2. Basic Image Processing

Image Filters, Convolution, Correlation, Fourier transform

Images are functions

Different types of images

- Radiance images, where a pixel value corresponds to the radiance from some point in the scene in the direction of the camera.
- X-Ray, MRI, Ultrasound, Light Microscopy, Electron Microscopy

$f(x,y) = [0, 255]^M \quad R^2 \rightarrow R^M$

$x = [a,b]$, $y = [c,d]$

Can be transformed with other functions

$f: [a,b] \times [c,d] \rightarrow [0,255]$

Domain support

range

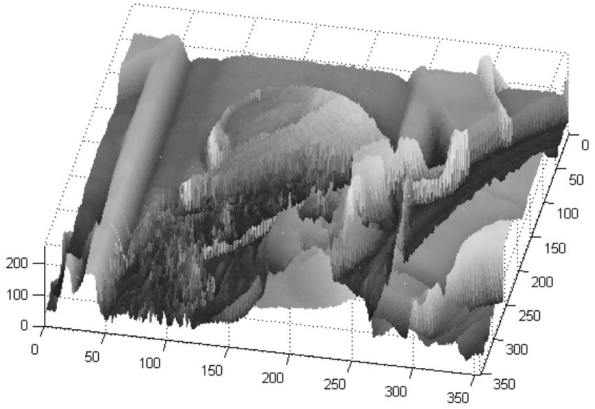


Image processing goals

- Image Compression
  - JPEG, JPEG2000, MPEG..

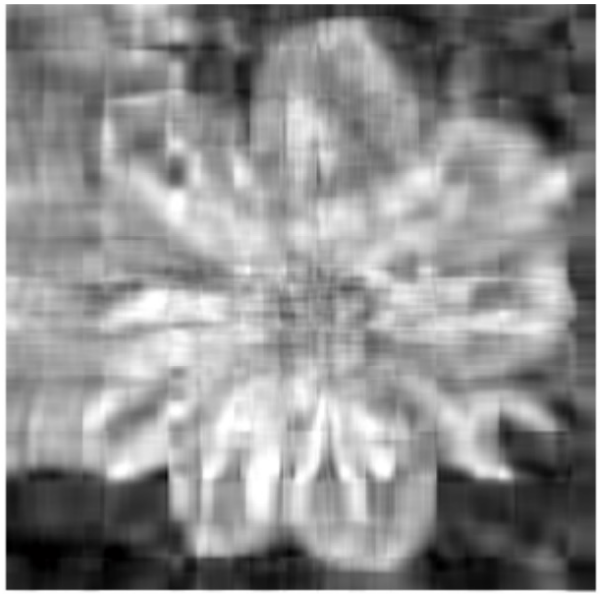


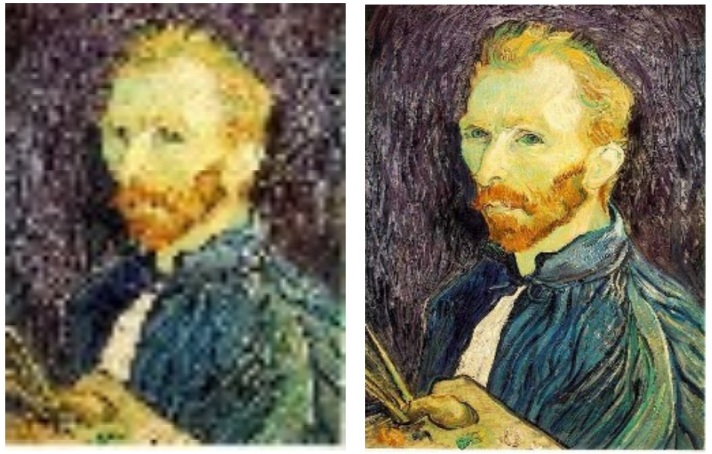
Image processing goals

- Image Restoration
  - denoising
  - deblurring (super-resolution)
  - In-painting

De-noising

Salt and pepper noise

Super-resolution

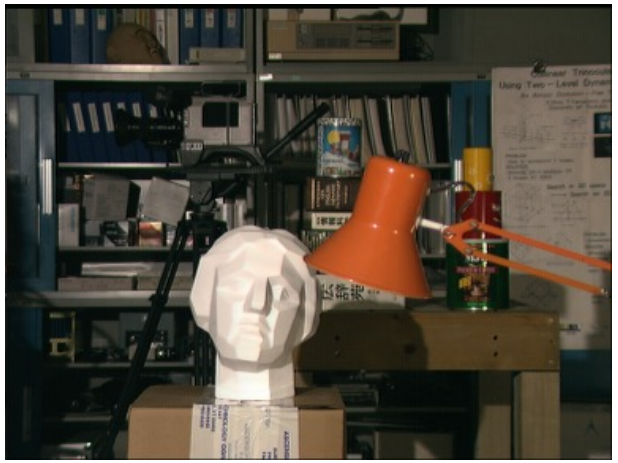


In-painting

Bertamio et al

Image processing goals

- Computing Field Properties
  - orientation
  - optical flow
  - disparity
- Locating Structural Features
  - corners
  - Edges
  - …



