BMB5113
COMPUTER VISION

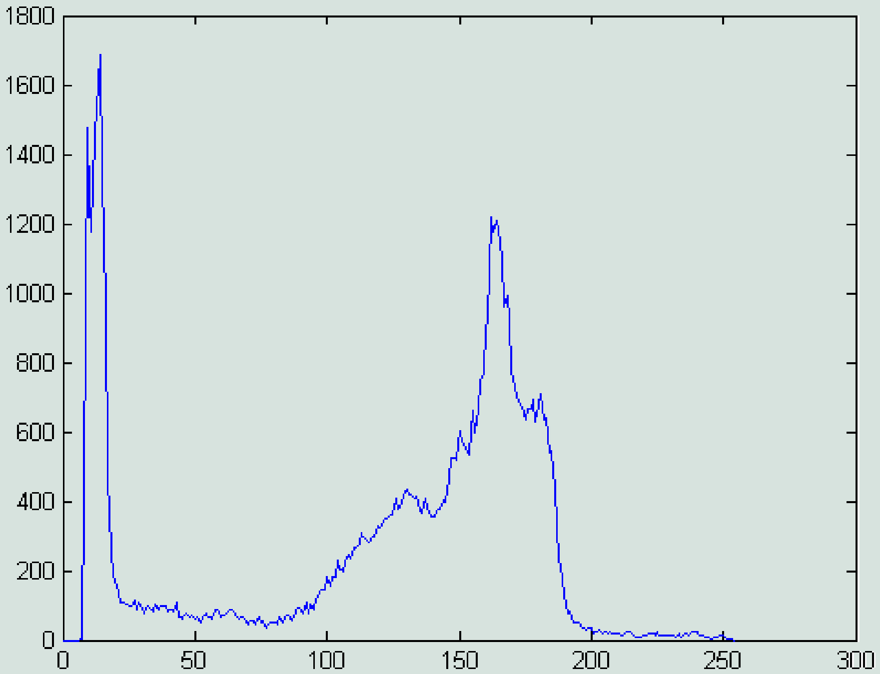
IMAGE FEATURES

Histograms

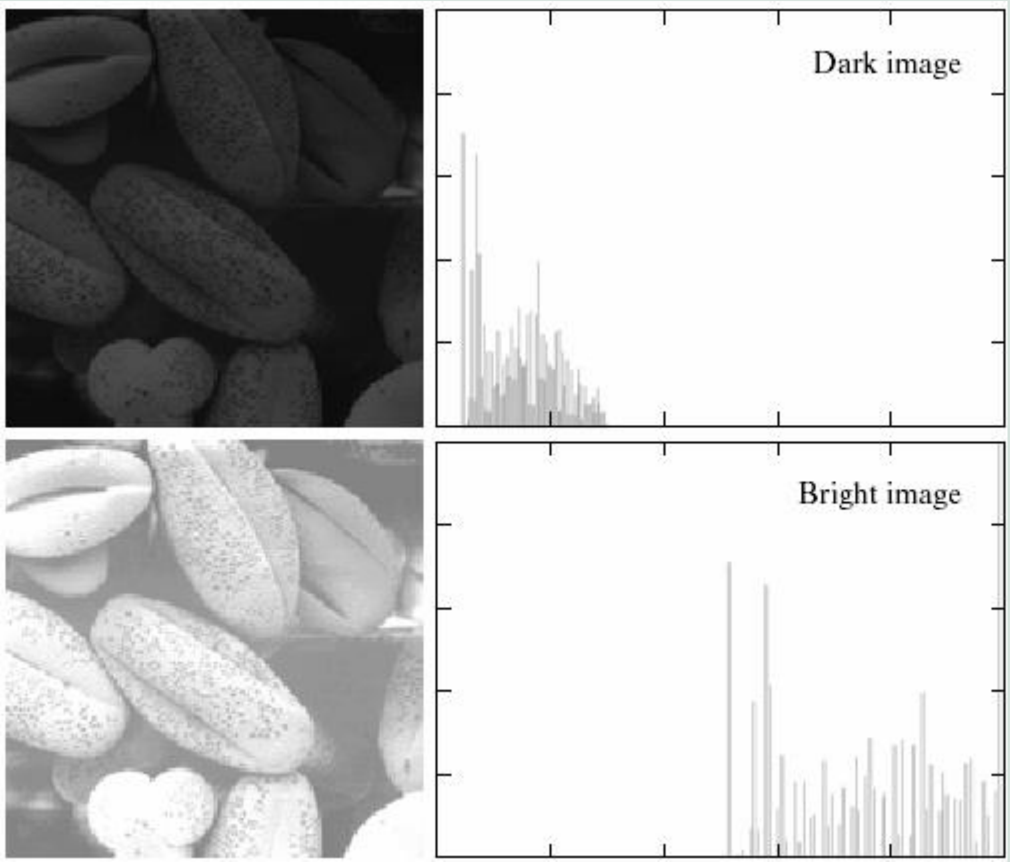
- The histogram of a digital image with intensity levels in the range [0, L-1] is a discrete function $h(r_k)=n_k$
  - r_k is the k^{th} intensity value
  - n_k is the number of pixels in the image with intensity r_k
- It is common practice to normalize a histogram by dividing each of its components by the total number of pixels in the image, denoted by the product MN.

An Image Histogram

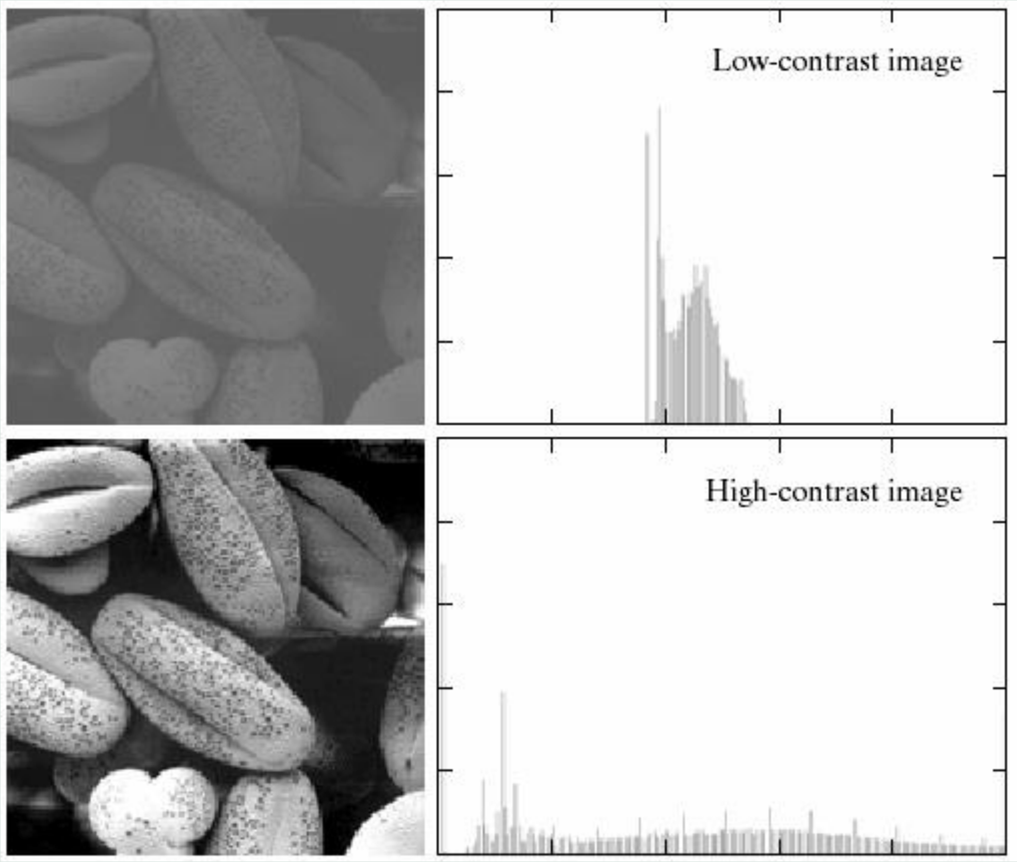
- Python function: hist(im.flatten(),128) #pylab
  histogram(im.flatten(),nbr_bins) #numpy



