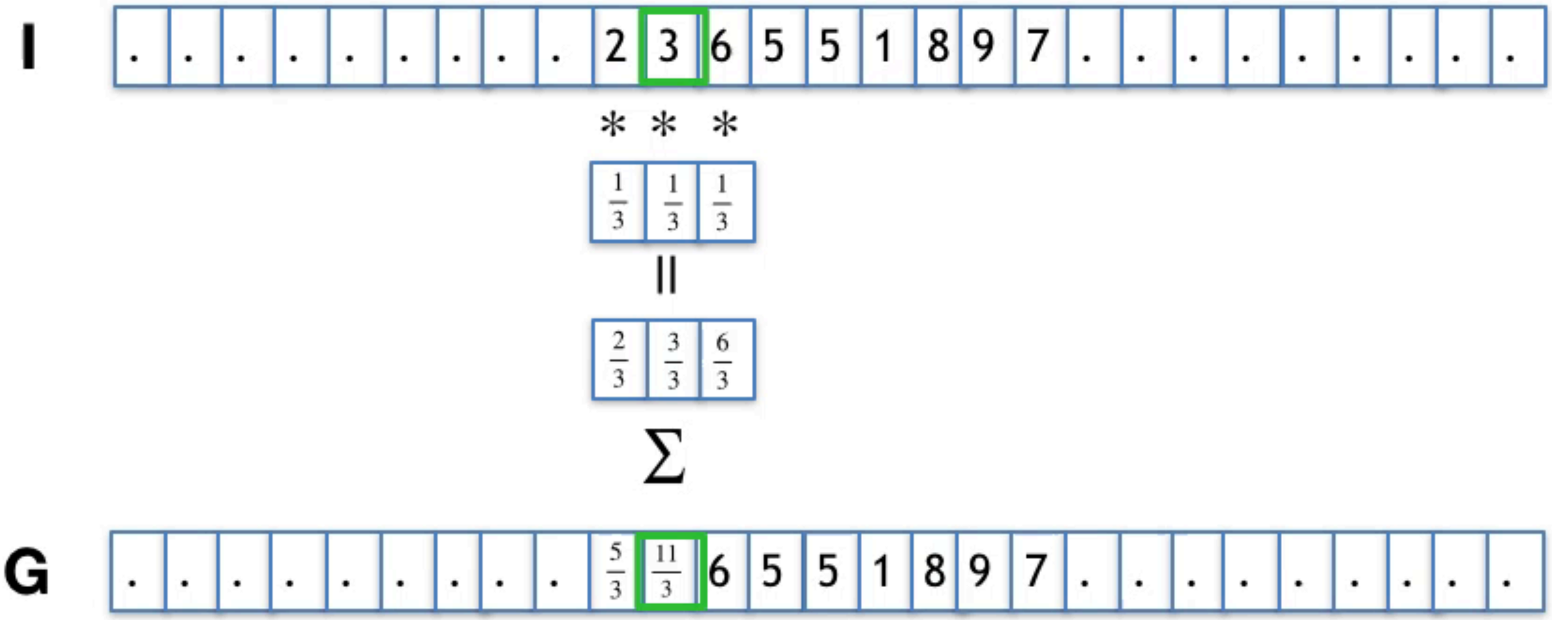
Correlation & Convolution

Mohammad Nayeem Teli

Correlation Example - 1D



Cross-Correlation - Mathematically

$1D$

$$G = F \circ I[i] = \sum_{u=-k}^{k} F[u]I[i+u]$$ F has $2k+1$ elements

Box filter $F[u] = \dfrac{1}{3}$ for $u = -1,0,1$ and 0 otherwise

Cross-correlation filtering - 2D

Let's write this down as an equation. Assume the averaging window is (2k+1)x(2k+1):

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

We can generalize this idea by allowing different weights for different neighboring pixels:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u,v] I[i+u, j+v]$$

This is called a **cross-correlation** operation and written:

$$G = F \circ I$$

F is called the "filter," "kernel," or "mask."

Convolution

Filter is flipped before correlating

$1D$ F has $2k+1$ elements

$$G = F * I[i] = \sum_{u=-k}^{k} F[u]I[i-u]$$

Box filter $F[u] = \frac{1}{3}$ for $u = -1,0,1$ and 0 otherwise

for example, convolution of 1D image with the filter [3,5,2]

is exactly the same as correlation with the filter [2,5,3]

Convolution filtering - 2D

For 2D the filter is flipped and rotated

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[u,v] I[i-u, j-v]$$

Correlation and convolution are identical for symmetrical filters

Convolution with the filter

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

is the same as Correlation with the filter

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

Correlation and Convolution Terminology

We used

G for correlation/convolution output

I for image - In literature sometimes F is used for image

F for filter - In literature sometimes H is used for filter

$$G = H \circ F$$

$$G = H * F$$

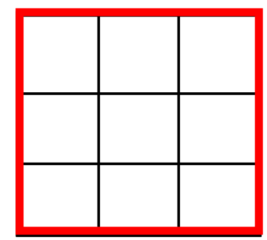
Filter Image

Mean kernel

What’s the kernel for a 3x3 mean filter?