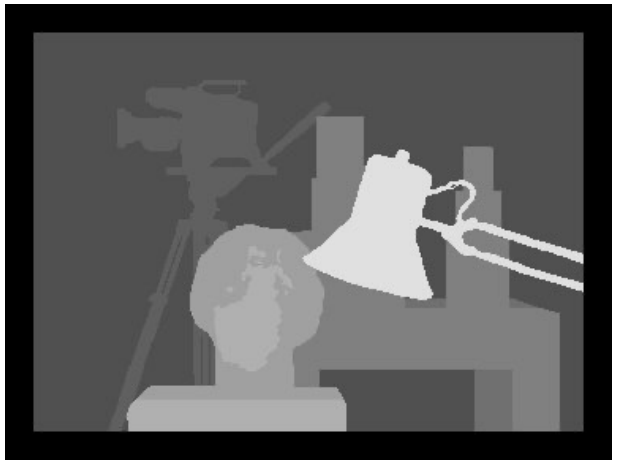


Image processing goals

- Locating Structural Features
  - corners
  - Edges
  - …

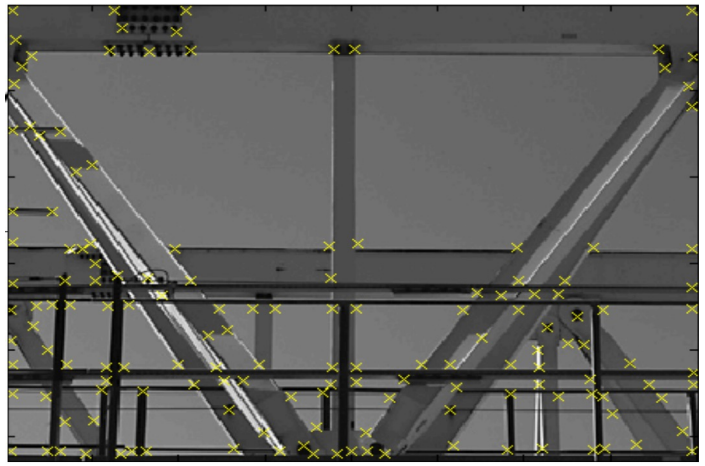
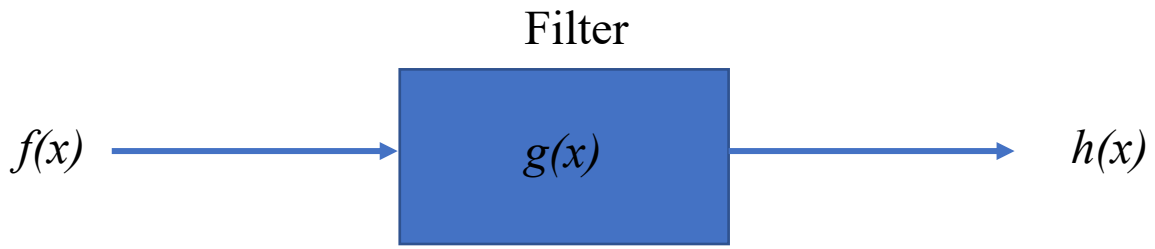
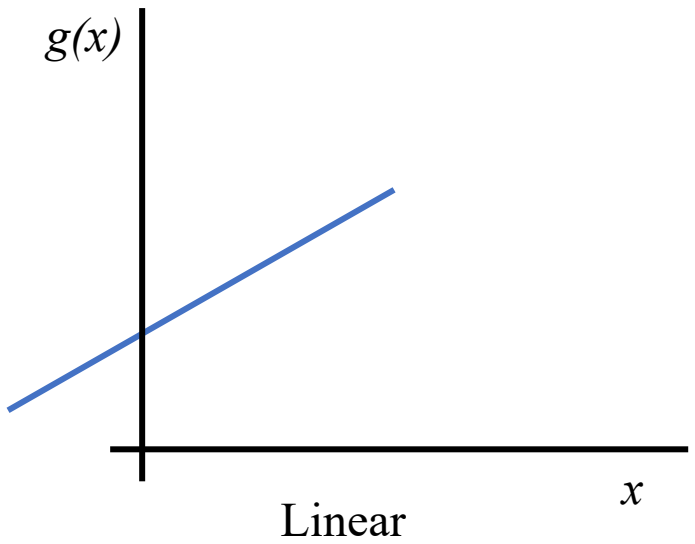
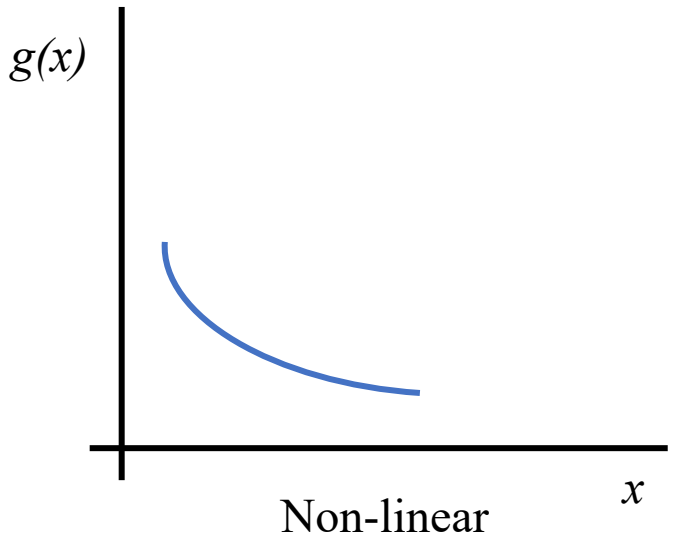


