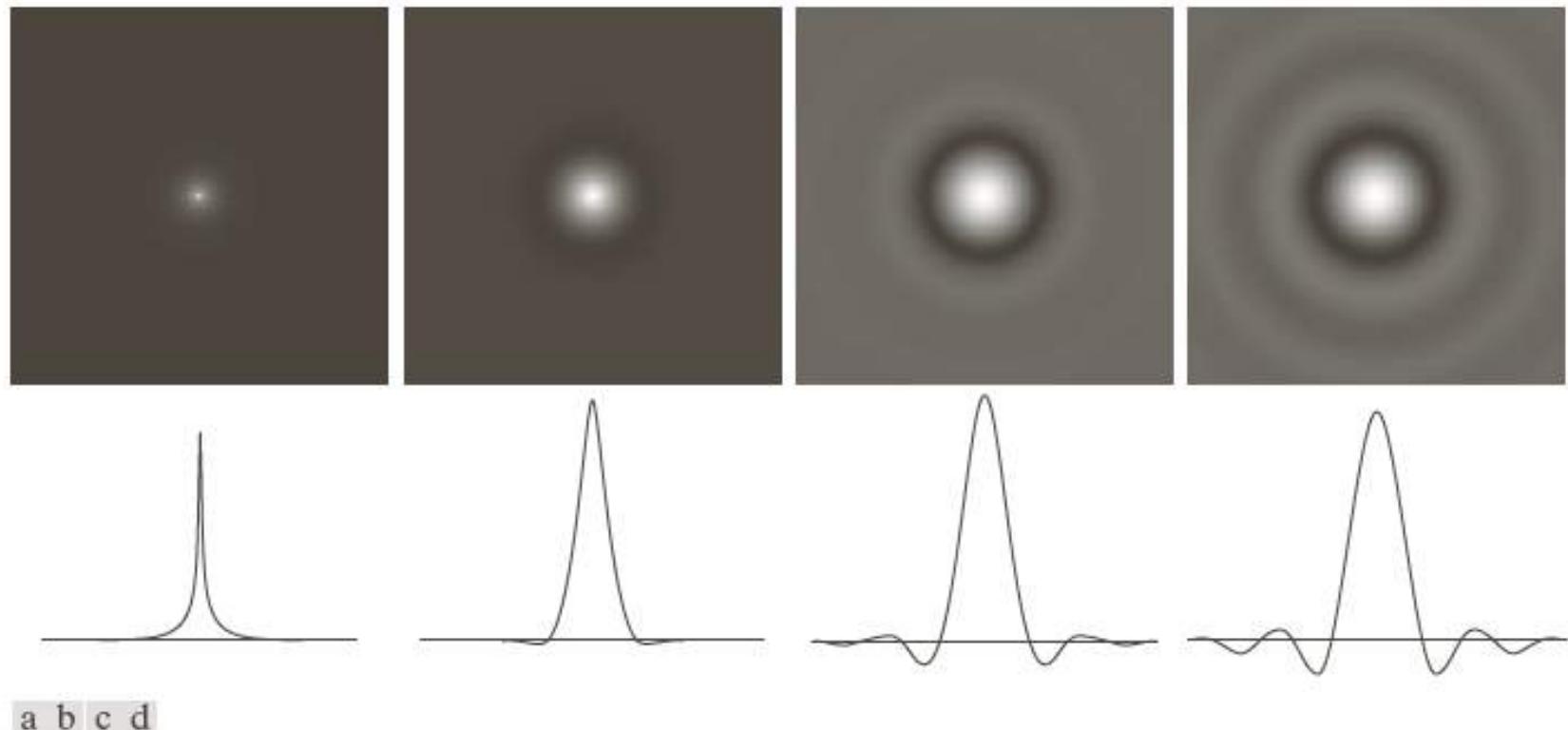
FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.



a b
c d
e f

FIGURE 4.45 (a) Original image. (b)–(f) Results of filtering using BLPFs of order 2, with cutoff frequencies at the radii shown in Fig. 4.41. Compare with Fig. 4.42.

Smoothing Filters: Spatial Representation of BLPF



a b c d

FIGURE 4.46 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding intensity profiles through the center of the filters (the size in all cases is 1000×1000 and the cutoff frequency is 5). Observe how ringing increases as a function of filter order.

Smoothing Filters: Gaussian Lowpass Filters (GLPF)

Gaussian Lowpass Filters (GLPF) in two dimensions is given

$$H(u, v) = e^{-D^2(u, v)/2\sigma^2}$$

By letting $\sigma = D_0$

$$H(u, v) = e^{-D^2(u, v)/2D_0^2}$$

Smoothing Filters: GLPF

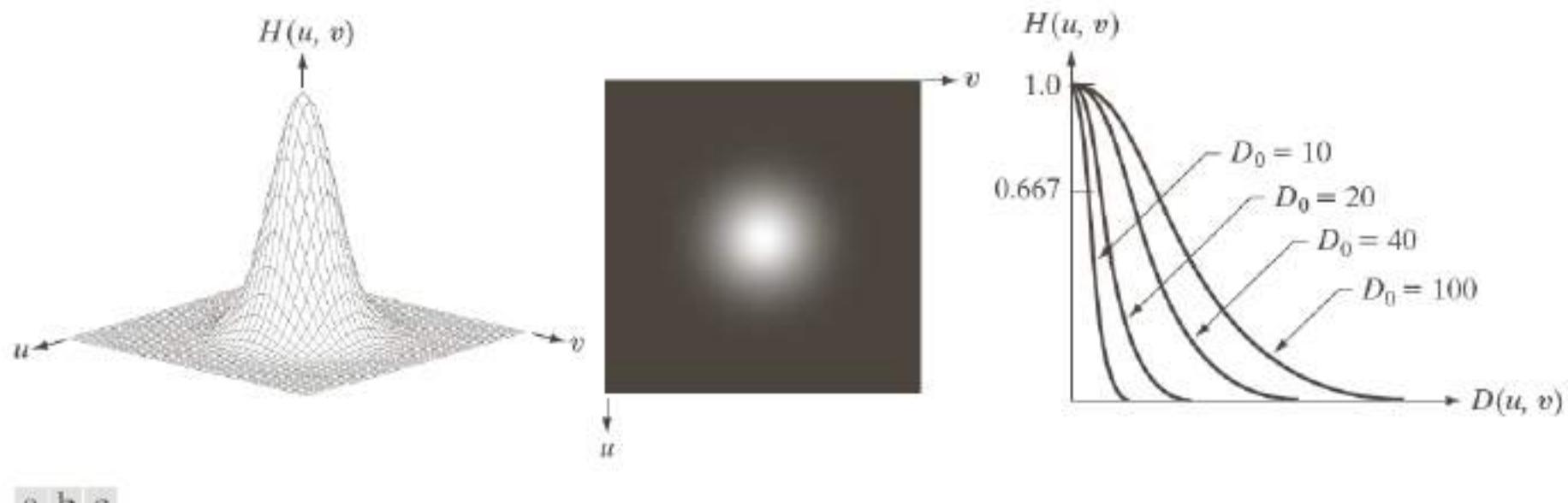


FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0 .

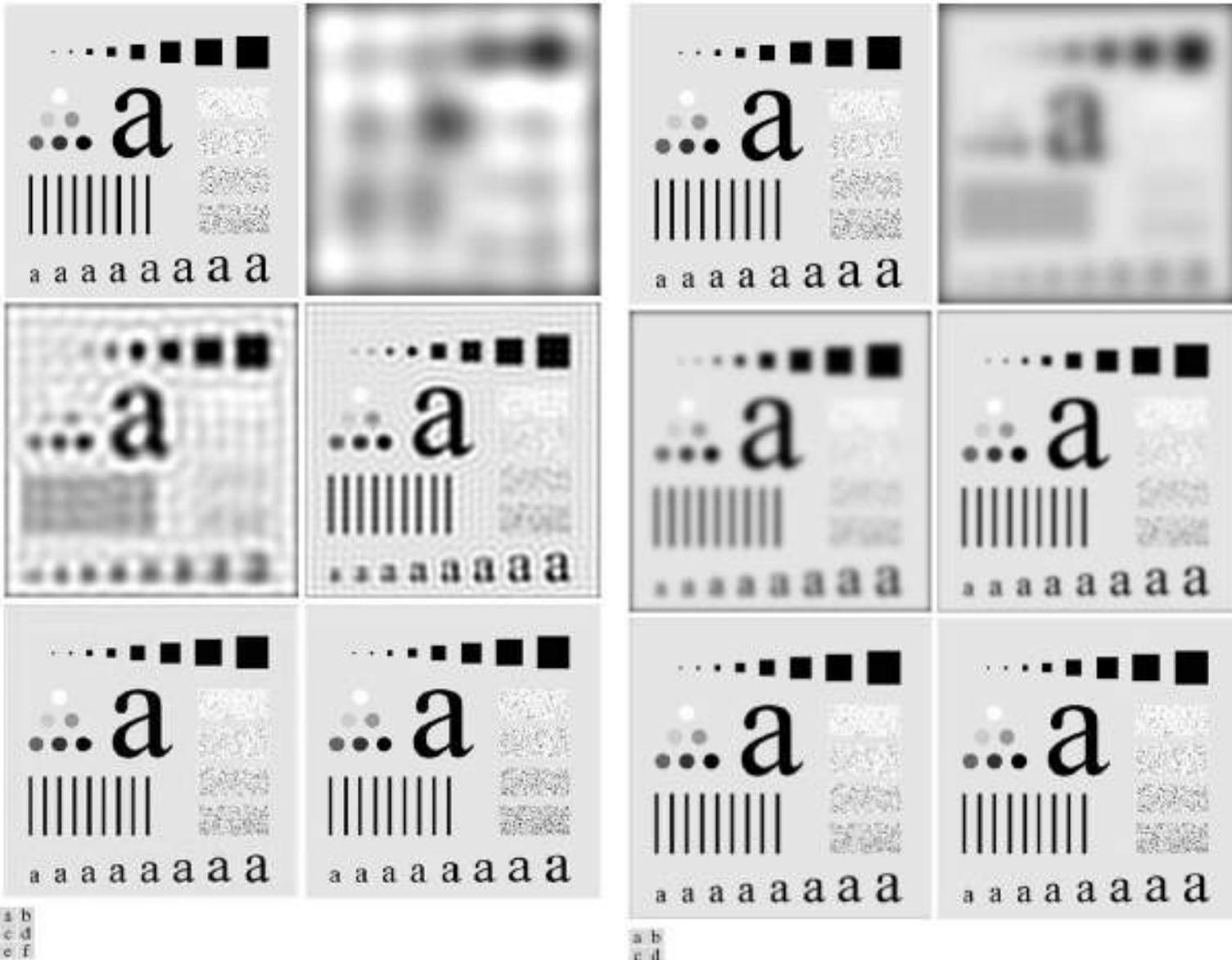


FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.



FIGURE 4.48 (a) Original image. (b)–(f) Results of filtering using GLPFs with cutoff frequencies at the radii shown in Fig. 4.41. Compare with Figs. 4.42 and 4.45.

Smoothing Filters: Example of smoothing by GLPF

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



a b

FIGURE 4.49

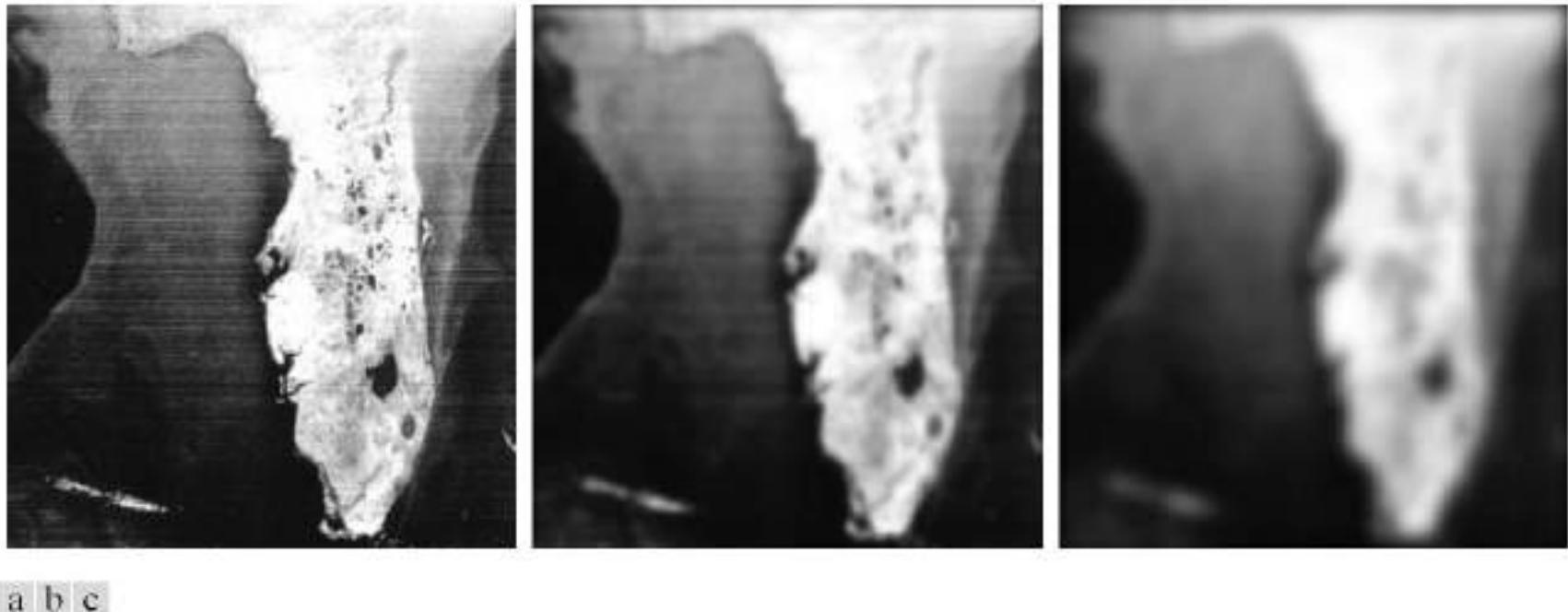
(a) Sample text of low resolution (note broken characters in magnified view).
(b) Result of filtering with a GLPF (broken character segments were joined).

Smoothing Filters: Example of smoothing by GLPF



FIGURE 4.50 (a) Original image (784×732 pixels). (b) Result of filtering using a GLPF with $D_0 = 100$. (c) Result of filtering using a GLPF with $D_0 = 80$. Note the reduction in fine skin lines in the magnified sections in (b) and (c).

Smoothing Filters: Example of smoothing by GLPF



a b c

FIGURE 4.51 (a) Image showing prominent horizontal scan lines. (b) Result of filtering using a GLPF with $D_0 = 50$. (c) Result of using a GLPF with $D_0 = 20$. (Original image courtesy of NOAA.)

Sharpening Filters

- Image sharpening can be achieved in the frequency domain by highpass filtering.
- It attenuates low-frequency components without disturbing high-frequency information in the Fourier transform.

Sharpening Filters

A highpass filter is obtained from a given lowpass filter using

$$H_{HP}(u, v) = 1 - H_{LP}(u, v)$$

A 2-D ideal highpass filter (IHPL) is defined as

$$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$$

Sharpening Filters

A 2-D Butterworth highpass filter (BHPL) is defined as

$$H(u, v) = \frac{1}{1 + [D_0 / D(u, v)]^{2n}}$$

A 2-D Gaussian highpass filter (GHPL) is defined as

$$H(u, v) = 1 - e^{-D^2(u, v)/2D_0^2}$$

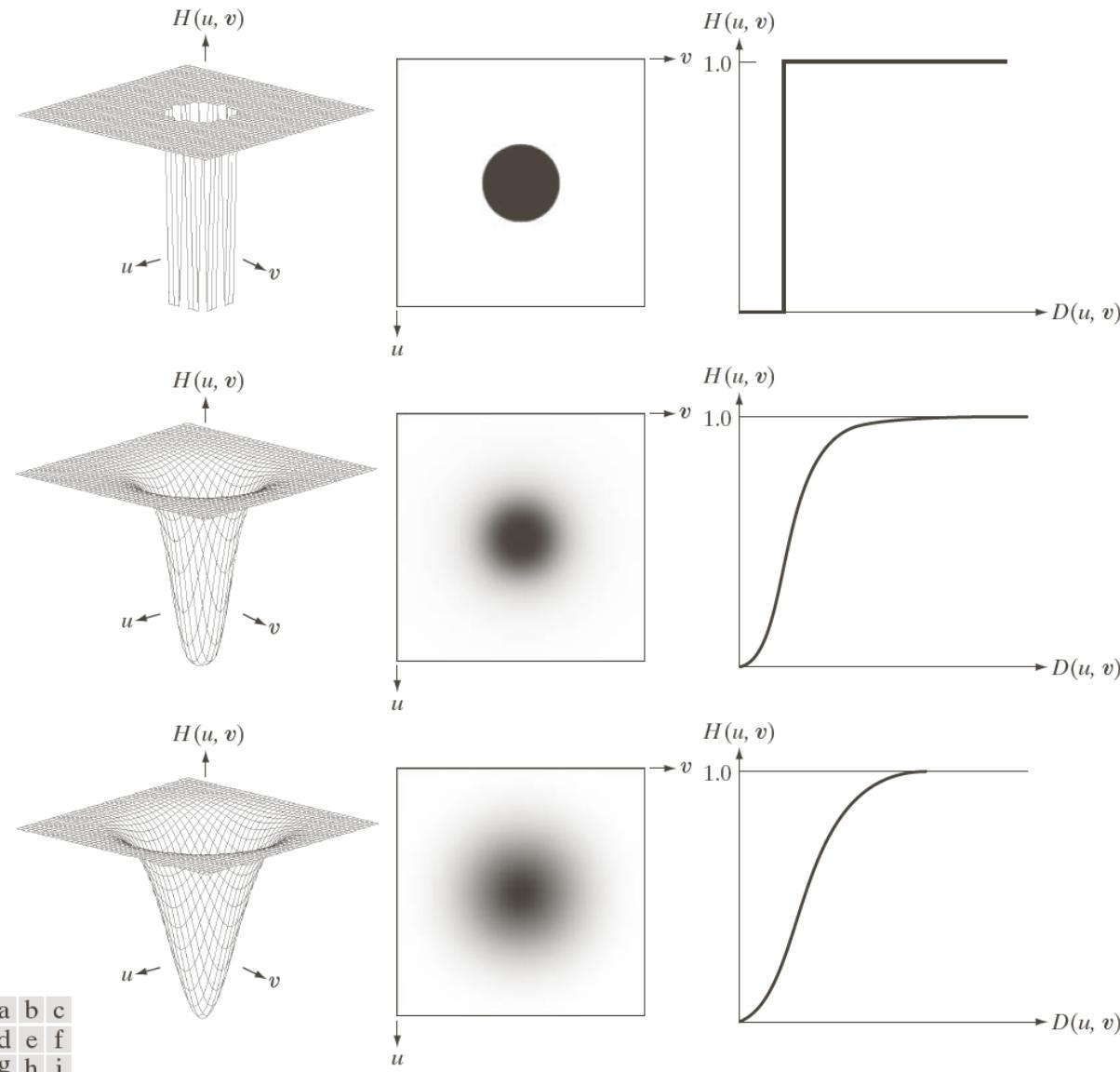


FIGURE 4.52 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Sharpening Filters: Spatial Representation

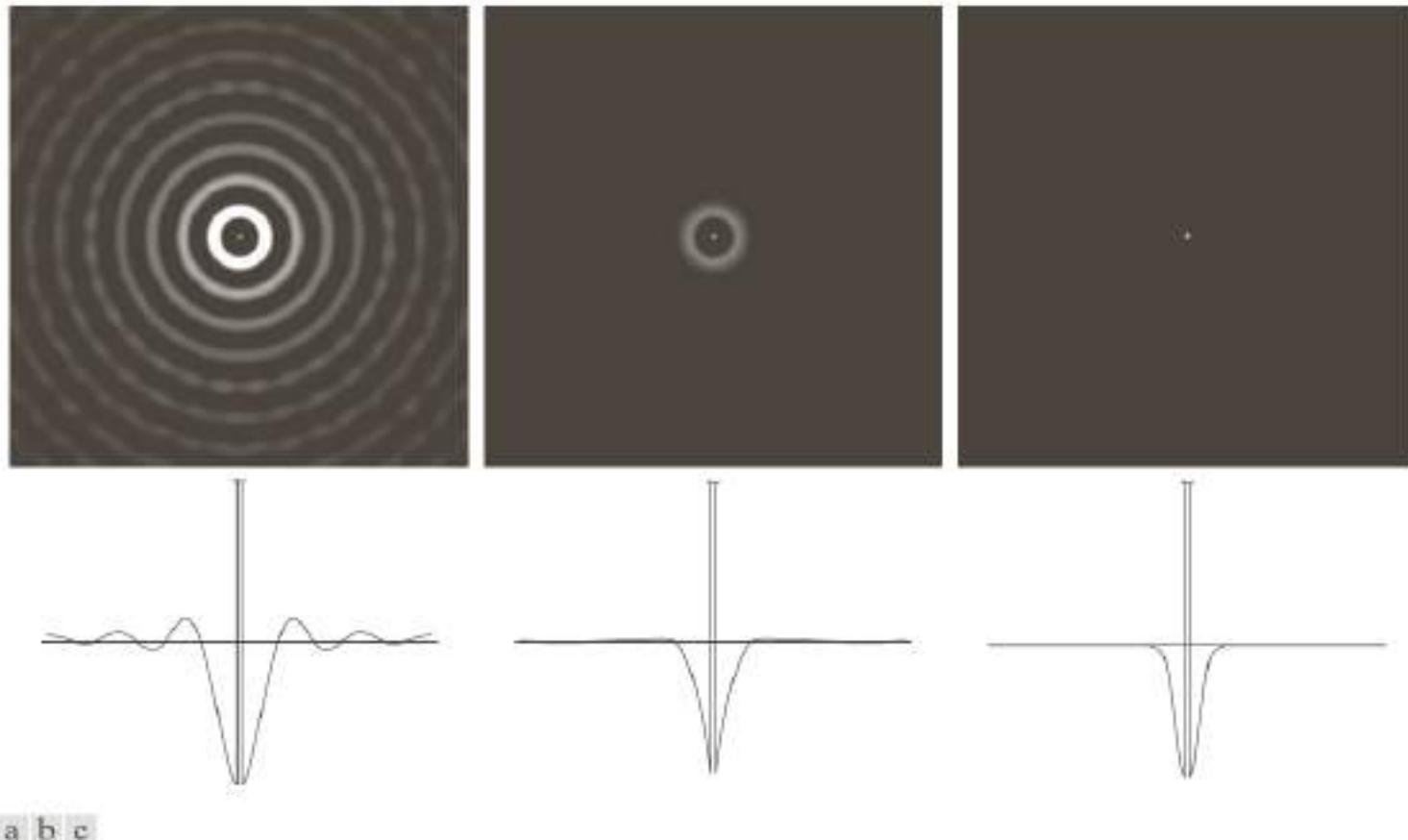


FIGURE 4.53 Spatial representation of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding intensity profiles through their centers.

Sharpening Filters: Result of IHPF



a b c

FIGURE 4.54 Results of highpass filtering the image in Fig. 4.41(a) using an IHPF with $D_0 = 30, 60$, and 160 .

Sharpening Filters: Result of BHPF



a b c

FIGURE 4.55 Results of highpass filtering the image in Fig. 4.41(a) using a BHPF of order 2 with $D_0 = 30, 60$, and 160 , corresponding to the circles in Fig. 4.41(b). These results are much smoother than those obtained with an IHPF.

Sharpening Filters: Result of GHPF



a b c

FIGURE 4.56 Results of highpass filtering the image in Fig. 4.41(a) using a GHPF with $D_0 = 30, 60$, and 160 , corresponding to the circles in Fig. 4.41(b). Compare with Figs. 4.54 and 4.55.

Laplacian in Frequency Domain

$$H(u, v) = -4\pi^2(u^2 + v^2)$$

$$\begin{aligned} H(u, v) &= -4\pi^2 \left[(u - P/2)^2 + (v - Q/2)^2 \right] \\ &= -4\pi^2 D^2(u, v) \end{aligned}$$

The Laplacian image

$$\nabla^2 f(x, y) = \mathfrak{F}^{-1} \{ H(u, v) F(u, v) \}$$

Enhancement is obtained

$$g(x, y) = f(x, y) + c \nabla^2 f(x, y) \quad c = -1$$

Laplacian in Frequency Domain

The enhanced image

$$\begin{aligned} g(x, y) &= \mathcal{I}^{-1} \left\{ F(u, v) - H(u, v)F(u, v) \right\} \\ &= \mathcal{I}^{-1} \left\{ [1 - H(u, v)]F(u, v) \right\} \\ &= \mathcal{I}^{-1} \left\{ [1 + 4\pi^2 D^2(u, v)]F(u, v) \right\} \end{aligned}$$

Laplacian in Frequency Domain



a b

FIGURE 4.58
(a) Original,
blurry image.
(b) Image
enhanced using
the Laplacian in
the frequency
domain. Compare
with Fig. 3.38(e).

Thank You

Questions?

Computer Vision

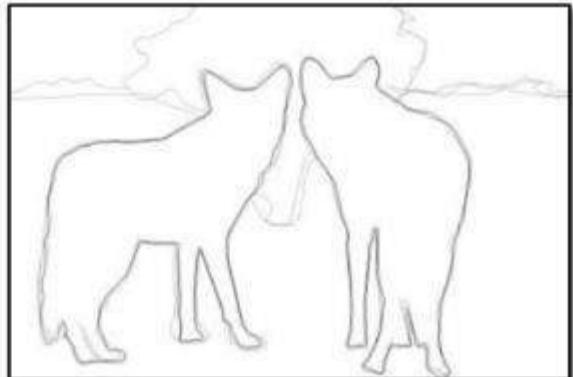
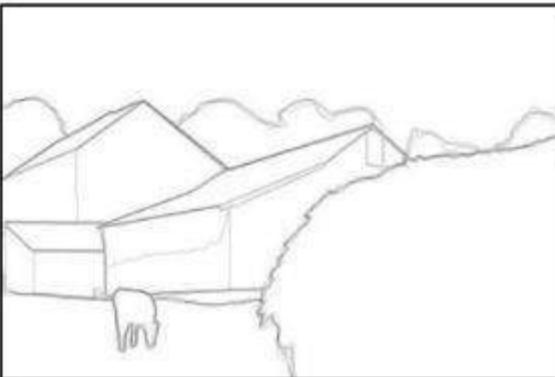
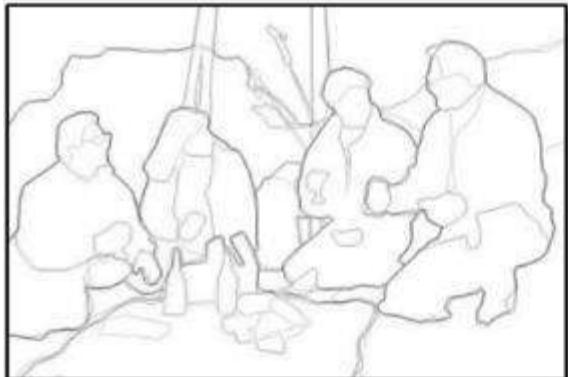
Edge Detection

Dr. Mrinmoy Ghorai

**Indian Institute of Information Technology
Sri City, Chittoor**



Today's Agenda: Edge Detection



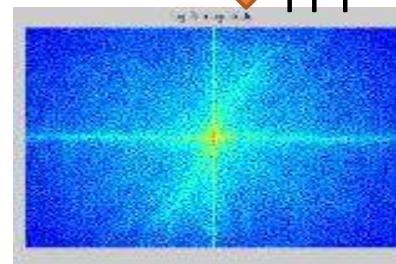
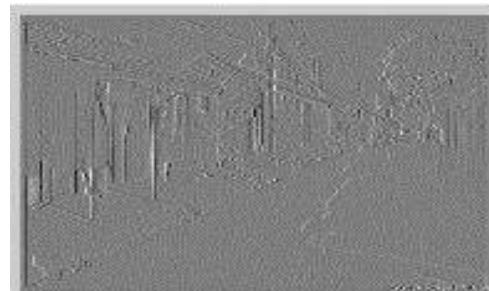
Previous classes: Image Filtering

Spatial domain

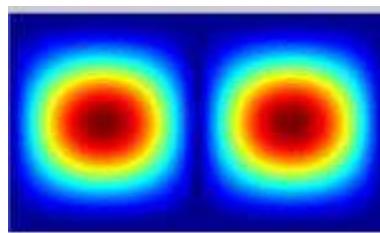
- Smoothing, sharpening, measuring texture



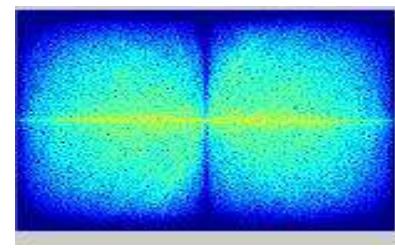
$$\text{Input image} * \text{Filter} = \text{Output image}$$



$$\times$$



$$=$$



Inverse FFT

Frequency domain

- Denoising, sampling

Today's class

- Detecting edges

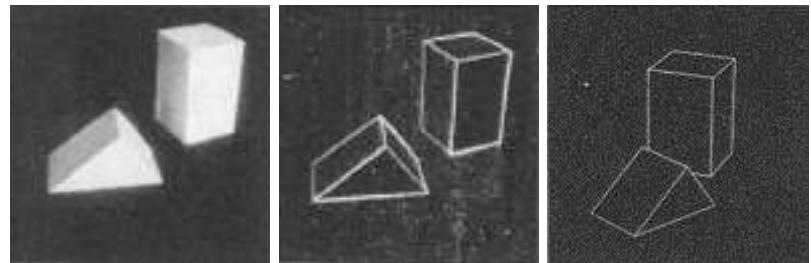


- Finding straight lines



Why finding edges is important?