Histogram Assessment I

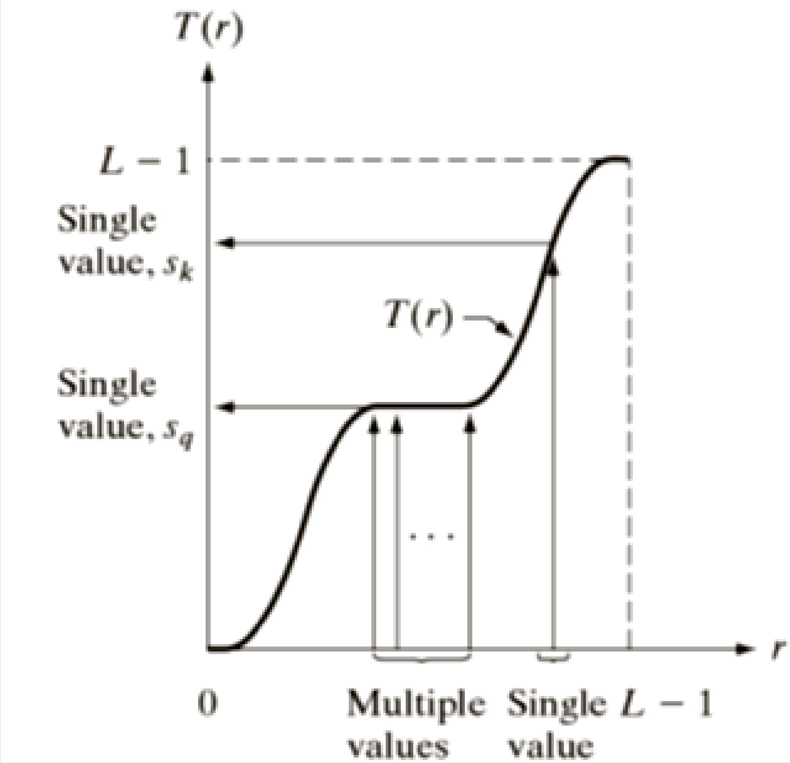


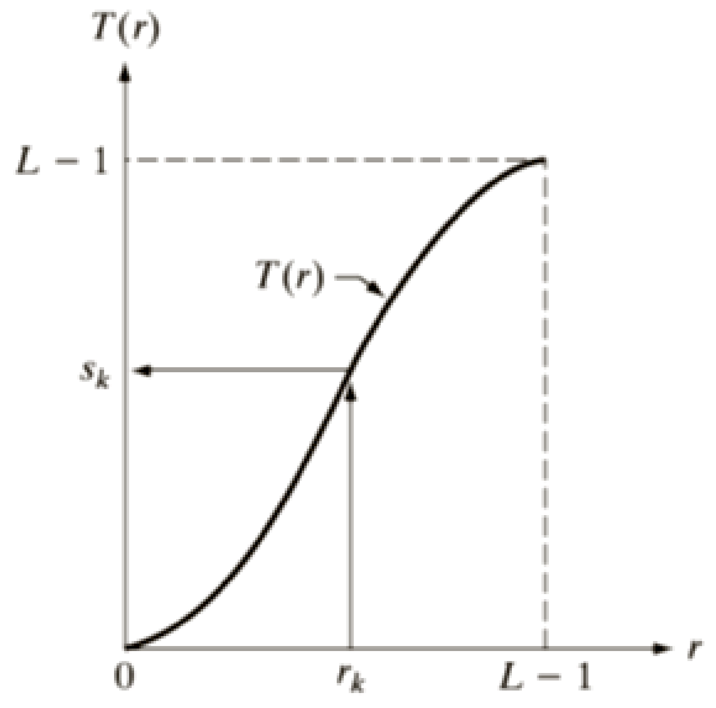
Histogram Assessment II



Histogram Equalization: Aim

1. One-to-one mapping between the image values



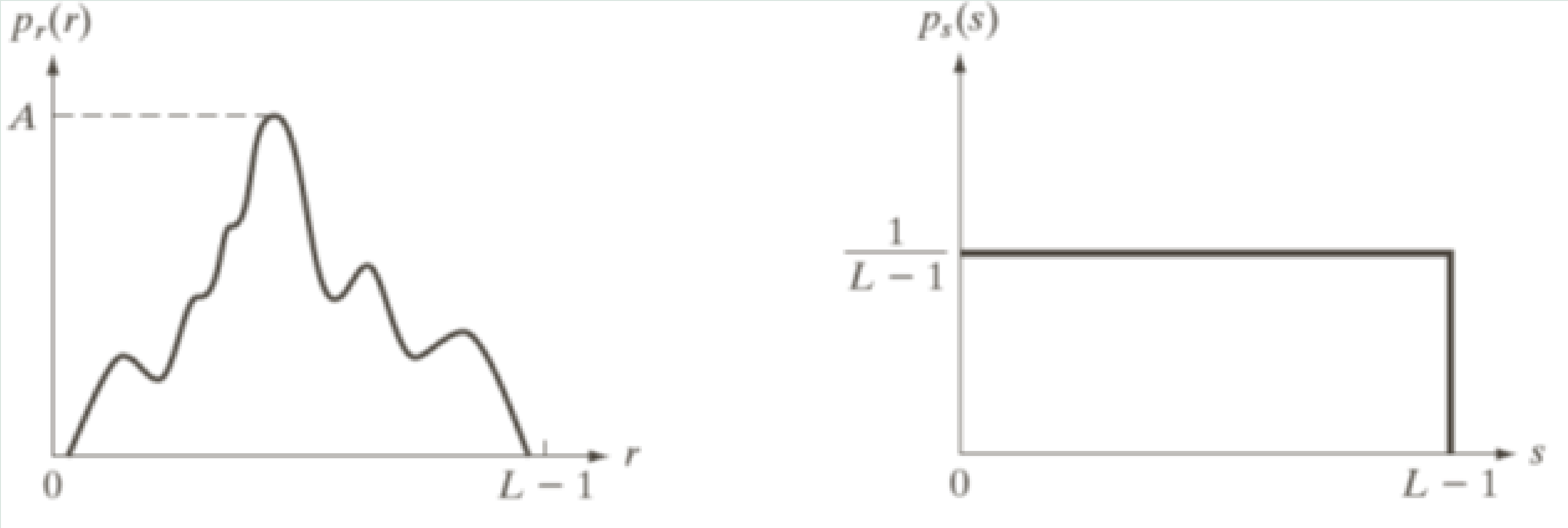


Monotonically increasing function, multiple values map to a single value

Strictly monotonically increasing function, one to one mapping between values

Histogram Equalization: Aim

2. A uniform probability density function (histogram)



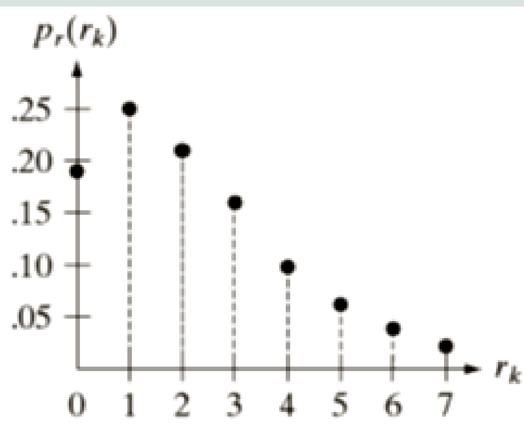
A Sample Probability Density Function

PDF of a 3-bit image with M=64 and N=64

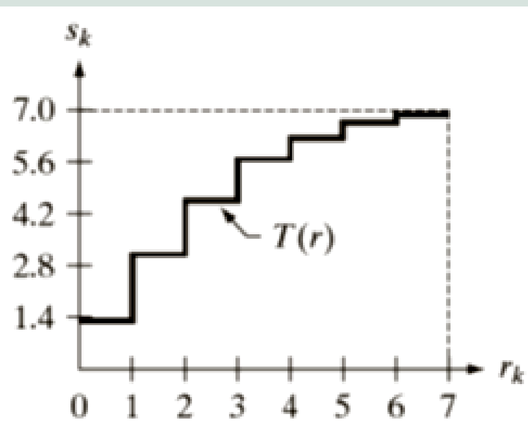
r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