Image filters

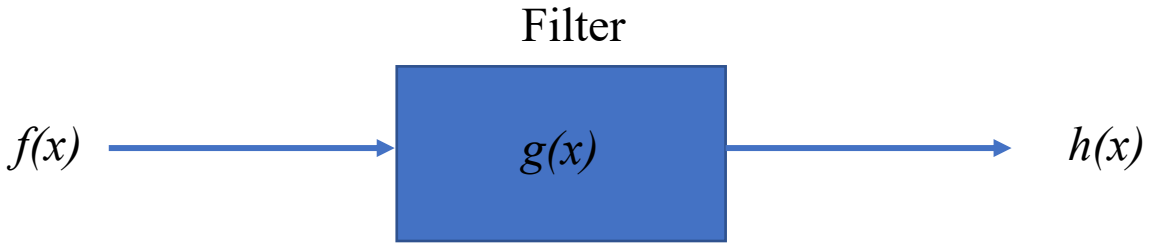


$h(x) = g(f(x)) = g(x) * f(x)$





Linear filters

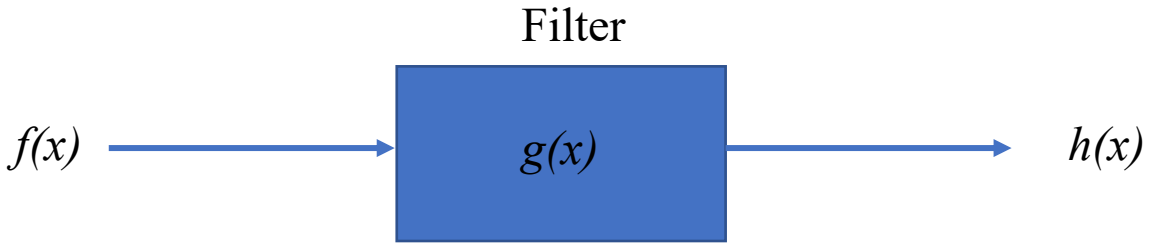


$h(x) = g(f(x)) = g(x)*f(x)$

Superposition: $g(a*f_1 + b*f_2) = a*g(f_1) + b*g(f_2)$

Shift Invariance: if $h(x) = g(f(x))$, then $h(x-x_0) = g(f(x-x_0))$

Translation Invariance

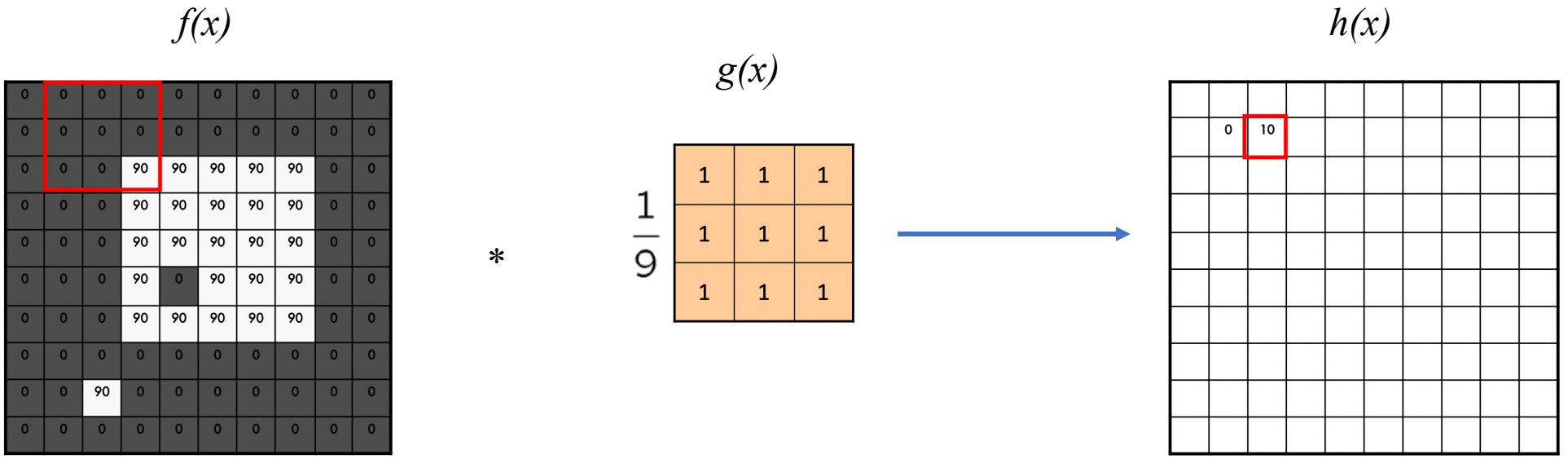


$h(x) = g(f(x)) = g(x) * f(x)$

Shift Invariance: if $h(x) = g(f(x))$, then $h(x-x_0) = g(f(x-x_0))$

Moving average filter (3X3)

- Replaces each pixel with an average of its neighborhood
- Achieves smoothing, reduces sharpness



Moving average filter

f(x) *h(x)*

Moving average filter (3X3)

$$h[n,m] = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n-k, m-l]$$

What kind of impact would a 5X5 or 7X7 filter have?
To handle boundaries, you can either reduce size of output image or do zero-padding, or do edge replication etc..
Matlab conv2 does zero padding

Thresholding filter

- Image segmentation by simple thresholding

$$h[n,m] = \begin{cases} 255, & f[n,m] > 100 \\ 0, & \text{otherwise.} \end{cases}$$

C&C: Convolution, Correlation (& auto-Correlation)

Convolution: generic name of kernel/filter-based processing to design sharpening, blurring filters etc

Discrete convolution:

1. Take a filter. Flip by origin such that g(x,y)=g(-x,-y)
2. Shift the folded results by x_0, y_0 to form $g(x_0 - x, y_0 - y)$
3. Multiply by signal $f(x_0, y_0)$ and sum all quantities

Convolution in 2D

Convolution: generic name of kernel/filter-based processing to design sharpening, blurring filters etc

Represented by *

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right) [2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

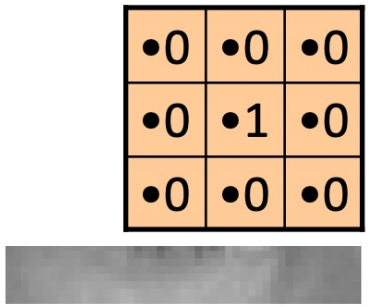
Original

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

(Note that filter sums to 1)

"details of the image"

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



Original

Shifted right
By 1 pixel

Convolution in 2D

Original

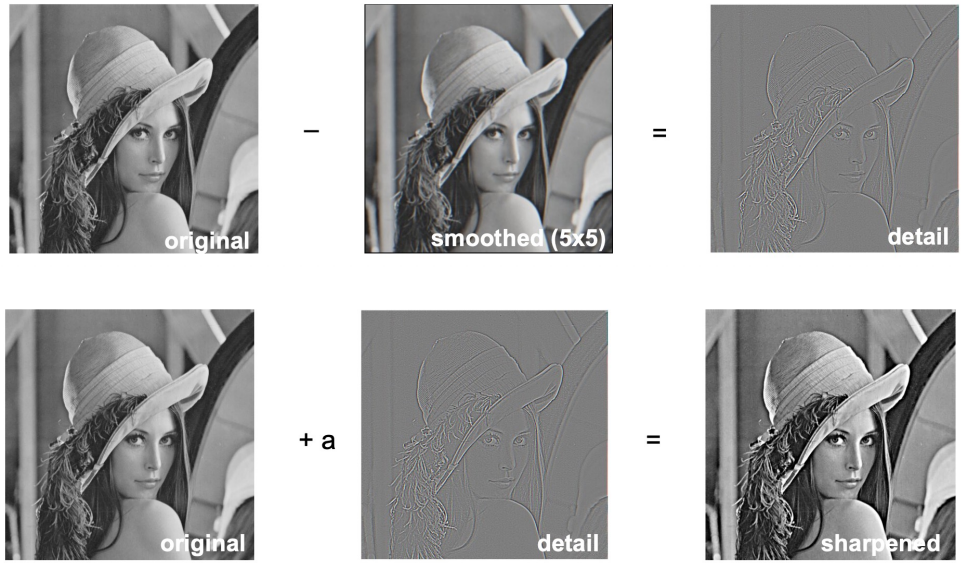
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

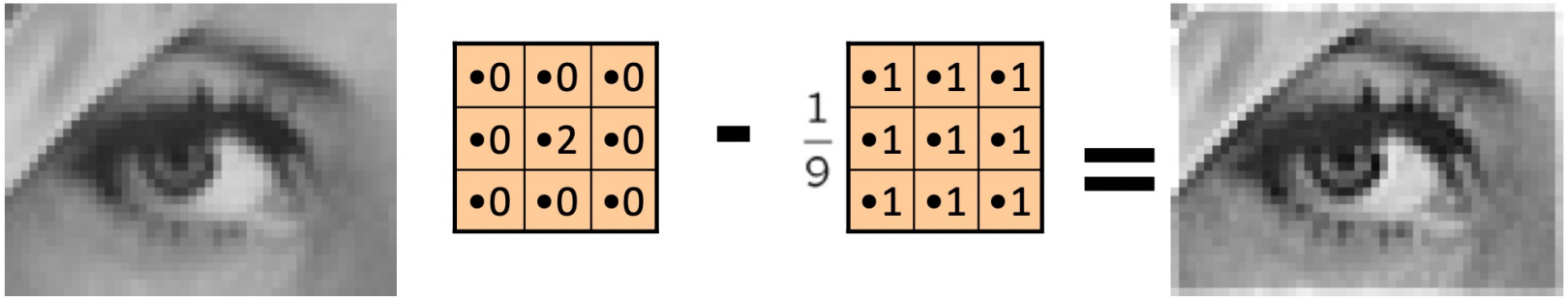
(Note that filter sums to 1)

"details of the image"