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

$F[i, j]$



$H[u, v]$

Mean filtering (average over a neighborhood)

$F[x, y]$

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

$G[x, y]$

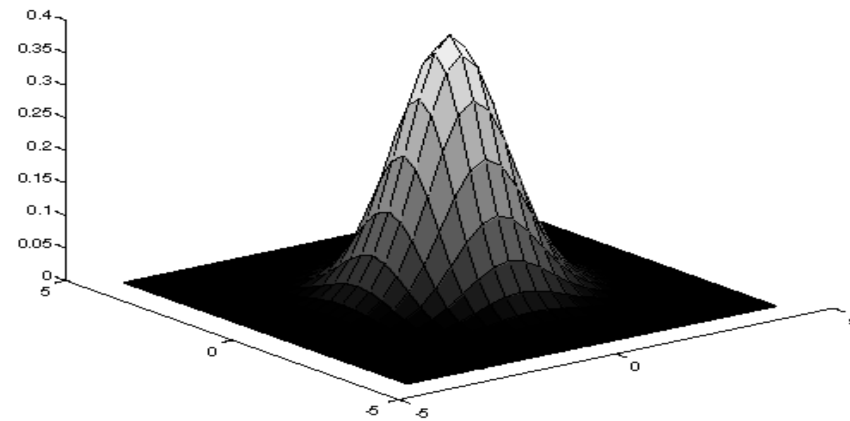
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Gaussian Averaging

Rotationally symmetric.

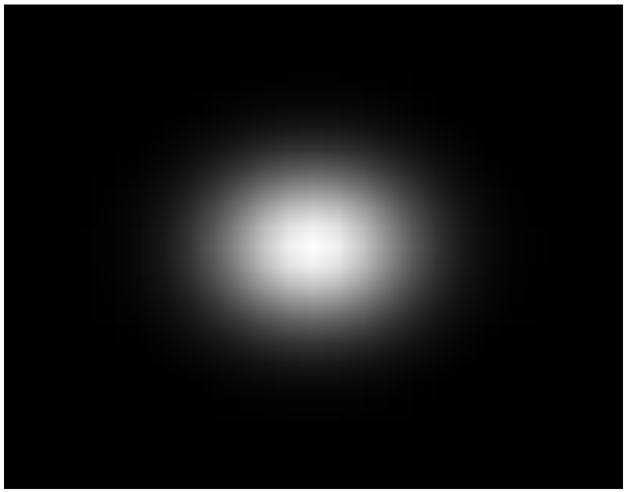
Weights nearby pixels more than distant ones.

- This makes sense as probabilistic inference.



A Gaussian gives a good model of a fuzzy blob

An Isotropic Gaussian



The picture shows a smoothing kernel proportional to

$$\exp\left(-\left(\frac{x^2+y^2}{2\sigma^2}\right)\right)$$

(which is a reasonable model of a circularly symmetric fuzzy blob)

Gaussian Filtering

A Gaussian kernel gives less weight to pixels further from the center of the window

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

$F[x, y]$

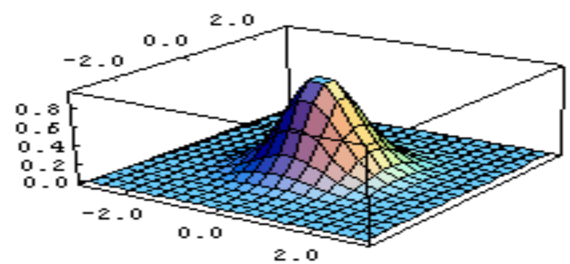
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

This kernel is an approximation of a

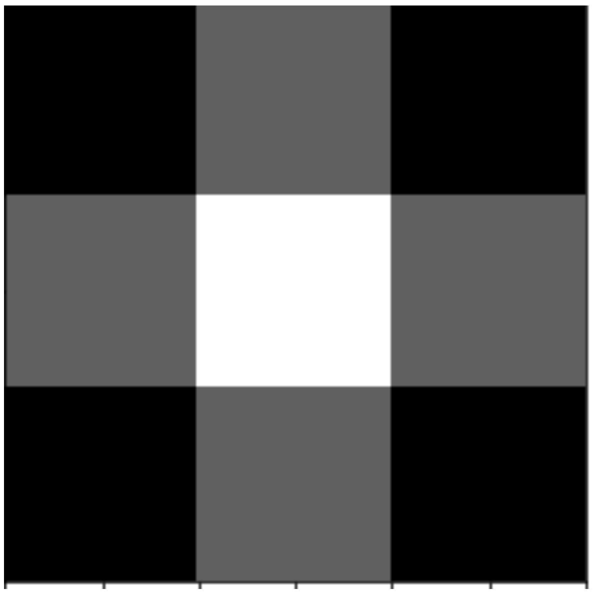
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