- Python function:
  - im2,cdf = imtools.histeq(im)
  - where im is the input image
  - the number of intensity levels can be specified

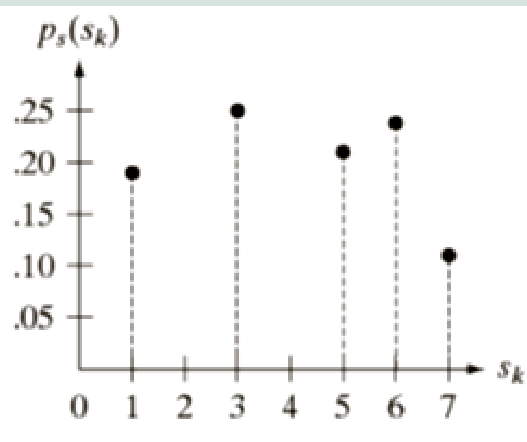
Original histogram



Transformation function



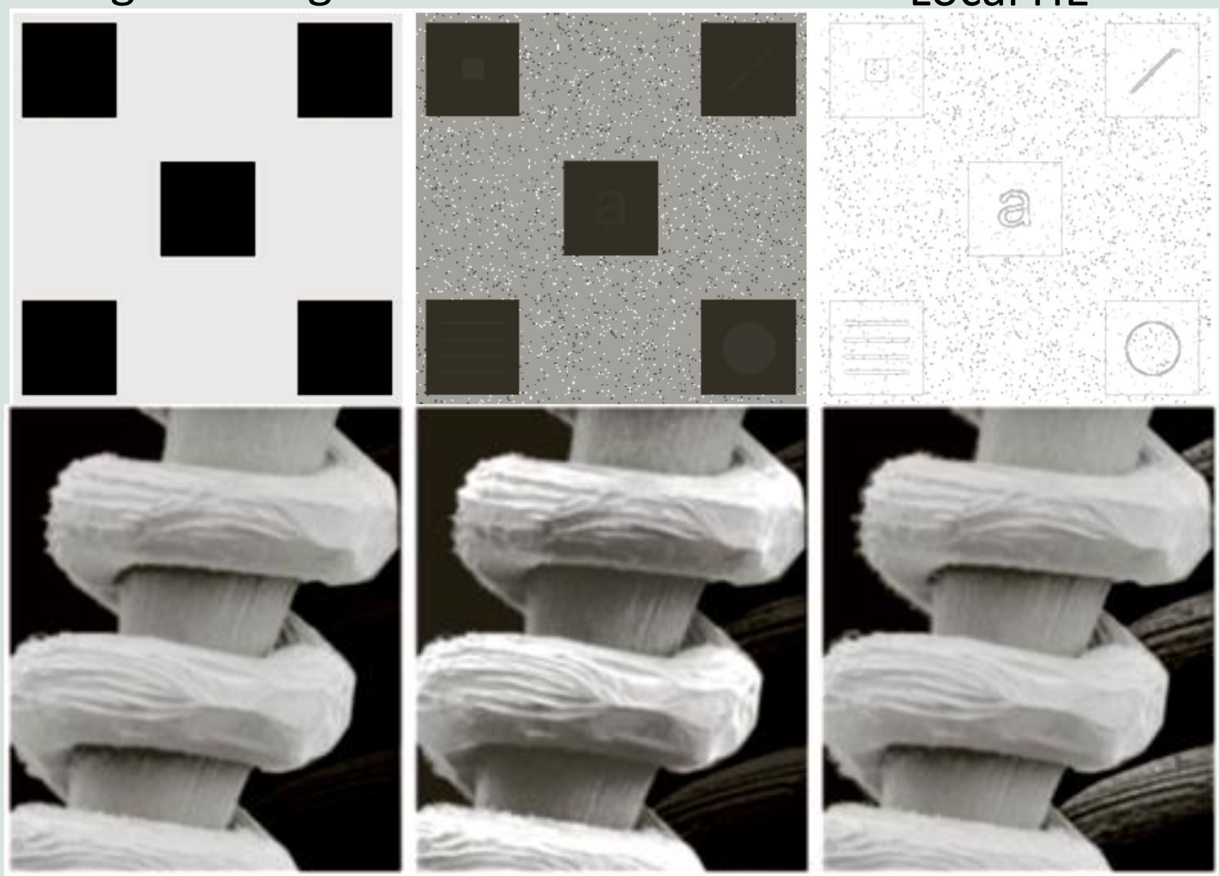
Equalized histogram



Local and Global Histogram Equalization

- Local histogram equalization:
  - in a neighborhood of a pixel, e.g. 3x3 neighborhood

Original Image Global HE Local HE

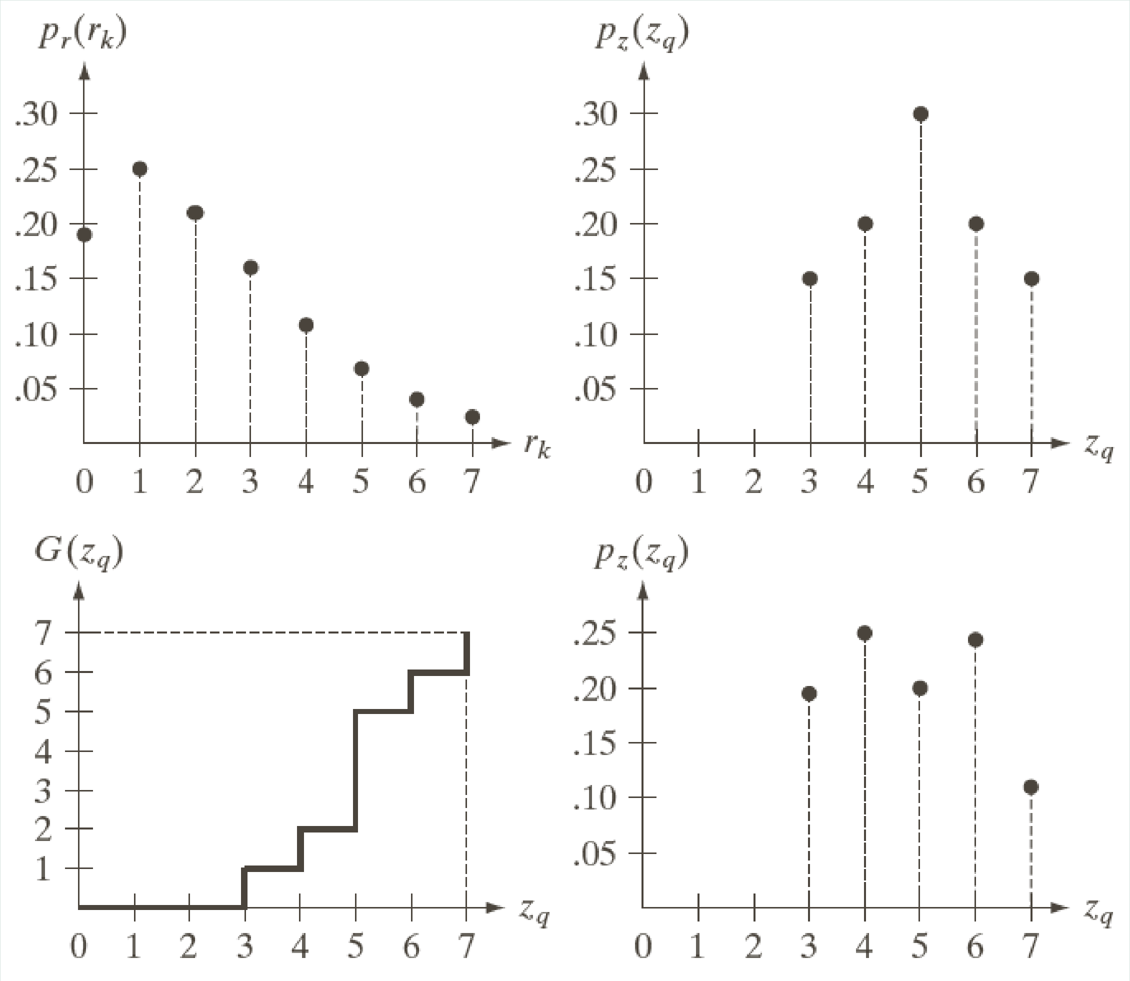


Local Image Statistics

Histogram Matching (Histogram Specification)