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

Convolution in 2D





Original

Sharpened image

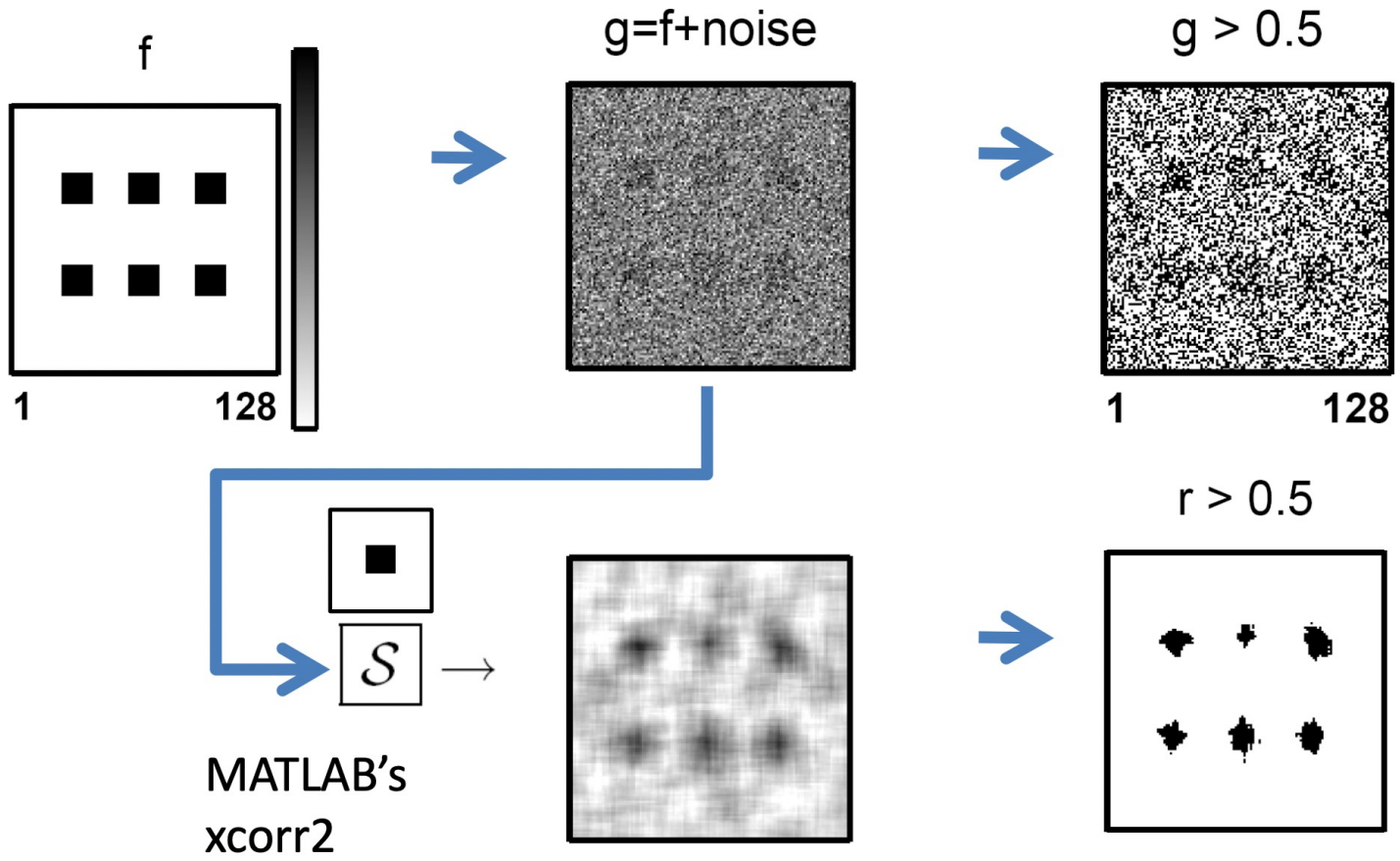
Cross-correlation and auto-correlation

Correlation = convolution without a flip

Cross-correlation: between 2 signals (f and g)

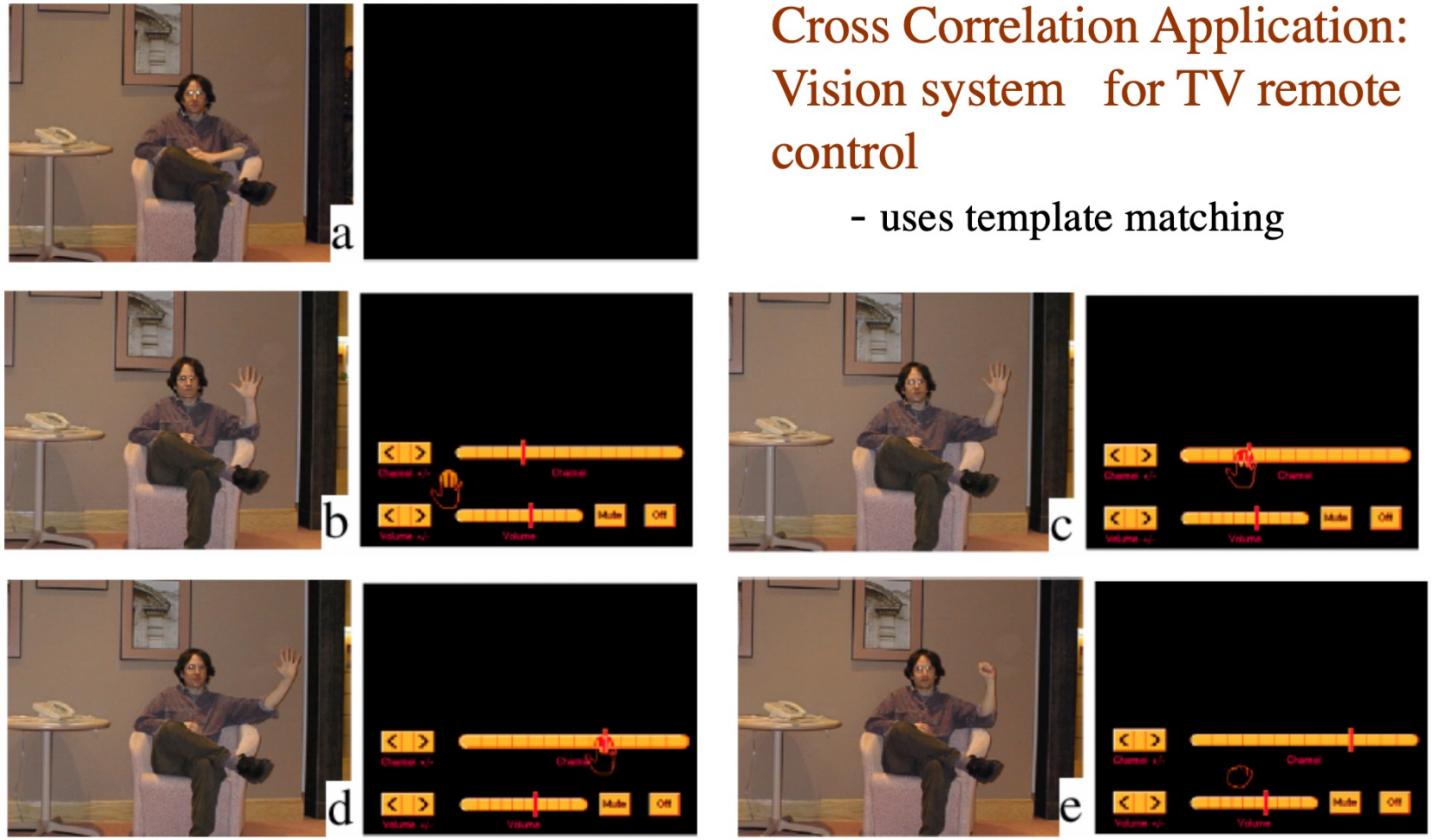
Auto-correlation: between same signal (f & f)