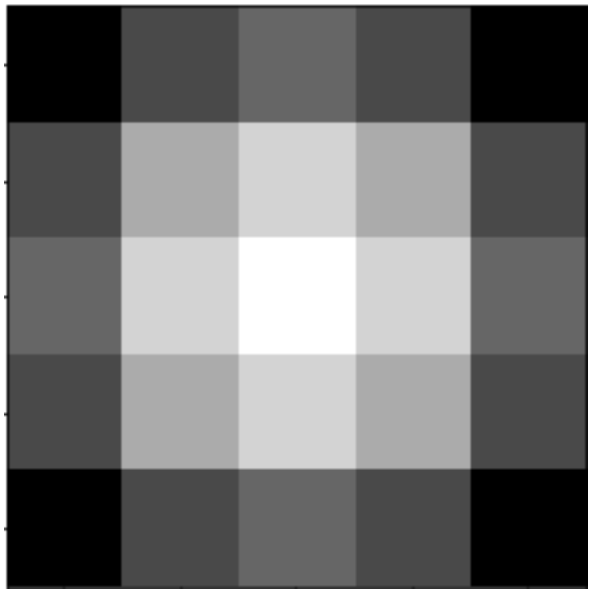


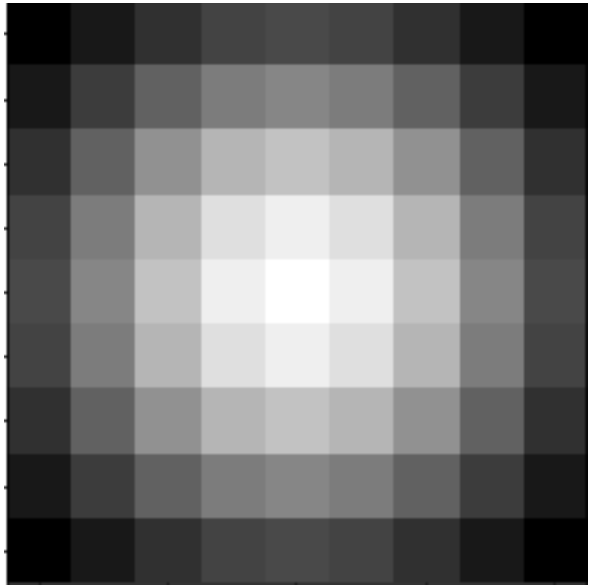
The size of the mask

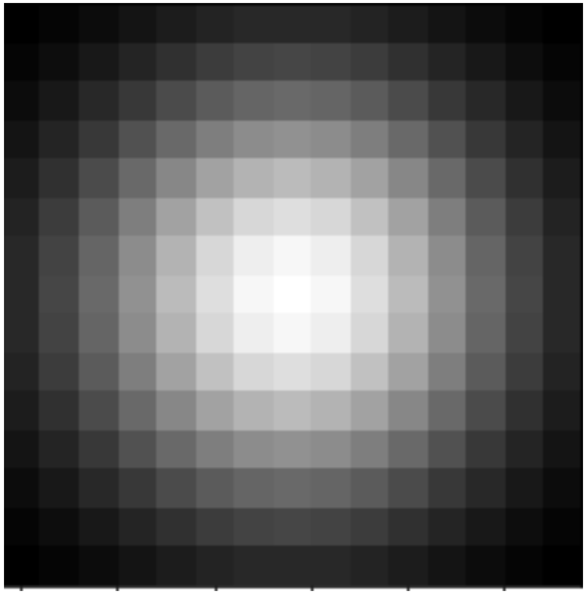
- Bigger mask:
  - more neighbors contribute.
  - smaller noise variance of the output.
  - bigger noise spread.
  - more blurring.
  - more expensive to compute.

Gaussians masks of different sizes



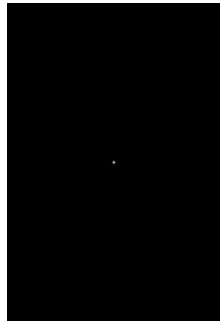






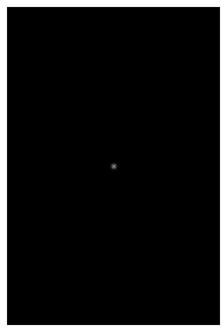
Convolution with masks of different sizes

$*$



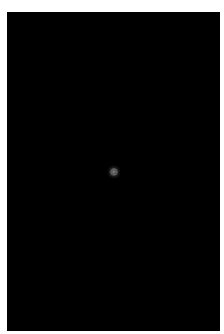
$\sigma = 1$

$*$



$\sigma = 2$

$*$



$\sigma = 3$

Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
- Convolution with self is another Gaussian
- Separable kernel
  - Factors into product of two 1D Gaussians

Separability of the Gaussian filter

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

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

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

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

Source: D. Lowe

Separability example

2D convolution
(center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors into
a product of 1D
filters:

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

Perform convolution
along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} \; & 11 & \; \\ \; & 18 & \; \\ \; & 18 & \; \end{bmatrix}$$

Followed by convolution
along the remaining column:

Source: K.
Grauman

Efficient Implementation

Both, the BOX filter and the Gaussian filter are separable:

- First convolve each row with a 1D filter
- Then convolve each column with a 1D filter.