Current histogram

Transformation function for specified histogram



Specified histogram

Result of performing histogram specification

Histogram Matching (Histogram Specification)

Specified and actual histograms

z_q	Specified $p_z(z_q)$	Actual $p_z(z_k)$
$z_0 = 0$	0.00	0.00
$z_1 = 1$	0.00	0.00
$z_2 = 2$	0.00	0.00
$z_3 = 3$	0.15	0.19
$z_4 = 4$	0.20	0.25
$z_5 = 5$	0.30	0.21
$z_6 = 6$	0.20	0.24
$z_7 = 7$	0.15	0.11

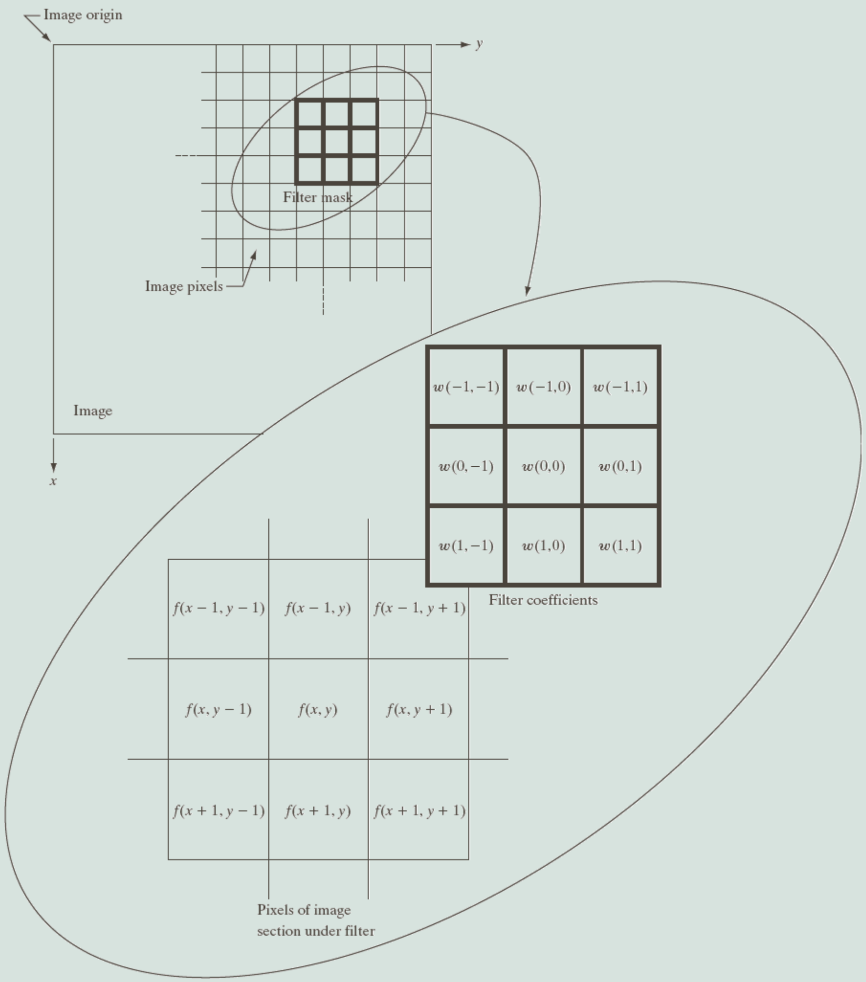
All possible values of transformation function G scaled, rounded and ordered with respect to z

z_q	$G(z_q)$
$z_0 = 0$	0
$z_1 = 1$	0
$z_2 = 2$	0
$z_3 = 3$	1
$z_4 = 4$	2
$z_5 = 5$	5
$z_6 = 6$	6
$z_7 = 7$	7

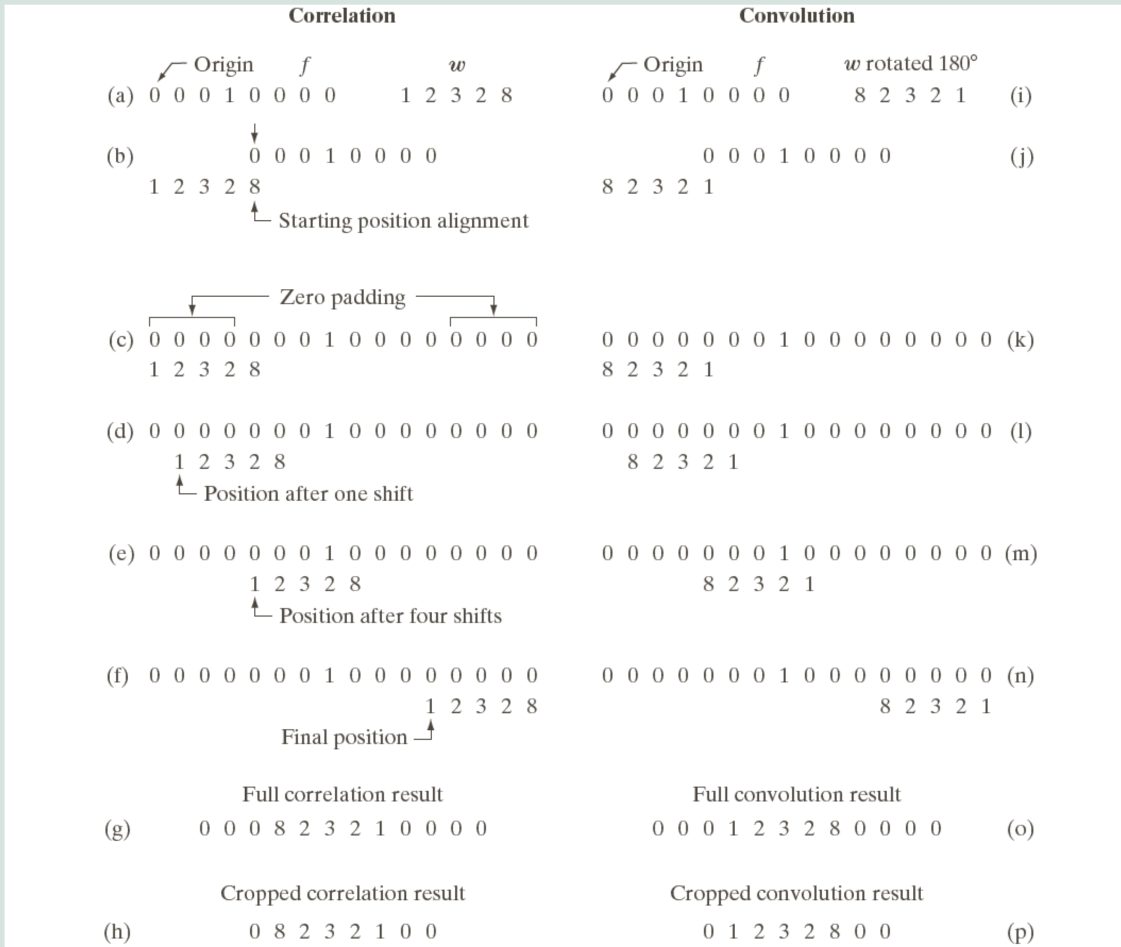
s_k	$\rightarrow$	z_q
1	$\rightarrow$	3
3	$\rightarrow$	4
5	$\rightarrow$	5
6	$\rightarrow$	6
7	$\rightarrow$	7