Cross-correlation and auto-correlation



The correlation result reaches a maximum at the time when the two signals match best.

Cross-correlation



Cross Correlation Application: Vision system for TV remote control

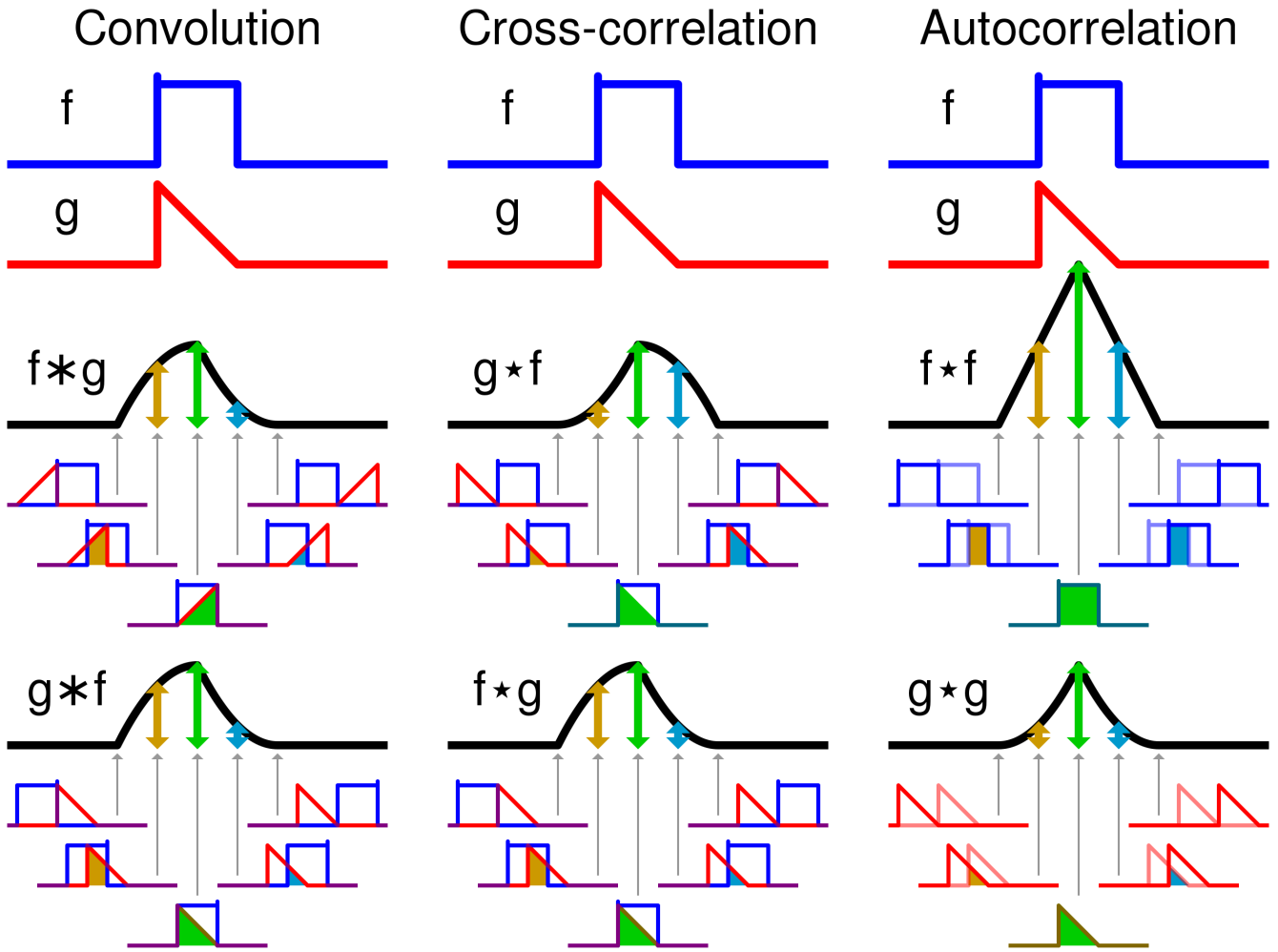
- uses template matching

Figure from "Computer Vision for Interactive Computer Graphics," W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

Convolution and Cross-correlation

- A convolution is a filtering operation that expresses the amount of overlap of one function as it is shifted over another function.
- Correlation compares the similarity of two sets of data. It computes a measure of similarity of two input signals as they are shifted by one another.

Cross-correlation and auto-correlation



Edge Detection

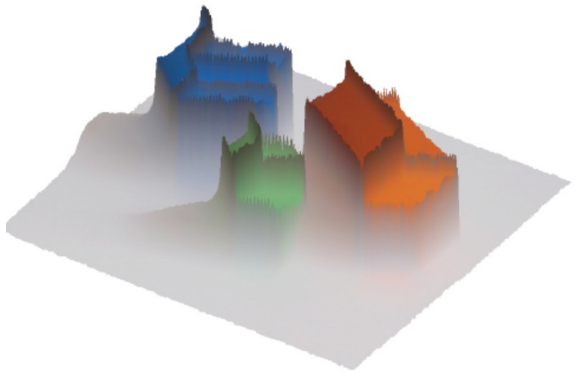


Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

Approximation image derivative:

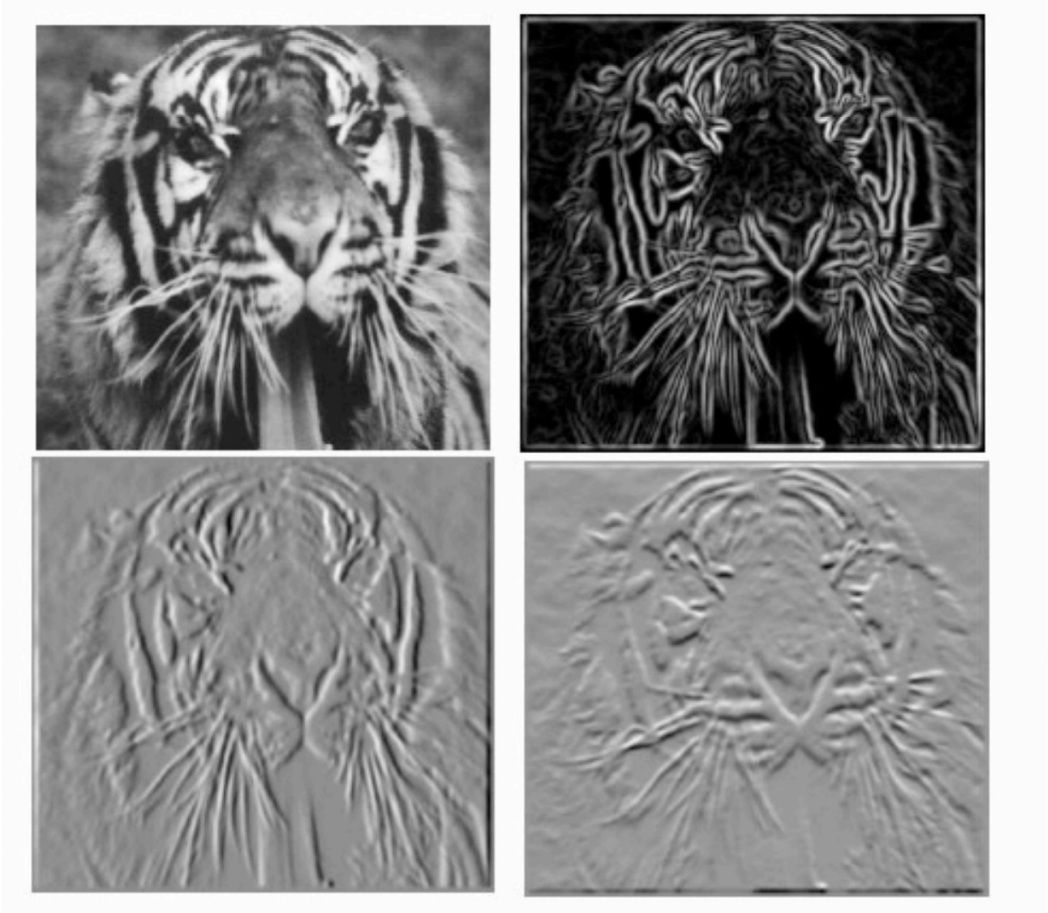
$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

Edge strength $E(x, y) = |\nabla \mathbf{I}(x, y)|$

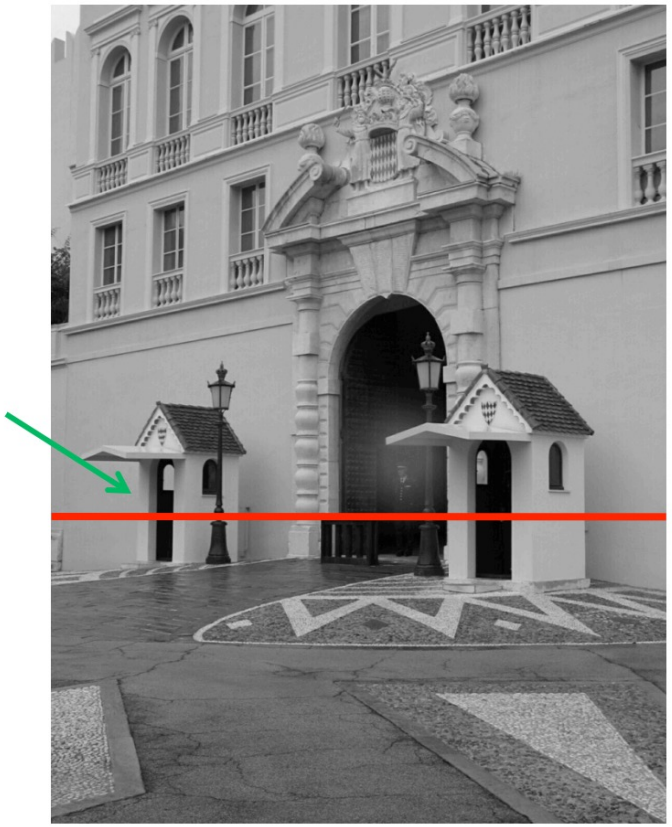
Edge orientation: $\theta(x, y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I} / \partial y}{\partial \mathbf{I} / \partial x}$

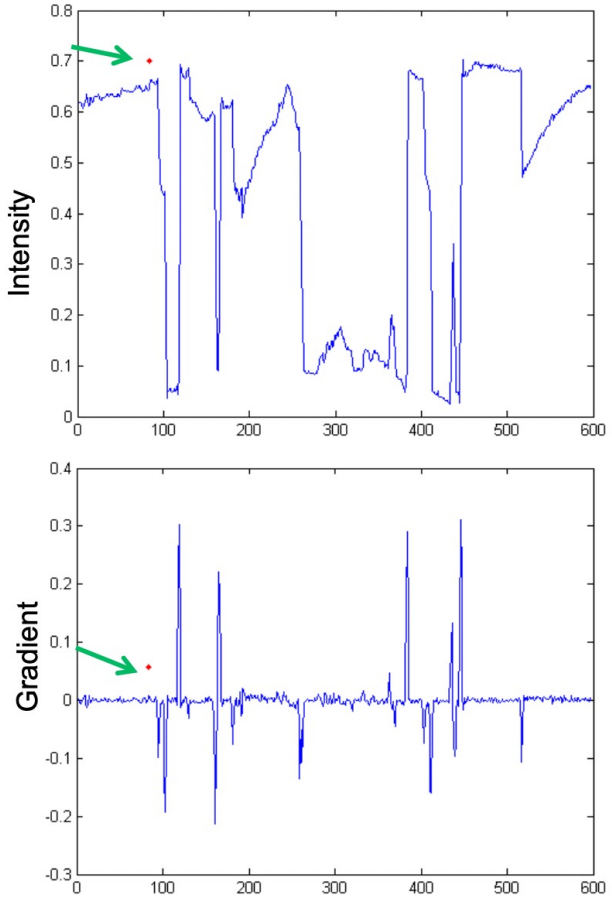
Edge normal: $\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$