Correlation & Convolution

- Basic operation to extract information from an image.

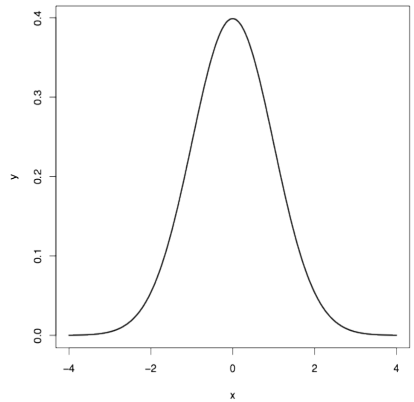
- These operations have two key features:

    - shift invariant
    - linear

- Applicable to 1-D and multi dimensional images.

Convolution

Image

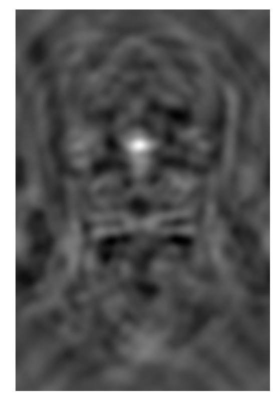


Gaussian

Modified Image

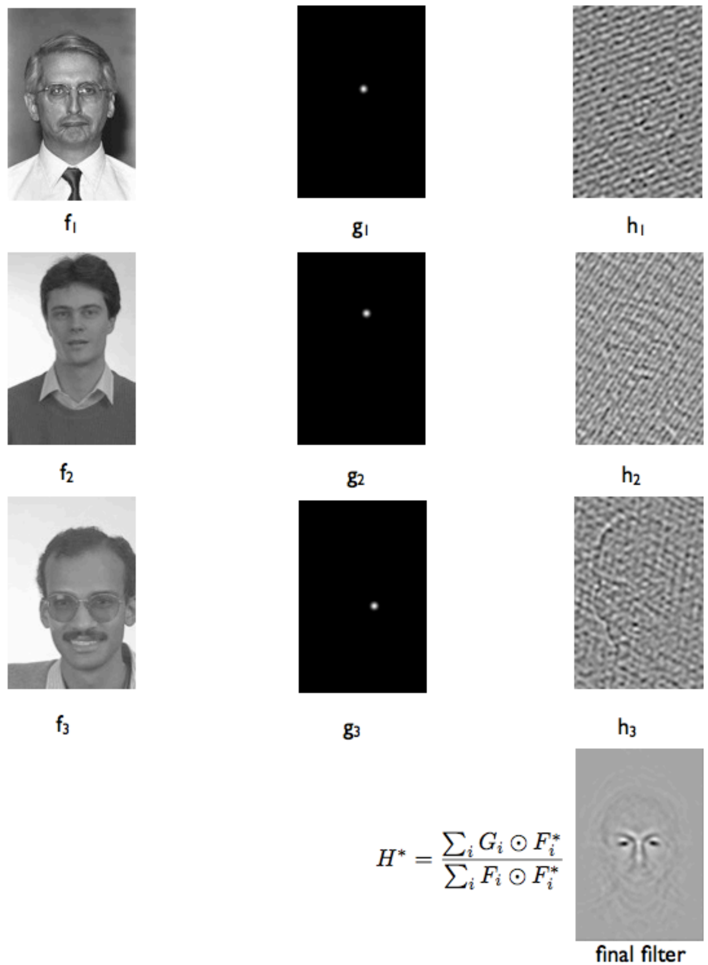