Mapping of all values of s_k into corresponding values z_q

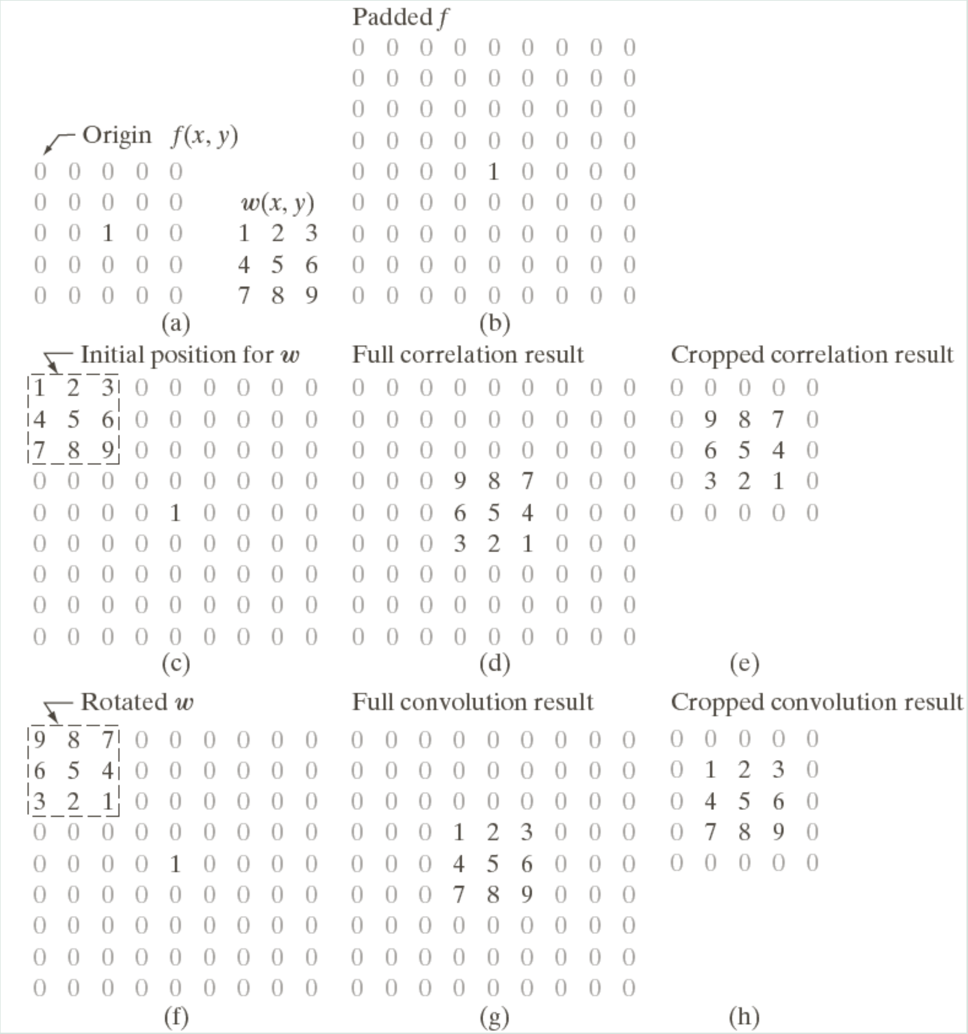
Spatial Filtering



Correlation and Convolution in 1-D



Correlation and Convolution in 2-D



Some Filters (Masks)

- Averaging filters

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Prewitt filters

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

- Sobel filters

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

Some Filters

- High-pass filters

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

,

-1	-2	-1
-2	12	-2
-1	-2	-1

- Low-pass filters

0	1	0
1	2	1
0	1	0

,

1	4	1
4	16	4
1	4	1

,

1	2	1
2	4	2
1	2	1

,

1	2	4	2	1
2	4	8	4	2
4	8	16	8	4
2	4	8	4	2
1	2	4	2	1

Python Implementation

```
import scipy.ndimage
import numpy as np
scipy.ndimage.correlate(A, B, mode='constant').transpose()
scipy.ndimage.correlate(A, B, mode='nearest').transpose()
scipy.ndimage.convolve(A, B, mode='nearest')

# Gaussian Filtering
im_filt = scipy.ndimage.filters.gaussian_filter(im,3)
```

- For scipy.ndimage library functions

https://docs.scipy.org/doc/scipy/reference/ndimage.html

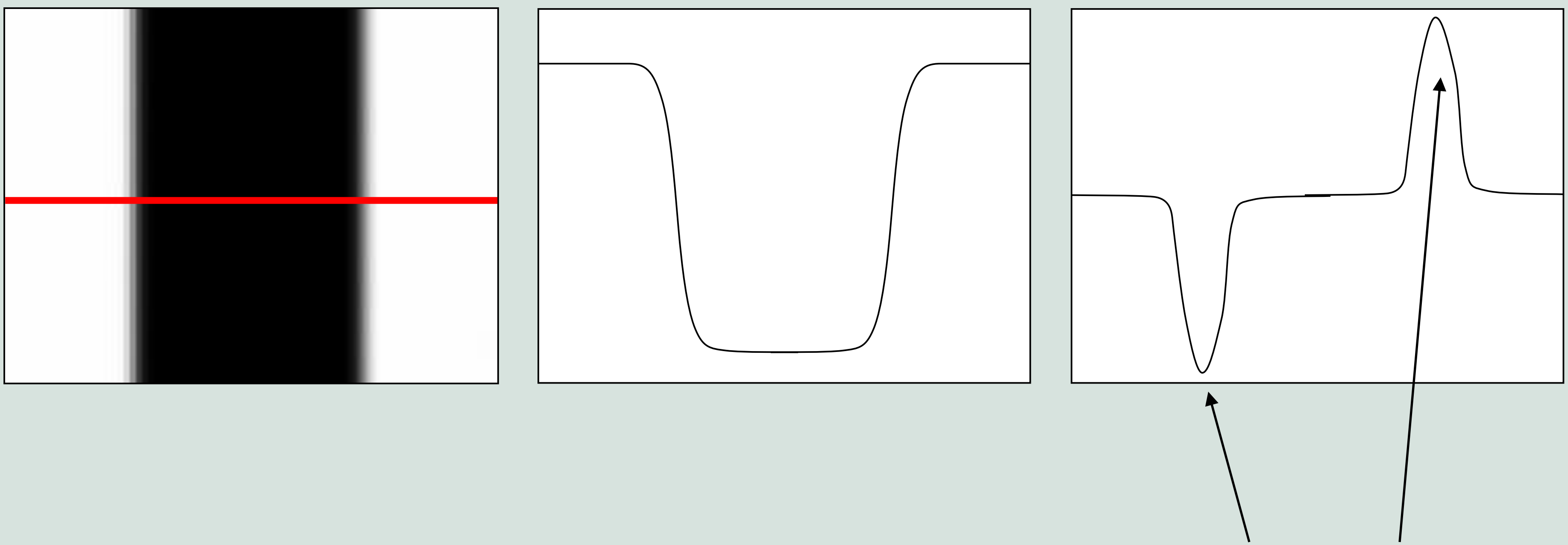
Derivatives and Edges

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

image

intensity function
(along horizontal scanline)

first derivative



edges correspond to
extrema of derivative

Differentiation and Convolution

For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

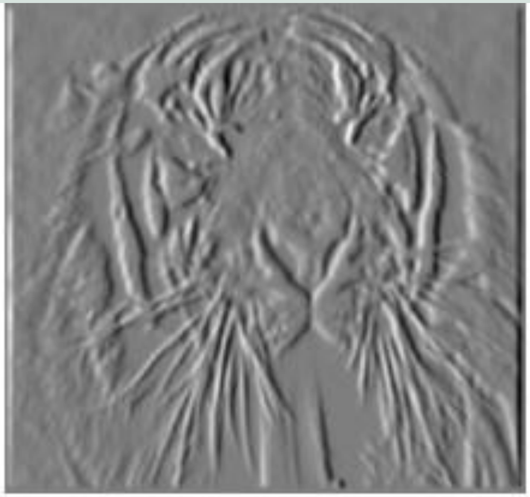
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

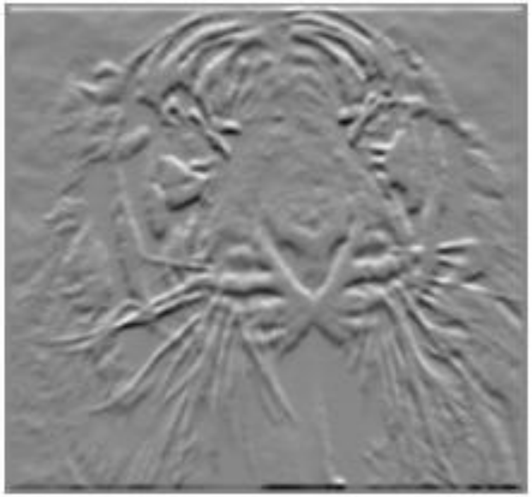
To implement above as convolution, what would be the associated filter?