Edge Detection



Edge Detection



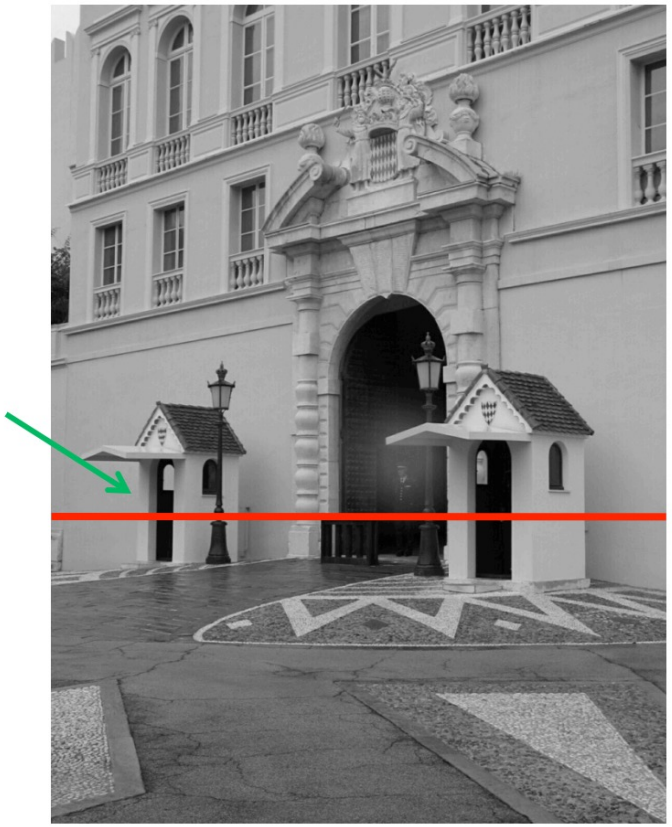


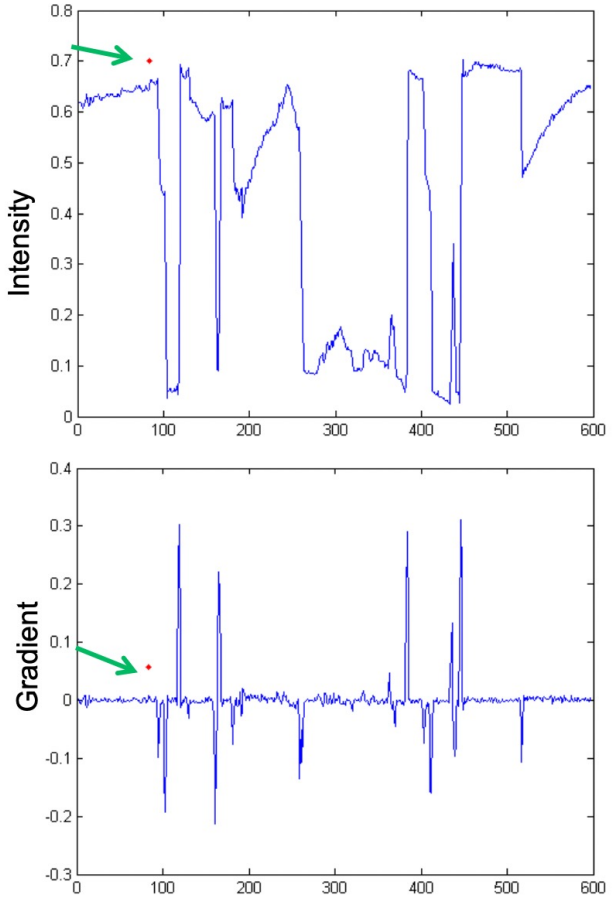
Finite differences are very sensitive to noise.

Therefore, it is not a good idea to compute edges directly on images.

Solution: smooth first

Edge Detection



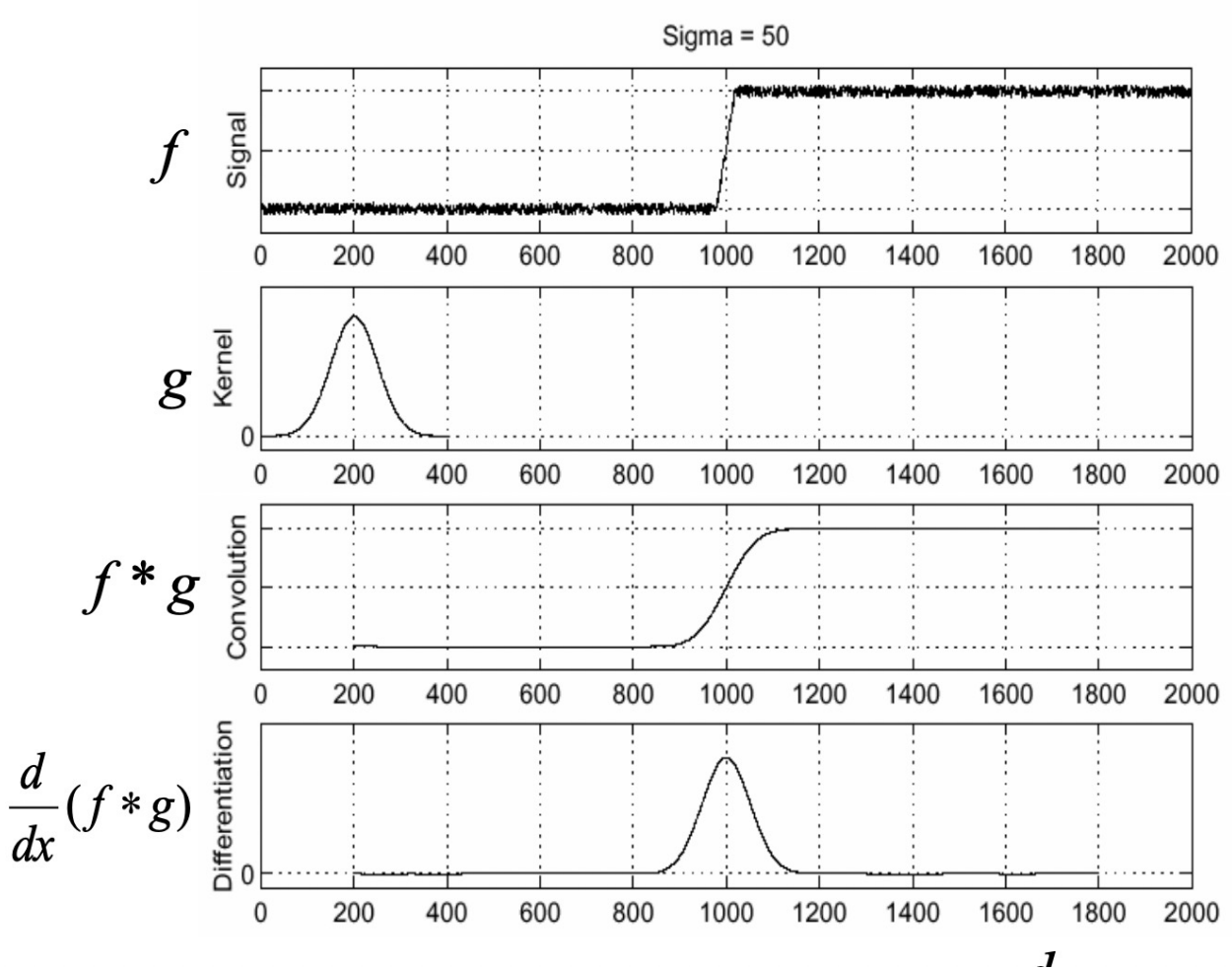


Finite differences are very sensitive to noise.