Image

Filter



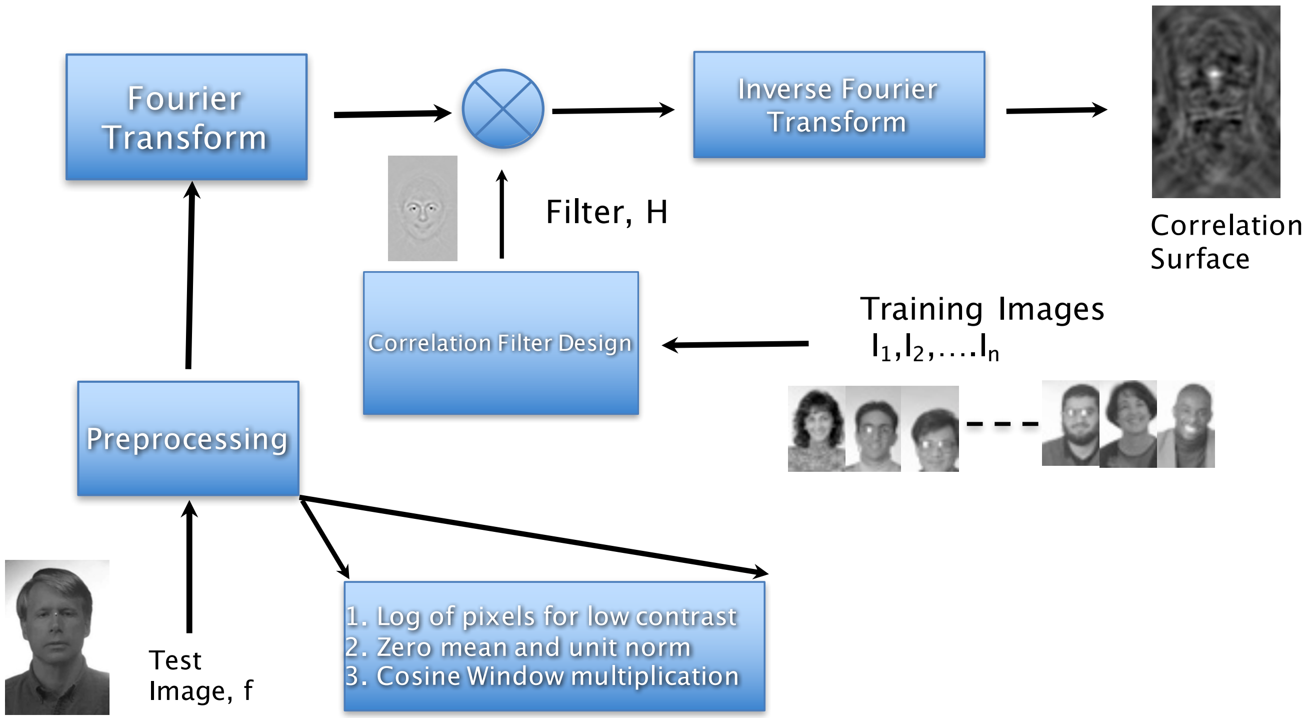
Correlation
Surface

MOSSE* Filter



Bolme et al. CVPR, 2010

Face Localization



Median filters

A **Median Filter** operates over a window by selecting the median intensity in the window.

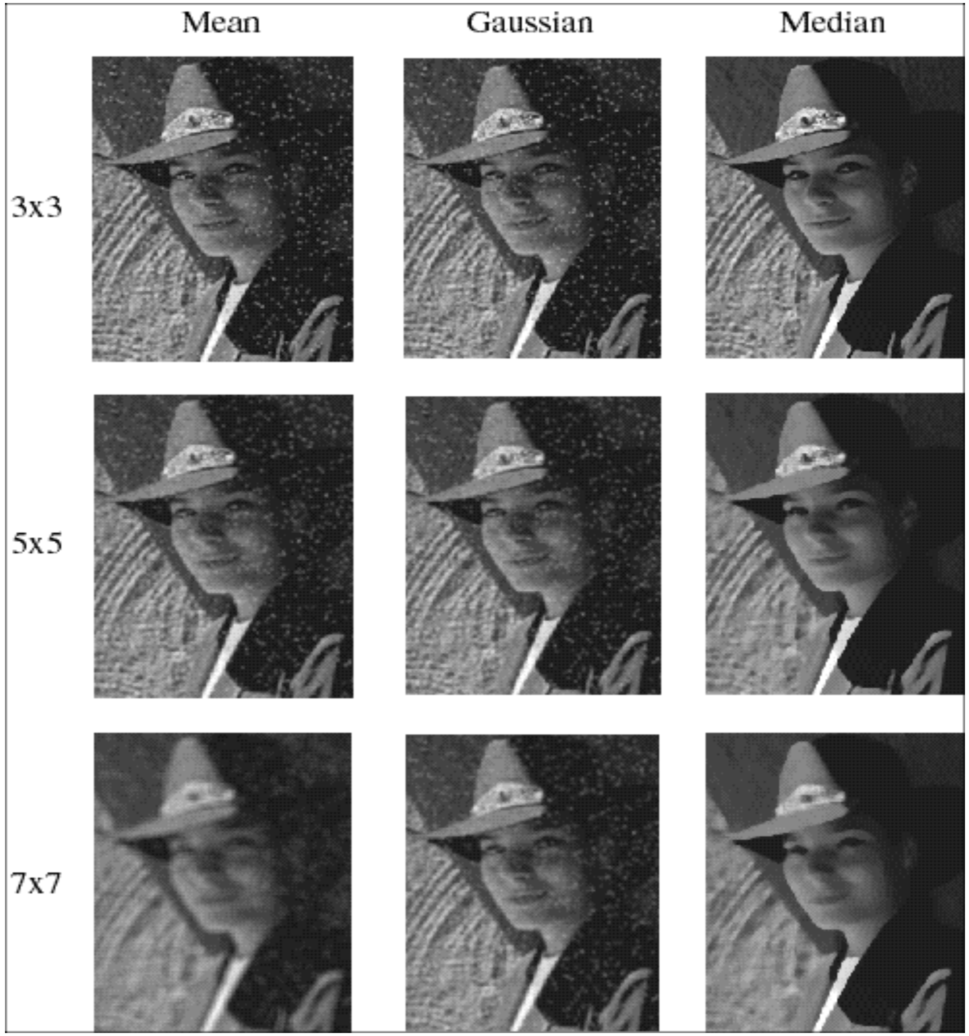
What advantage does a median filter have over a mean filter?