- Cues for 3D shape



- Group pixels into objects or parts

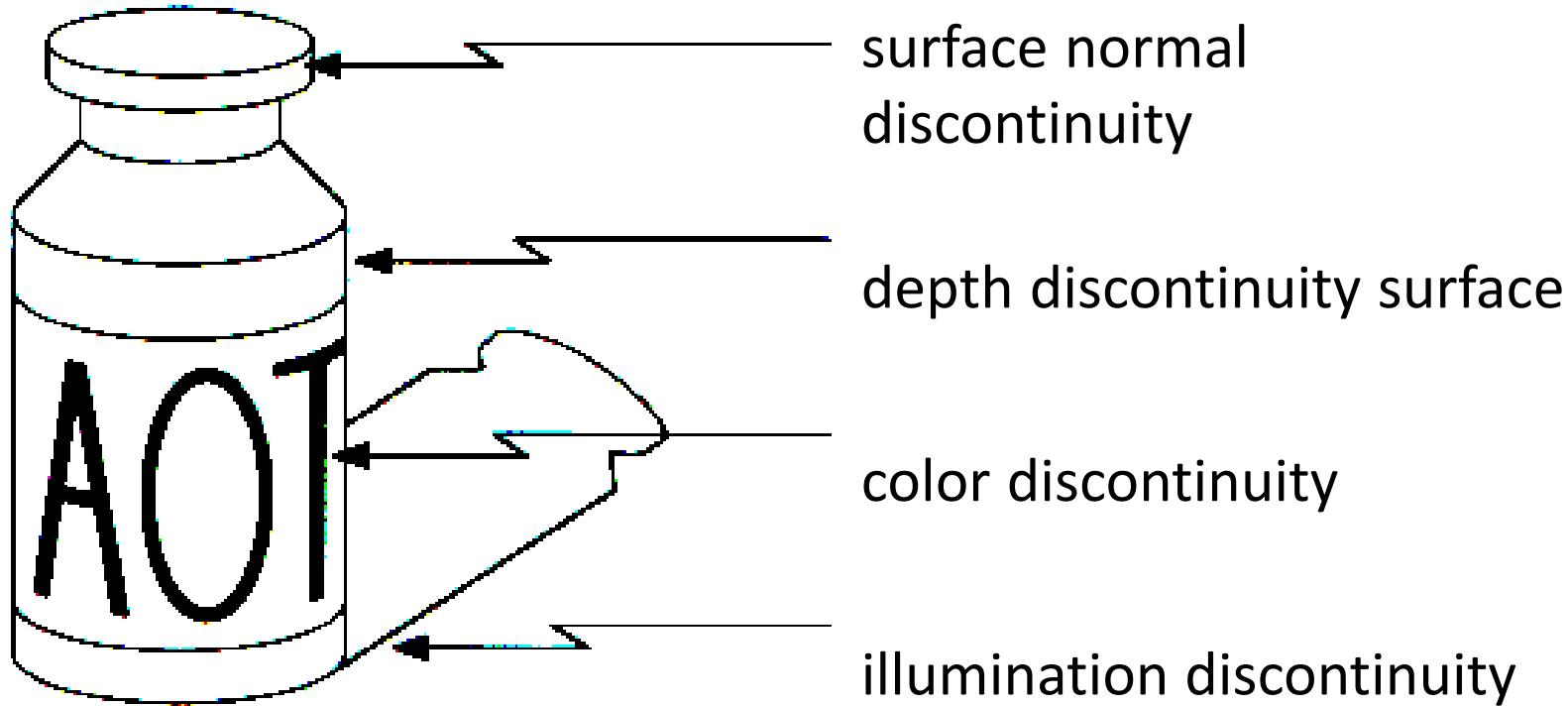


- Shape analysis

- Recover geometry and viewpoint



Origin of Edges

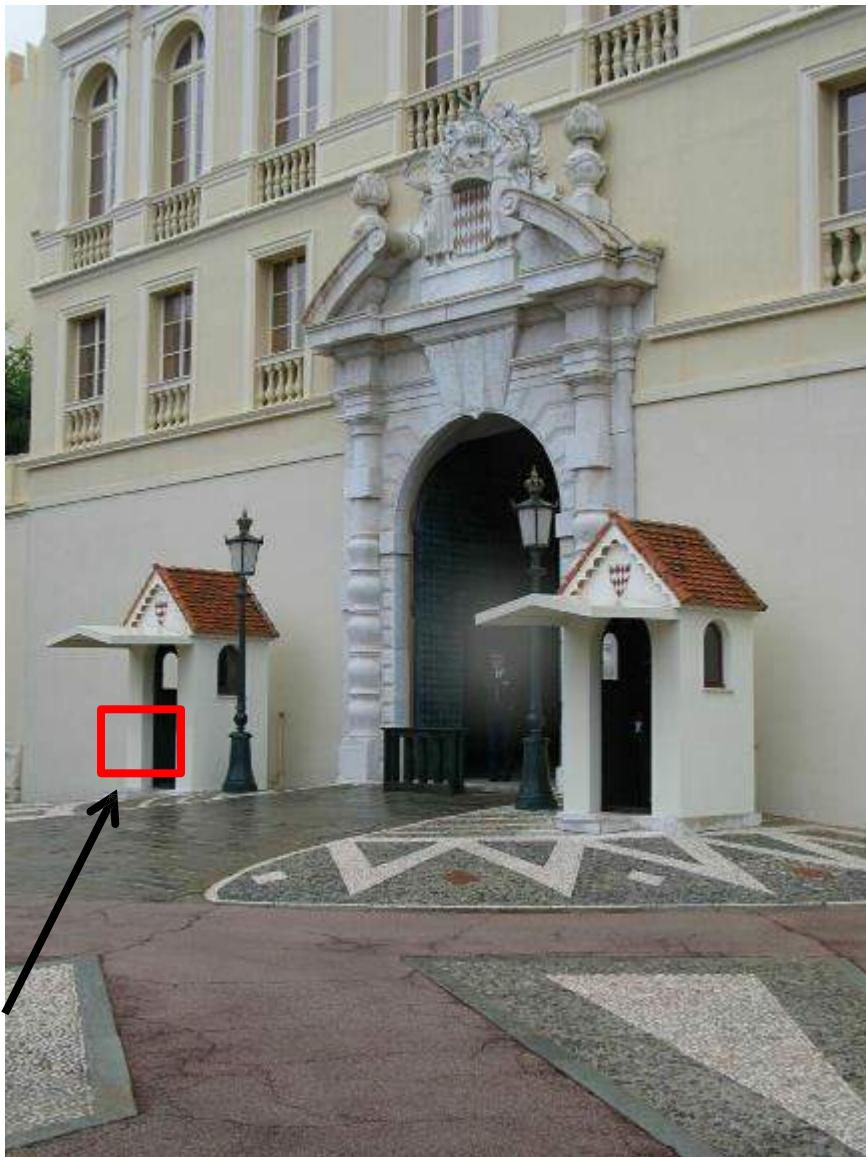


Edges are caused by a variety of factors

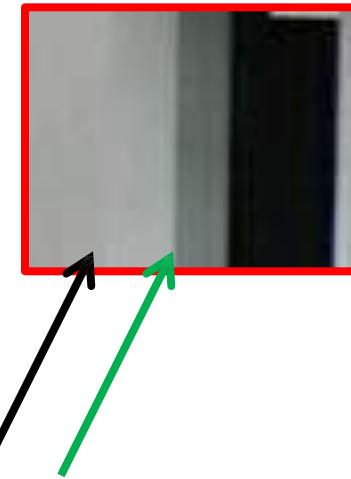
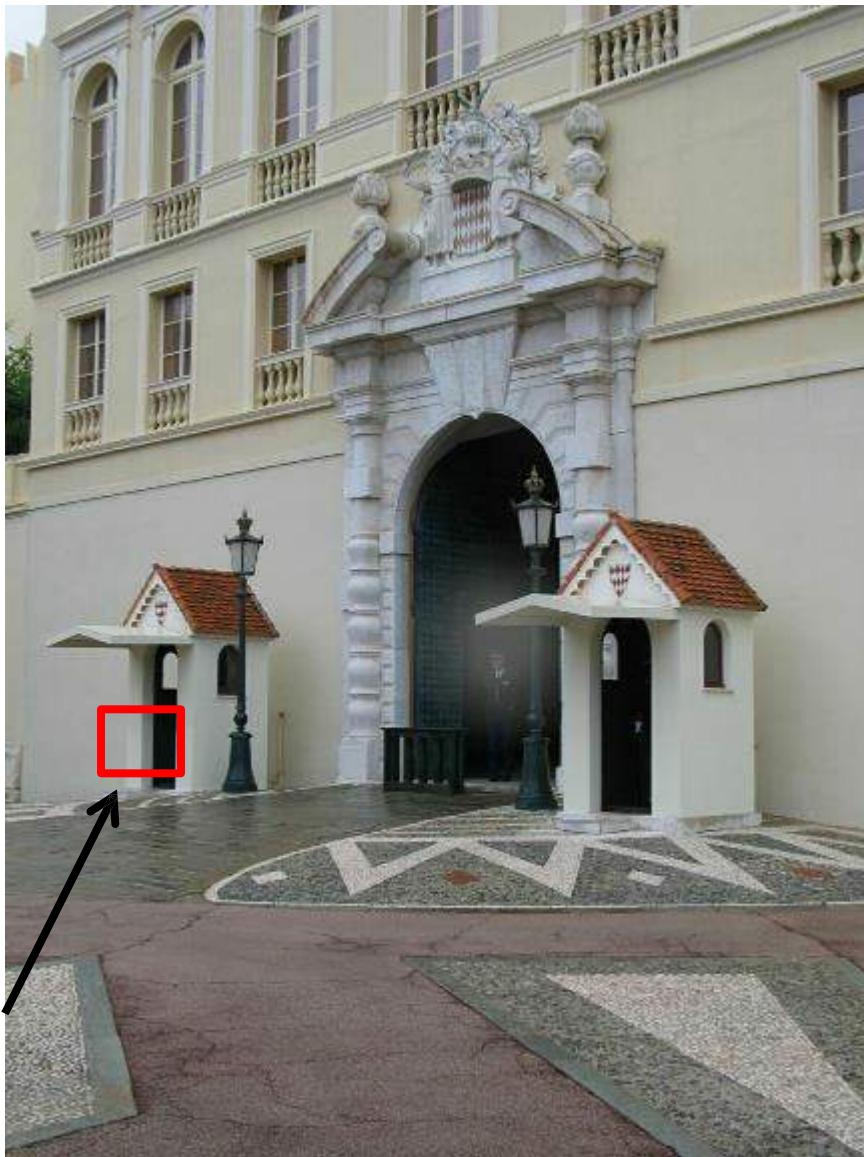
Closeup of edges



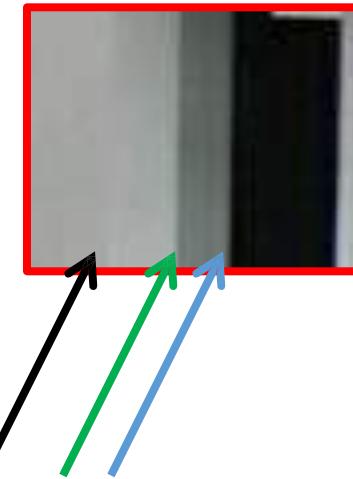
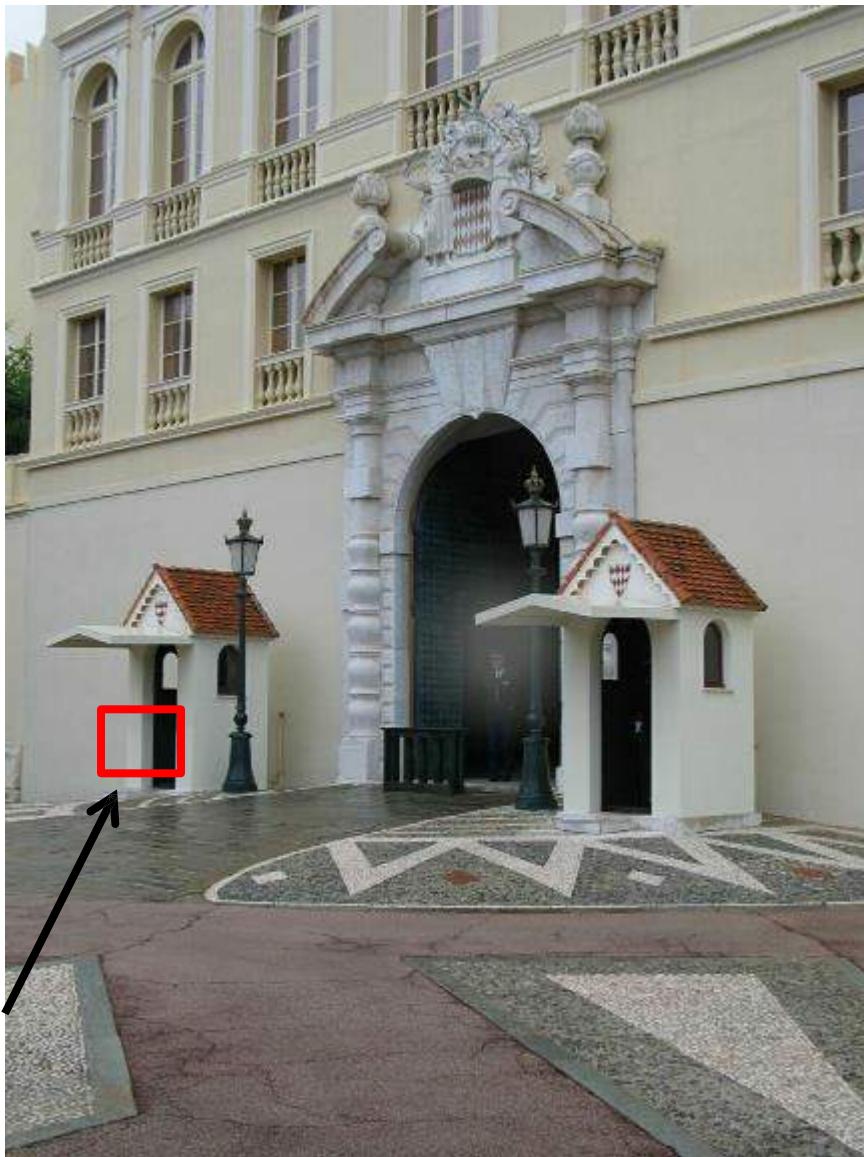
Closeup of edges



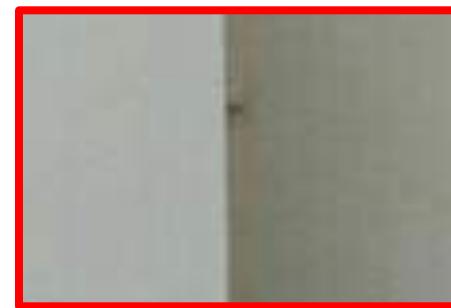
Closeup of edges



Closeup of edges



Closeup of edges



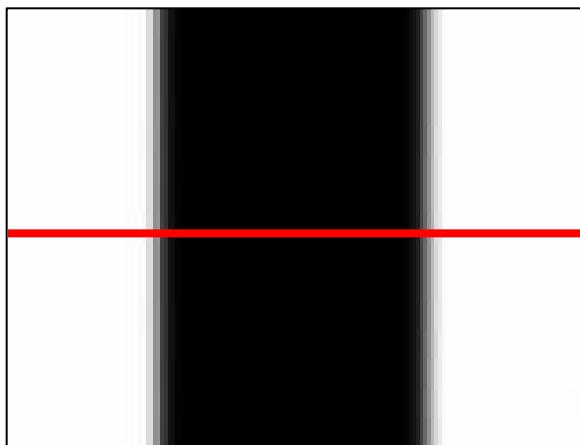
Closeup of edges



Characterizing edges

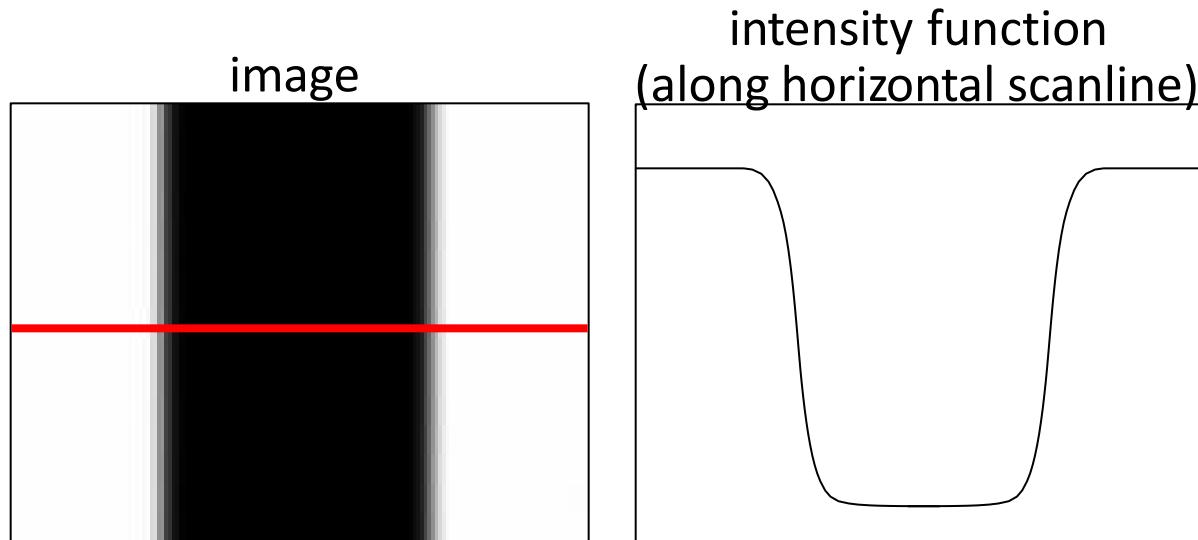
- An edge is a place of rapid change in the image intensity function

image



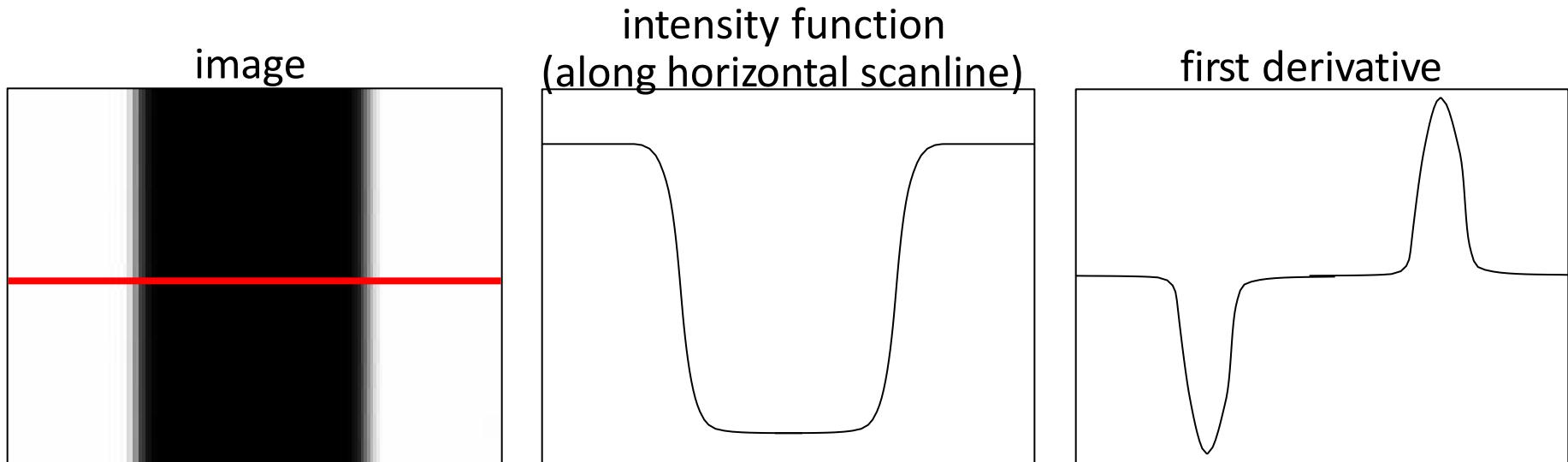
Characterizing edges

- An edge is a place of rapid change in the image intensity function



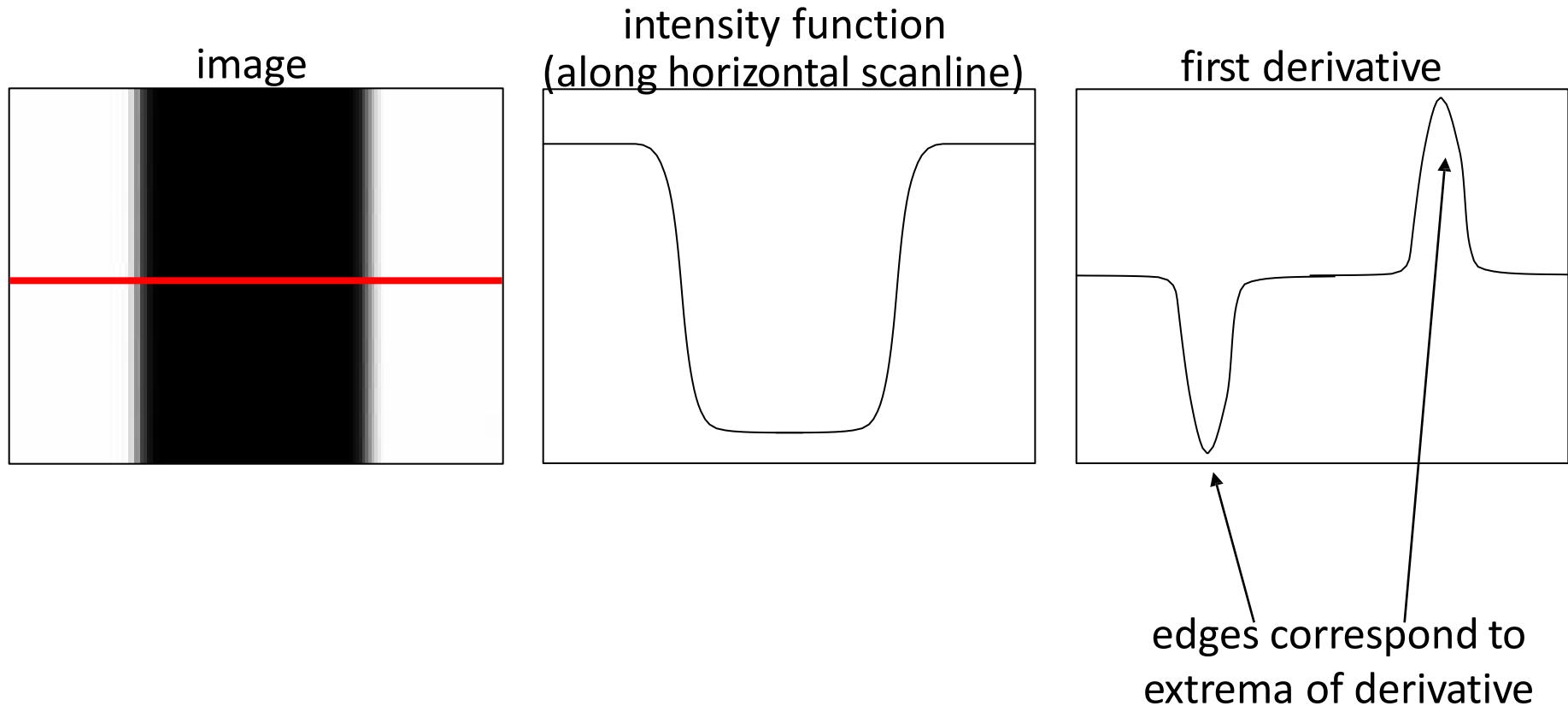
Characterizing edges

- An edge is a place of rapid change in the image intensity function

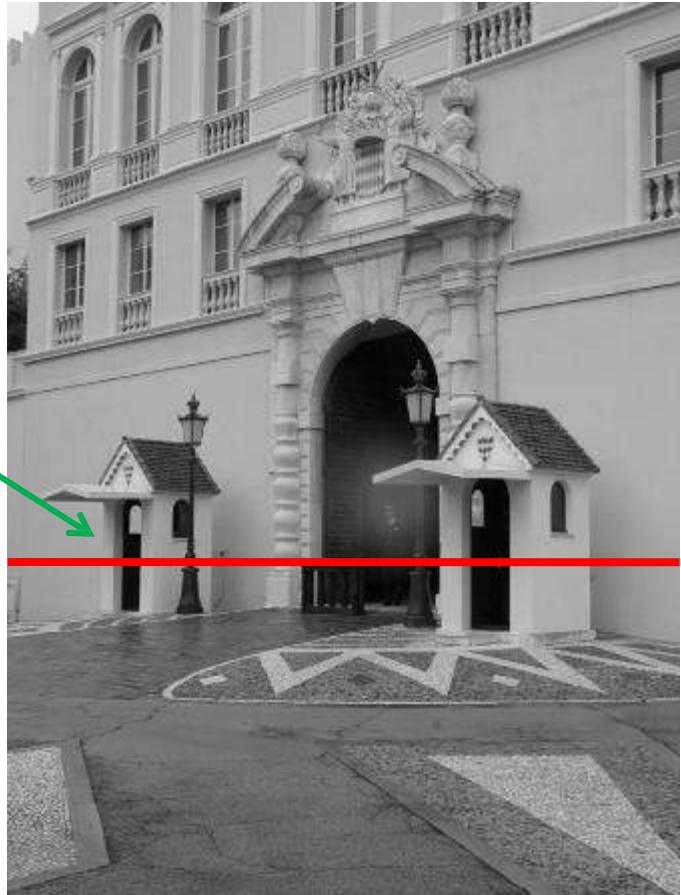


Characterizing edges

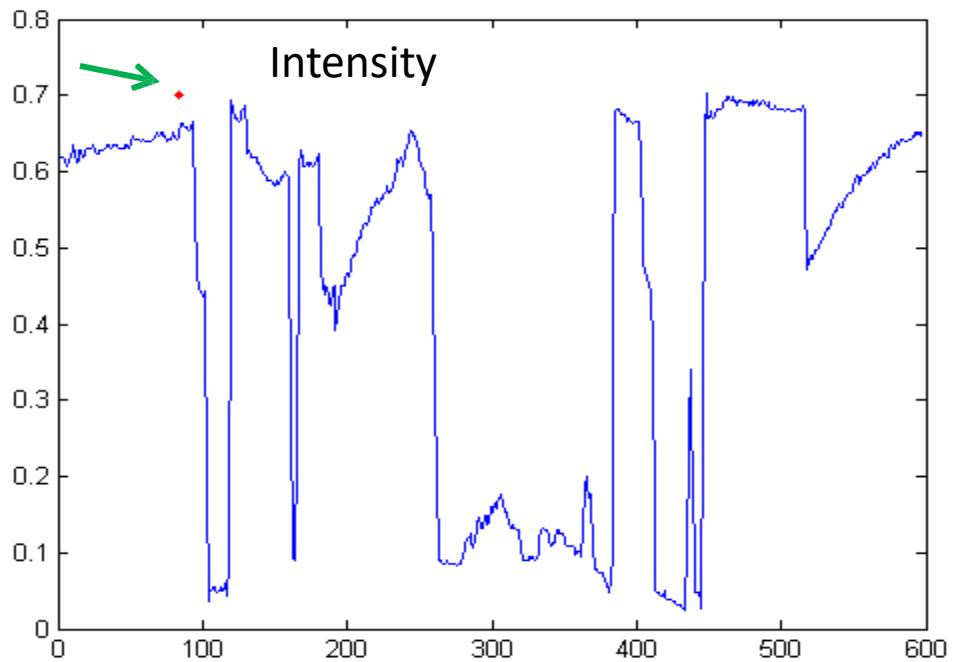
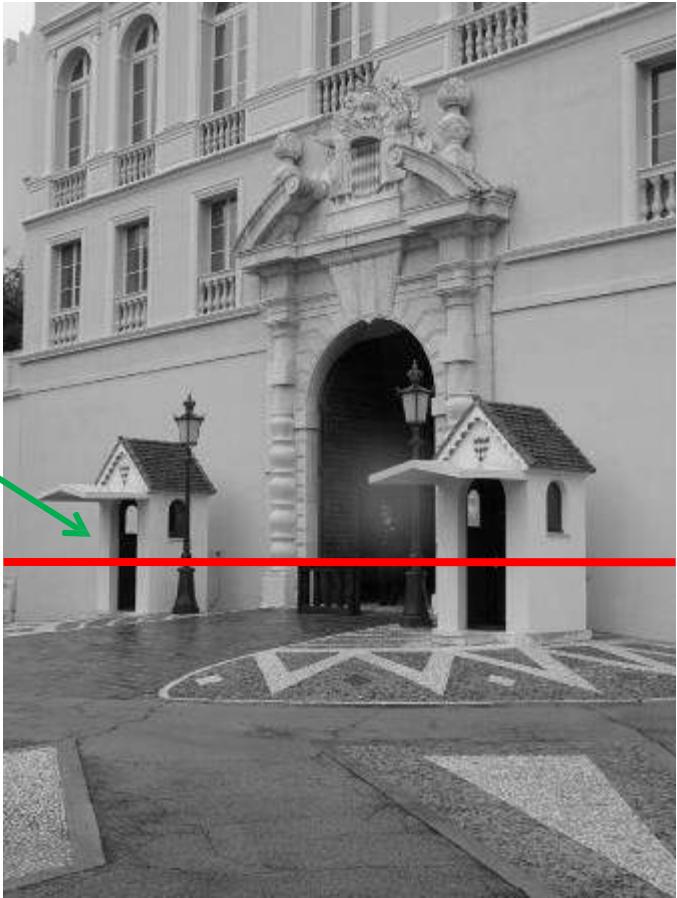
- An edge is a place of rapid change in the image intensity function



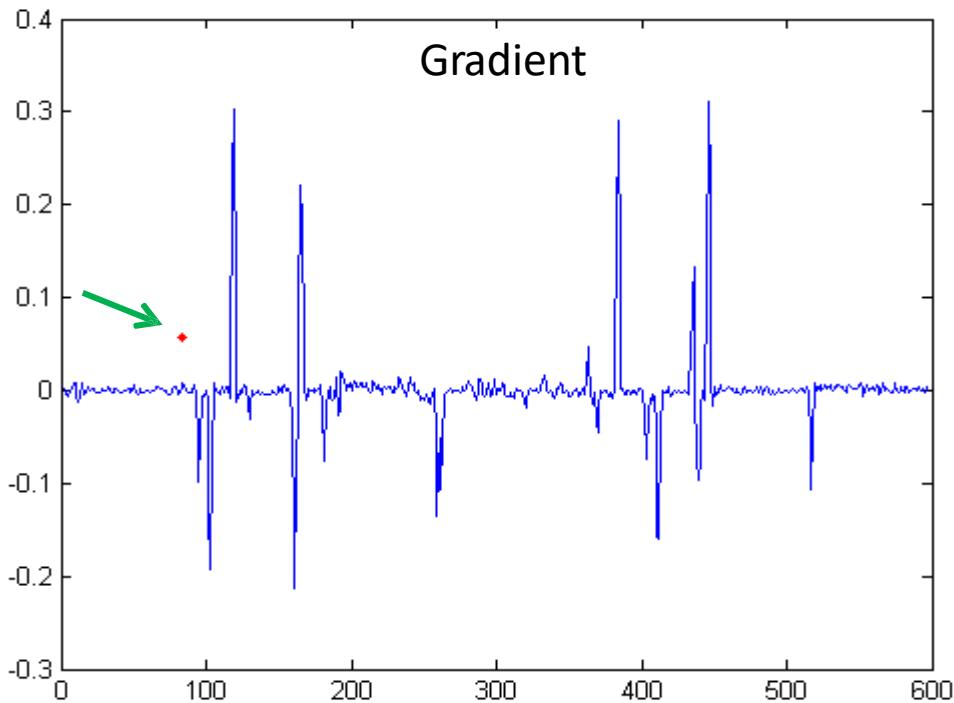
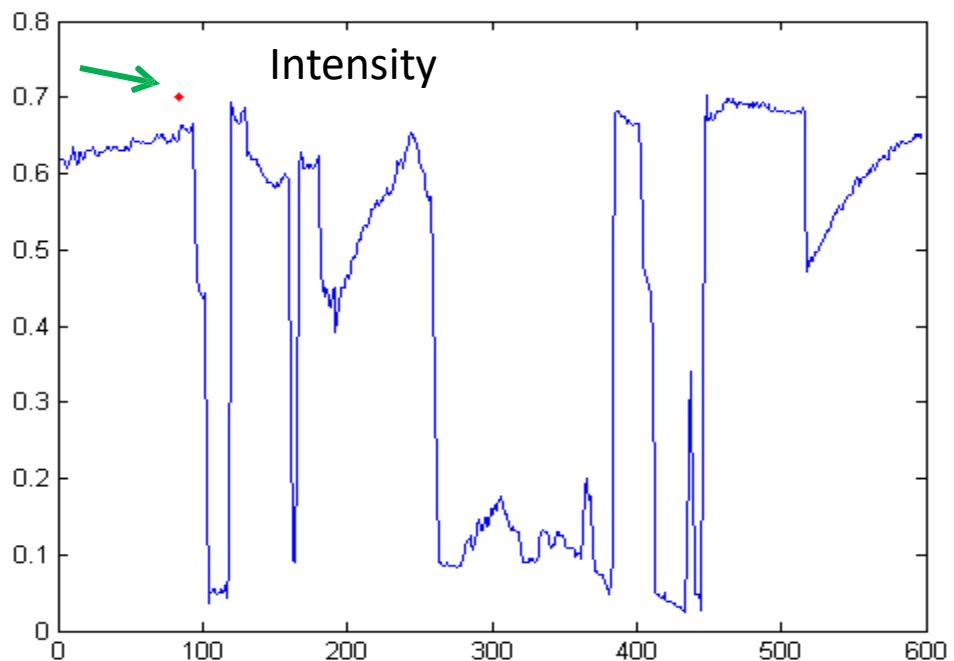
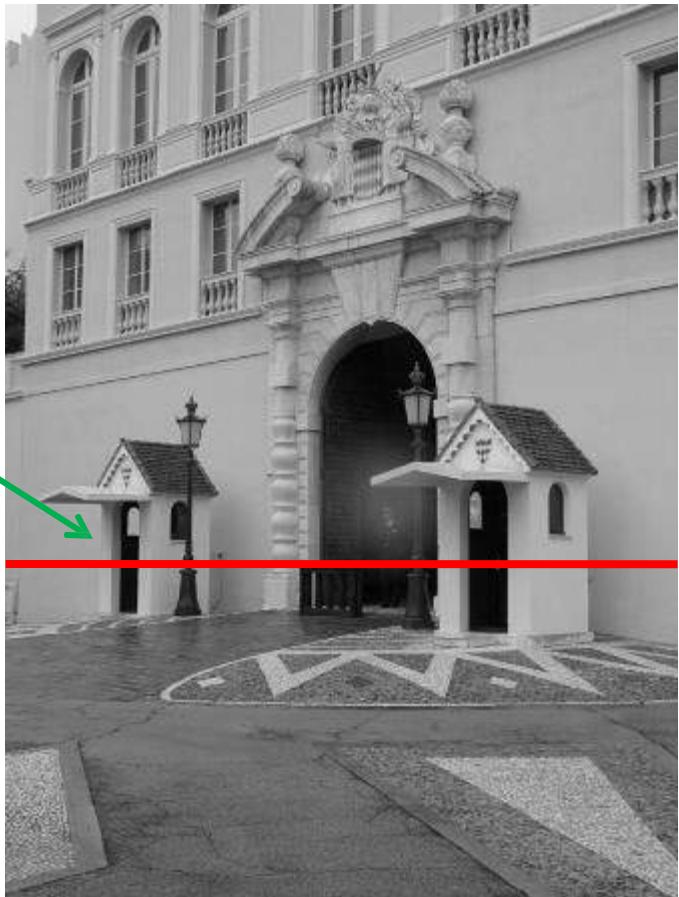
Intensity profile