Partial Derivatives of an Image

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$





-1	1

-1
1

? or

1
-1

Which shows changes with respect to x?

(showing flipped filters)

Assorted Finite Difference Filters

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $\boxed{M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

```
# Sobel
sy = ndimage.sobel(im, axis=0, mode='constant')
sx = ndimage.sobel(im, axis=1, mode='constant')
sob = np.hypot(sx, sy)
```

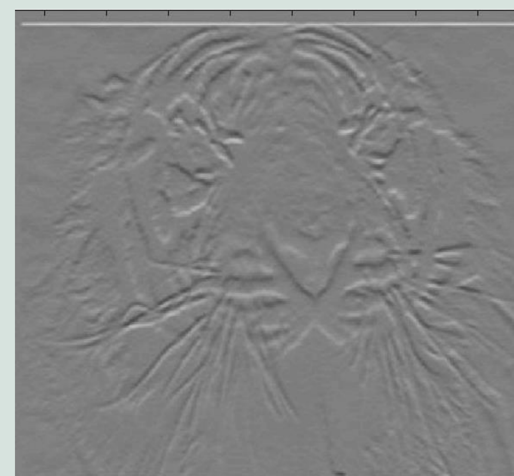
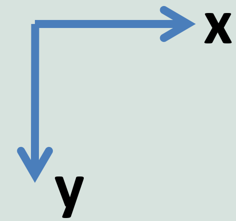


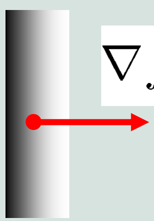
Image gradient

The gradient of an image:

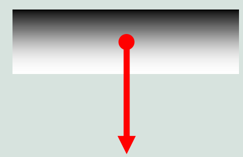
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



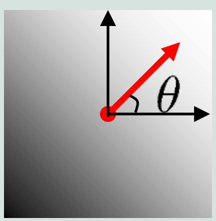
The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient direction (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

The *edge strength* is given by the gradient magnitude

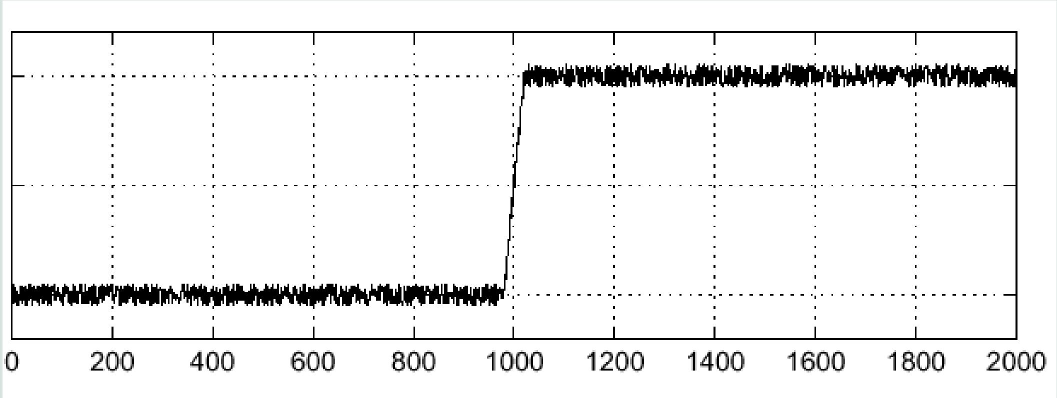
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Effects of noise

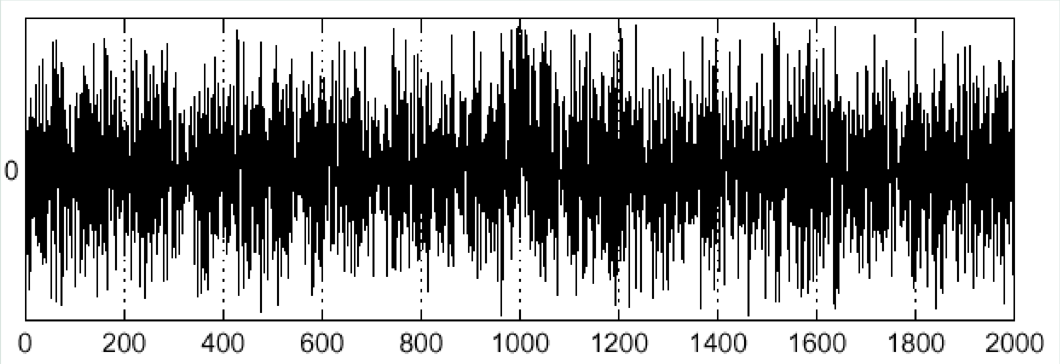
Consider a single row or column of the image