Is a median filter a kind of convolution?

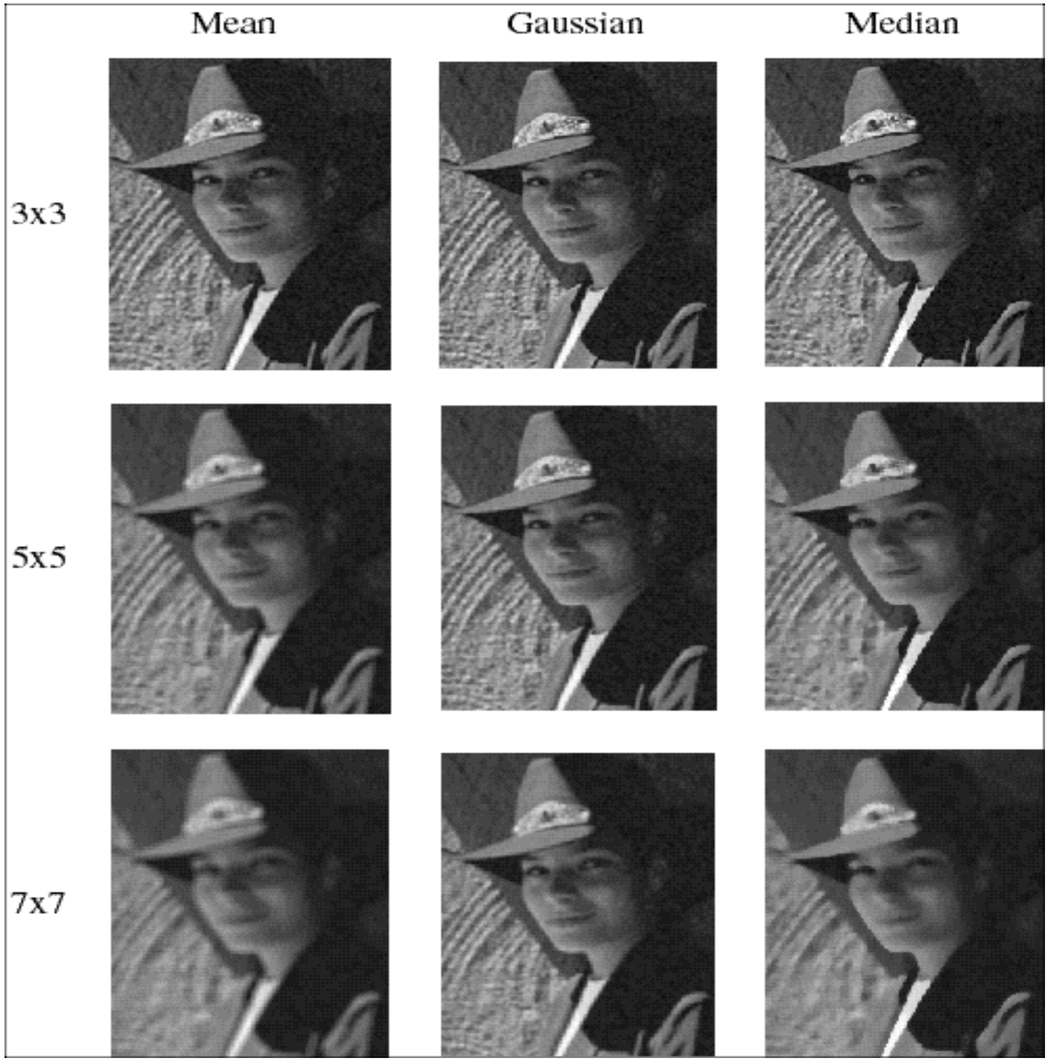
Median filter is non linear

Median filter

Comparison: salt and pepper noise

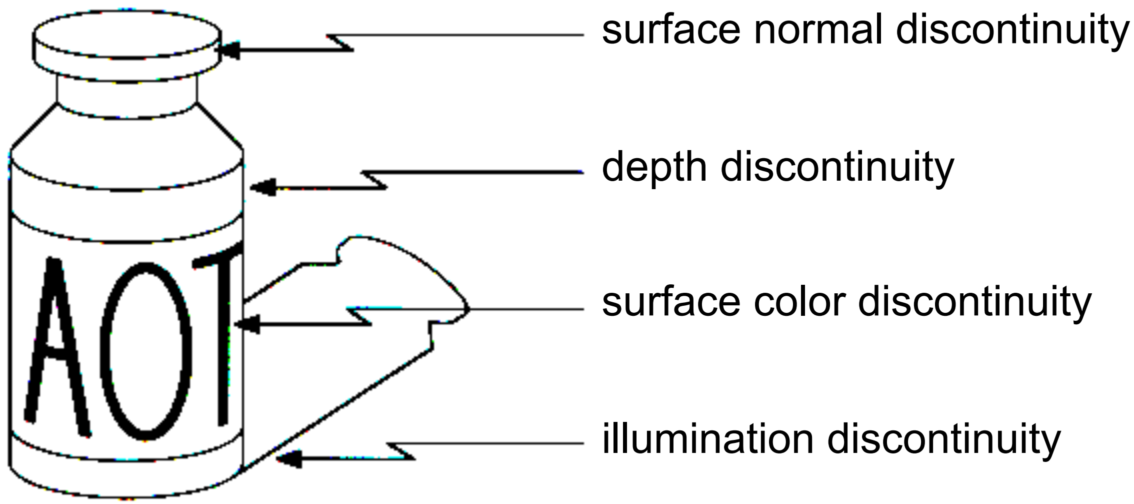


Comparison: Gaussian noise



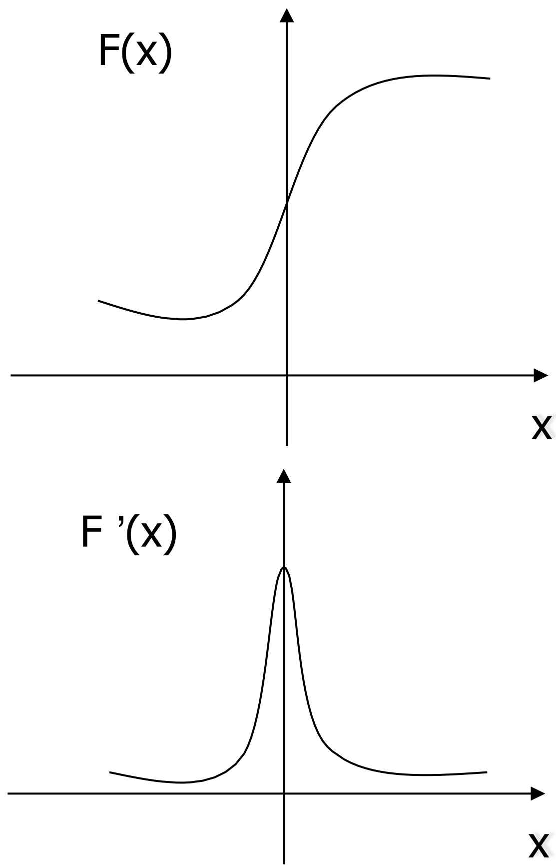
Edge Detection

Origin of Edges

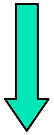