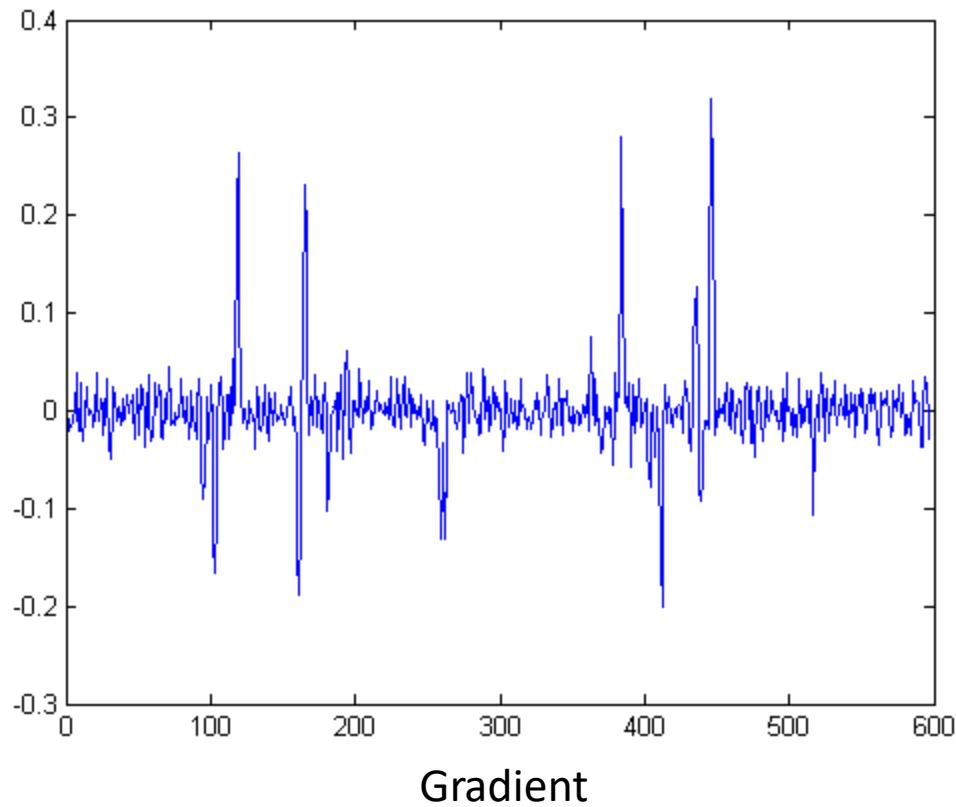
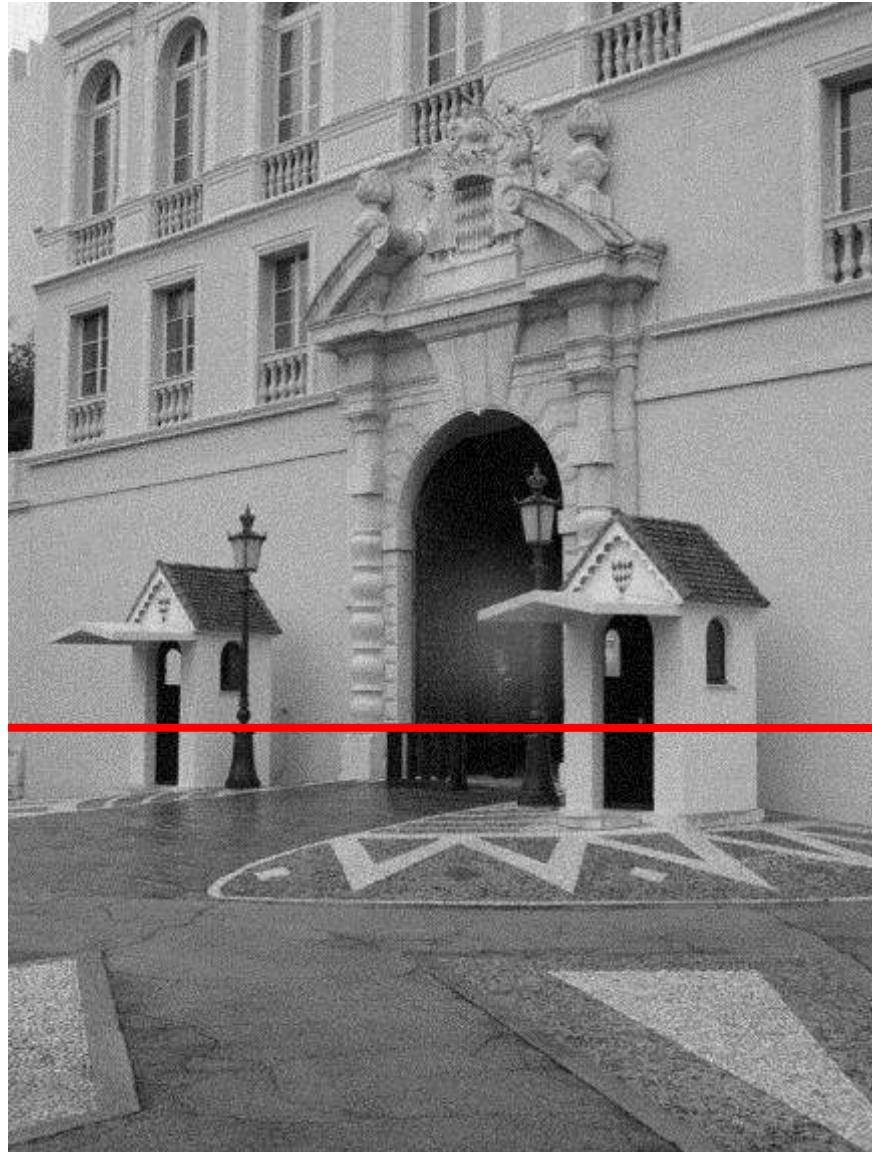
Intensity profile



Intensity profile

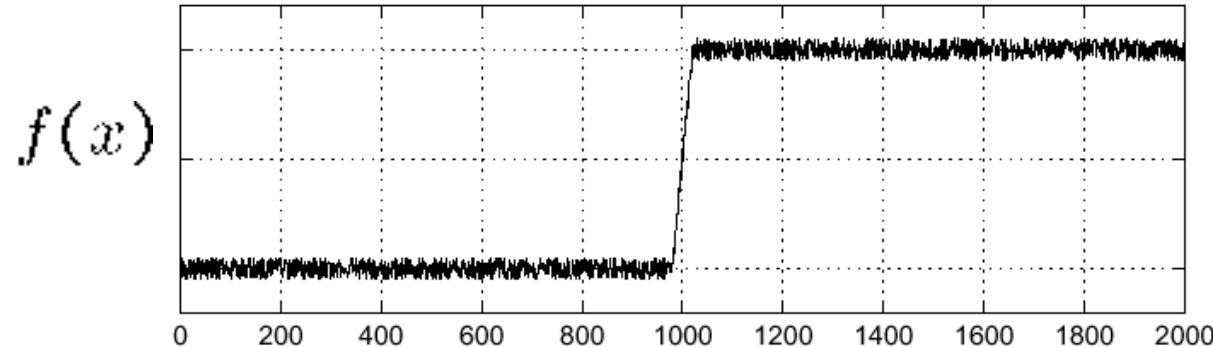


With a little Gaussian noise



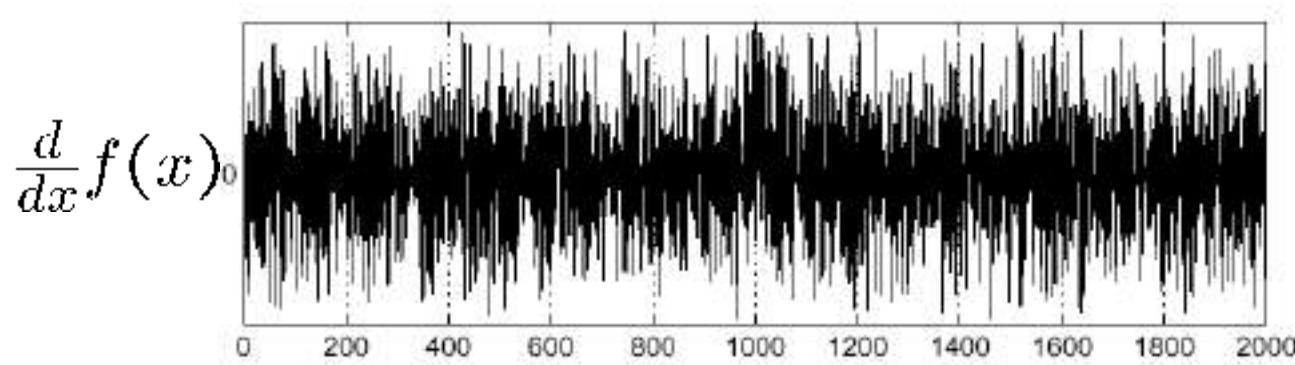
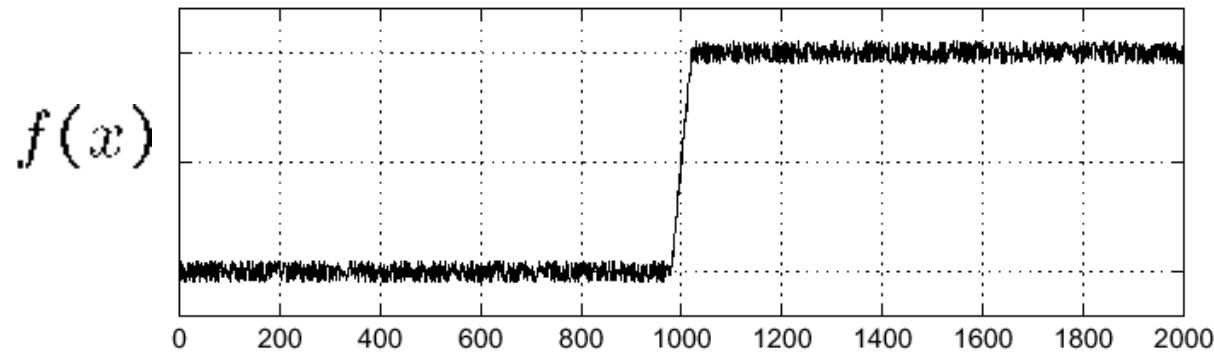
Effects of noise

- Consider a single row or column of the image
 - Plotting intensity as a function of position gives a signal



Effects of noise

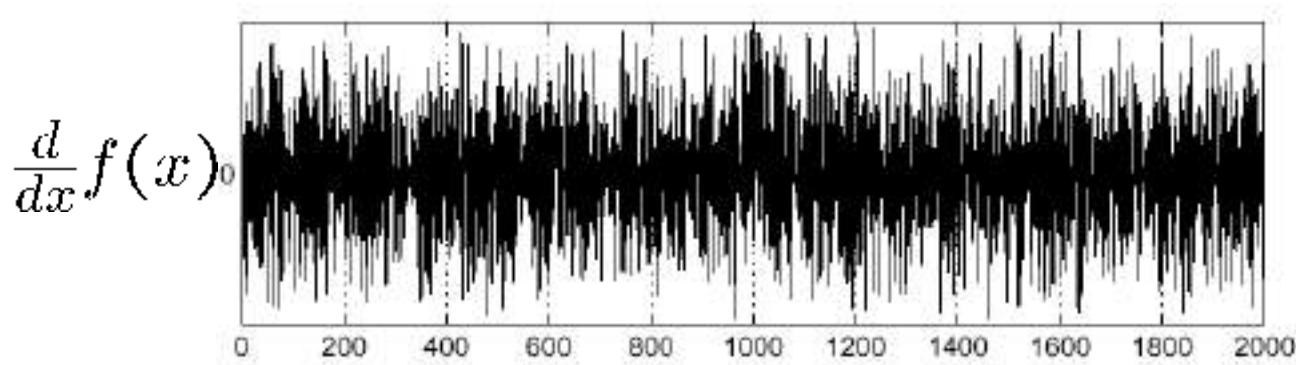
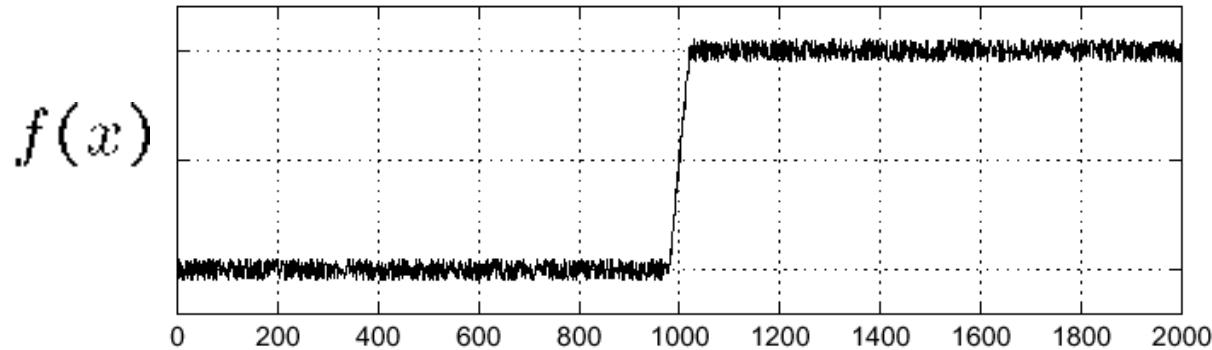
- Consider a single row or column of the image
 - Plotting intensity as a function of position gives a signal



Where is the edge?

Effects of noise

- Consider a single row or column of the image
 - Plotting intensity as a function of position gives a signal

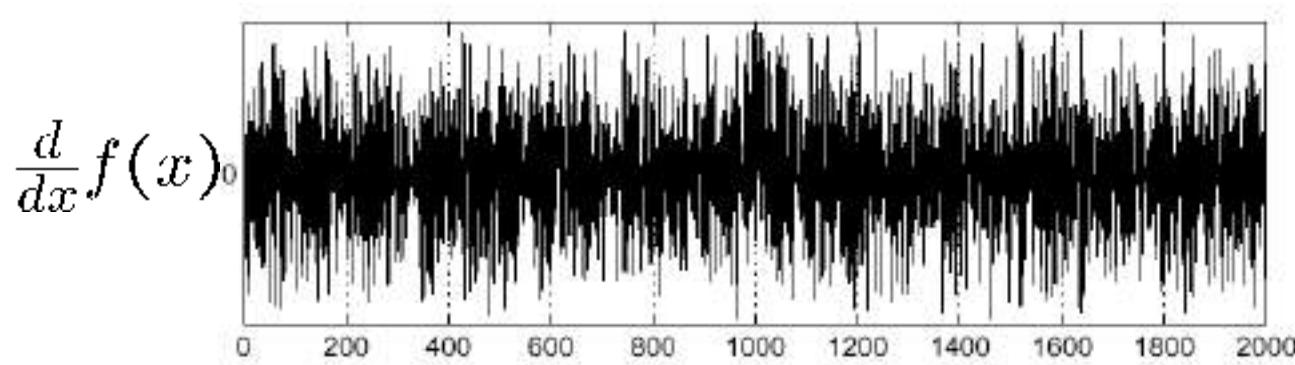
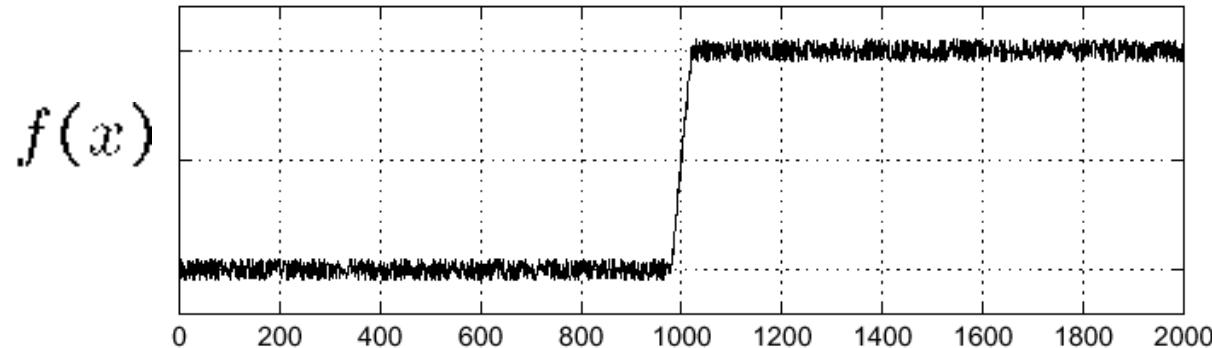


Where is the edge?

The larger the noise the stronger the response

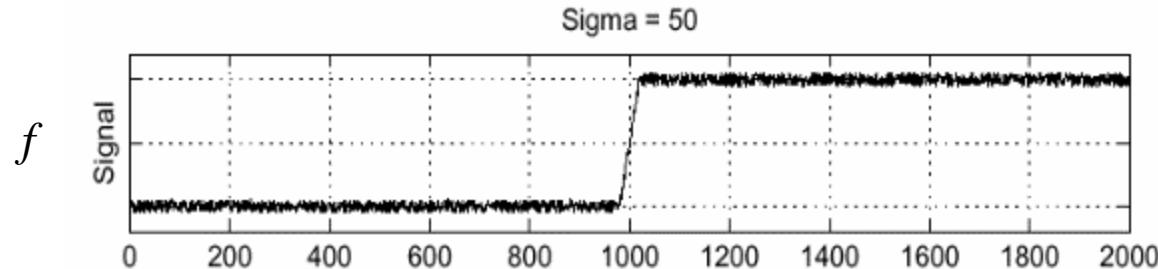
Effects of noise

- Consider a single row or column of the image
 - Plotting intensity as a function of position gives a signal

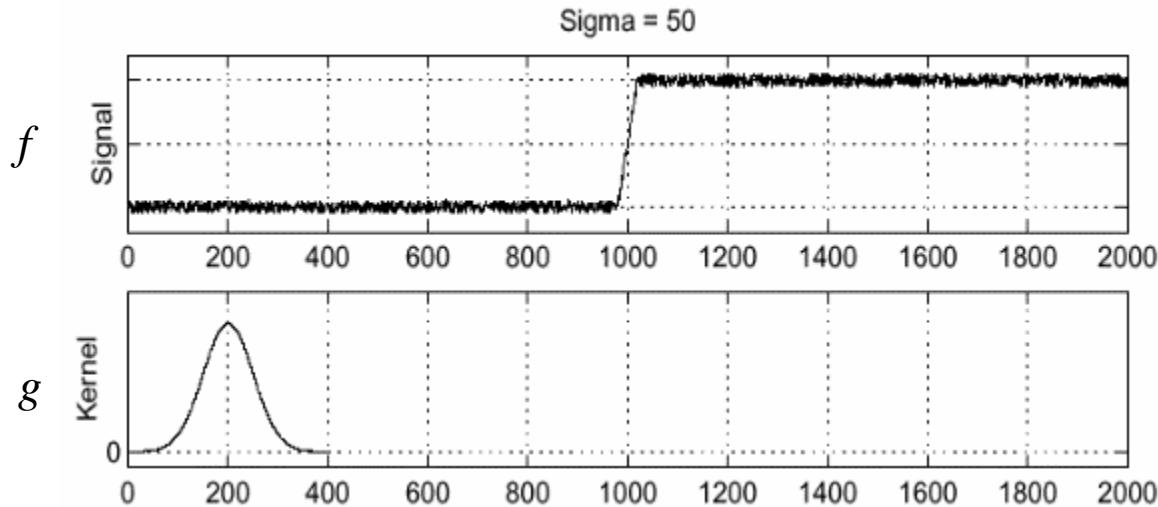


- **What can we do about it?**

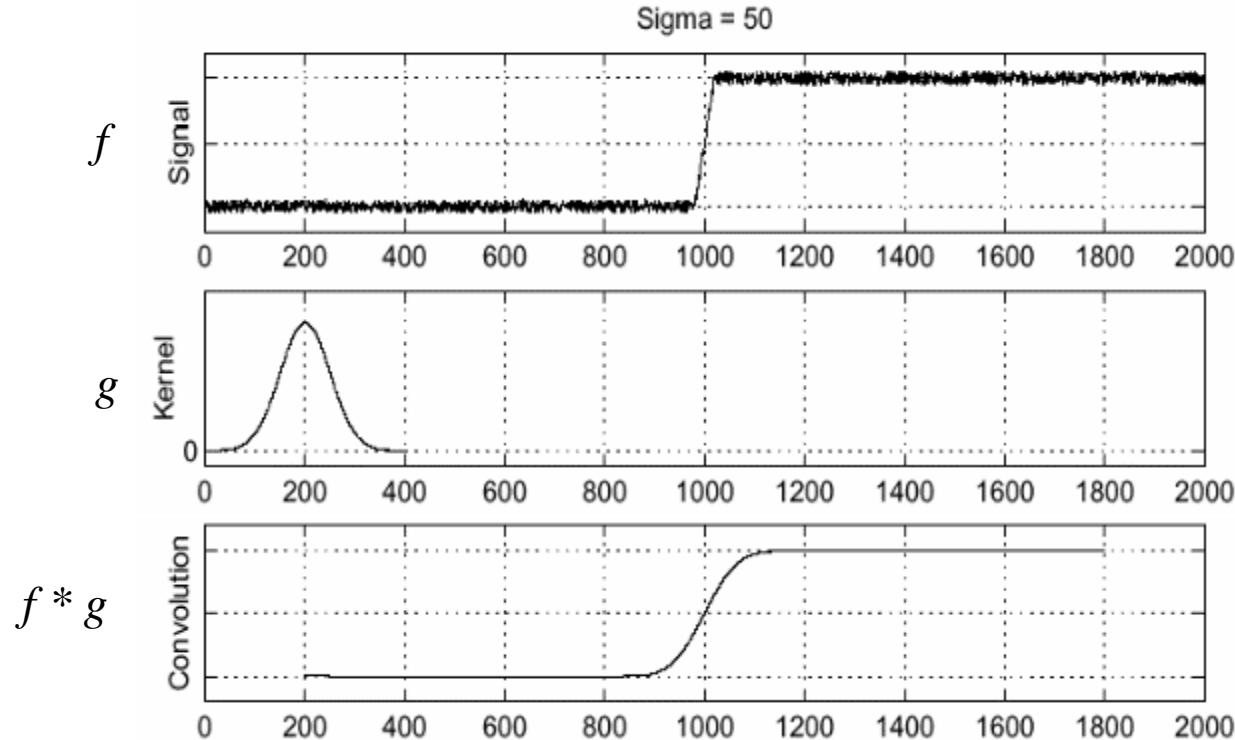
Solution: smooth first



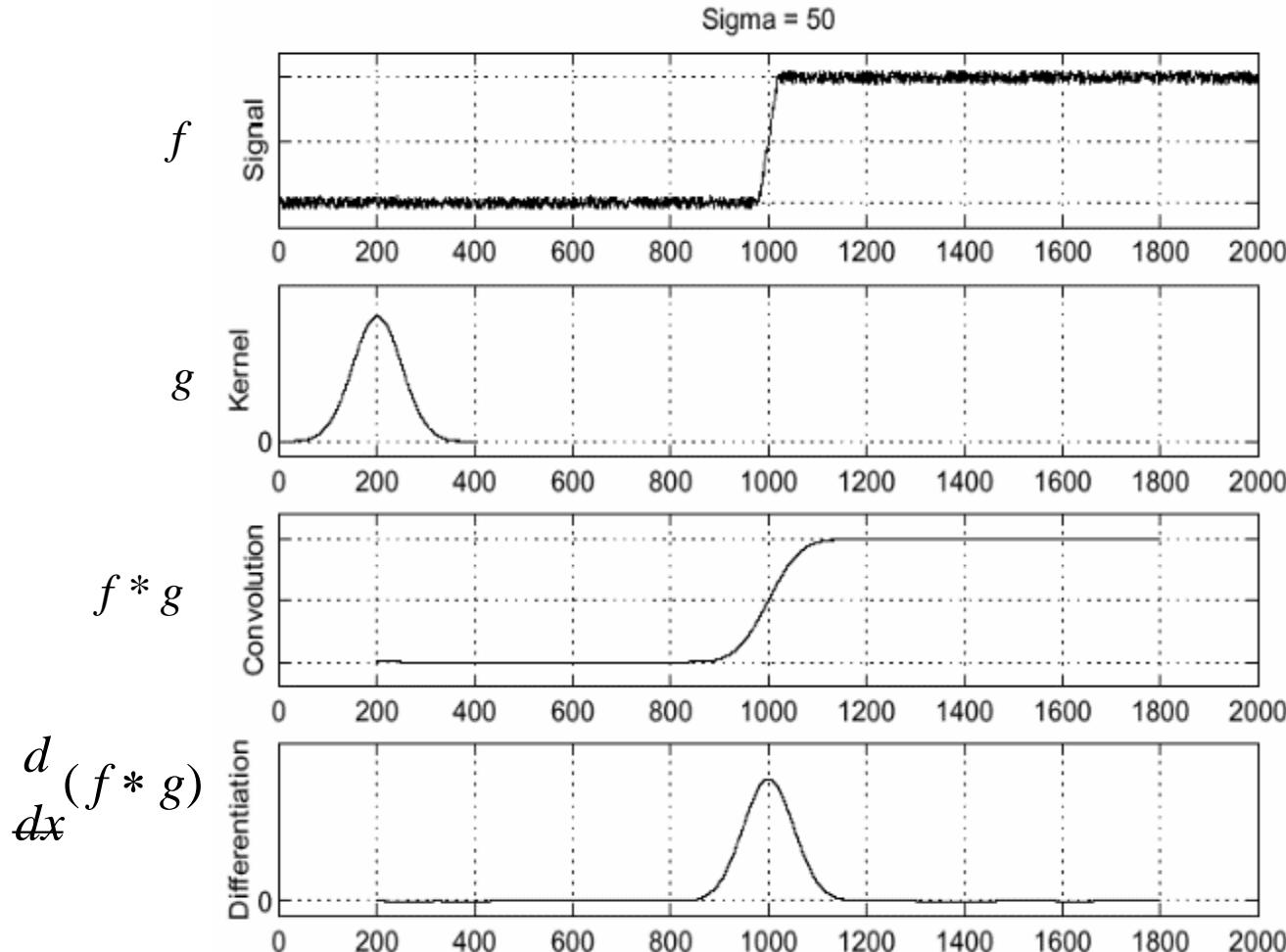
Solution: smooth first



Solution: smooth first



Solution: smooth first



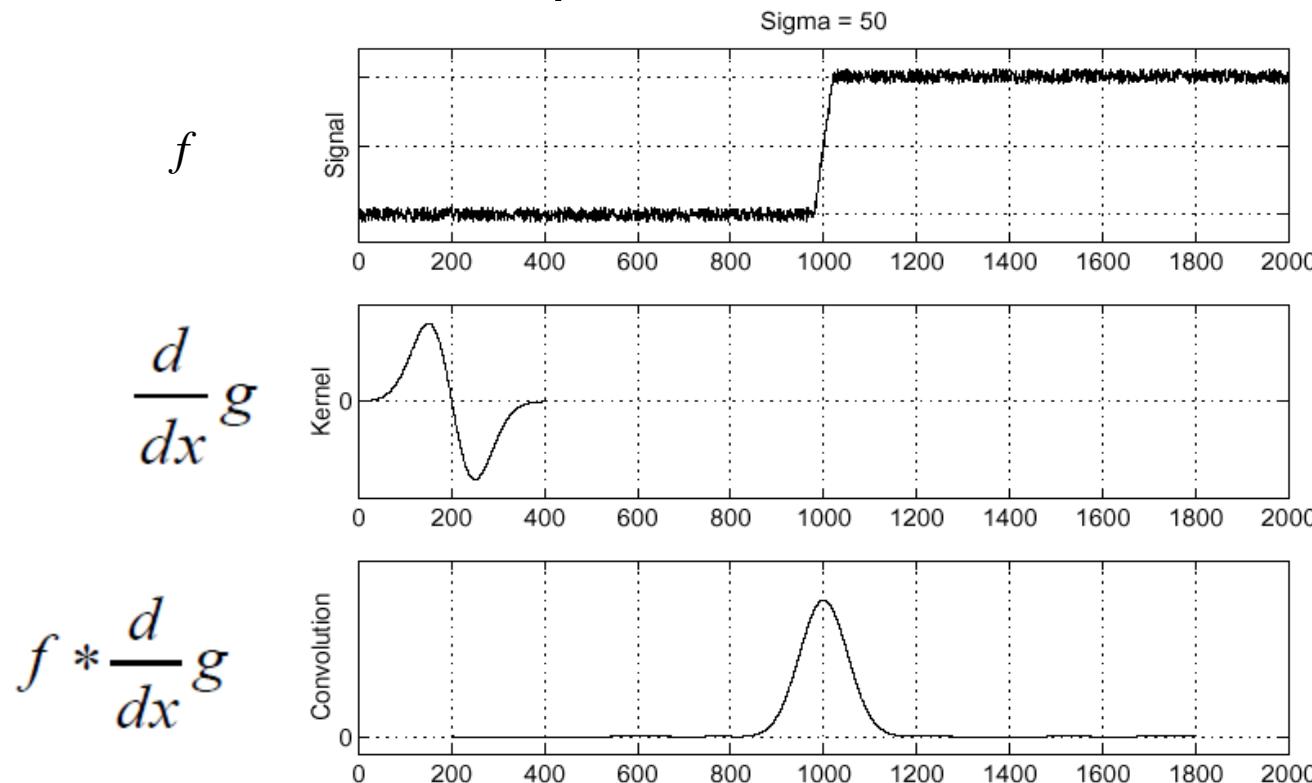
- To find edges, look for peaks in

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

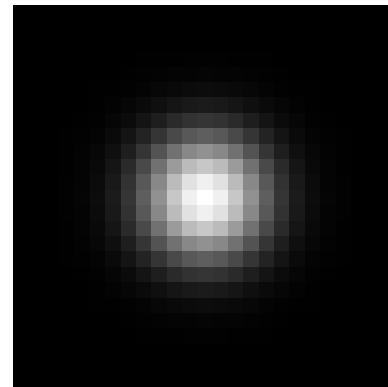
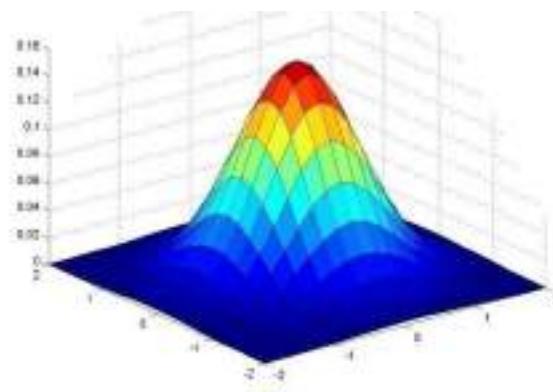
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:
- This saves us one operation:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$

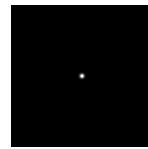
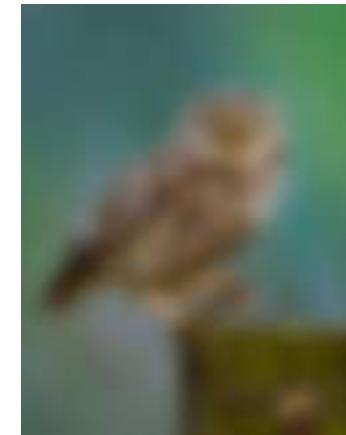
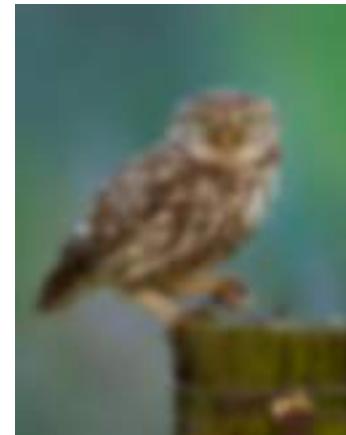


Gaussian Kernel

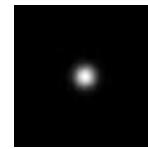


$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Gaussian filters



$\sigma = 1$ pixel



$\sigma = 5$ pixels

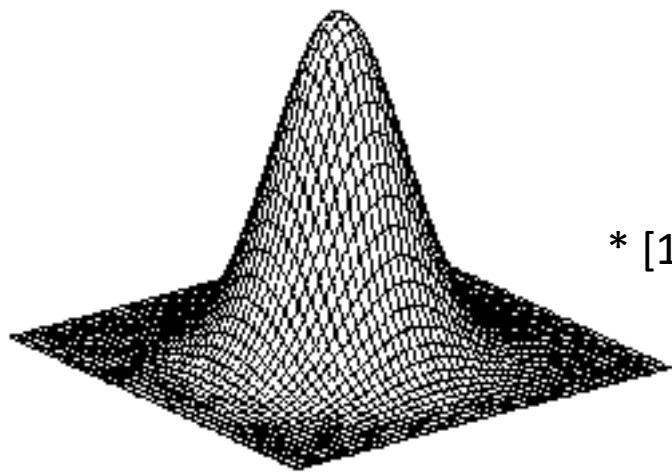


$\sigma = 10$ pixels

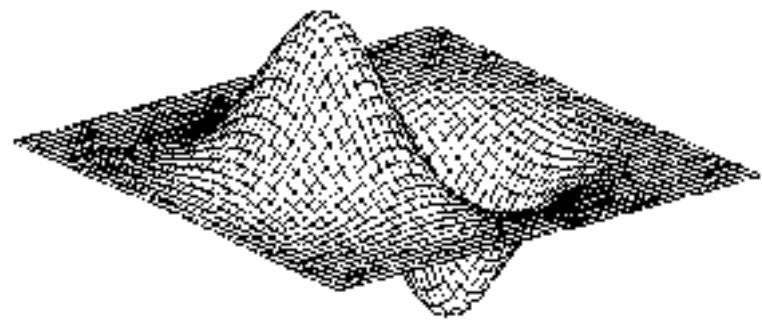


$\sigma = 30$ pixels

Derivative of Gaussian filter



* [1 0 -1] =



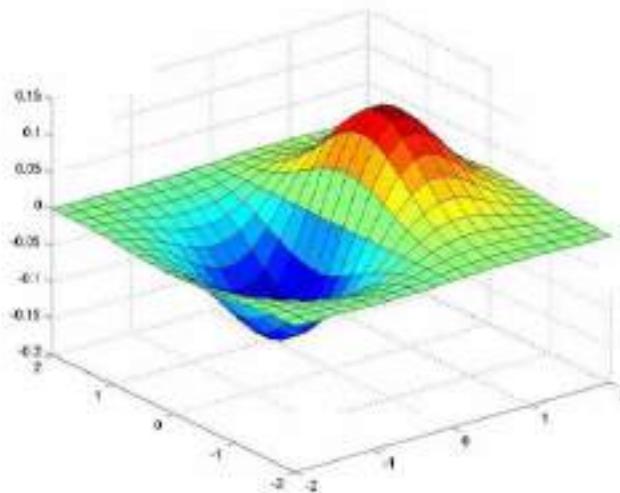
derivative of Gaussian (x)

Gaussian

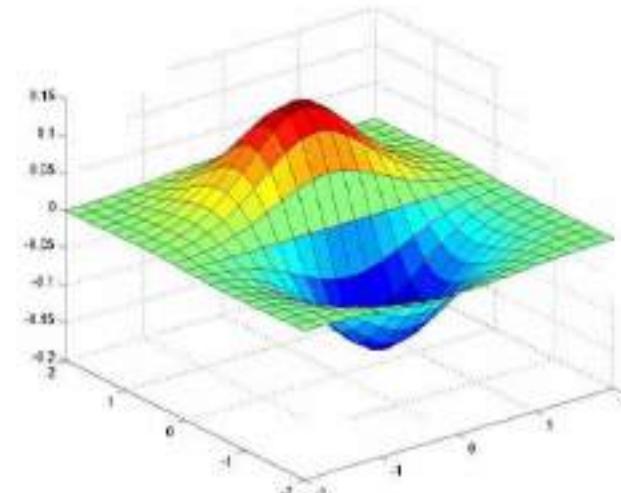
$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

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

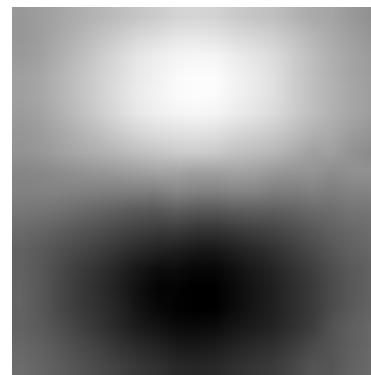
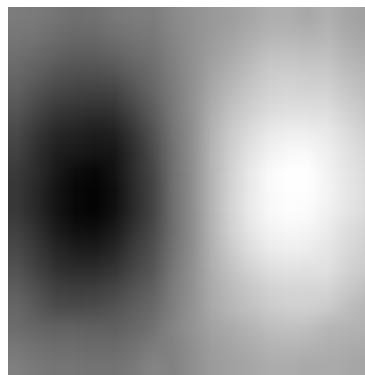
Derivative of Gaussian filter



x-direction



y-direction



Designing an edge detector

- Criteria for a good edge detector:
 - **Good detection:**
 - find all real edges, ignoring noise or other artifacts

Designing an edge detector

- Criteria for a good edge detector:
 - **Good detection:**
 - find all real edges, ignoring noise or other artifacts
 - **Good localization**
 - detect edges as close as possible to the true edges
 - return one point only for each true edge point

Canny edge detector

- The most widely used edge detector

A computational approach to edge detection

[J Canny - IEEE Transactions on pattern analysis and machine ...](#), 1986 - ieeexplore.ieee.org

Abstract: This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for ...