Therefore, it is not a good idea to compute edges directly on images.

Solution: smooth first

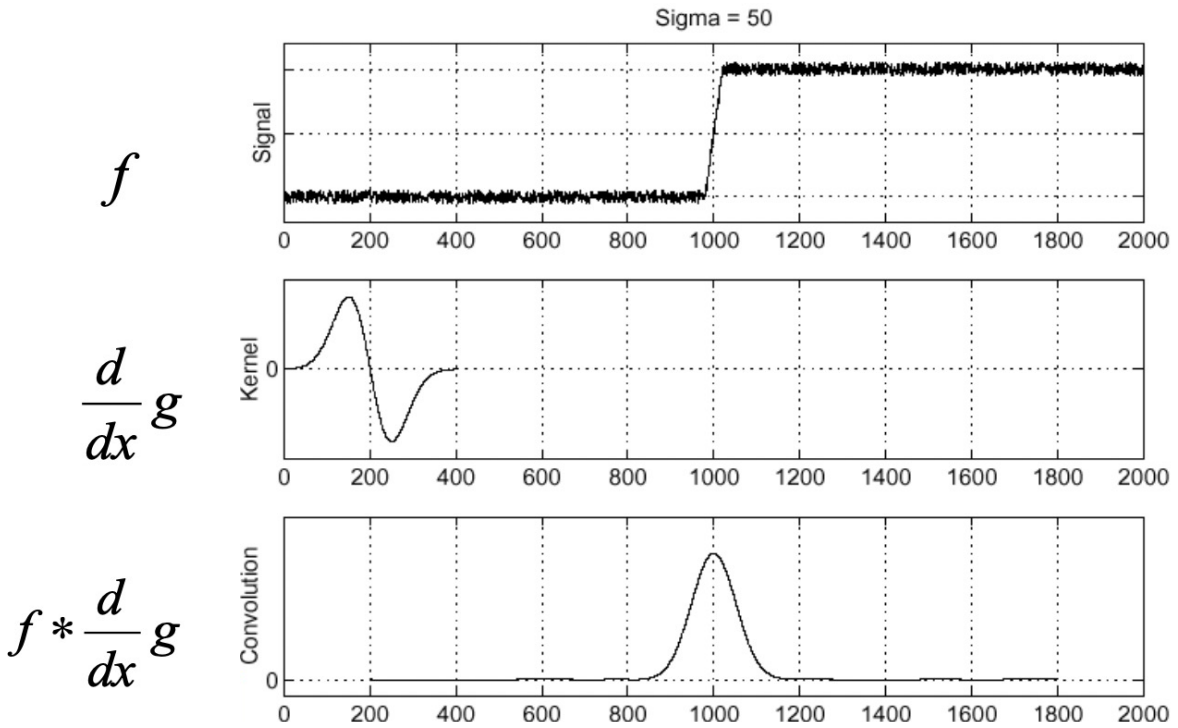
Edge Detection



Edge Detection

Convolution has the following property:

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



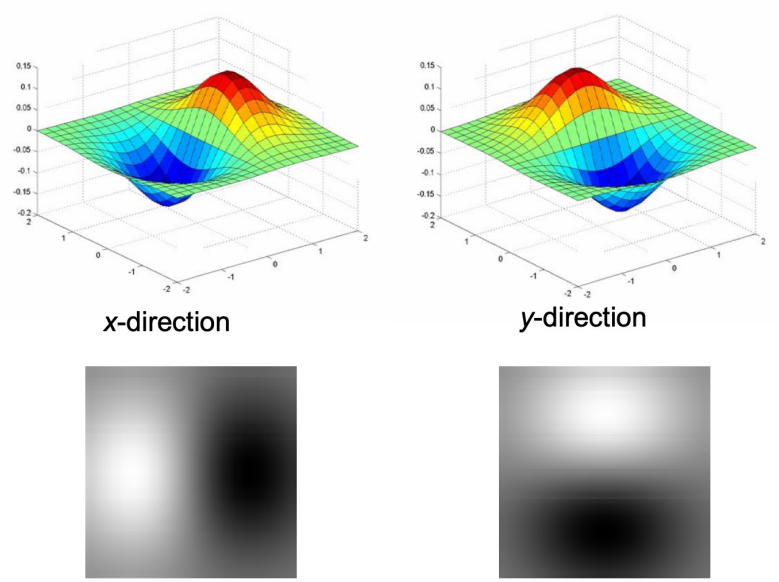
f

$\frac{d}{dx} g$

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

Canny Edge Detection

1. Apply Gaussian filter to smooth the image
2. Find the intensity gradients of the image



Canny Edge Detection

1. Apply Gaussian filter to smooth the image
2. Find the intensity gradients of the image
3. Apply gradient magnitude thresholding
4. Apply double threshold to determine potential edges
5. Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Canny Edge Detection

P1: Moving average and Thresholding

- Is moving average is shift invariant?
- Is thresholding shift –invariant?
- Is moving average linear?
- Is thresholding linear?

Hint: try how $a*f_1 + b*f_2$ transform to check linearity

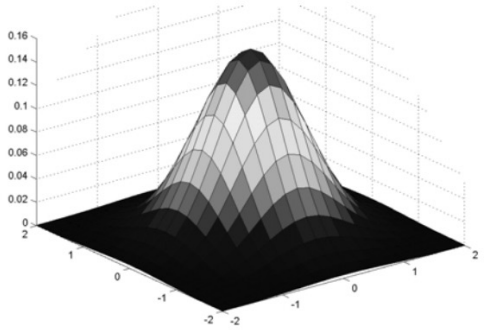
And how $f(n-n_0,m-m_0)$ transform to check shift invariance

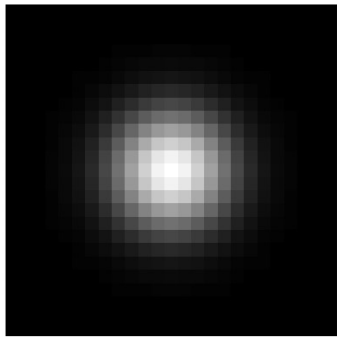
→ Formulate a simple example to check whether your result is correct.

P2: Averaging using gaussian filter (3X3)

Obtain image smoothing using gaussian filter. How do you find the result to be different from moving average filter.

- Idea: Weight contributions of neighboring pixels by nearness





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

5 x 5, $\sigma = 1$

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

P3: Finite differences as filters

Implement gradients using finite differences (df/dx = f(x+h)-f(x)/h) as a filter.

Bonus: Is there a better way of computing gradients. Implement with filters and show the difference in result.

P4: C&C

Comment on following propertites (take examples if need be):

Associativity

Distributivity

Commutativity

Shift invariance

Linearity