– Plotting intensity as a function of position gives a signal

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

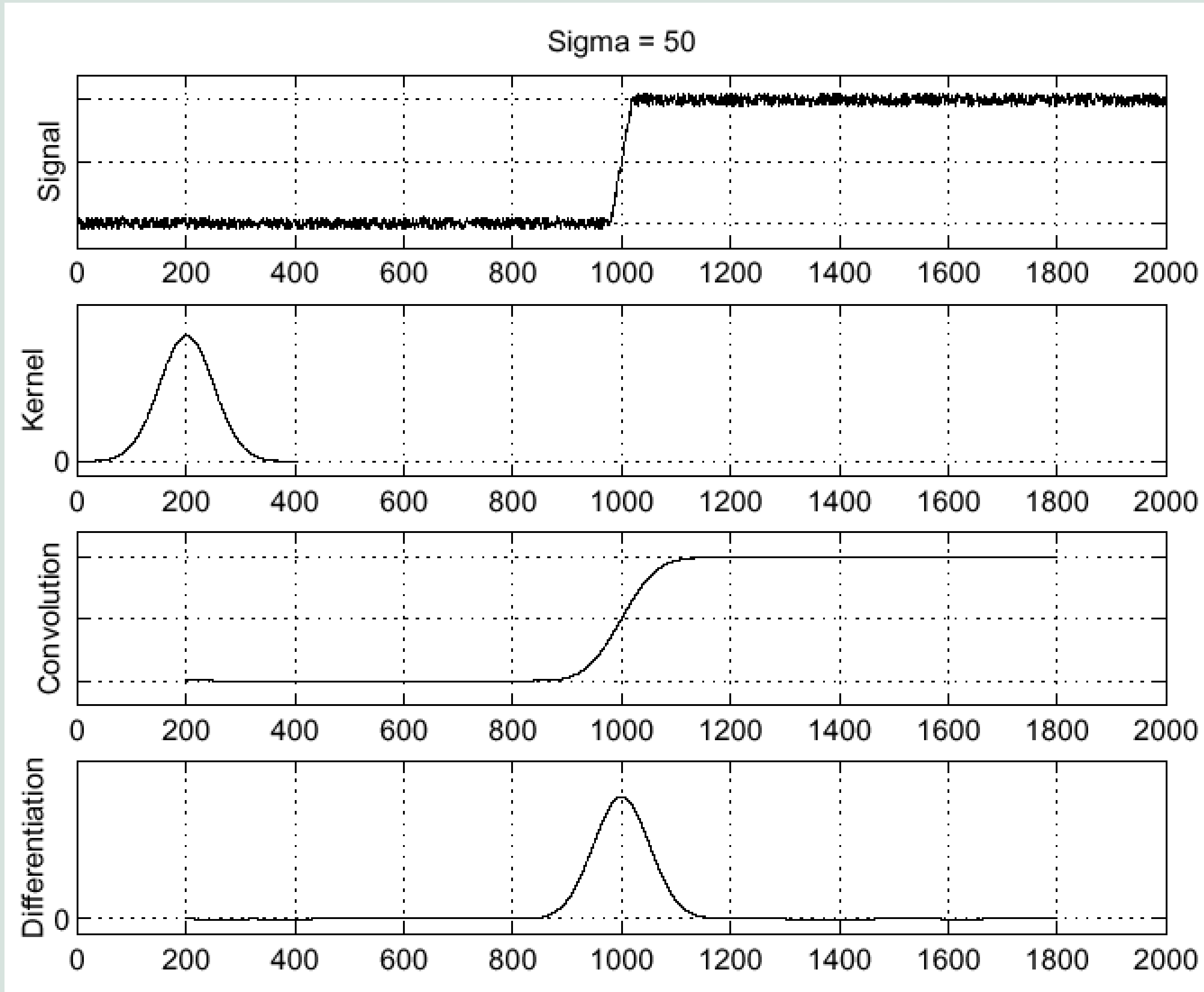
Solution: smooth first

$$f$$

$$h$$

$$h \star f$$

$$\frac{\partial}{\partial x}(h \star f)$$



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

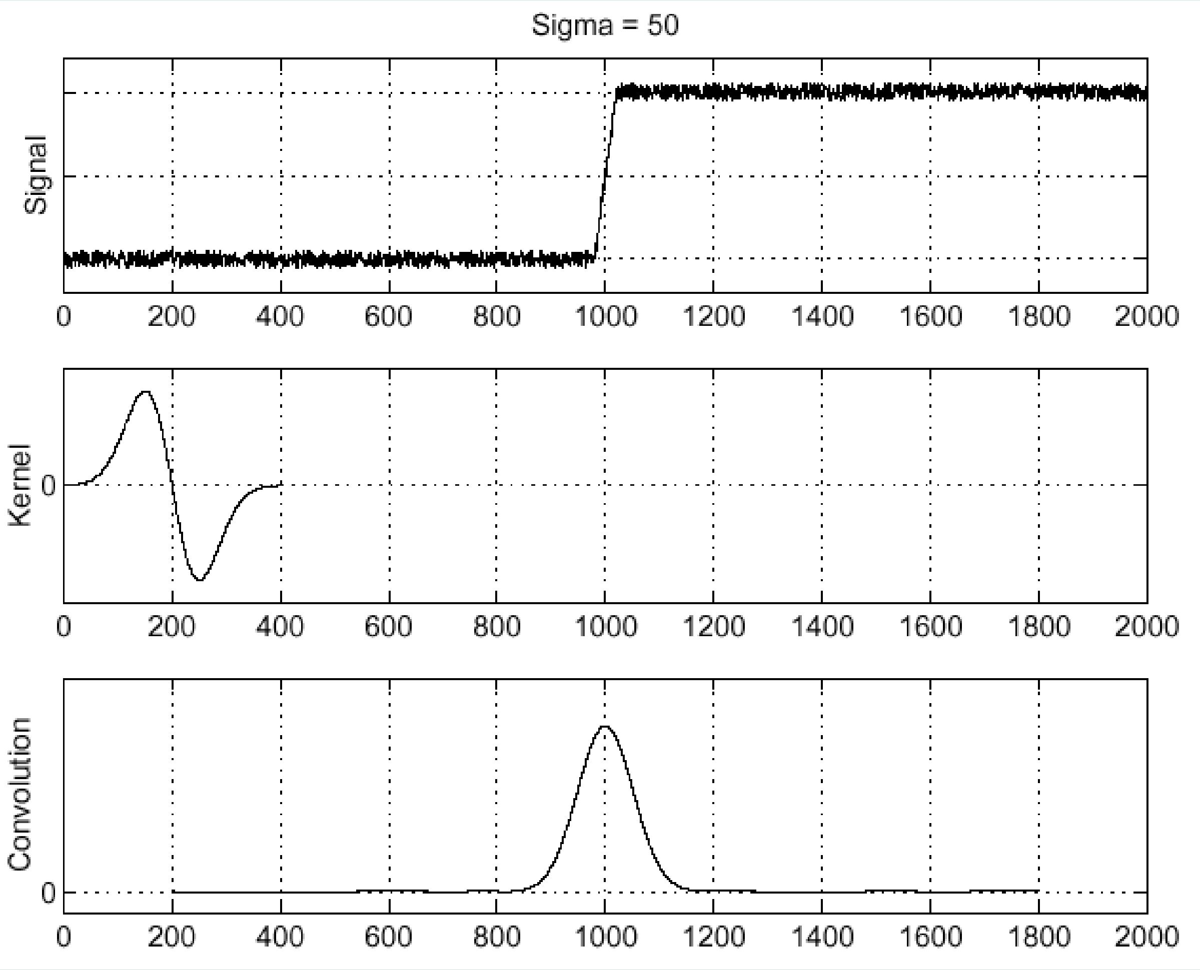
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

f

$\frac{\partial}{\partial x}h$

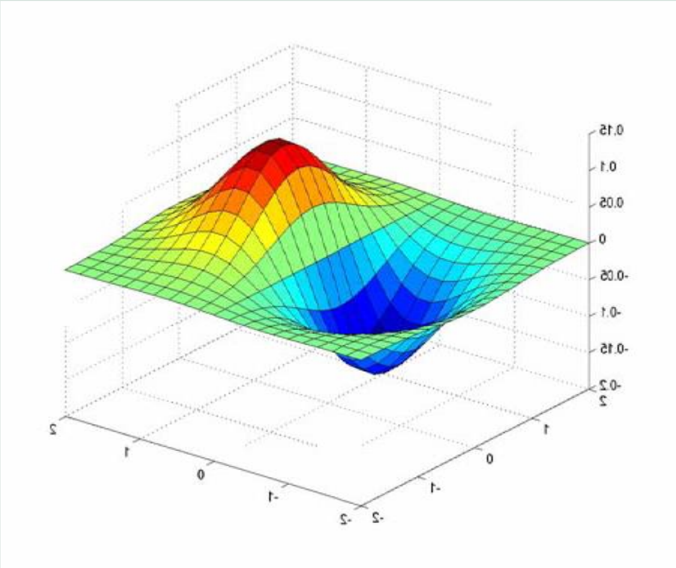
$(\frac{\partial}{\partial x}h) \star f$



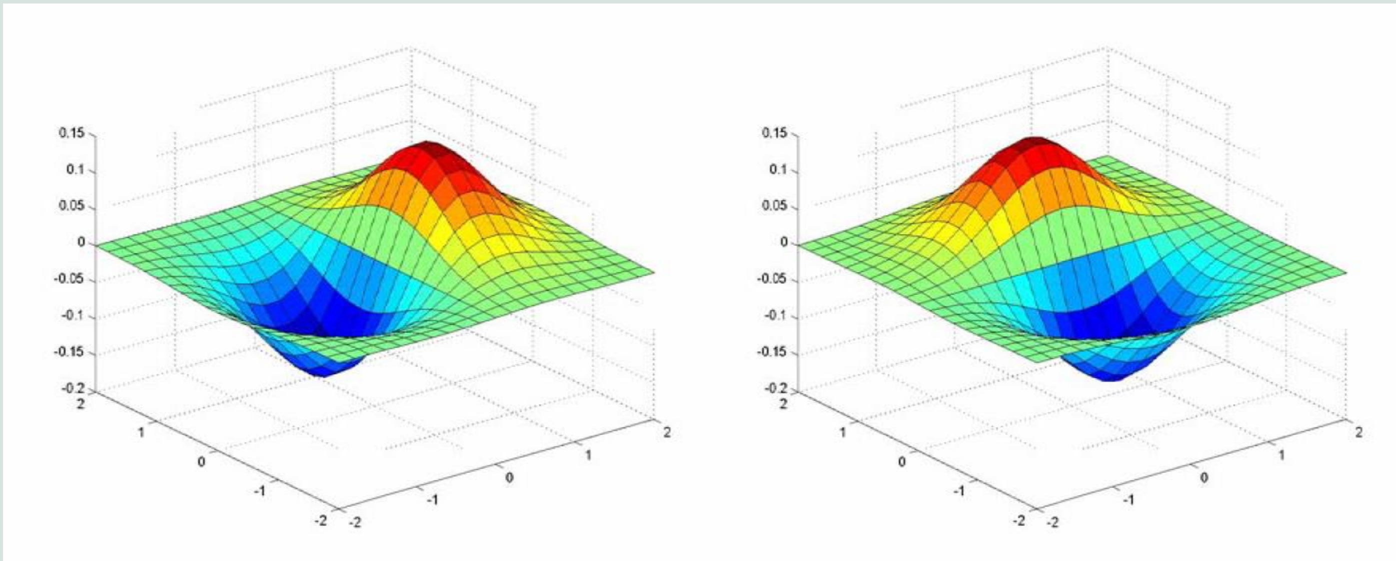
Derivative of Gaussian filter

$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

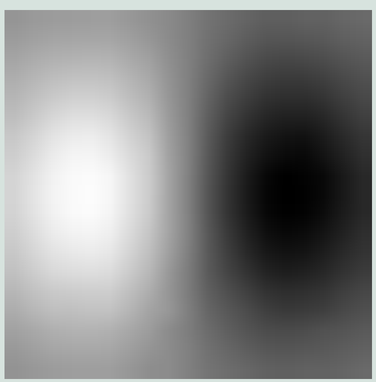
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$