Cited by 27743 Related articles All 27 versions Import into BibTeX Save More

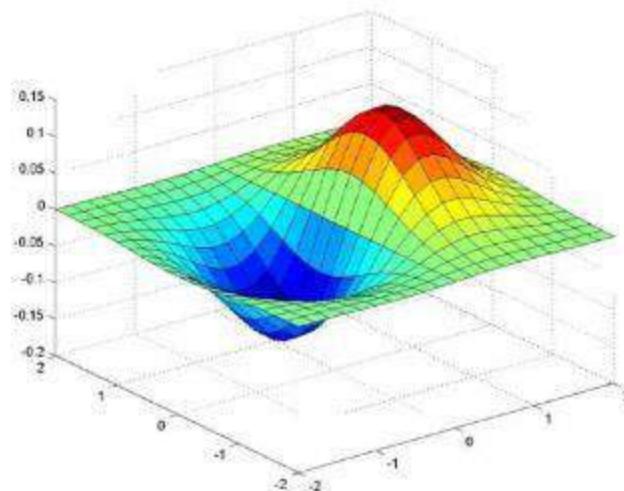
J. Canny, [*A Computational Approach To Edge Detection*](#), IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Example

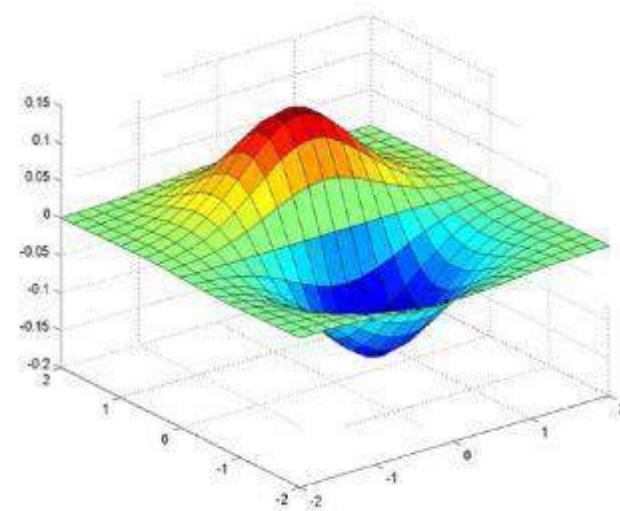


input image (“Lena”)

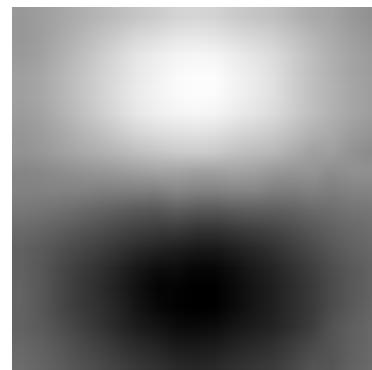
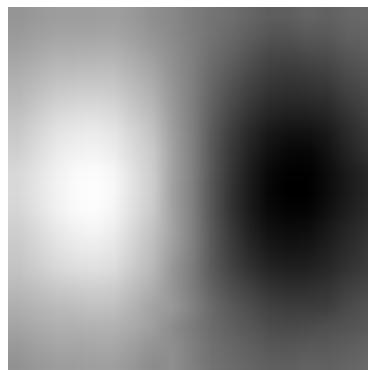
Derivative of Gaussian filter



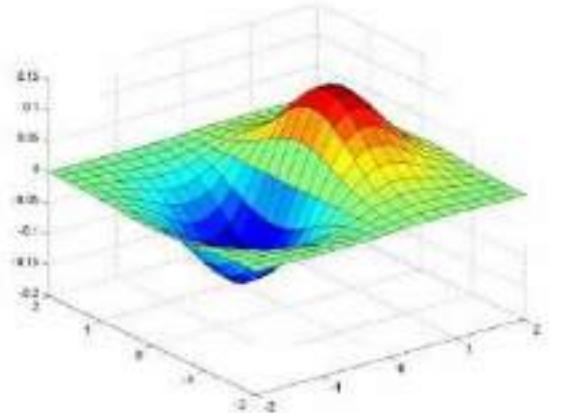
x-direction



y-direction



Compute Gradients (DoG)

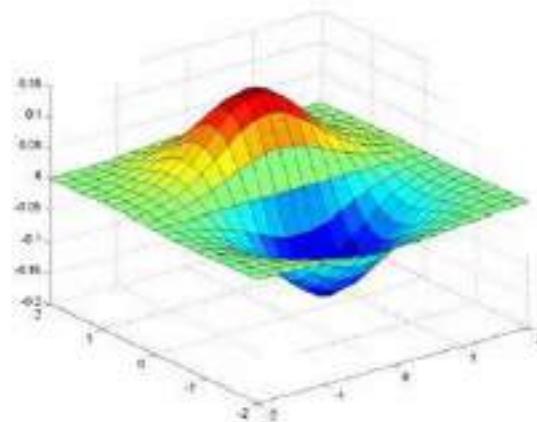
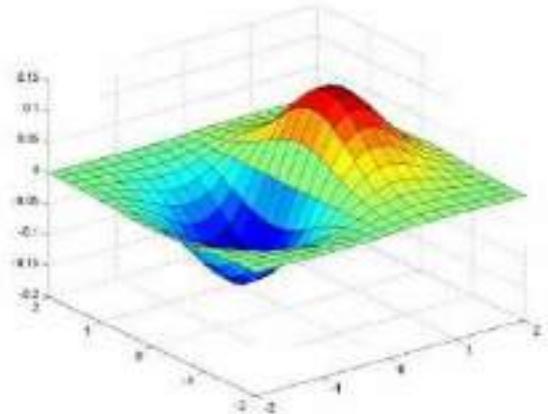


Input Image



X-Derivative of Gaussian

Compute Gradients (DoG)



Input Image

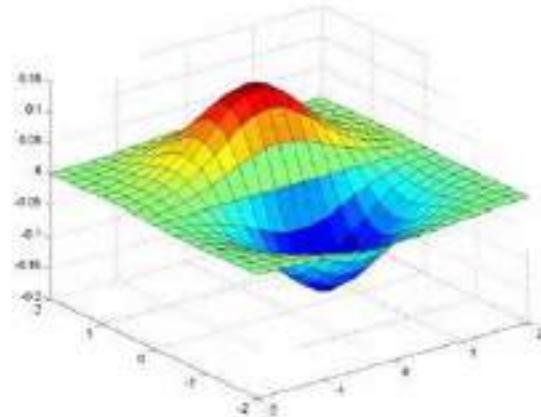
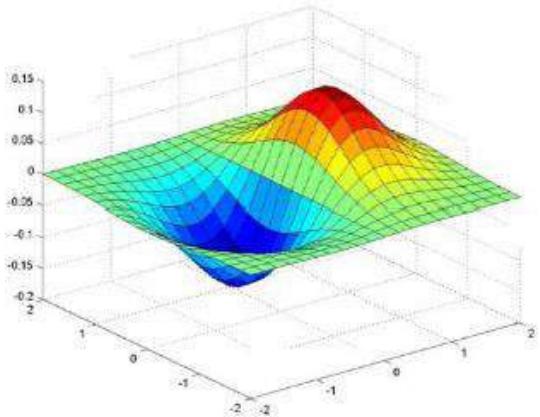


X-Derivative of Gaussian



Y-Derivative of Gaussian

Compute Gradients (DoG)



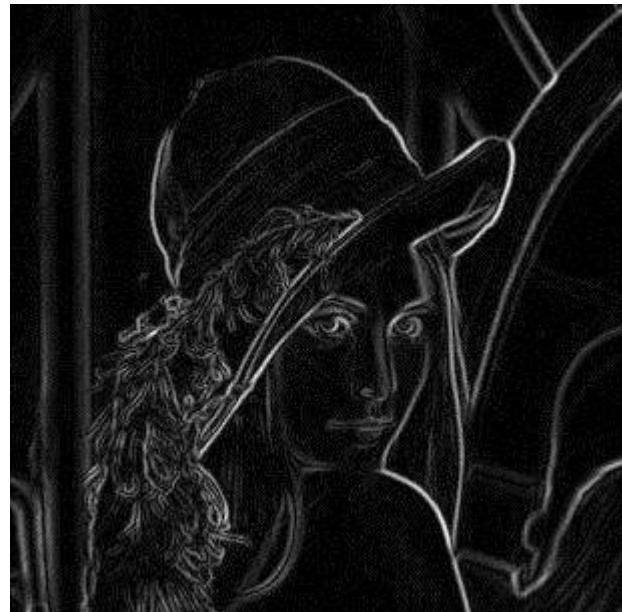
Input Image



X-Derivative of Gaussian



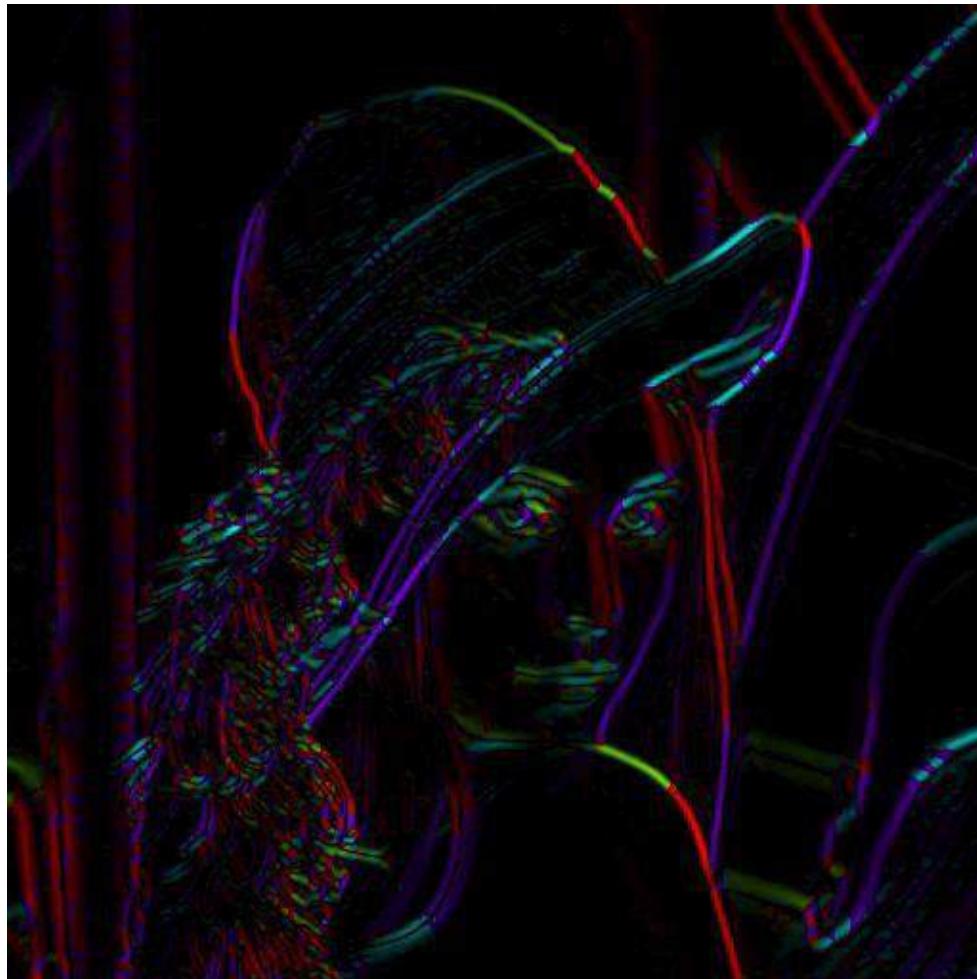
Y-Derivative of Gaussian



Gradient Magnitude

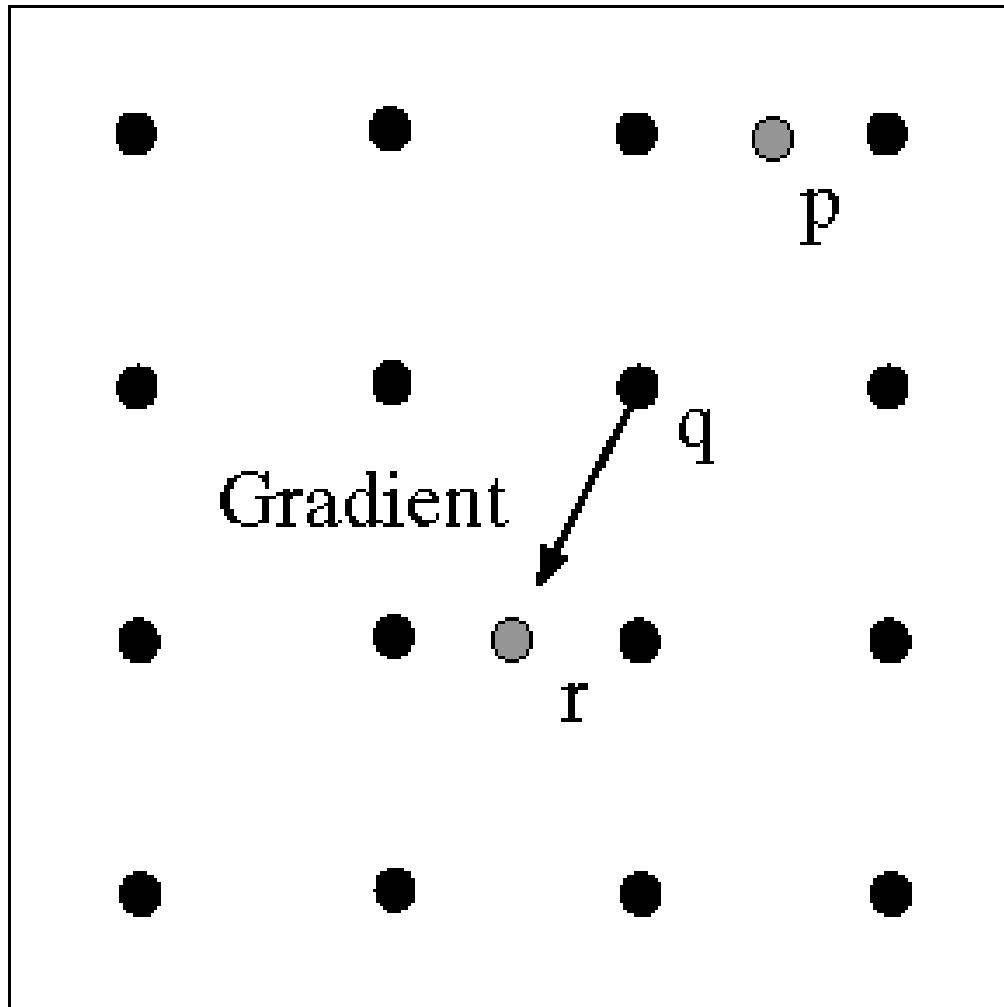
Get Orientation at Each Pixel

- Get orientation

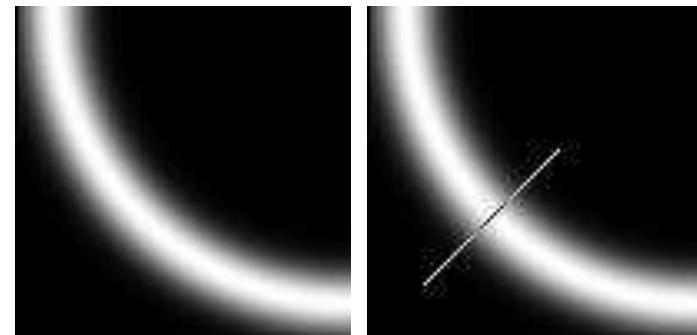


$\theta = \text{atan2}(gy, gx)$

Non-maximum suppression for each orientation

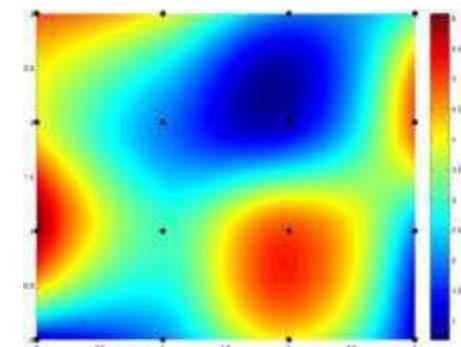
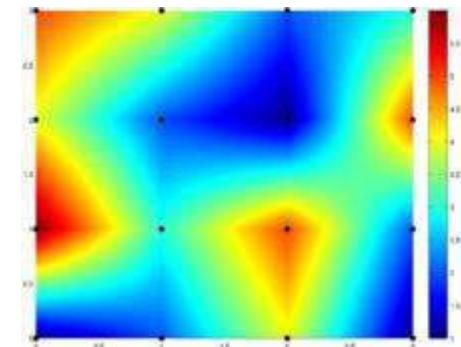
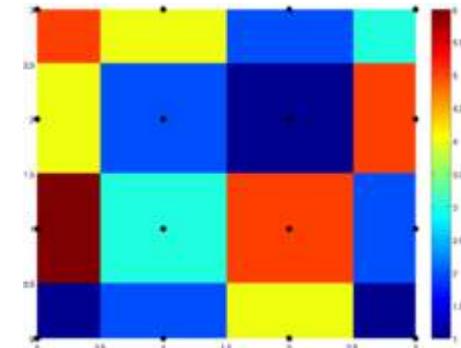


At q , we have a maximum if the value is larger than those at both p and at r .
Interpolate to get these values.



Sidebar: Interpolation options

- ‘nearest’
 - Copy value from nearest known
 - Very fast but creates blocky edges
- ‘bilinear’
 - Weighted average from four nearest known pixels
 - Fast and reasonable results
- ‘bicubic’ (default)
 - Non-linear smoothing over larger area
 - Slower, visually appealing, may create negative pixel values



Before Non-max Suppression



After non-max suppression



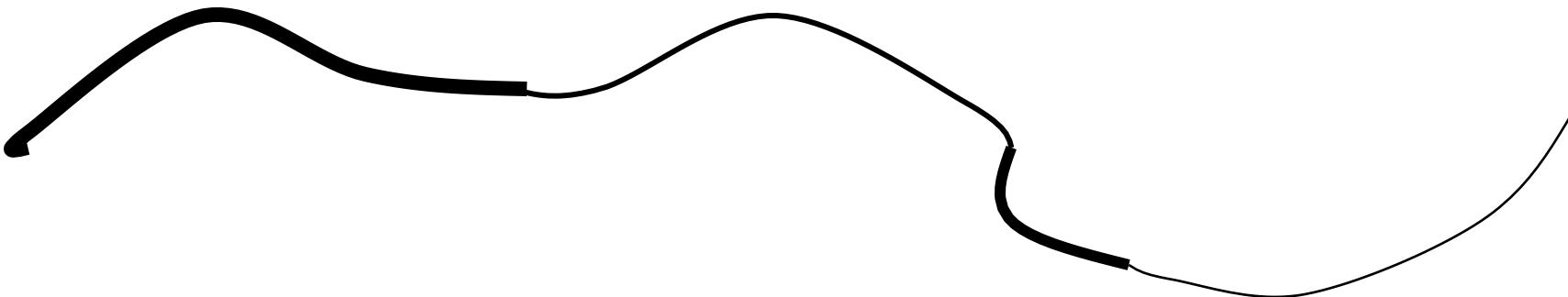
Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels



Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
 - drop-outs? use **hysteresis**
 - use a high threshold to start edge curves and a low threshold to continue them.



Final Canny Edges



Canny edge detector

1. Filter image with x, y derivatives of Gaussian

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
 - Thin multi-pixel wide “ridges” down to single pixel width

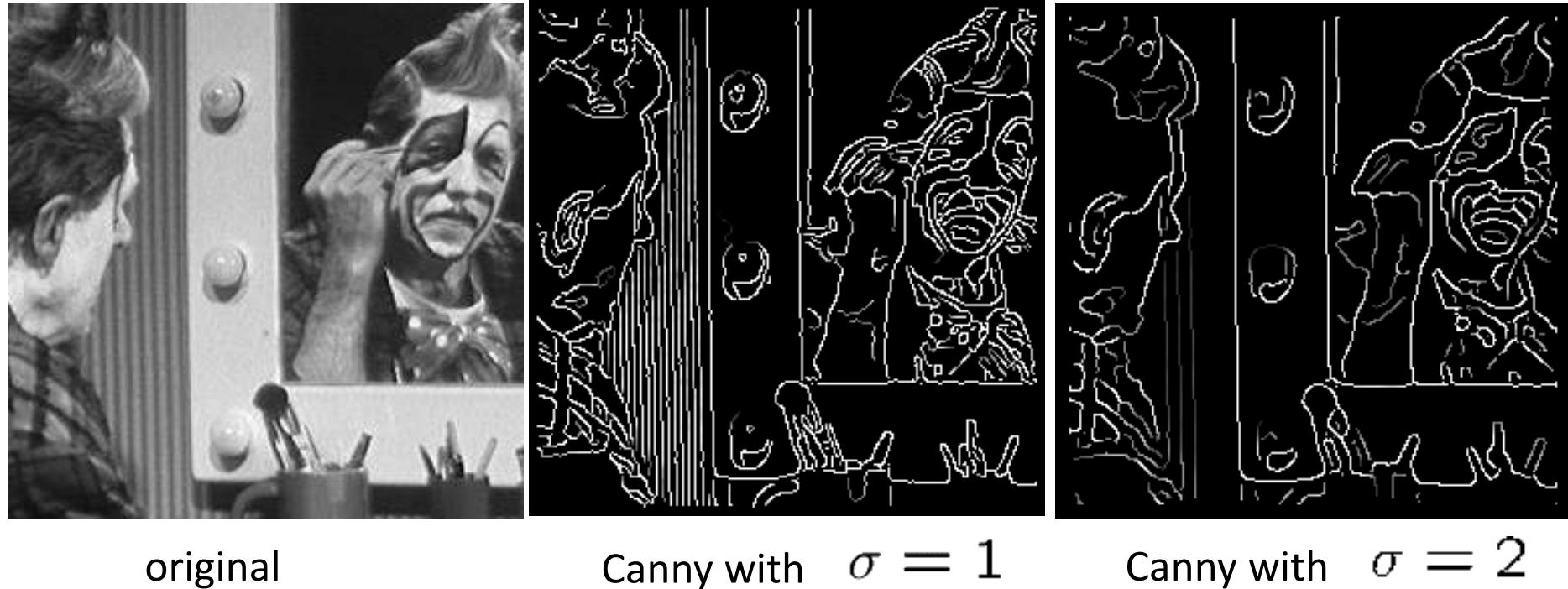
Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
 - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
 2. Find magnitude and orientation of gradient
 3. Non-maximum suppression:
 - Thin multi-pixel wide “ridges” down to single pixel width
 4. Thresholding and linking (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them
-
- MATLAB: `edge(image, 'canny')`

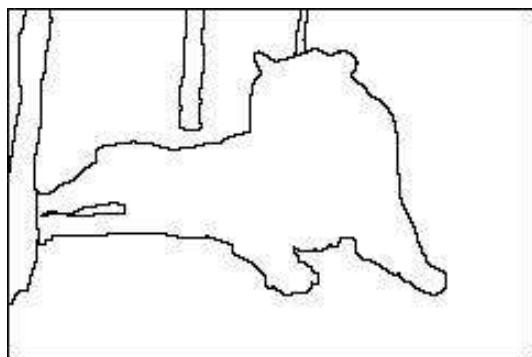
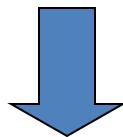
Effect of σ (Gaussian kernel spread/size)



The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Why edges?



Reduce dimensionality of data

Preserve content information

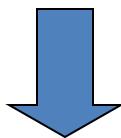
Useful in applications such as:

object detection

structure from motion

tracking

Why not edges?



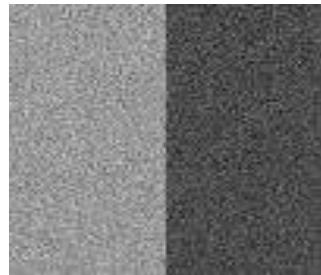
But, not that useful, **why**?

Difficulties:

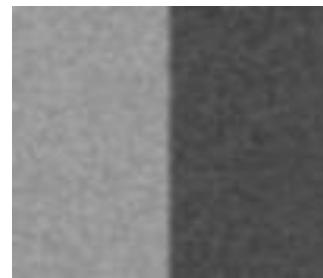
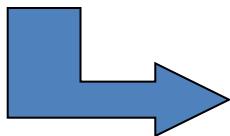
1. Modeling assumptions
2. Parameters
3. Multiple sources of information
(brightness, color, texture, ...)
4. Real world conditions

Is edge detection even well defined?

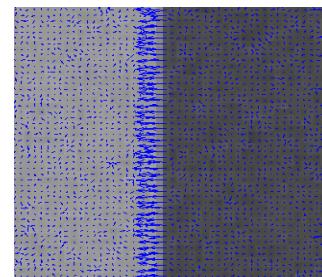
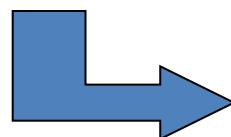
Canny edge detection



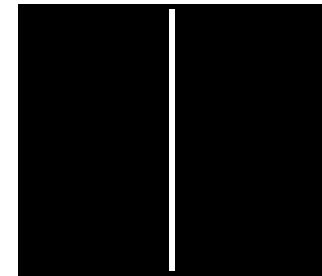
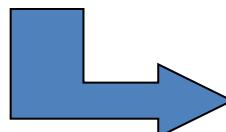
1. smooth



2. gradient

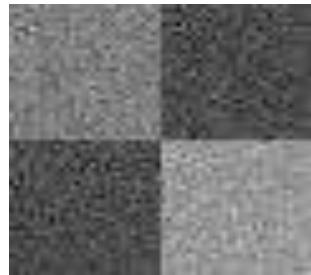


3. thresh, suppress, link

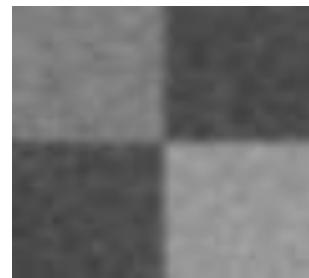
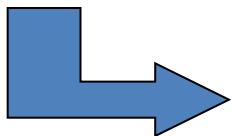


Canny is optimal w.r.t. some model.

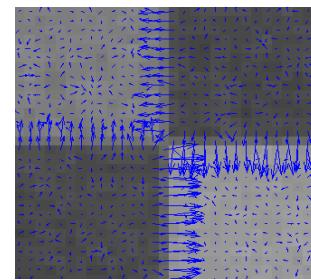
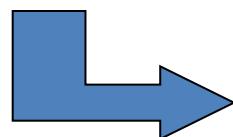
Canny edge detection



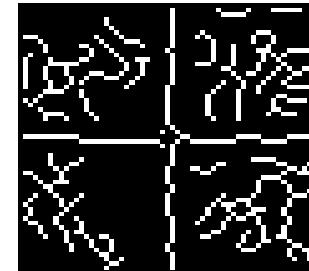
1. smooth



2. gradient



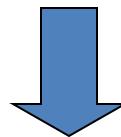
3. thresh, suppress, link



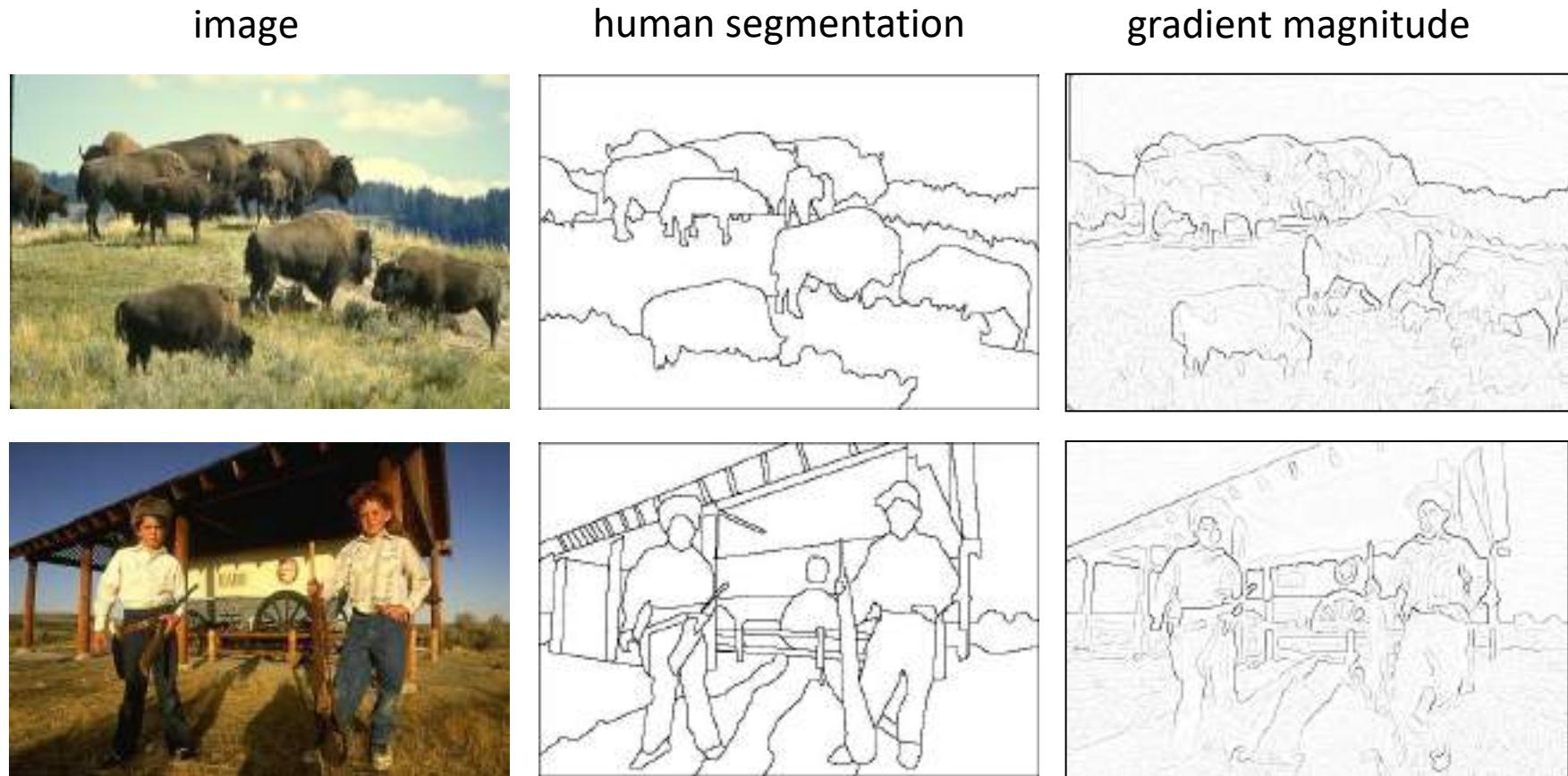
And yet...

Canny difficulties

1. Modeling assumptions
Step edges, junctions, etc.
2. Parameters
Scales, threshold, etc.
3. Multiple sources of information
Only handles brightness
4. Real world conditions
Gaussian iid noise? Texture...