Edges are caused by a variety of factors

Edge detection (1D)



Edge= sharp variation



Large first derivative

Edge is Where Change Occurs

Change is measured by derivative in 1D

Biggest change, derivative has maximum magnitude

Or 2nd derivative is zero.

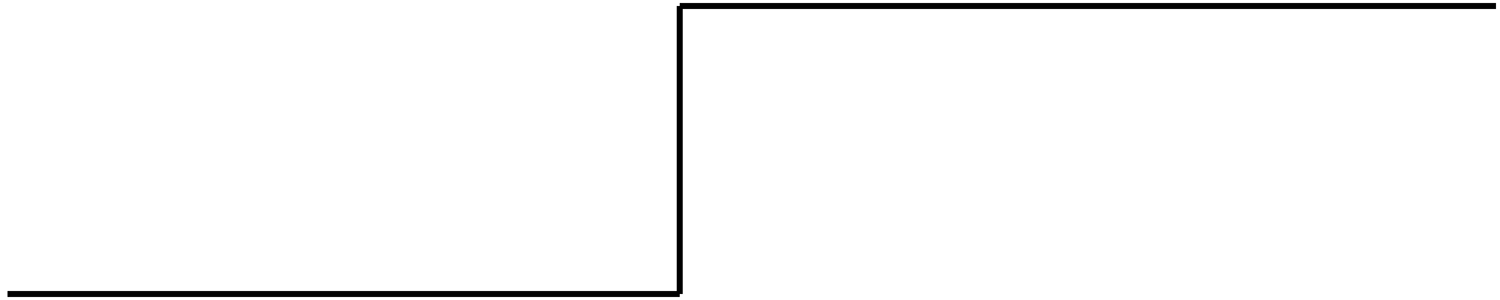
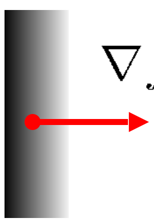