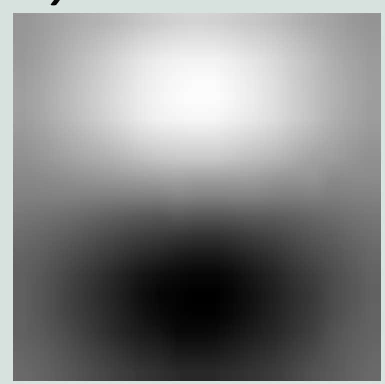
Derivative of Gaussian filters



x-direction



y-direction



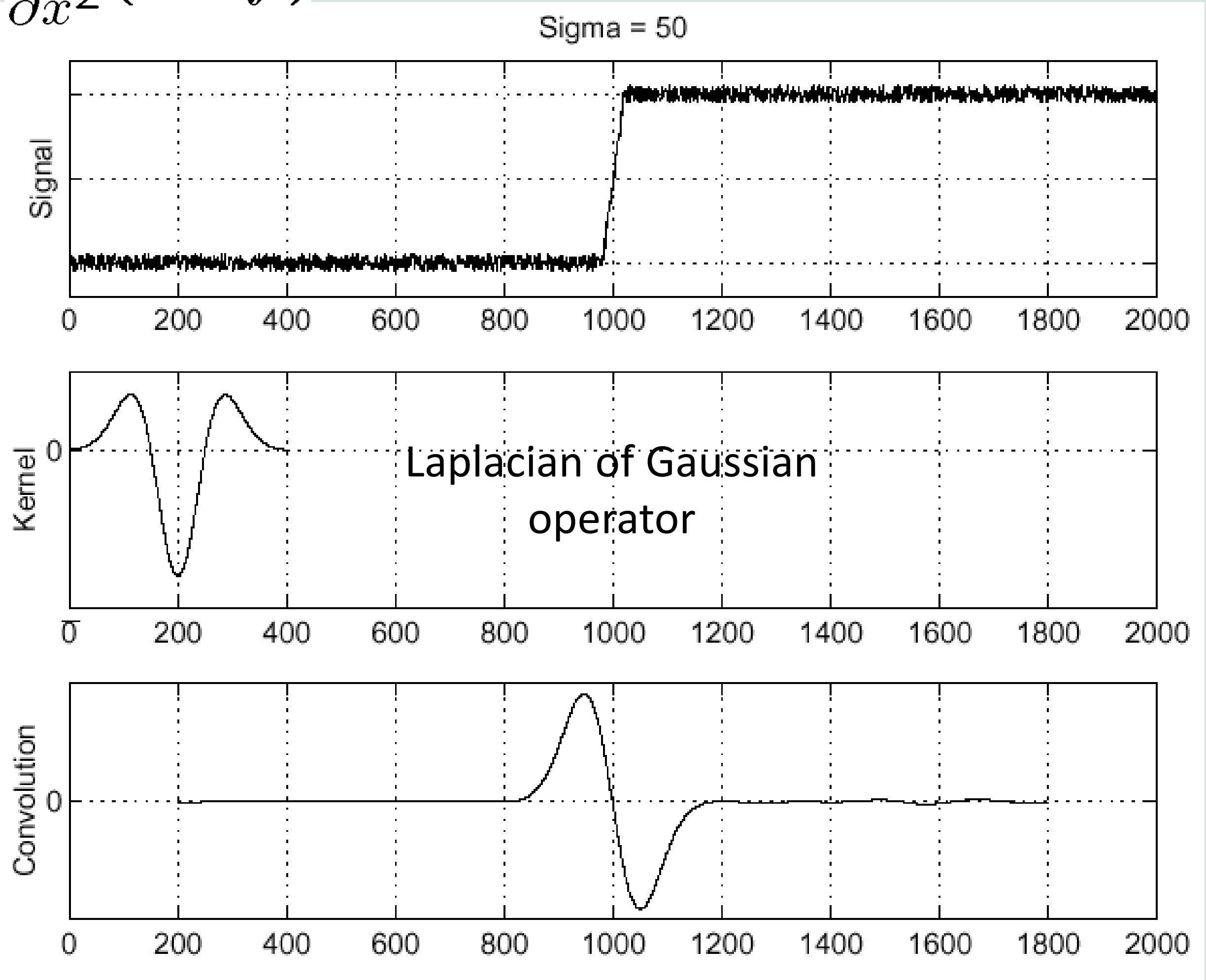
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

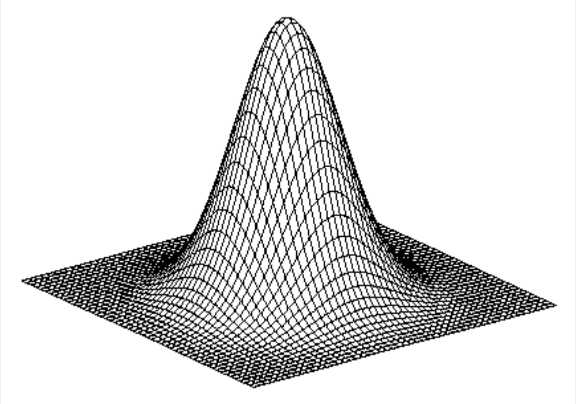
$(\frac{\partial^2}{\partial x^2}h) \star f$



Where is the edge?

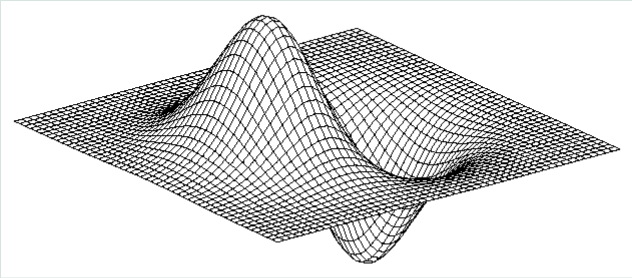
Zero-crossings of bottom graph

2D edge detection filters



Gaussian

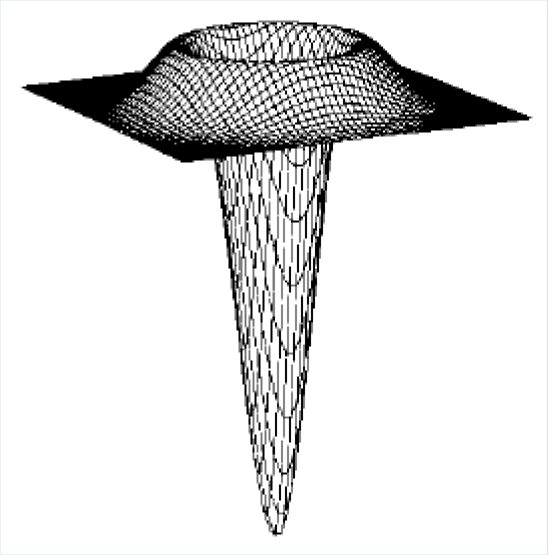
$$G^{\sigma}(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial G^{\sigma}(x,y)}{\partial x}$$

Laplacian of Gaussian



$$\nabla^2 G^{\sigma}(\mathrm{x},\mathrm{y})$$

- ∇^2 is the Laplacian operator:

$$\nabla^2 G^{\sigma}(\mathrm{x},\mathrm{y})=\frac{\partial^2 G}{\partial x^2}+\frac{\partial^2 G}{\partial y^2}$$

Mask properties

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- Derivatives
  - Opposite signs used to get high response in regions of high contrast
  - Sum to 0 → no response in constant regions
  - High absolute value at points of high contrast
- Filters act as templates
  - Highest response for regions that “look the most like the filter”
  - Dot product as correlation

Gradients -> edges

Primary edge detection steps:

1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization

Determine which local maxima from filter output are actually edges vs. noise

- Threshold, Thin