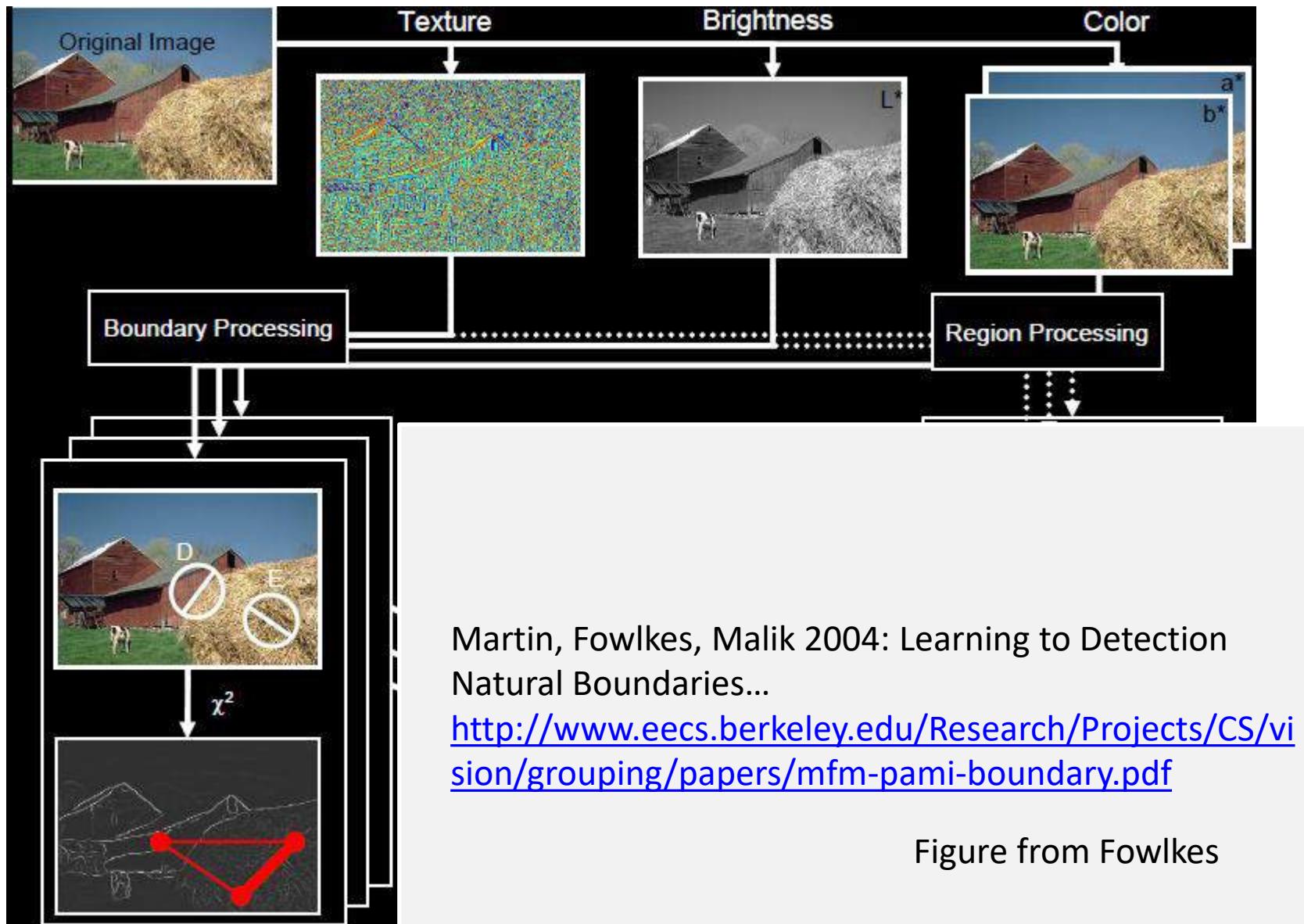
Learning to detect boundaries



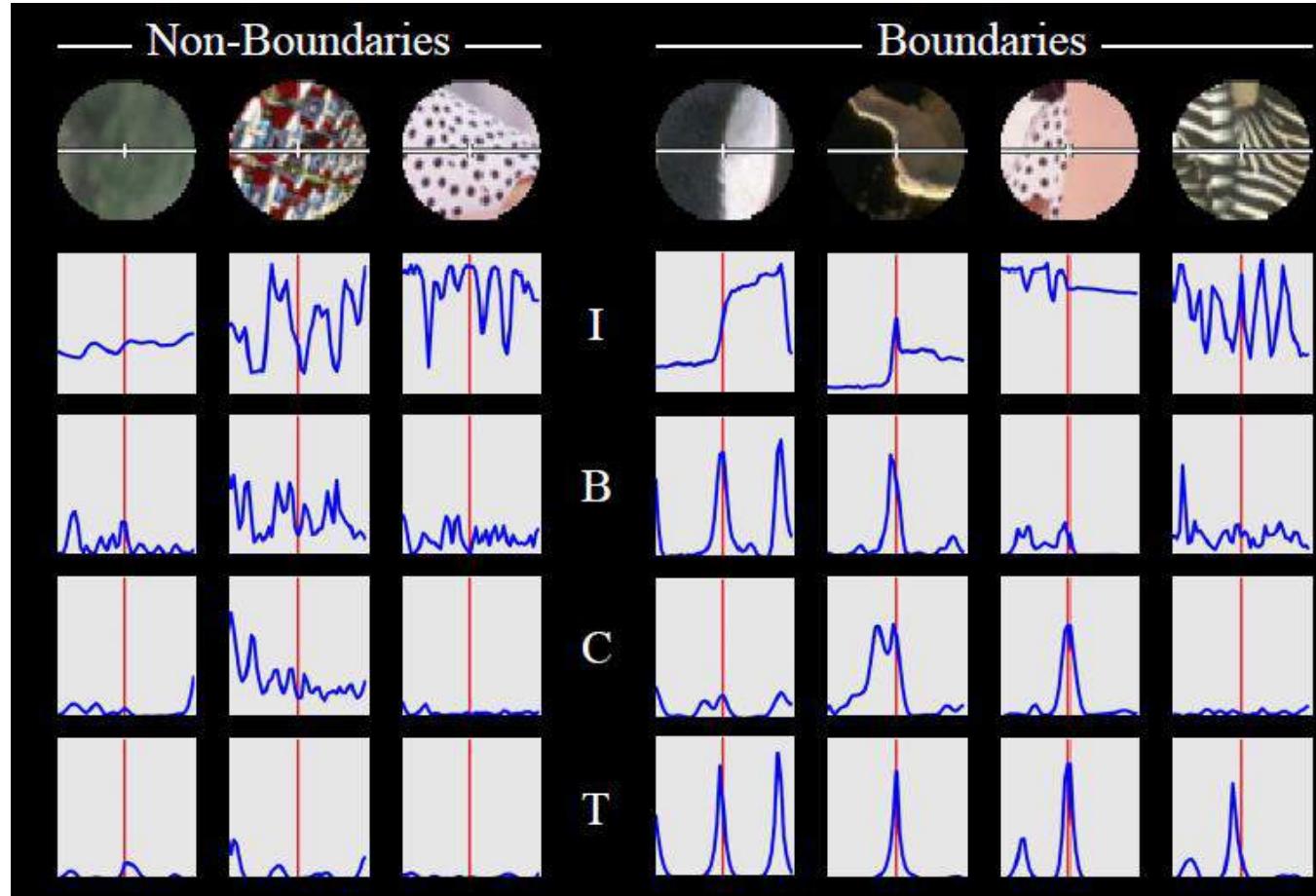
Berkeley segmentation database:

<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/>

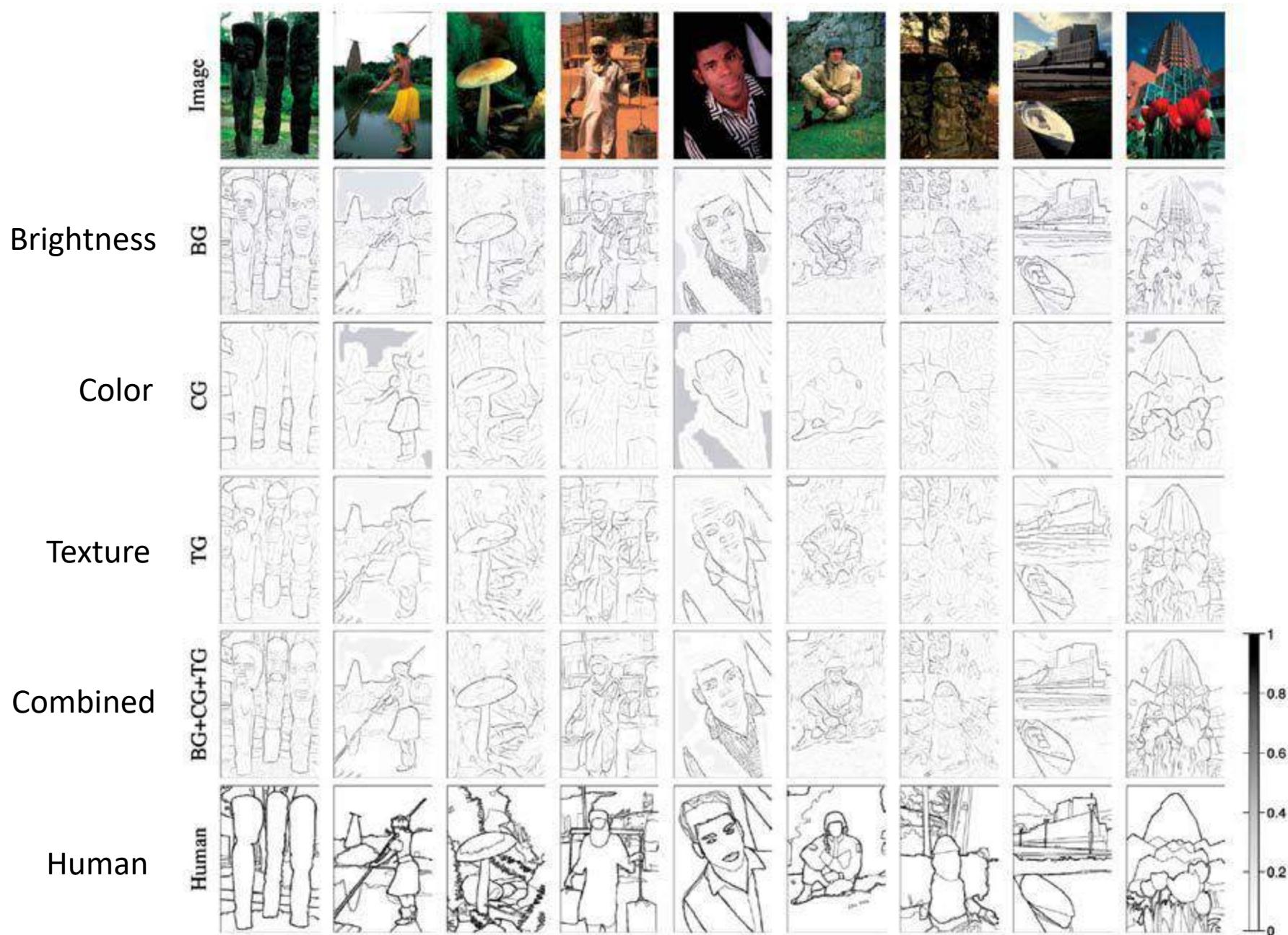
pB boundary detector



pB Boundary Detector



- Estimate Posterior probability of boundary passing through centre point based on local patch based features
- Using a Supervised Learning based framework



Features

Brightness oriented energy,

$$\text{OE}_{\theta,\sigma} = (I * f_{\theta,\sigma}^e)^2 + (I * f_{\theta,\sigma}^o)^2$$

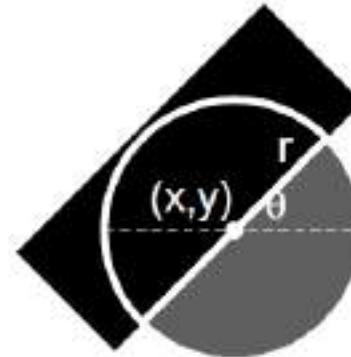
← → Gaussian second derivative

Gradients computed from two disc halves:

Brightness gradient

Color gradient

Texture gradient



Texture features



(a)

Filterbank (13 filters)

Martin, Fowlkes, Malik, 2004: Berkeley (Pb) edge detector

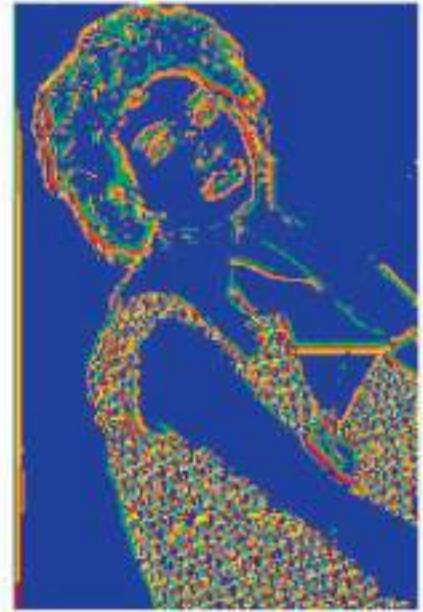
(b)

Universal textons (64)



(c)

Image



(d)

Texton map (color-coded)

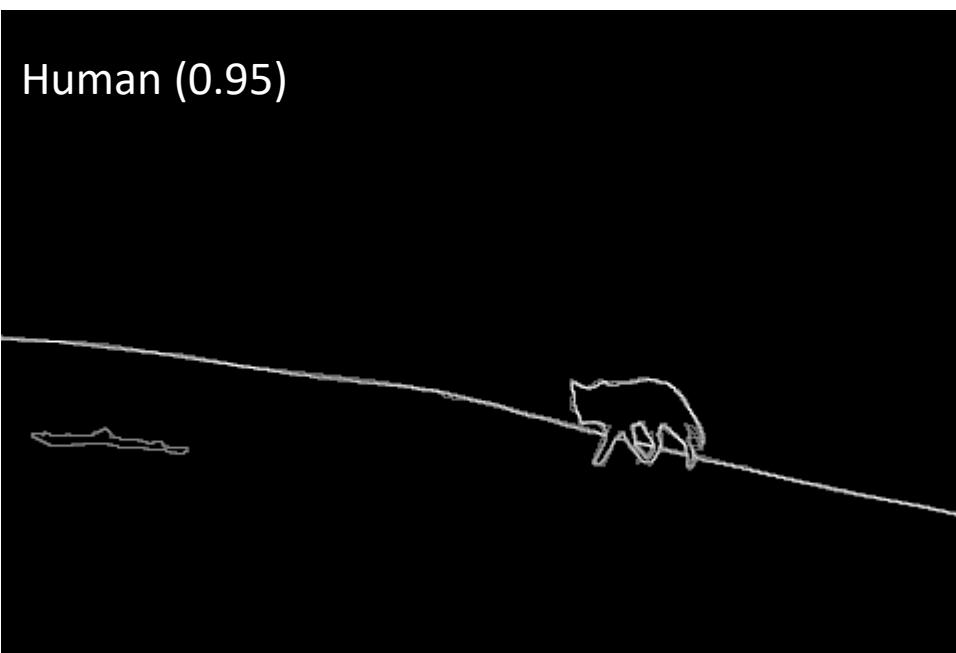
Localization

- edges (due to large filters) are poorly localized; double peaks
- Improve Localization by using derived feature
- Divide by distance to nearest maximum

$$\hat{f}(x) = \bar{f}(x) \cdot \left(\frac{-f''(x)}{|f'(x)| + \epsilon} \right)$$

where $f(x)$ is feature and the estimated distance to the nearest maximum of $f(x)$ is
 $d(x) = -|f'(x)|/f''(x)$

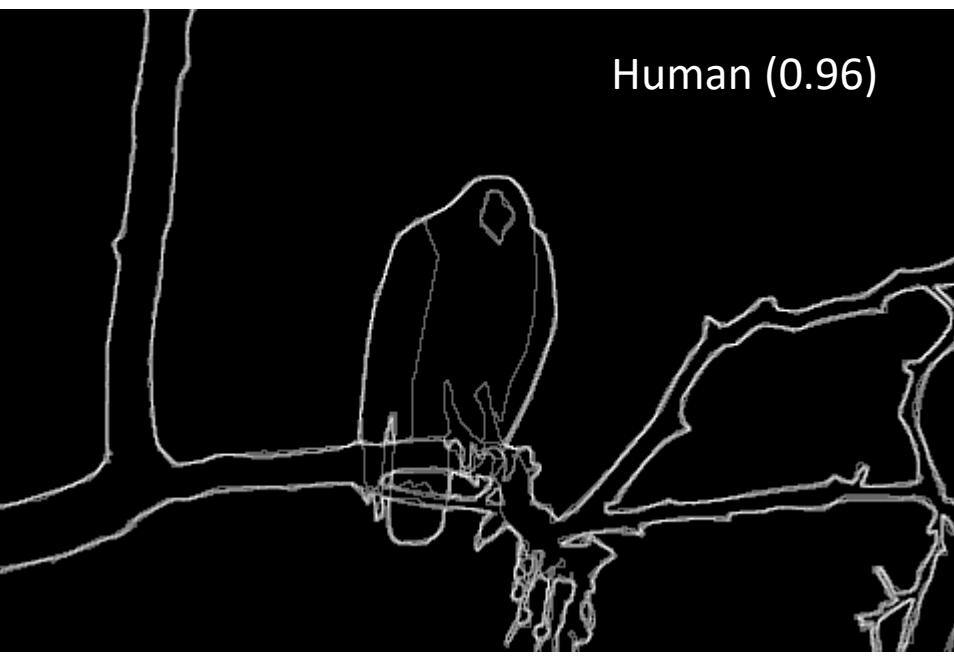
Results



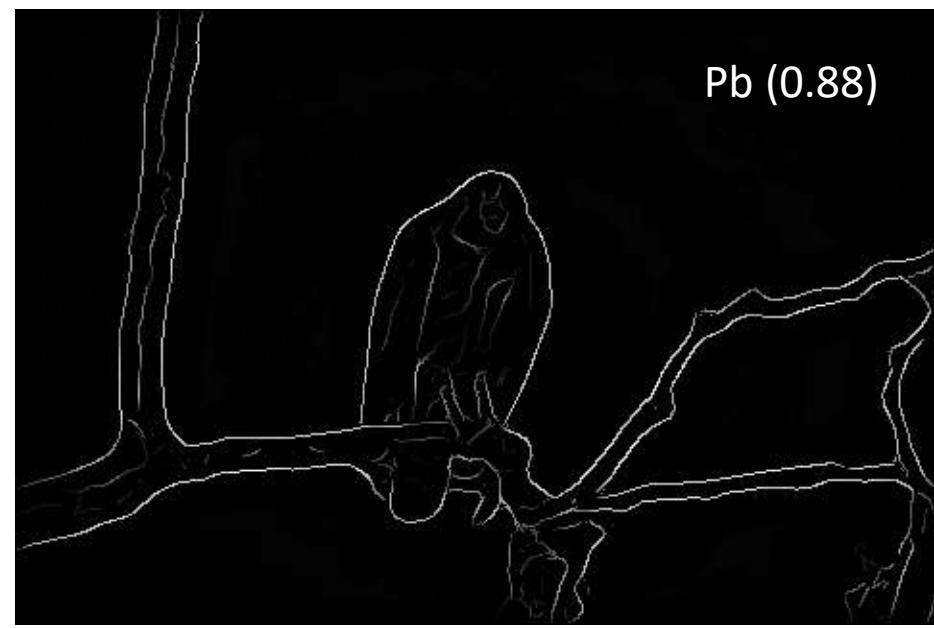
Results

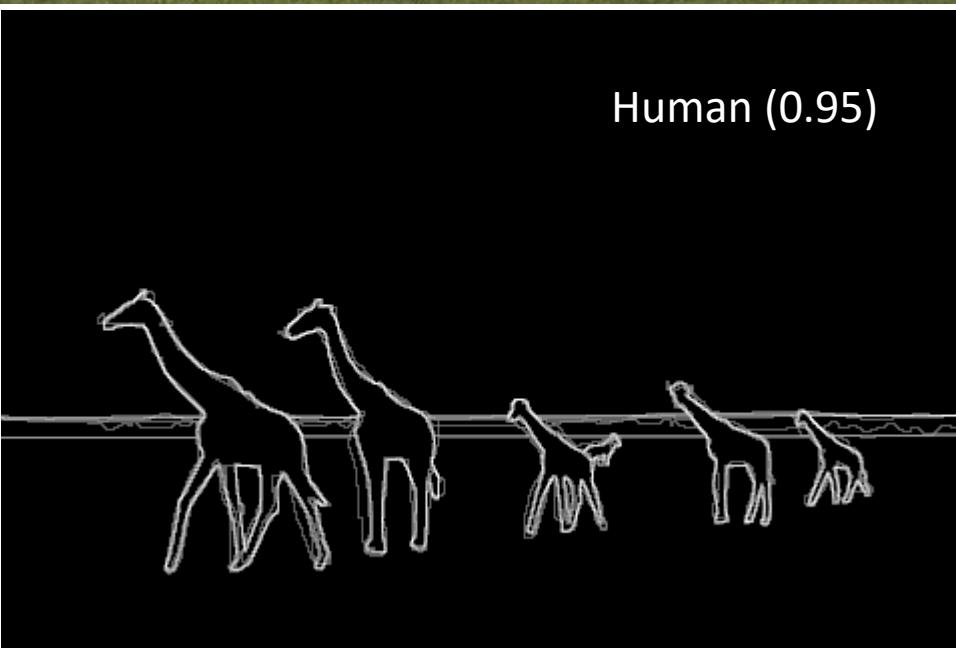


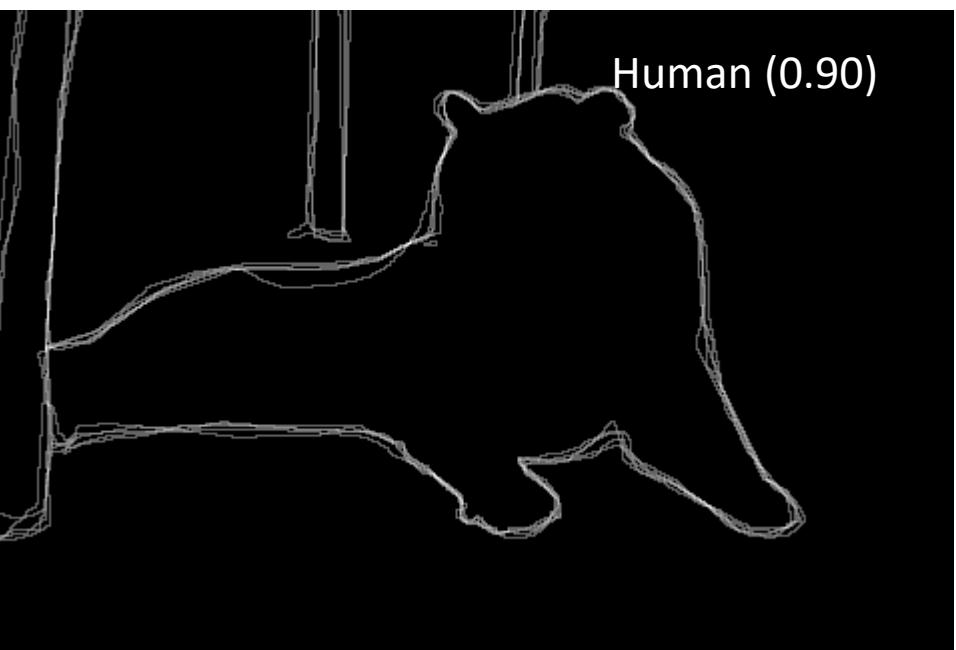
Human (0.96)



Pb (0.88)







For more:

<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/108082-color.html>

State of edge detection

- Local edge detection is mostly solved
 - Intensity gradient, color, texture
- Often used in combination with object detectors or region classifiers
- Deep learning approach is more common nowadays

Finding straight lines



Finding line segments using connected components

1. Compute canny edges
 - Compute: gx, gy (DoG in x,y directions)
 - Compute: $\theta = \text{atan}(gy / gx)$

Finding line segments using connected components

1. Compute canny edges
 - Compute: gx, gy (DoG in x,y directions)
 - Compute: $\theta = \text{atan}(gy / gx)$
2. Assign each edge to one of 8 directions

Finding line segments using connected components

1. Compute canny edges
 - Compute: gx, gy (DoG in x, y directions)
 - Compute: $\theta = \text{atan}(gy / gx)$
2. Assign each edge to one of 8 directions
3. For each direction d , get edgelets:
 - find connected components for edge pixels with directions in $\{d-1, d, d+1\}$

Finding line segments using connected components

1. Compute canny edges
 - Compute: gx, gy (DoG in x, y directions)
 - Compute: $\theta = \text{atan}(gy / gx)$
2. Assign each edge to one of 8 directions
3. For each direction d , get edgelets:
 - find connected components for edge pixels with directions in $\{d-1, d, d+1\}$
4. Compute straightness and theta of edgelets using eig of x, y 2nd moment matrix of their points

Finding line segments using connected components

1. Compute canny edges
 - Compute: gx, gy (DoG in x, y directions)
 - Compute: $\theta = \text{atan}(gy / gx)$
2. Assign each edge to one of 8 directions
3. For each direction d , get edgelets:
 - find connected components for edge pixels with directions in $\{d-1, d, d+1\}$
4. Compute straightness and theta of edgelets using eig of x, y 2nd moment matrix of their points

$$\mathbf{M} = \begin{bmatrix} \sum (x - \mu_x)^2 & \sum (x - \mu_x)(y - \mu_y) \\ \sum (x - \mu_x)(y - \mu_y) & \sum (y - \mu_y)^2 \end{bmatrix} \quad [v, \lambda] = \text{eig}(\mathbf{M})$$

Larger eigenvector
↓

$$\theta = \text{atan } 2(v(2,2), v(1,2))$$
$$conf = \lambda_2 / \lambda_1$$

Finding line segments using connected components

1. Compute canny edges
 - Compute: gx, gy (DoG in x, y directions)
 - Compute: $\theta = \text{atan}(gy / gx)$
2. Assign each edge to one of 8 directions
3. For each direction d , get edgelets:
 - find connected components for edge pixels with directions in $\{d-1, d, d+1\}$
4. Compute straightness and theta of edgelets using eig of x, y 2nd moment matrix of their points

$$\mathbf{M} = \begin{bmatrix} \sum (x - \mu_x)^2 & \sum (x - \mu_x)(y - \mu_y) \\ \sum (x - \mu_x)(y - \mu_y) & \sum (y - \mu_y)^2 \end{bmatrix} \quad [v, \lambda] = \text{eig}(\mathbf{M})$$

Larger eigenvector
↓

$$\theta = \text{atan } 2(v(2,2), v(1,2))$$
$$conf = \lambda_2 / \lambda_1$$

5. Threshold on straightness, store segment

Canny lines → ... → straight edges

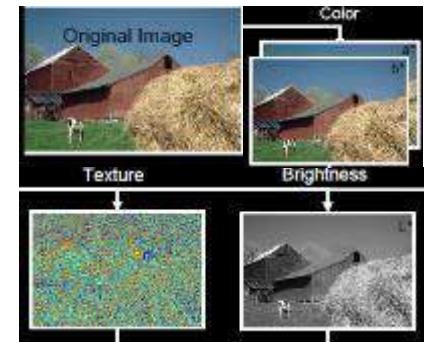


Things to remember

- Canny edge detector =
smooth → derivative → thin → threshold → link



- Pb: learns weighting of gradient, color, texture differences



- Straight line detector =
canny + gradient orientations → orientation binning
→ linking → check for straightness



Acknowledgements

- Thanks to the following researchers for making their teaching/research material online
 - Forsyth
 - Steve Seitz
 - Noah Snavely
 - J.B. Huang
 - Derek Hoiem
 - D. Lowe
 - A. Bobick
 - S. Lazebnik
 - K. Grauman
 - R. Zaleski

Thank you: Question?

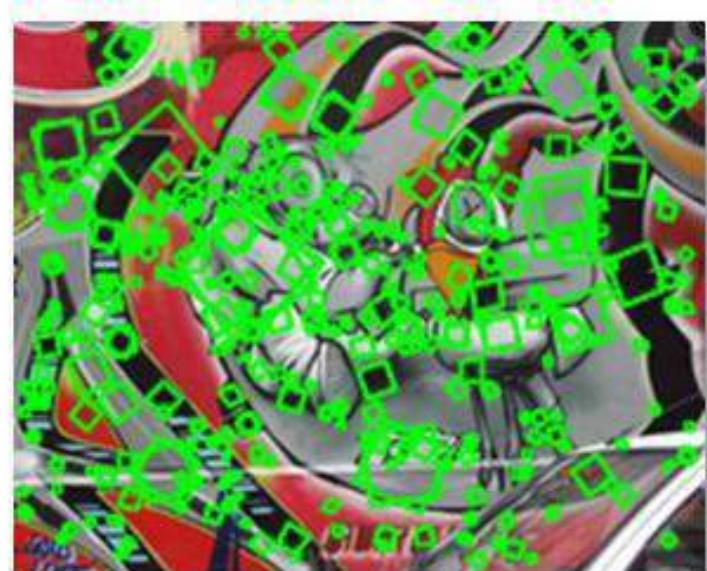
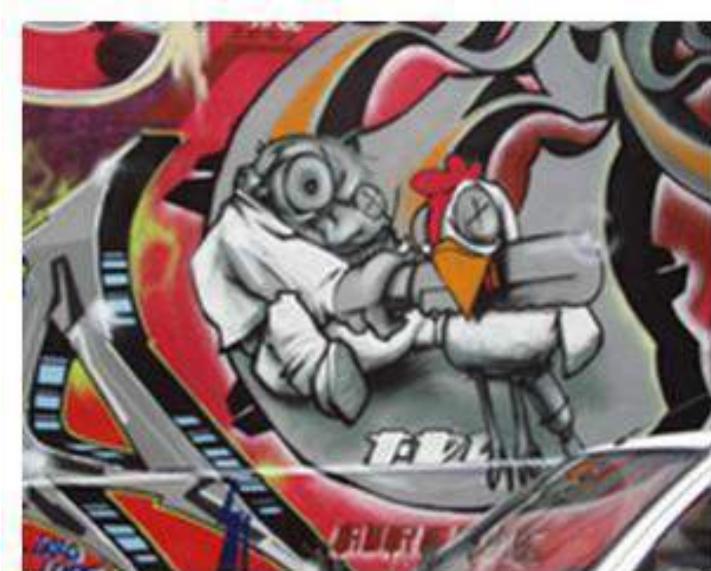
Computer Vision Interest Points

Dr. Mrinmoy Ghorai

**Indian Institute of Information Technology
Sri City, Chittoor**

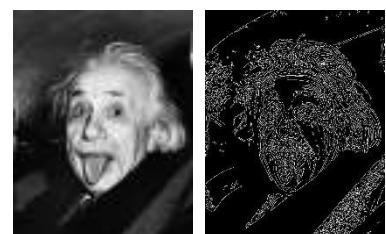
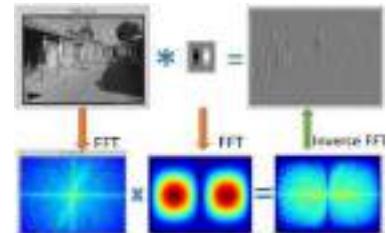
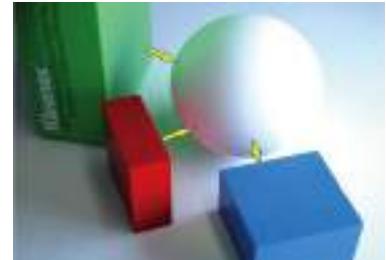


Interest Points



What have we learned so far?

- Light and color
 - What an image records
- Filtering in spatial domain
 - Filtering = weighted sum of neighboring pixels
 - Smoothing, sharpening, measuring texture
- Filtering in frequency domain
 - Filtering = change frequency of the input image
 - Denoising, sampling, image compression
- Image pyramid (Gaussian and Laplacian)
 - Multi-scale analysis
- Edge detection
 - Canny edge = smooth -> derivative -> thin -> threshold -> link
 - Finding straight lines



Today's class

- What is interest point?
- Corner detection
- Handling scale and orientation

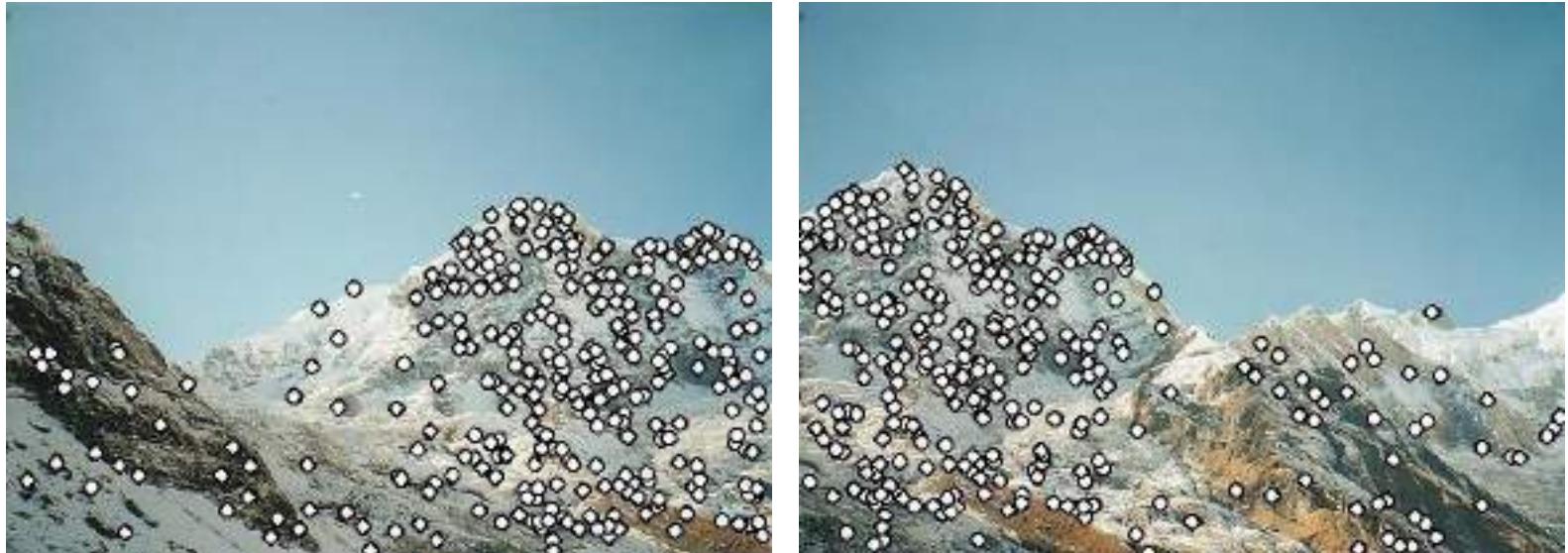
Why extract features?

- Motivation: panorama stitching
 - We have two images – how do we combine them?



Why extract features?

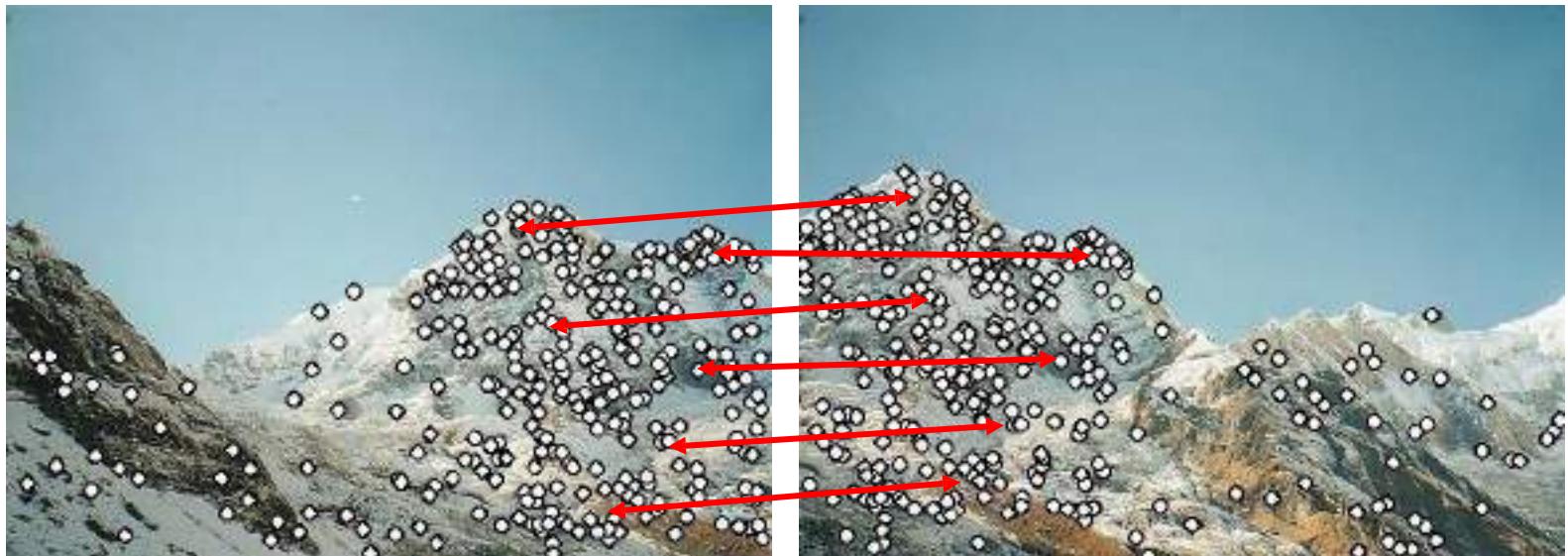
- Motivation: panorama stitching
 - We have two images – how do we combine them?



Step 1: extract features

Why extract features?

- Motivation: panorama stitching
 - We have two images – how do we combine them?



Step 1: extract
features Step 2: match
features

Why extract features?

- Motivation: panorama stitching
 - We have two images – how do we combine them?



Step 1: extract
features Step 2: match
features Step 3: align
images

Applications

- Keypoints are used for:
 - Image alignment
 - 3D reconstruction
 - Motion tracking
 - Robot navigation
 - Indexing and database retrieval
 - Object recognition



Advantages of local features

Locality

- features are local, so robust to occlusion and clutter

Quantity

- hundreds or thousands in a single image

Distinctiveness:

- can differentiate a large database of objects

Efficiency

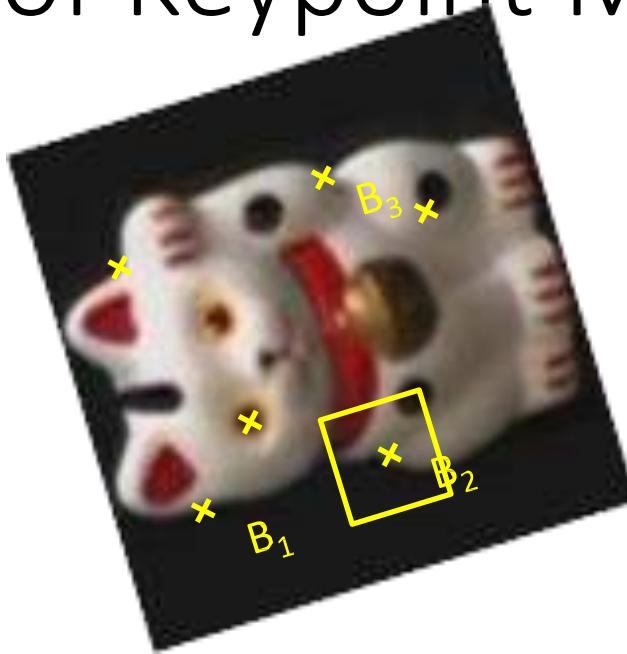
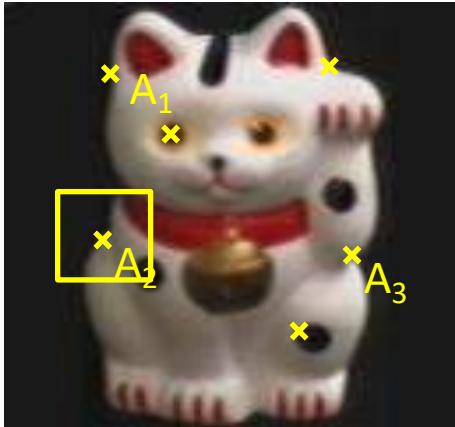
- real-time performance achievable

Overview of Keypoint Matching

1. Find a set of distinctive key-points



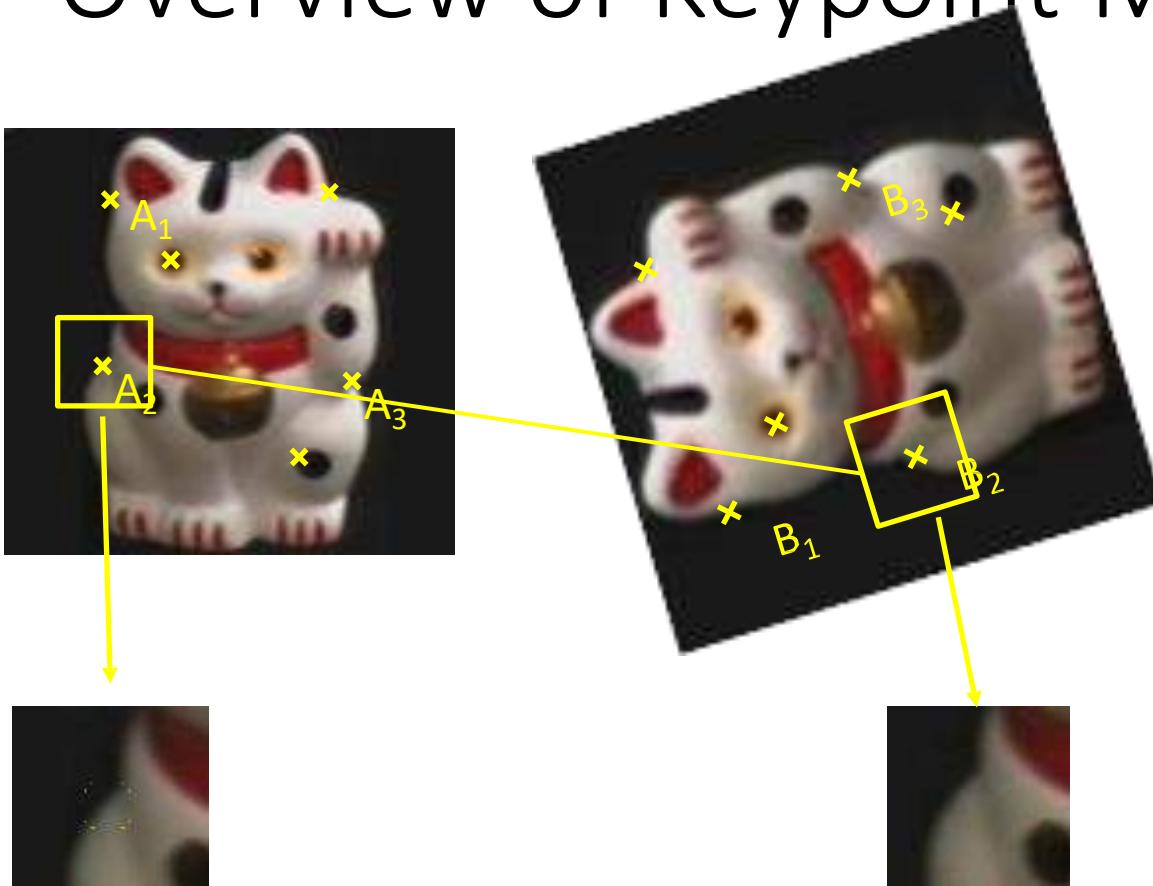
Overview of Keypoint Matching



1. Find a set of distinctive keypoints

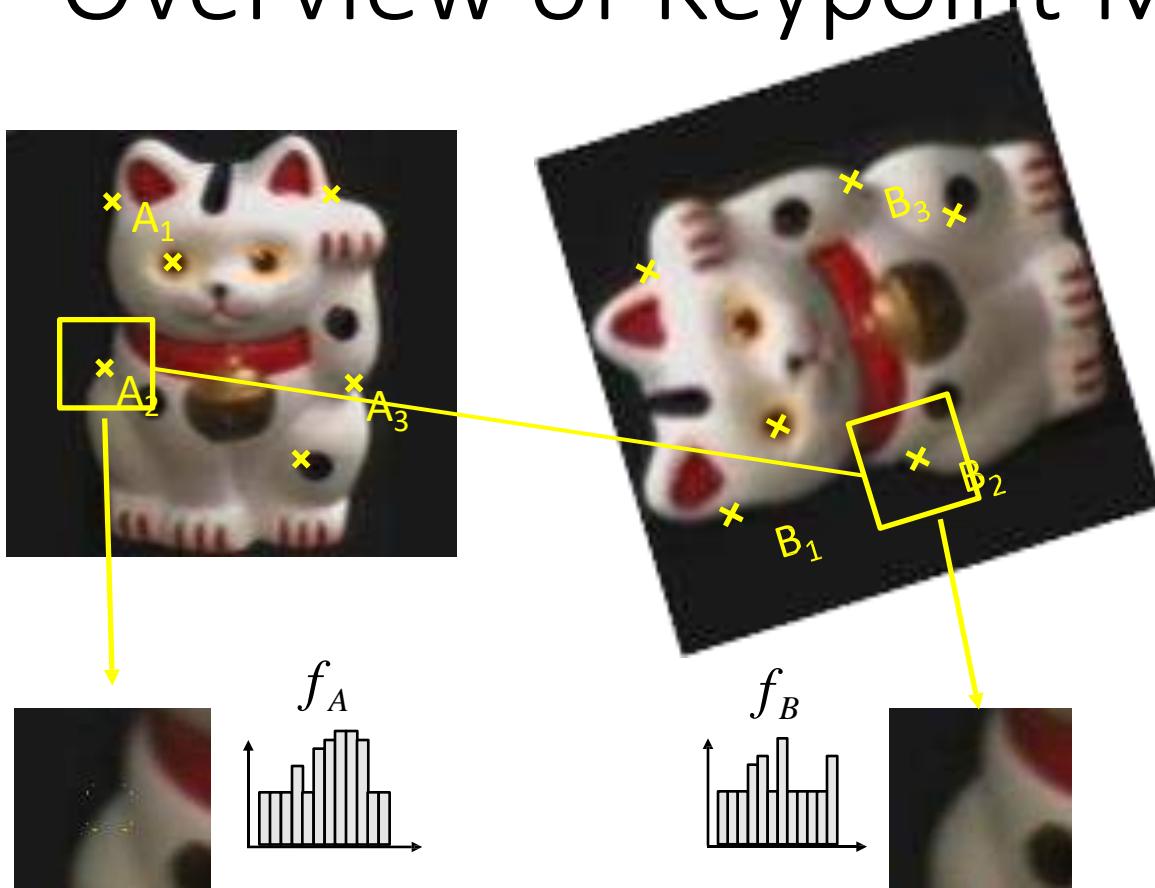
2. Define a region around each keypoint

Overview of Keypoint Matching



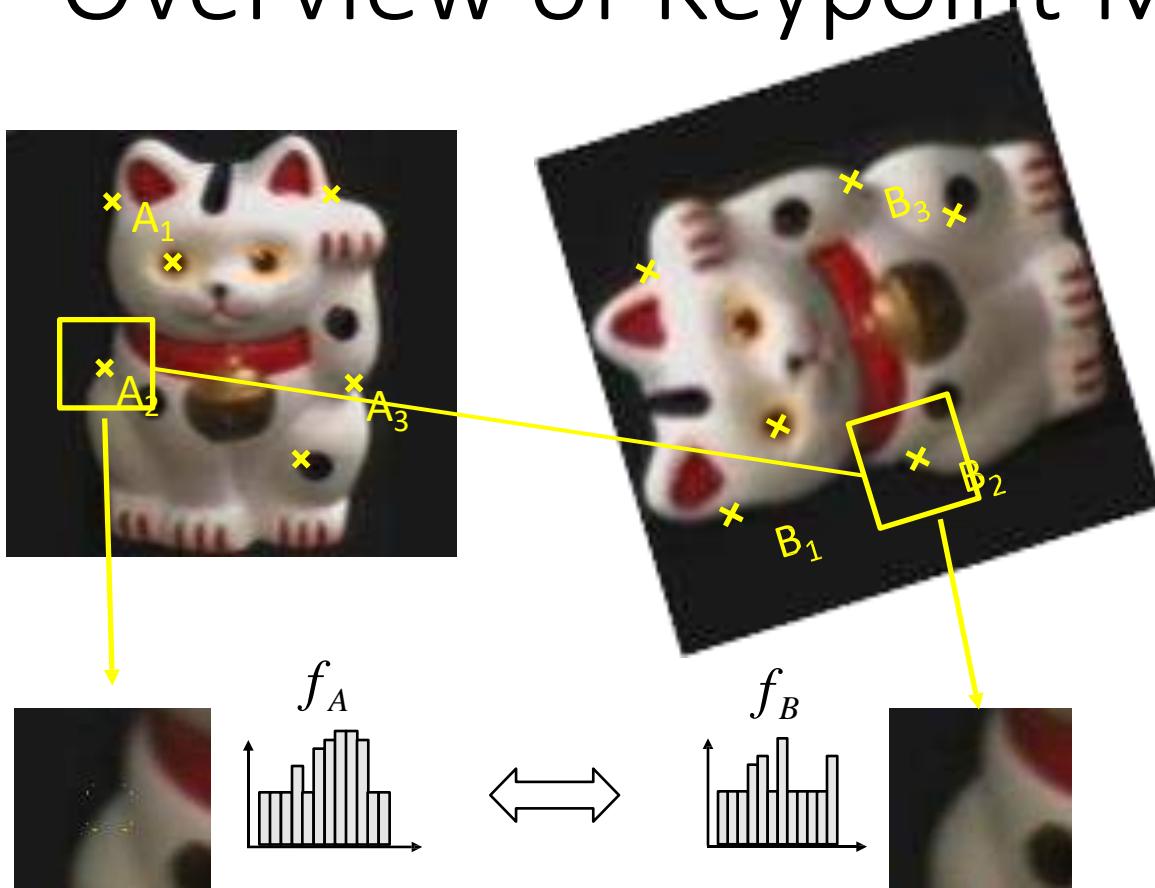
- 1. Find a set of distinctive key-points**
- 2. Define a region around each keypoint**
- 3. Extract and normalize the region content**

Overview of Keypoint Matching



- 1. Find a set of distinctive keypoints**
- 2. Define a region around each keypoint**
- 3. Extract and normalize the region content**
- 4. Compute a local descriptor from the normalized region**

Overview of Keypoint Matching



$$d(f_A, f_B) < T$$

1. Find a set of distinctive keypoints

2. Define a region around each keypoint

3. Extract and normalize the region content

4. Compute a local descriptor from the normalized region

5. Match local descriptors

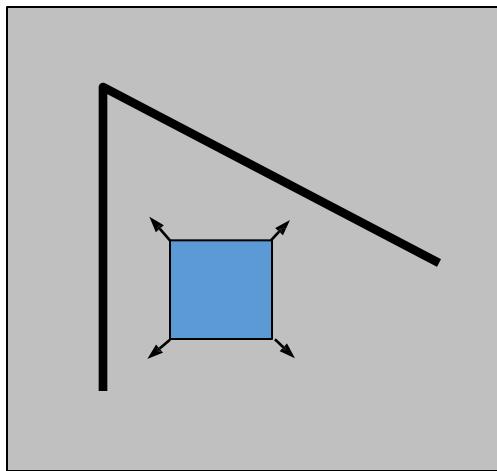
Goals for Keypoints



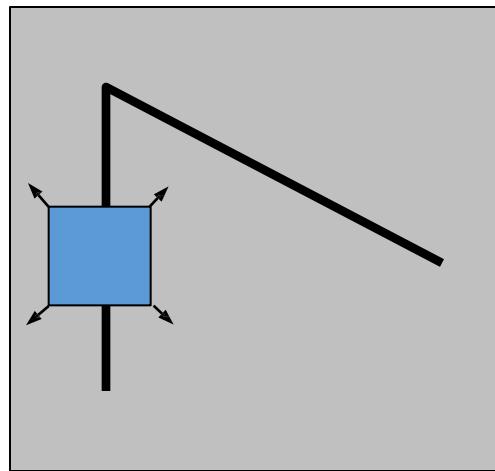
Detect points that are *repeatable* and *distinctive*

Corner Detection: Basic Idea

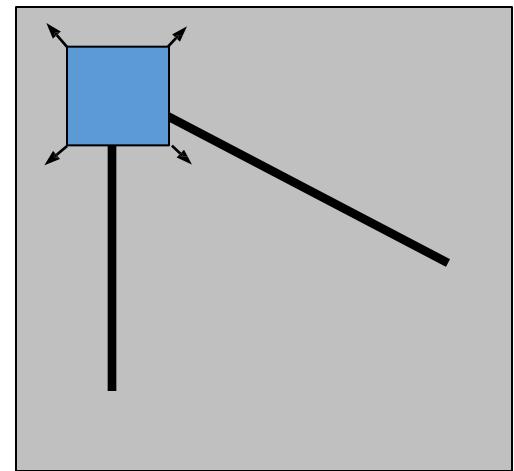
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



“flat” region:
no change in
all directions



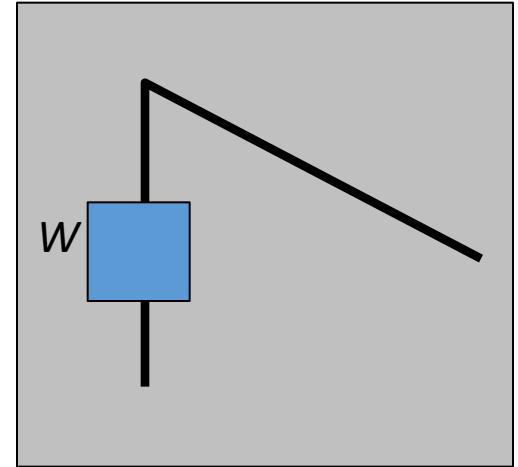
“edge”:
no change along the
edge direction



“corner”:
significant change in
all directions

Corner detection: the math

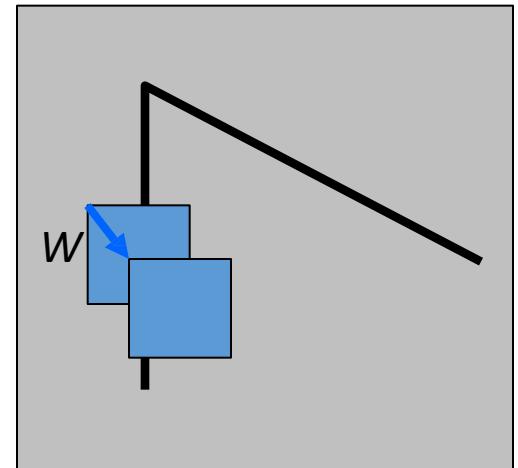
Consider shifting the window W by (u, v)



Corner detection: the math

Consider shifting the window W by (u, v)

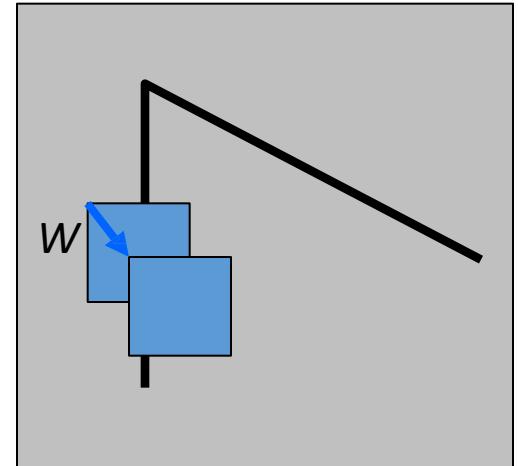
- how do the pixels in W change?



Corner detection: the math

Consider shifting the window W by (u, v)

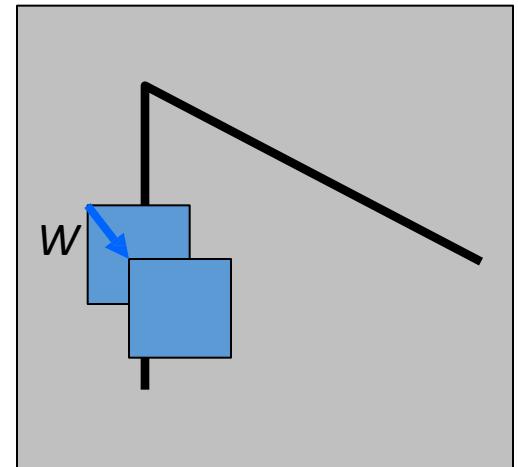
- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” $E(u, v)$:



Corner detection: the math

Consider shifting the window W by (u, v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” $E(u, v)$:



$$E(u, v) = \sum_{(x,y) \in W} (I(x + u, y + v) - I(x, y))^2$$

Small motion assumption

Taylor Series expansion of I :

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

Small motion assumption

Taylor Series expansion of I :

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (u, v) is small, then first order approximation is good

$$I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

Small motion assumption

Taylor Series expansion of I :

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (u, v) is small, then first order approximation is good

$$\begin{aligned} I(x + u, y + v) &\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

Small motion assumption

Taylor Series expansion of I :

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (u, v) is small, then first order approximation is good

$$\begin{aligned} I(x + u, y + v) &\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

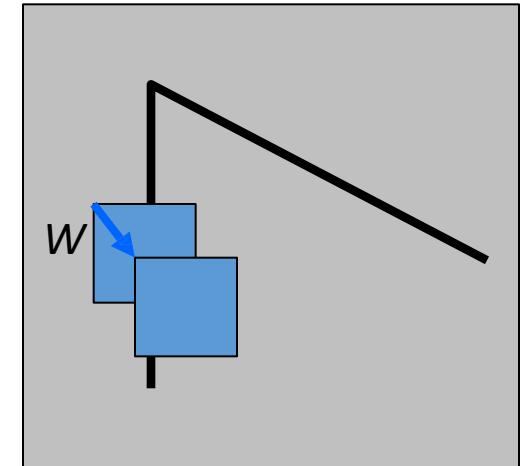
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide...

Corner detection: the math

Using the small motion assumption,
replace I with a linear approximation

(Shorthand: $I_x = \frac{\partial I}{\partial x}$)

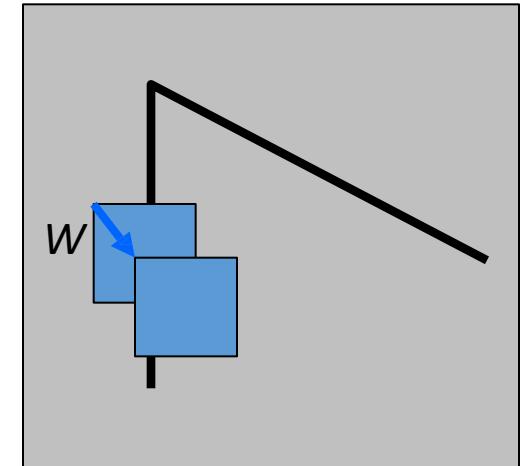


$$E(u, v) = \sum_{(x,y) \in W} (I(x + u, y + v) - I(x, y))^2$$

Corner detection: the math

Using the small motion assumption,
replace I with a linear approximation

(Shorthand: $I_x = \frac{\partial I}{\partial x}$)

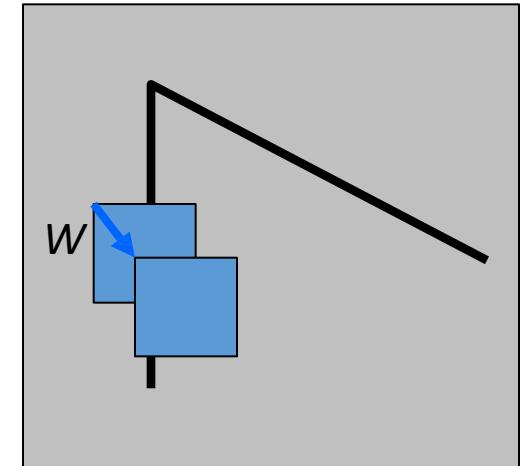


$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} (I(x+u, y+v) - I(x, y))^2 \\ &\approx \sum_{(x,y) \in W} (I(x, y) + I_x(x, y)u + I_y(x, y)v - I(x, y))^2 \end{aligned}$$

Corner detection: the math

Using the small motion assumption,
replace I with a linear approximation

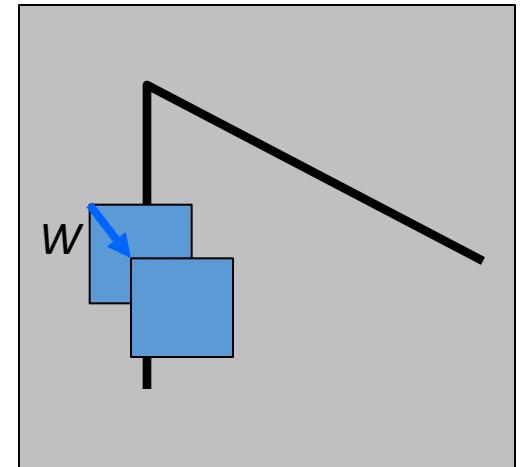
(Shorthand: $I_x = \frac{\partial I}{\partial x}$)



$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} (I(x+u, y+v) - I(x, y))^2 \\ &\approx \sum_{(x,y) \in W} (I(x, y) + I_x(x, y)u + I_y(x, y)v - I(x, y))^2 \\ &\approx \sum_{(x,y) \in W} (I_x(x, y)u + I_y(x, y)v)^2 \end{aligned}$$

Corner detection: the math

$$\begin{aligned} E(u, v) &\approx \sum_{(x,y) \in W} (I_x(x, y)u + I_y(x, y)v)^2 \\ &\approx \sum_{(x,y) \in W} (I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2) \end{aligned}$$



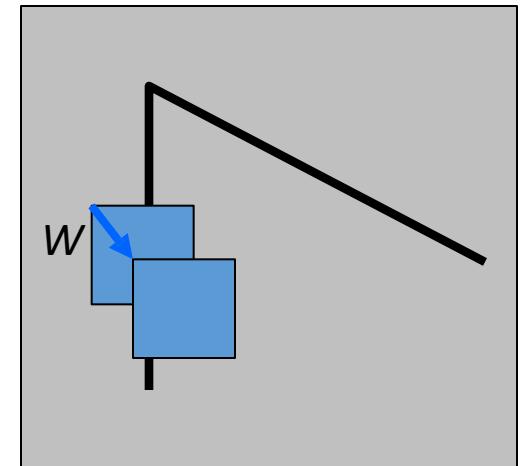
Corner detection: the math

$$E(u, v) \approx \sum_{(x,y) \in W} (I_x(x, y)u + I_y(x, y)v)^2$$

$$\approx \sum_{(x,y) \in W} (I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2)$$

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y) \in W} I_x^2 \quad B = \sum_{(x,y) \in W} I_x I_y \quad C = \sum_{(x,y) \in W} I_y^2$$



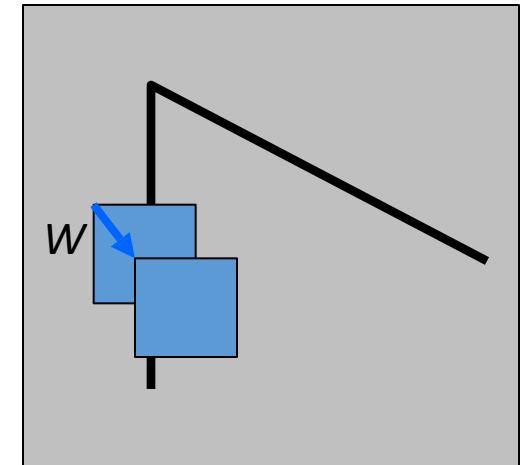
Corner detection: the math

$$E(u, v) \approx \sum_{(x,y) \in W} (I_x(x, y)u + I_y(x, y)v)^2$$

$$\approx \sum_{(x,y) \in W} (I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2)$$

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y) \in W} I_x^2 \quad B = \sum_{(x,y) \in W} I_x I_y \quad C = \sum_{(x,y) \in W} I_y^2$$



- Thus, $E(u, v)$ is locally approximated as a *quadratic form*

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u, v) \approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y) \in W} I_x^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2 \quad H$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

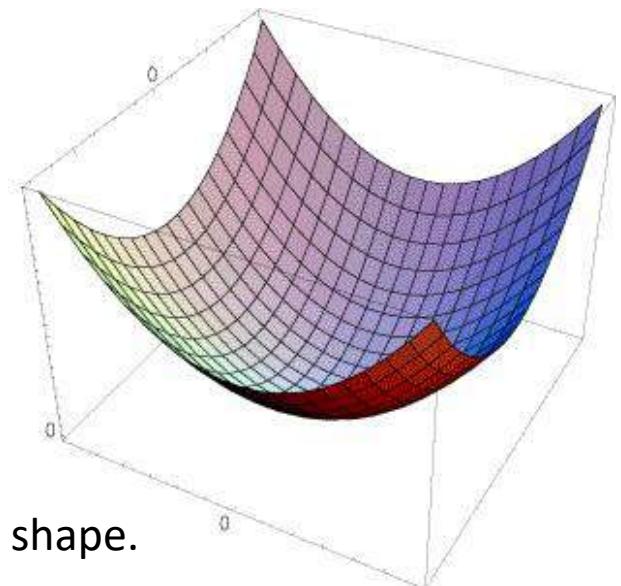
$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$

Let's try to understand its shape.

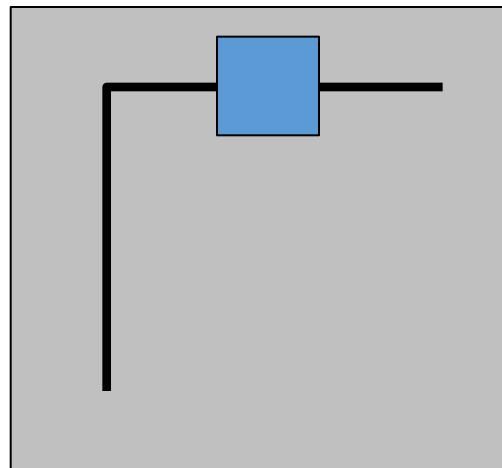


$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

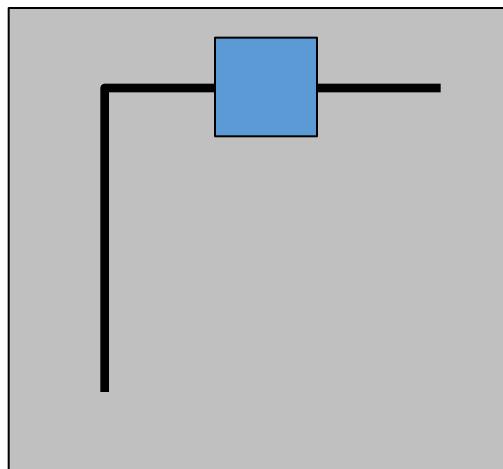
$$C = \sum_{(x,y) \in W} I_y^2$$



Horizontal edge: $I_x = 0$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\begin{aligned} A &= \sum_{(x,y) \in W} I_x^2 \\ B &= \sum_{(x,y) \in W} I_x I_y \\ C &= \sum_{(x,y) \in W} I_y^2 \end{aligned}$$



$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

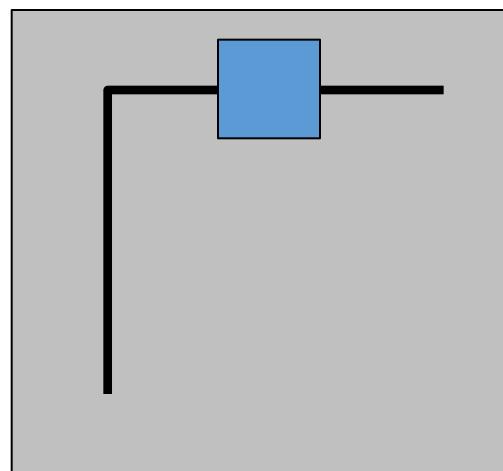
Horizontal edge: $I_x = 0$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2$$

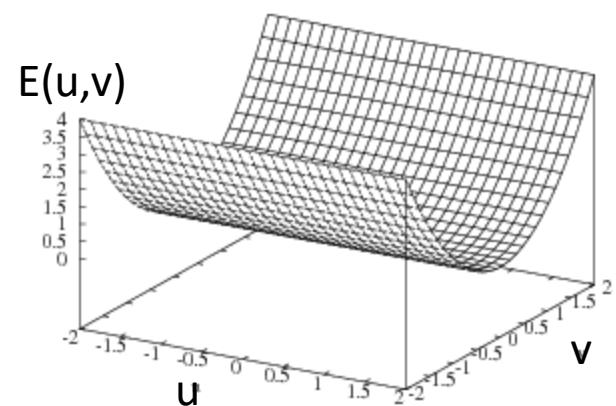
$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$



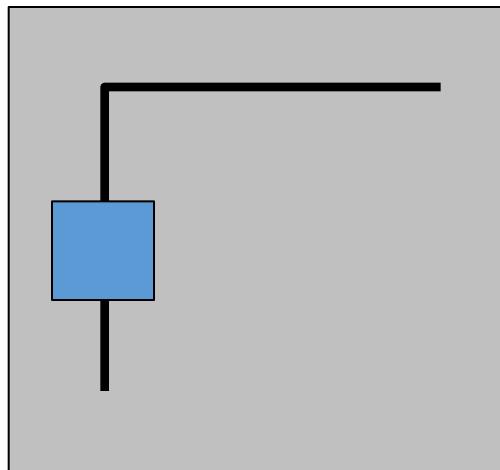
$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$\underbrace{\hspace{10em}}_H$

$$A = \sum_{(x,y) \in W} I_x^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

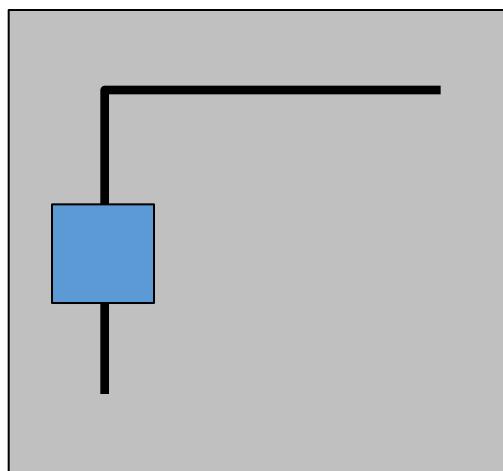
$$C = \sum_{(x,y) \in W} I_y^2$$



Vertical edge: $I_y = 0$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\begin{aligned} A &= \sum_{(x,y) \in W} I_x^2 \\ B &= \sum_{(x,y) \in W} I_x I_y \\ C &= \sum_{(x,y) \in W} I_y^2 \end{aligned}$$



$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

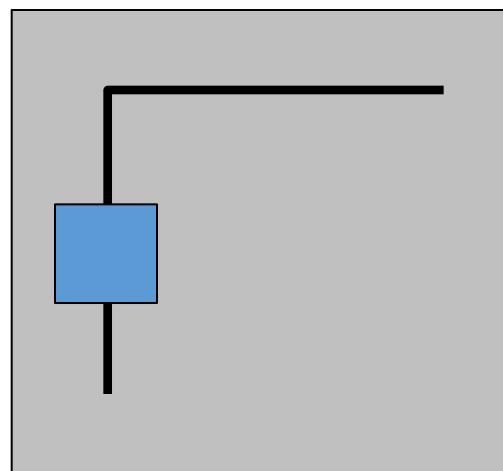
Vertical edge: $I_y = 0$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y) \in W} I_x^2$$

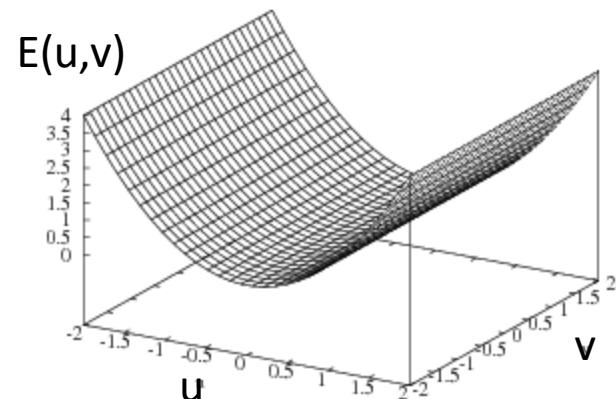
$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} I_y^2$$



Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$



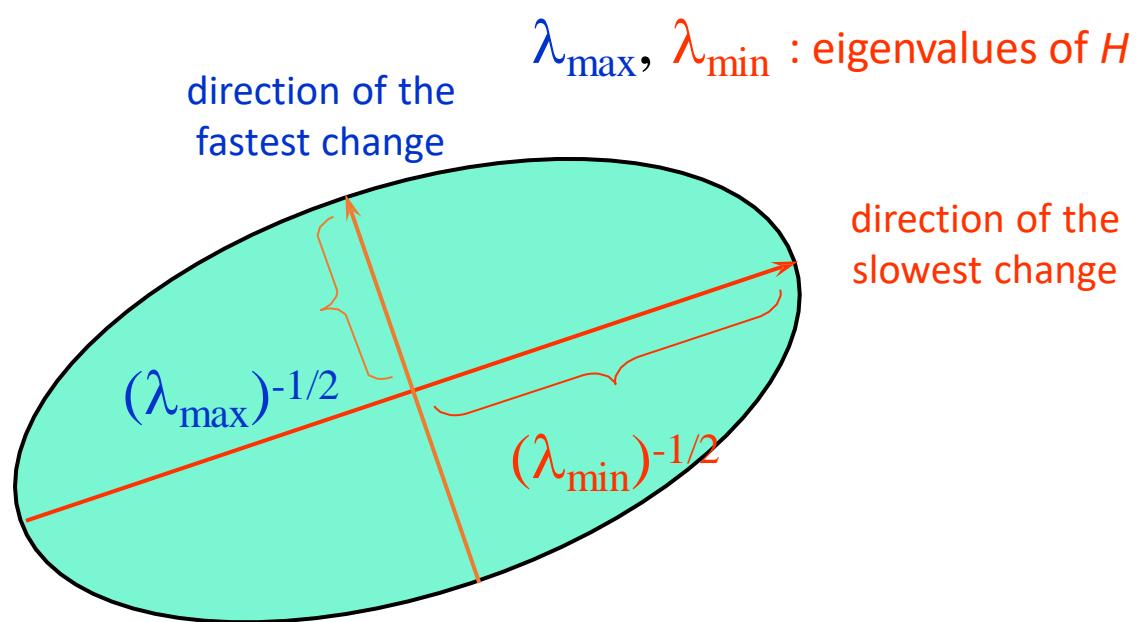
General case

The shape of H tells us something about the *distribution of gradients* around a pixel

We can visualize H as an ellipse with axis lengths determined by the *eigenvalues* of H and orientation determined by the *eigenvectors* of H

Ellipse equation:

$$[u \ v] H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

The scalar λ is the **eigenvalue** corresponding to **x**

Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix \mathbf{A} are the vectors \mathbf{x} that satisfy:

The scalar λ is the **eigenvalue** corresponding to \mathbf{x}

- The eigenvalues are found by solving:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix \mathbf{A} are the vectors \mathbf{x} that satisfy:

The scalar λ is the **eigenvalue** corresponding to \mathbf{x}

- The eigenvalues are found by solving:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

- In our case, $\mathbf{A} = \mathbf{H}$ is a 2x2 matrix, so we have

$$\det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix \mathbf{A} are the vectors \mathbf{x} that satisfy:

The scalar λ is the **eigenvalue** corresponding to \mathbf{x}

- The eigenvalues are found by solving:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

- In our case, $\mathbf{A} = \mathbf{H}$ is a 2x2 matrix, so we have

$$\det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix \mathbf{A} are the vectors \mathbf{x} that satisfy:

The scalar λ is the **eigenvalue** corresponding to \mathbf{x}

- The eigenvalues are found by solving:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

- In our case, $\mathbf{A} = \mathbf{H}$ is a 2x2 matrix, so we have

$$\det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

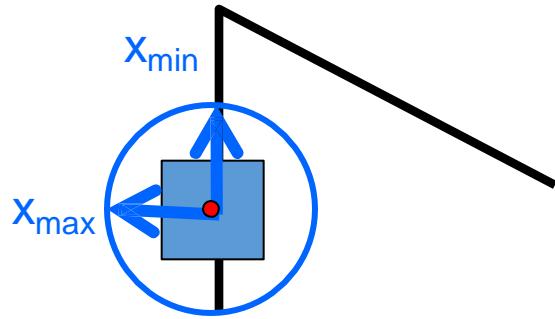
$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

Once you know λ , you find \mathbf{x} by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

Corner detection: the math

$$E(u, v) \approx [u \ v] \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$



H

$$Hx_{\max} = \lambda_{\max}x_{\max}$$

$$Hx_{\min} = \lambda_{\min}x_{\min}$$

Eigenvalues and eigenvectors of H

- Define shift directions with the smallest and largest change in error
- x_{\max} = direction of largest increase in E
- λ_{\max} = amount of increase in direction x_{\max}
- x_{\min} = direction of smallest increase in E
- λ_{\min} = amount of increase in direction x_{\min}

Corner detection: the math

How are λ_{\max} , x_{\max} , λ_{\min} , and x_{\min} relevant for feature detection?

- What's our feature scoring function?

Corner detection: the math

How are λ_{\max} , x_{\max} , λ_{\min} , and x_{\min} relevant for feature detection?

- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue (λ_{\min}) of H

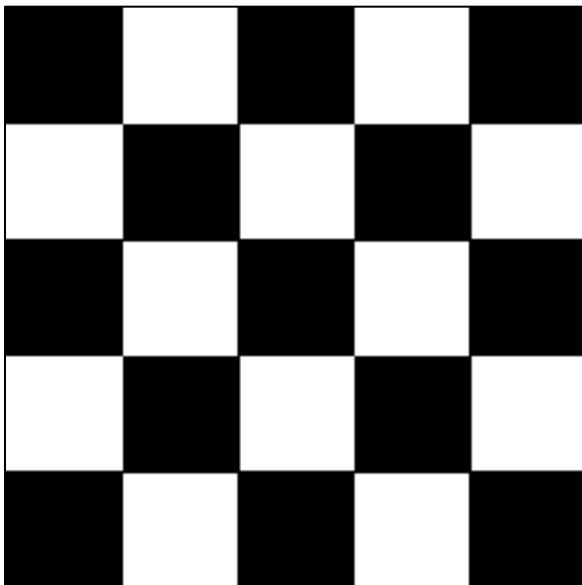
Corner detection: the math

How are λ_{\max} , x_{\max} , λ_{\min} , and x_{\min} relevant for feature detection?

- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue (λ_{\min}) of H



I

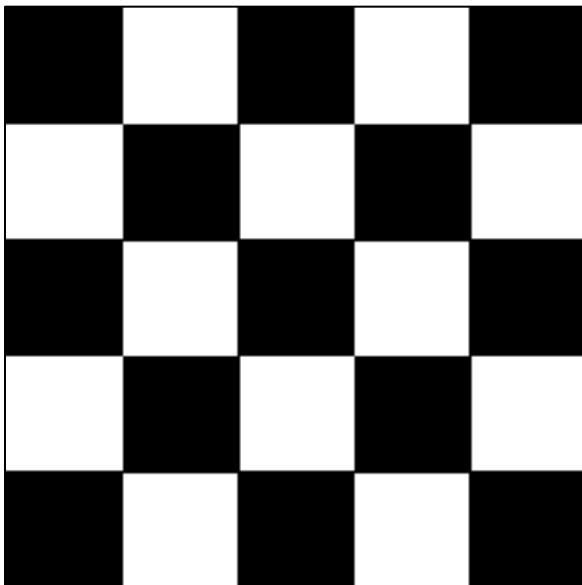
Corner detection: the math

How are λ_{\max} , x_{\max} , λ_{\min} , and x_{\min} relevant for feature detection?

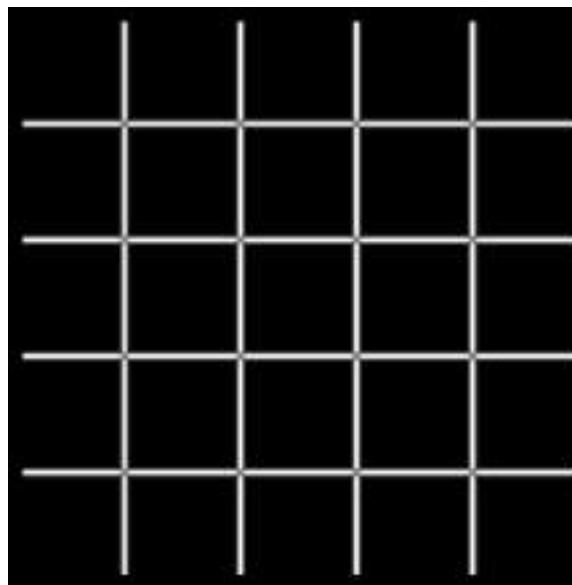
- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue (λ_{\min}) of H



I



λ_{\max}

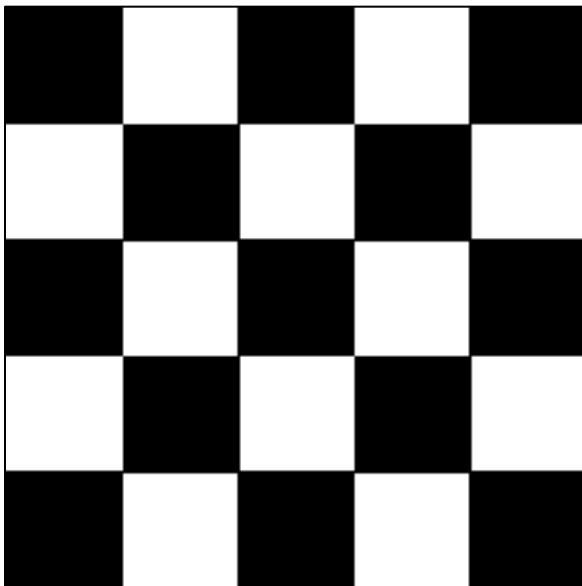
Corner detection: the math

How are λ_{\max} , x_{\max} , λ_{\min} , and x_{\min} relevant for feature detection?

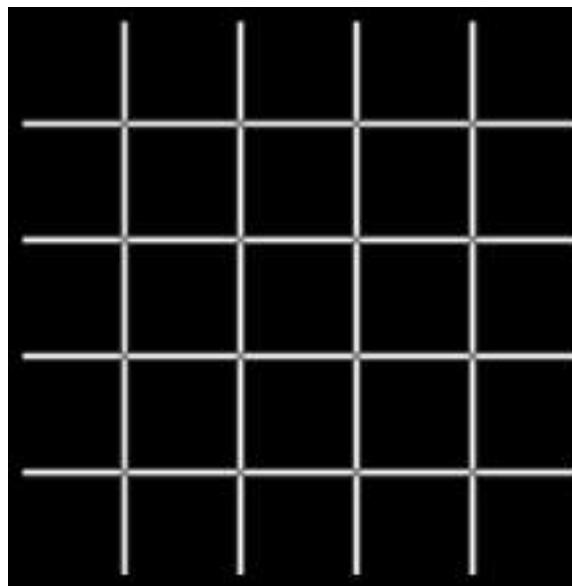
- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

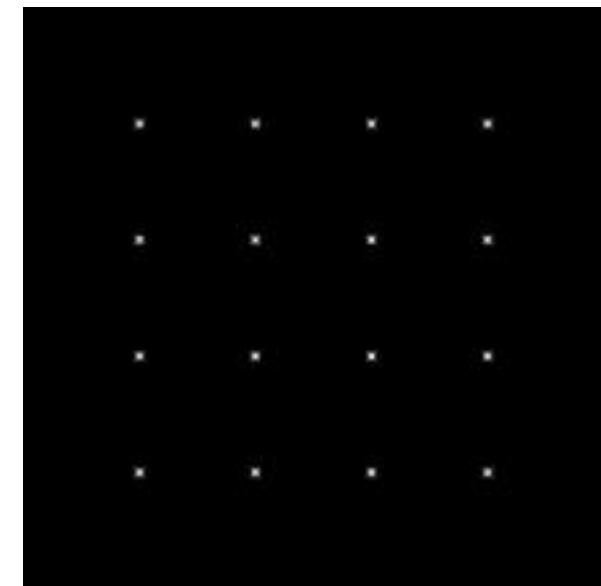
- the minimum of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue (λ_{\min}) of H



I



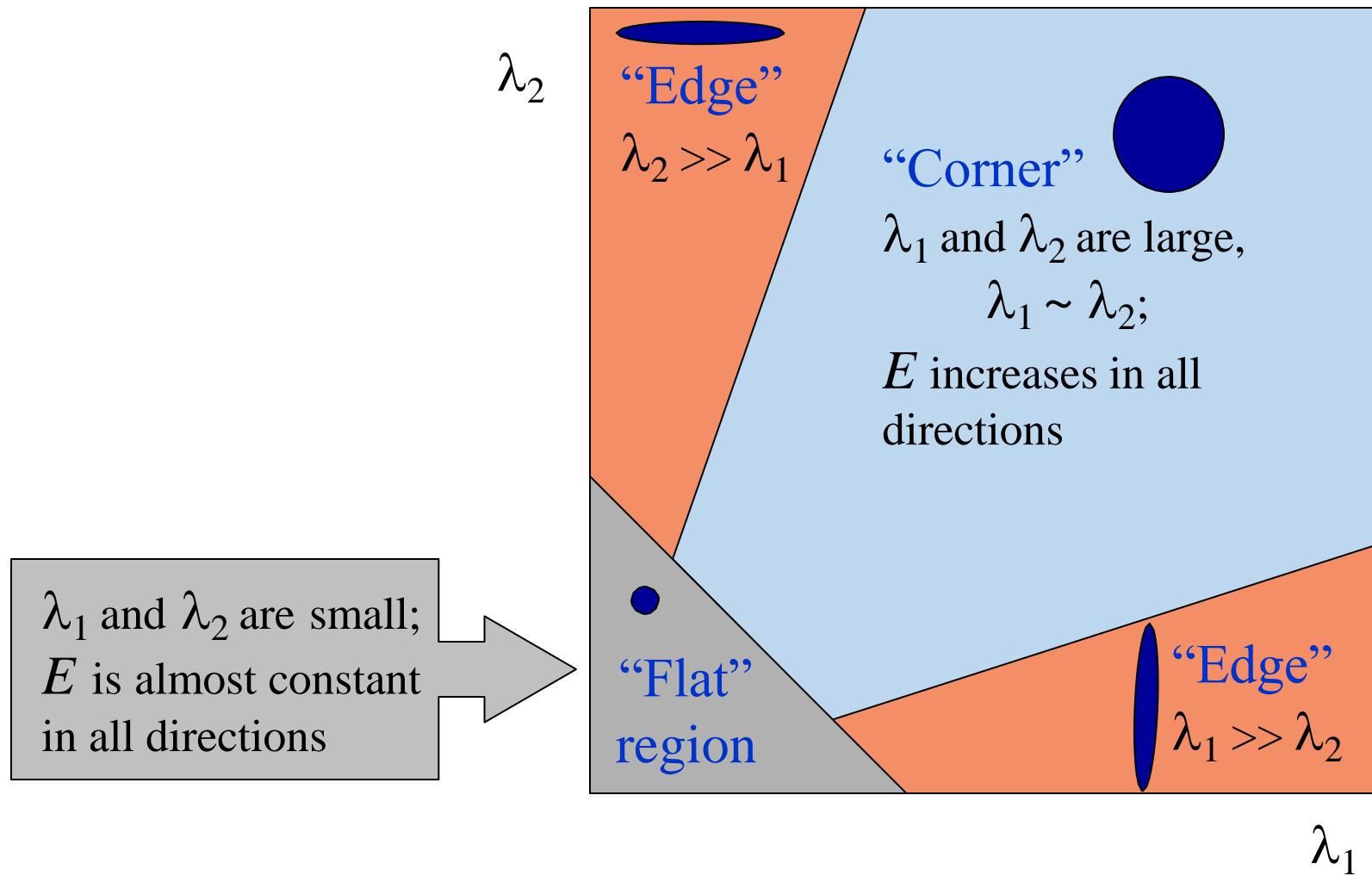
λ_{\max}



λ_{\min}

Interpreting the eigenvalues

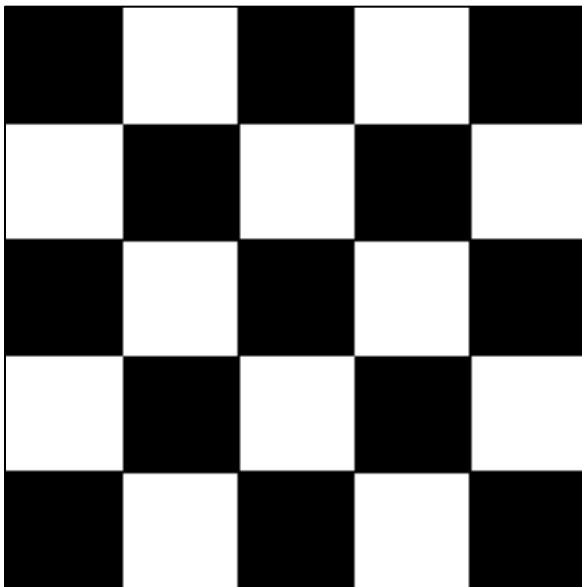
Classification of image points using eigenvalues of M :



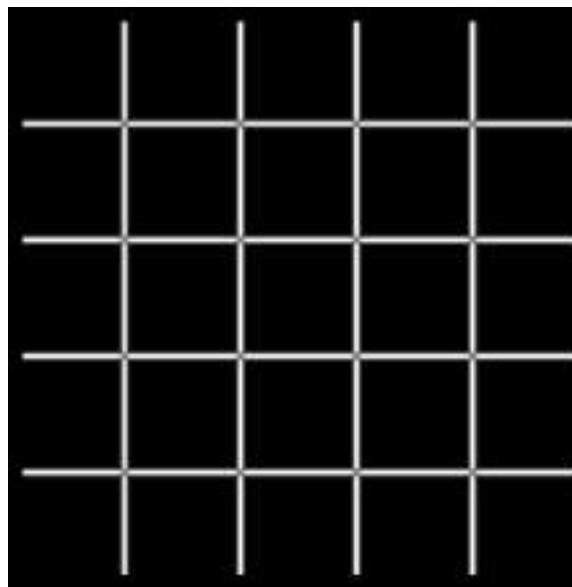
Corner detection summary

Here's what you do

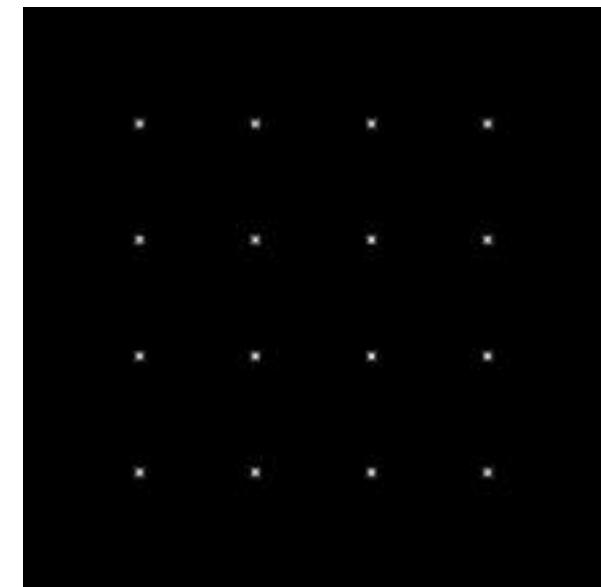
- Compute the gradient at each point in the image
- Create the H matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ($\lambda_{\min} > \text{threshold}$)
- Choose those points where λ_{\min} is a local maximum as features



I



λ_{\max}

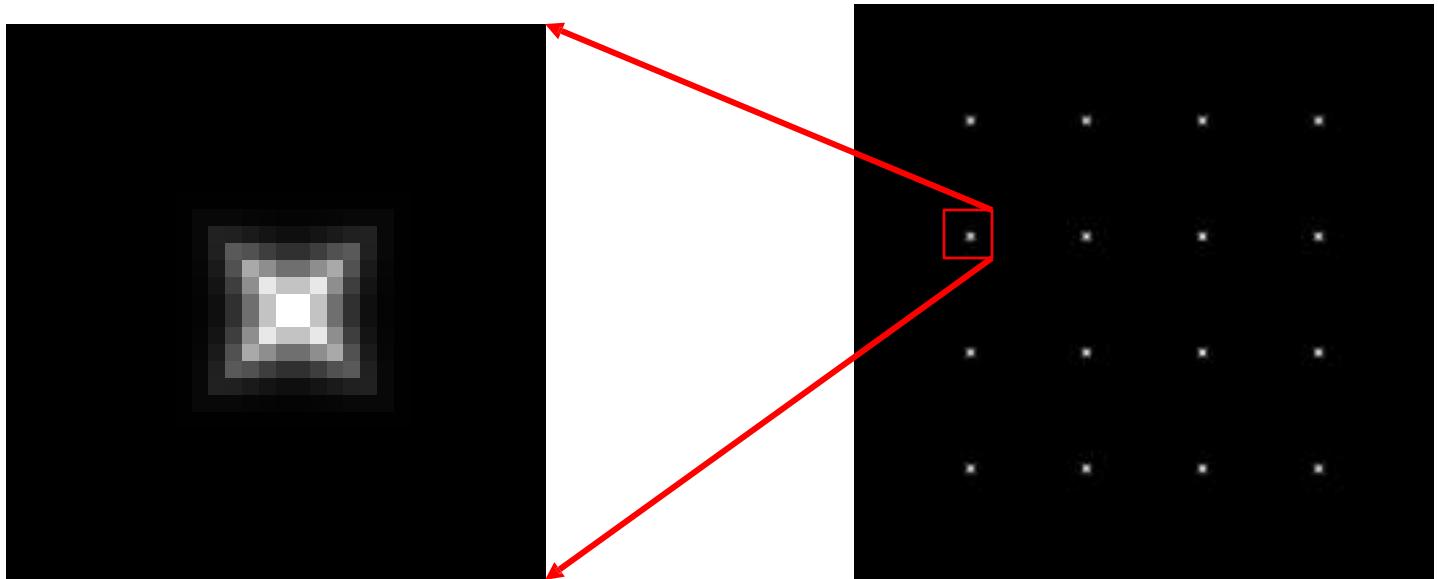


λ_{\min}

Corner detection summary

Here's what you do

- Compute the gradient at each point in the image
- Create the H matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ($\lambda_{\min} > \text{threshold}$)
- Choose those points where λ_{\min} is a local maximum as features



$$\lambda_{\min}$$

The Harris operator

λ_{\min} is a variant of the “Harris operator”¹ for feature detection

$$f = \lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2$$

$$= \text{determinant}(H) - \kappa (\text{trace}(H))^2$$

- The *trace* is the sum of the diagonals, i.e., $\text{trace}(H) = h_{11} + h_{22}$
- Called the “Harris Corner Detector” or “Harris Operator”
- Lots of other detectors, this is one of the most popular

¹C. Harris and M. Stephens (1988). ["A combined corner and edge detector"](#). *Proceedings of the 4th Alvey Vision Conference*. pp. 147–151.

Noble's corner operator

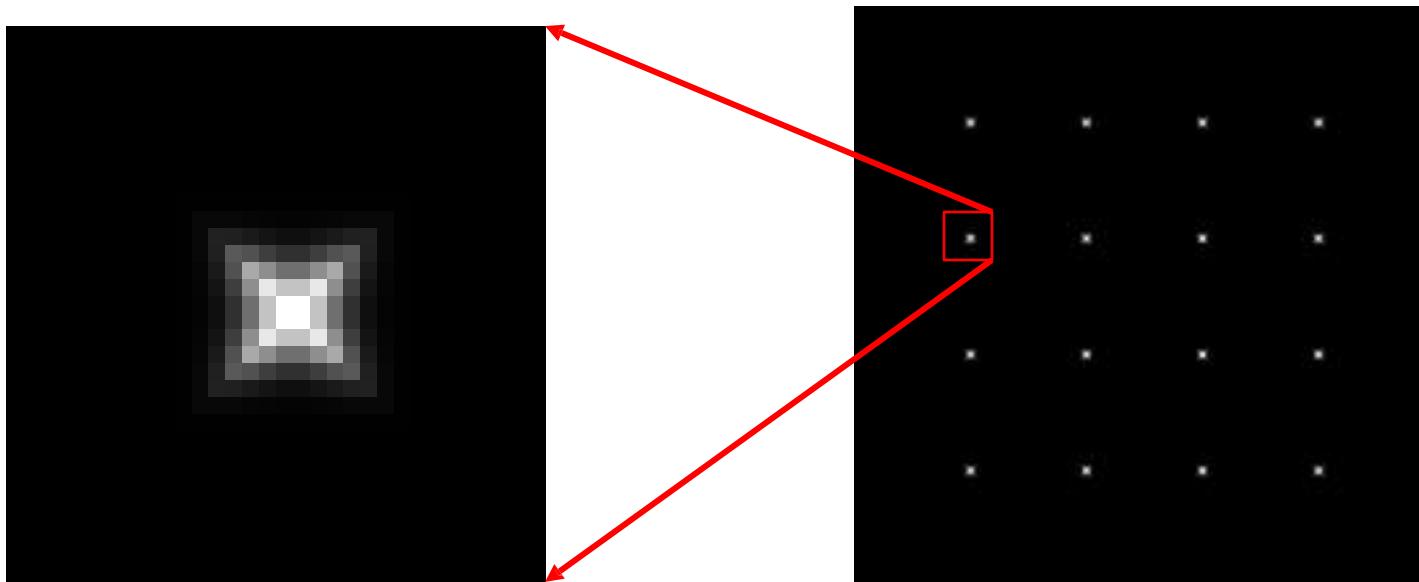
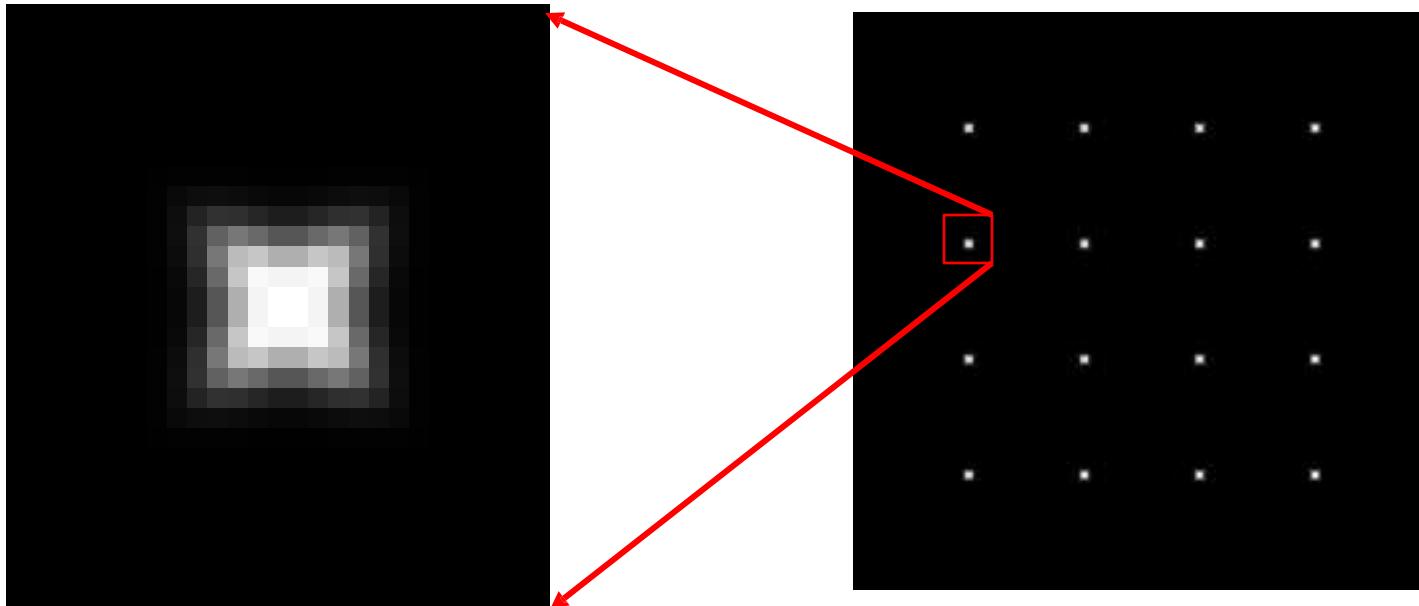
The “Noble’s operator”¹ for feature detection is:

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$
$$= \frac{\text{determinant}(H)}{\text{trace}(H)}$$

- The *trace* is the sum of the diagonals, i.e., $\text{trace}(H) = h_{11} + h_{22}$
- Very similar to λ_{\min} but less expensive (no square root)
- Called the “Noble’s Corner Detector”

¹A. Noble (1989). *Descriptions of Image Surfaces (Ph.D.). Department of Engineering Science, Oxford University.* p. 45.

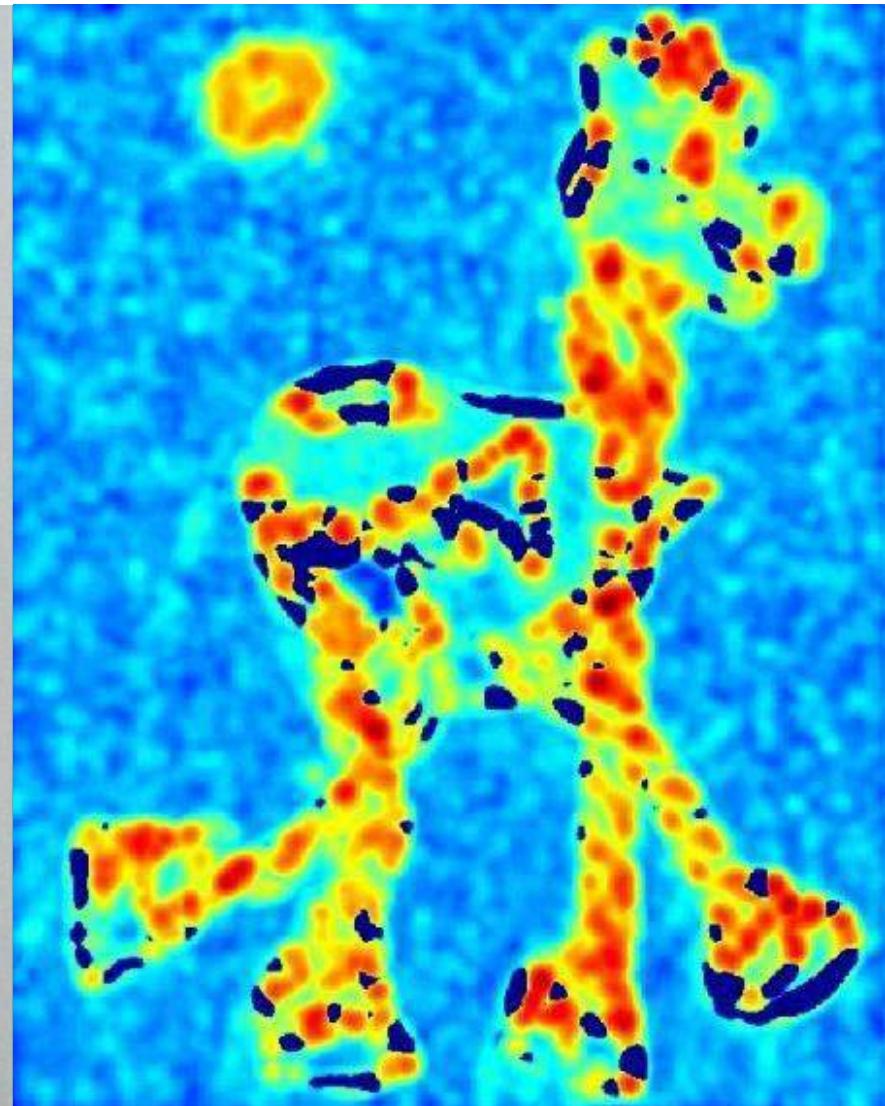
The Harris operator



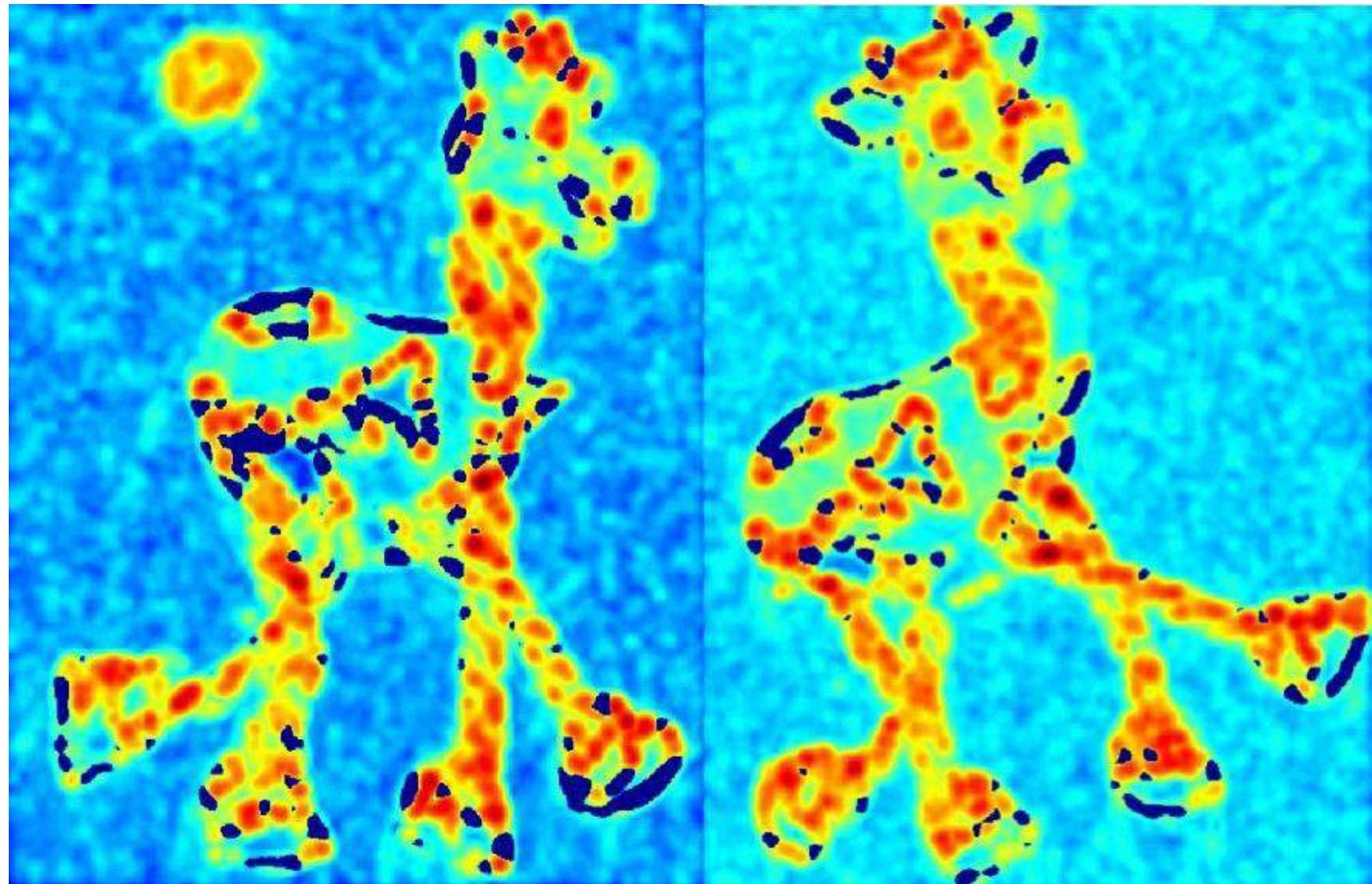
Harris detector example



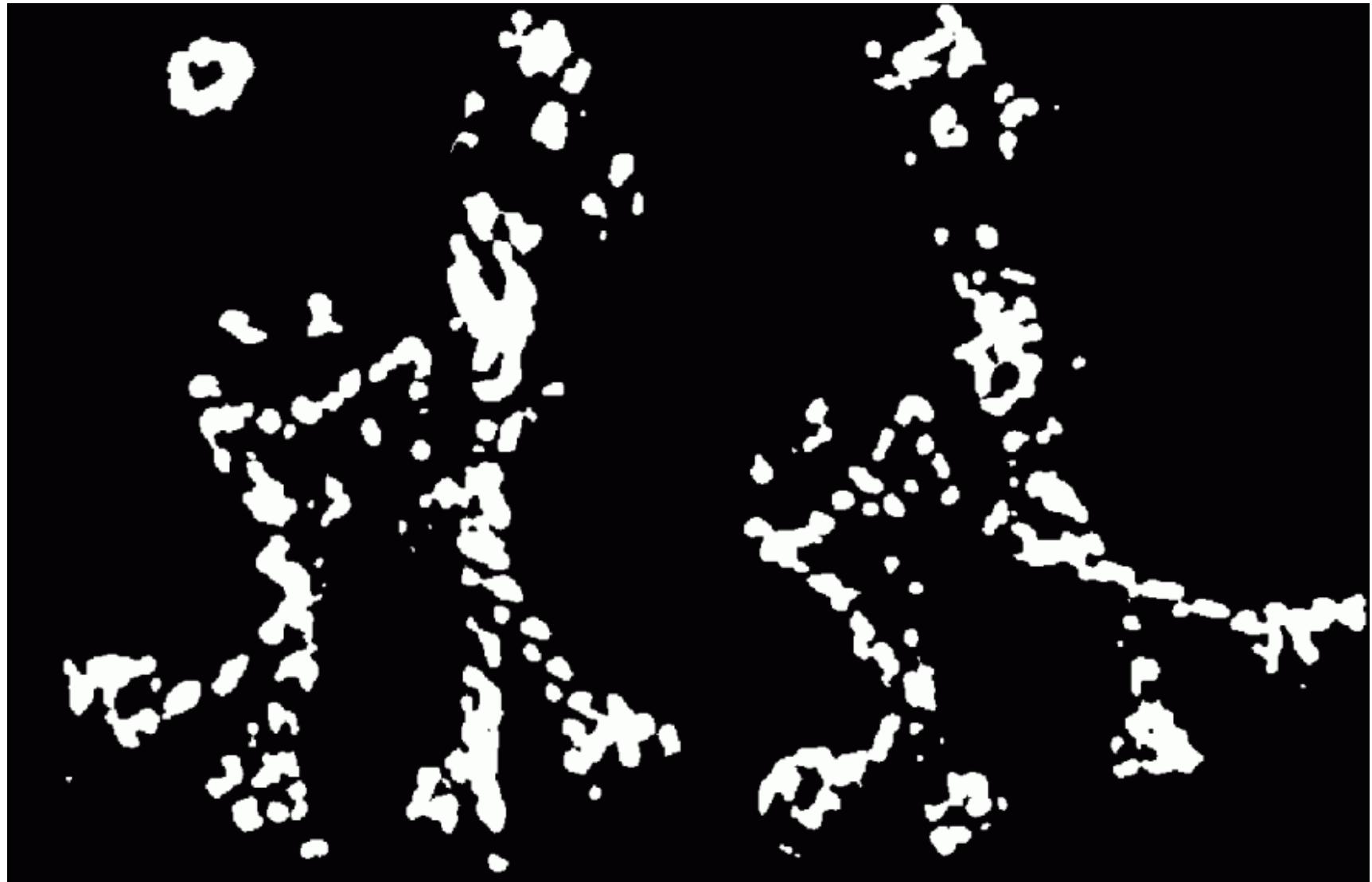
f value (red high, blue low)



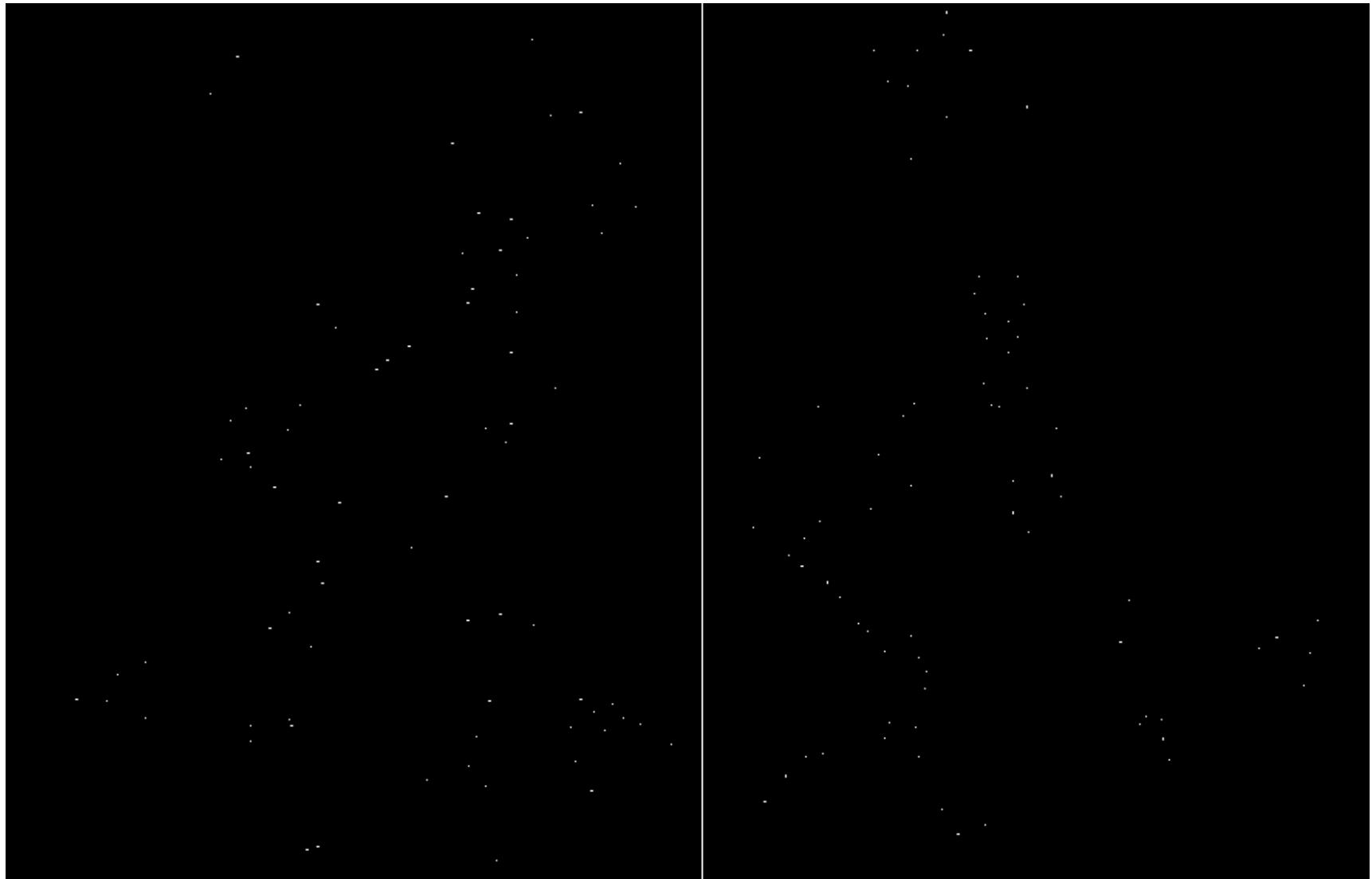
f value (red high, blue low)



Threshold ($f > \text{value}$)



Find local maxima of f



Harris features (in red)



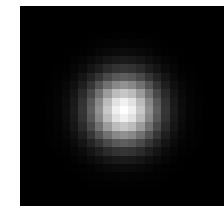
Weighting the derivatives

- In practice, using a simple window W doesn't work too well

$$H = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

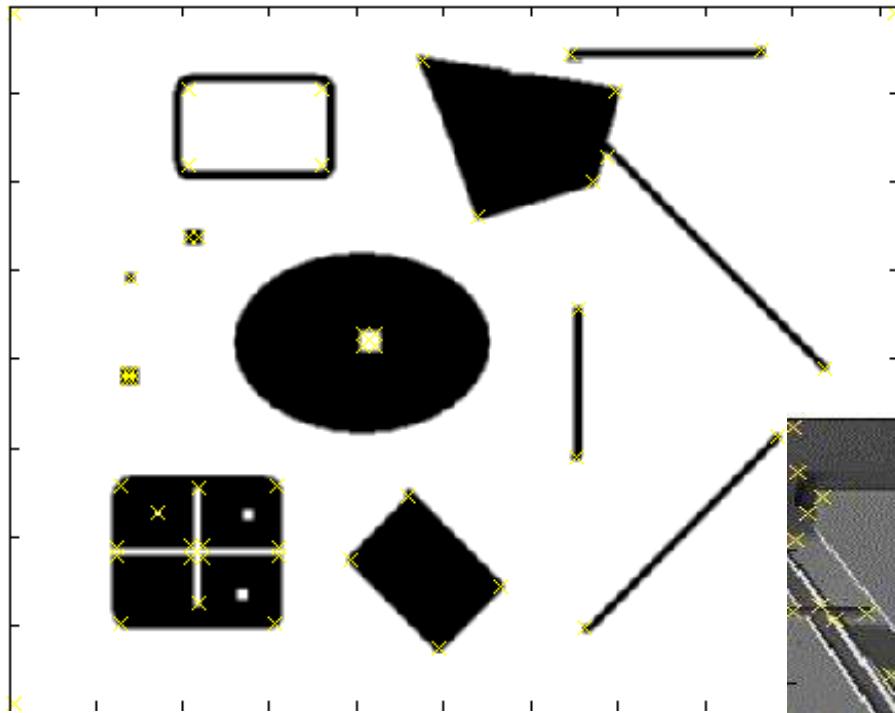
- Instead, we'll *weight* each derivative value based on its distance from the center pixel

$$H = \sum_{(x,y) \in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

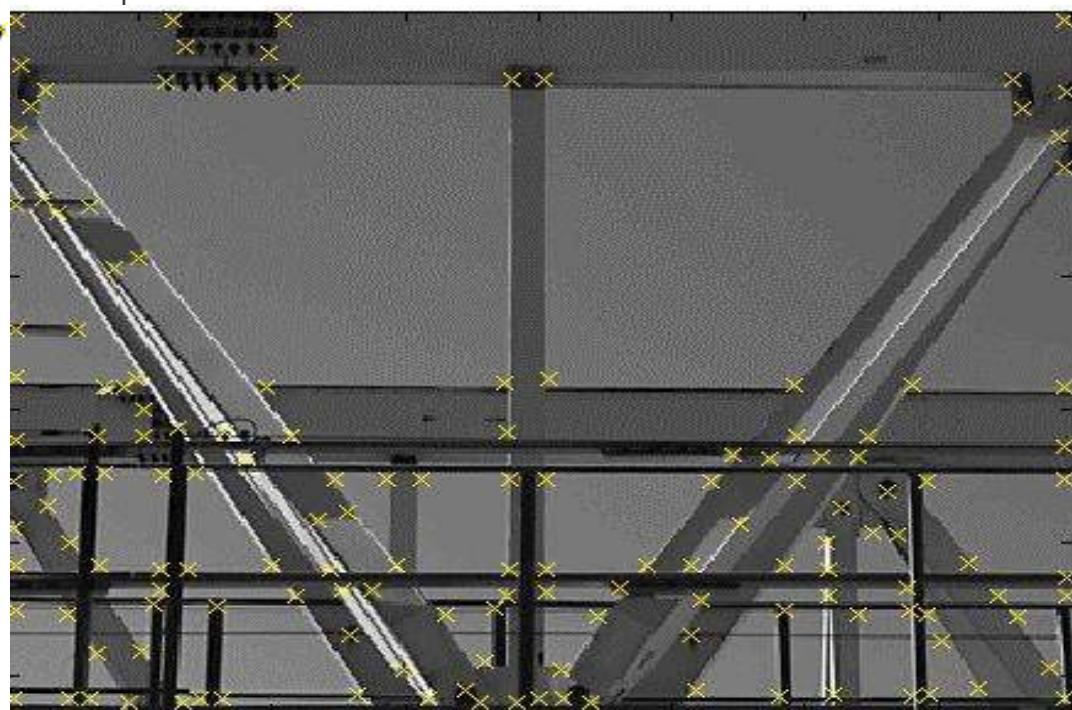


$w_{x,y}$

Harris Detector – Responses [Harris88]



Effect: A very precise corner detector.

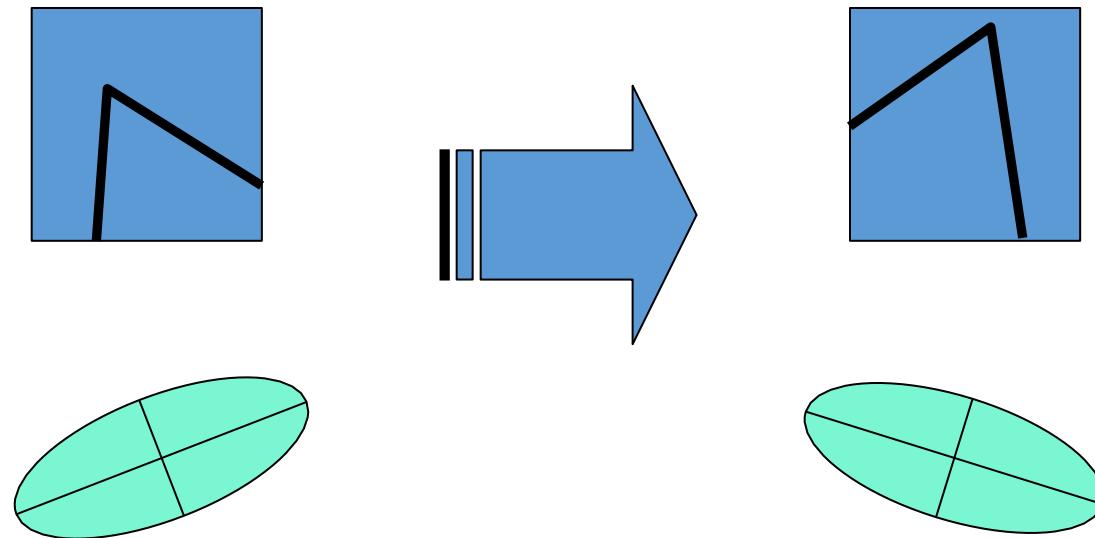


Harris Detector – Responses [Harris88]



Harris Detector: Invariance Properties

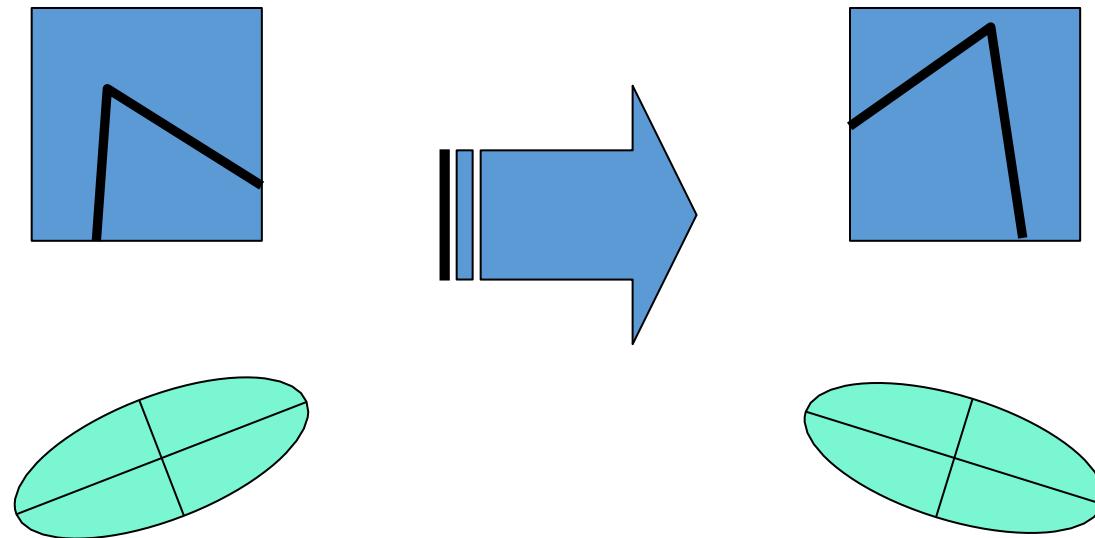
- Rotation



Ellipse rotates but its shape (i.e.
eigenvalues) remains the same

Harris Detector: Invariance Properties

- Rotation



Ellipse rotates but its shape (i.e.
eigenvalues) remains the same

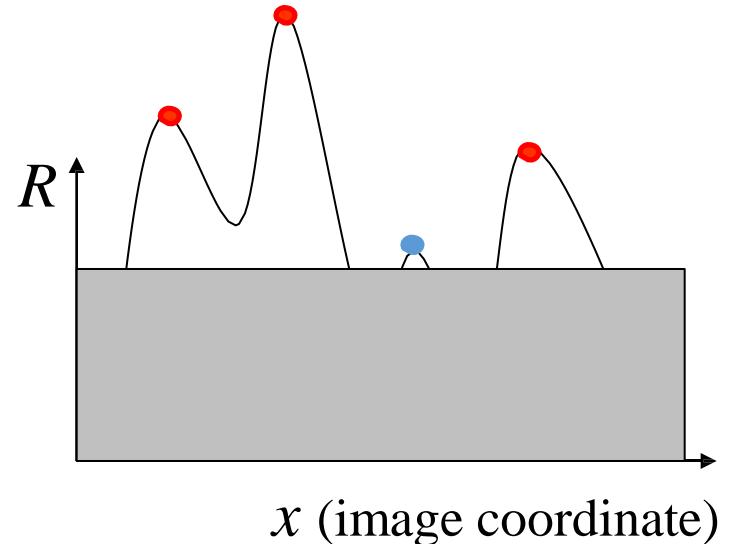
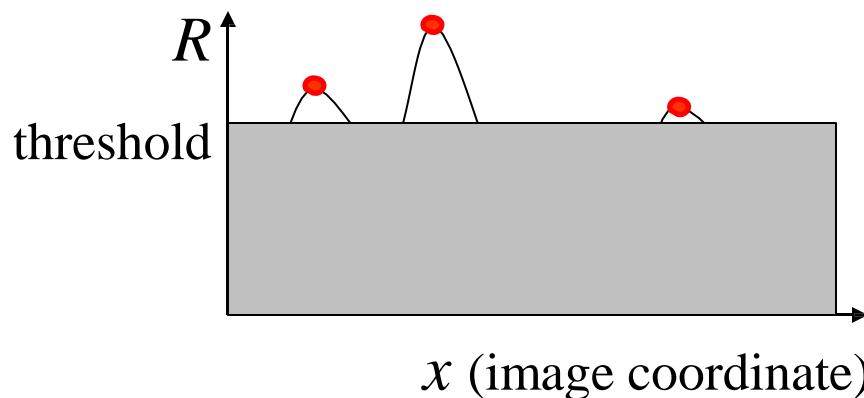
Corner response is invariant to image rotation

Harris Detector: Invariance Properties

- Affine intensity change: $I \rightarrow aI + b$
 - ✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

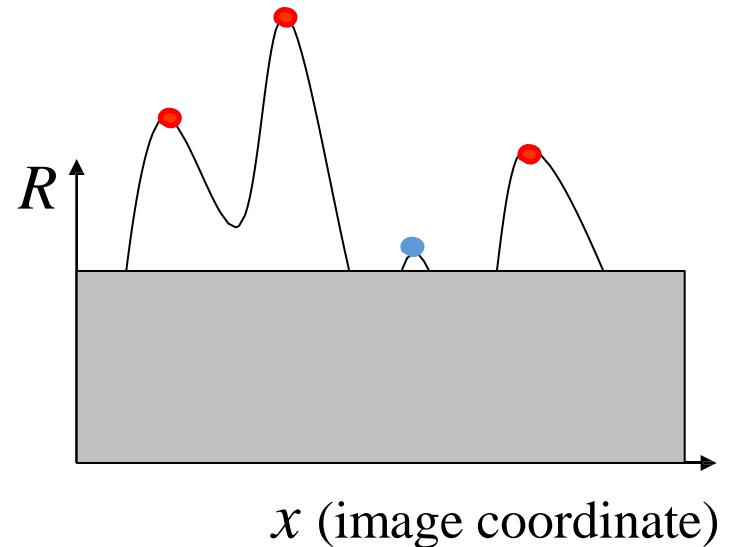
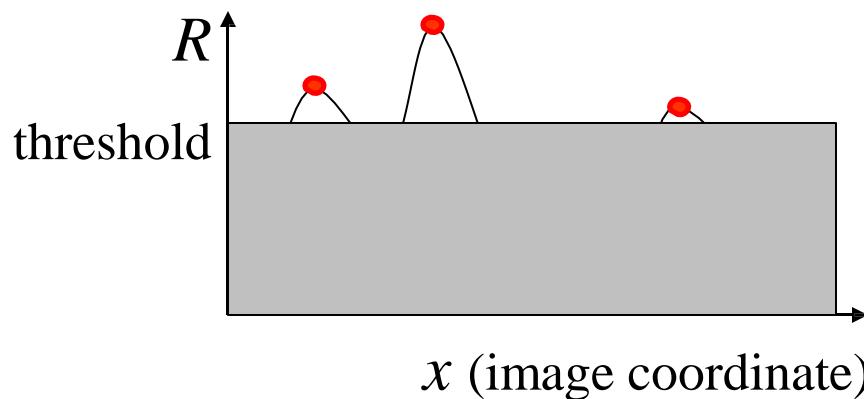
Harris Detector: Invariance Properties

- Affine intensity change: $I \rightarrow aI + b$
 - ✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
 - ✓ Intensity scale: $I \rightarrow aI$



Harris Detector: Invariance Properties

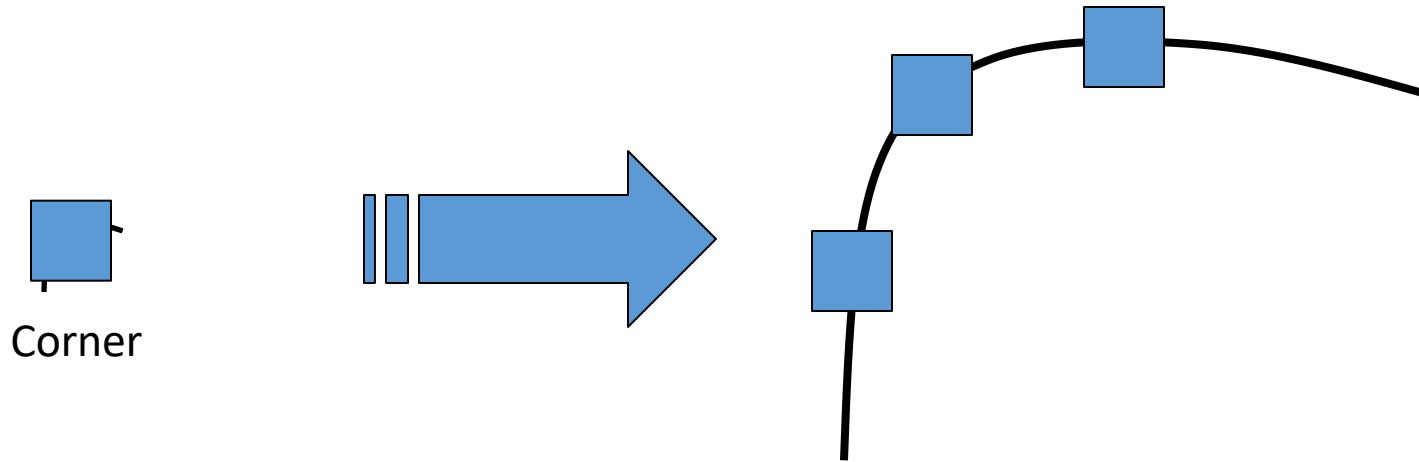
- Affine intensity change: $I \rightarrow aI + b$
 - ✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
 - ✓ Intensity scale: $I \rightarrow a I$



Partially invariant to affine intensity change

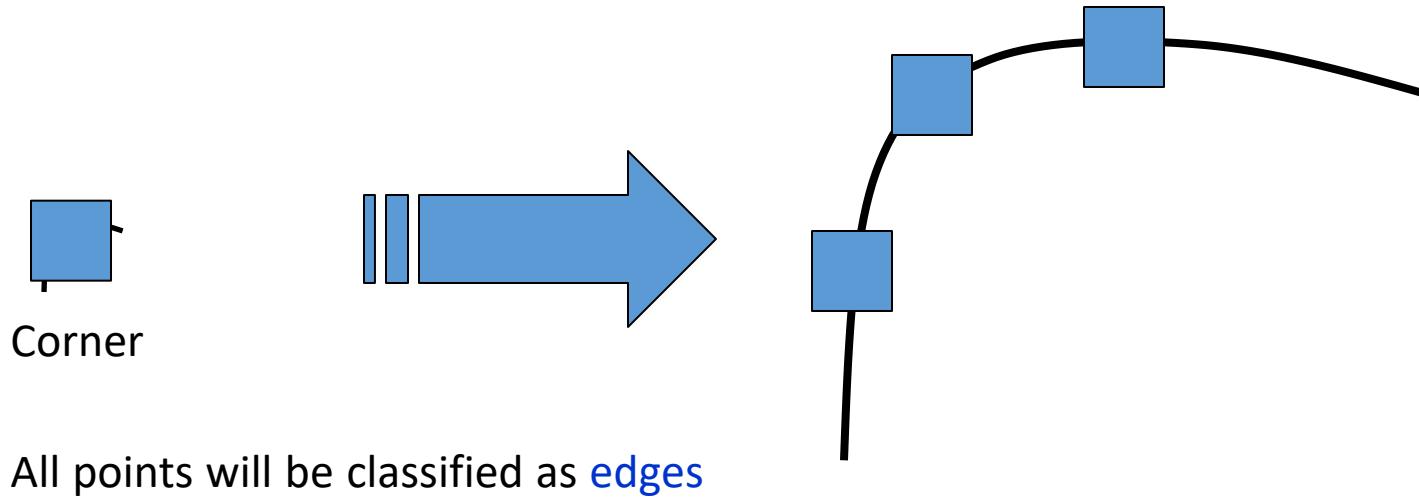
Harris Detector: Invariance Properties

- Scaling



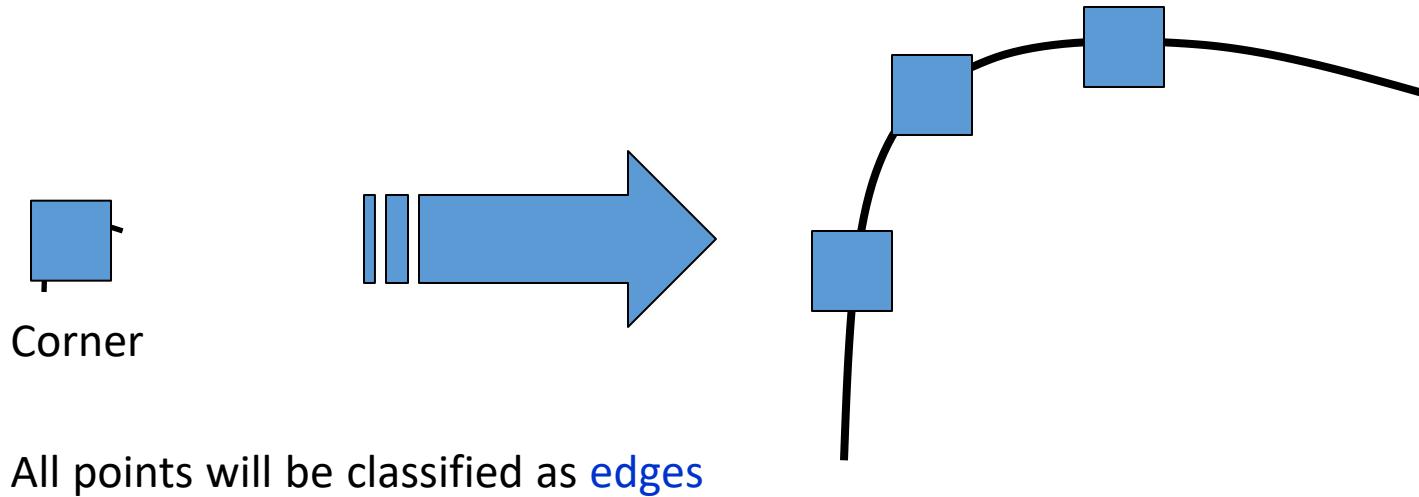
Harris Detector: Invariance Properties

- Scaling



Harris Detector: Invariance Properties

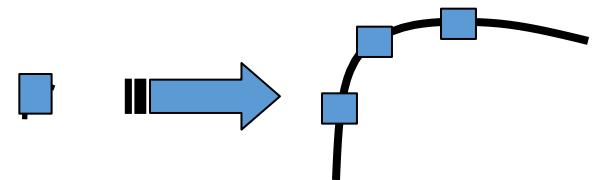
- Scaling



Not invariant to scaling

Things to remember

- Keypoint detection: repeatable and distinctive
 - Corners, Harris
 - Invariant to scale, rotation, etc.
- Harris Corner Detection
 - Rotation Invariant
 - Partial Intensity Change Invariant
 - *Not Invariant to Scale*



Acknowledgements

- Thanks to the following researchers for making their teaching/research material online
 - Forsyth
 - Steve Seitz
 - Noah Snavely
 - J.B. Huang
 - Derek Hoiem
 - D. Lowe
 - A. Bobick
 - S. Lazebnik
 - K. Grauman
 - R. Zaleski
 - Leibe

Thank you

- Next class: Region Detection and Local Descriptors