Image gradient

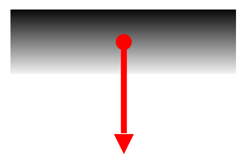
The gradient of an image:

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

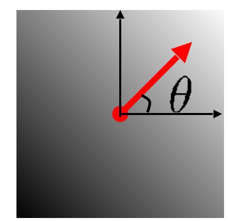
The gradient points in the direction of most rapid change in intensity



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



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



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

The gradient direction is given by:

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

- How does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

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

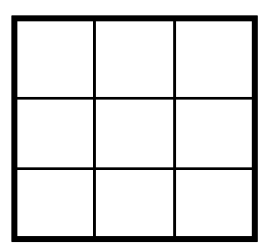
The discrete gradient

How can we differentiate a *digital* image f[x,y]?

- Option 1: reconstruct a continuous image, then take gradient
- Option 2: take discrete derivative (finite difference)

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

How would you implement this as a cross-correlation?



H

The Sobel operator

Better approximations of the derivatives exist

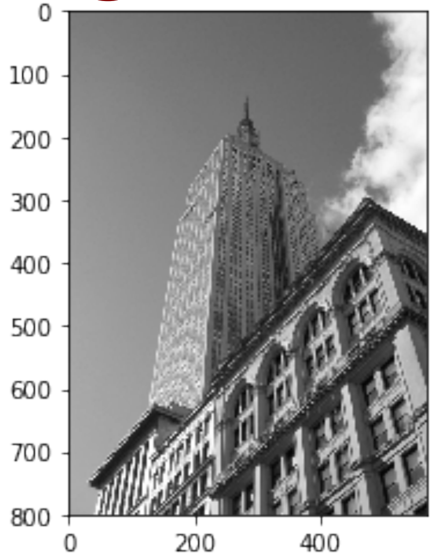
- The *Sobel* operators below are very commonly used

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad s_y$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value, however

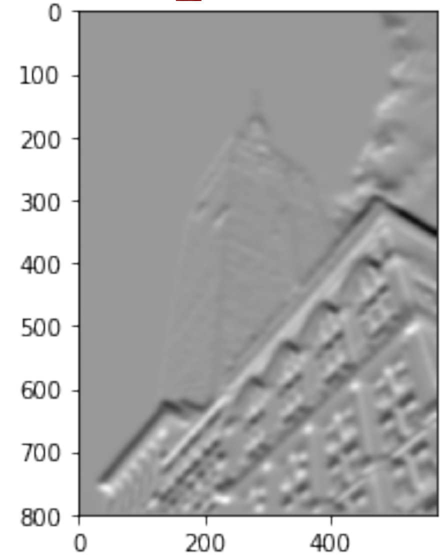
Edge Detection Using Sobel Operator



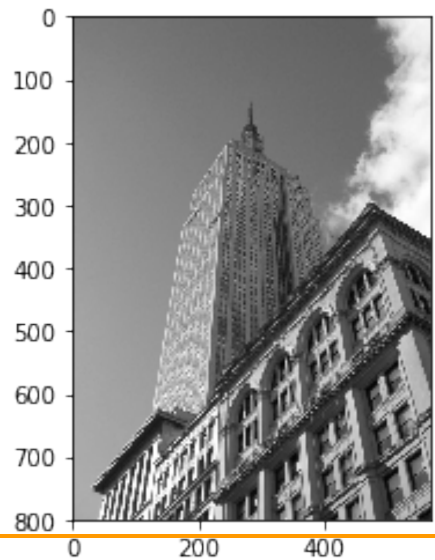
*

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

=



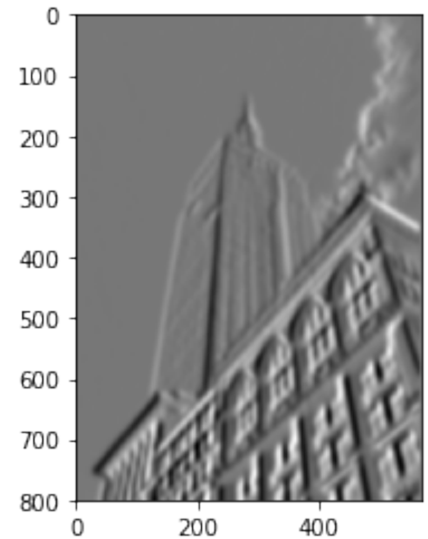
horizontal edge detector



*

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

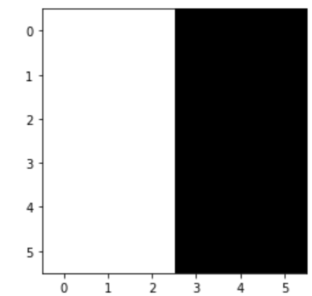
=



vertical edge detector

Vertical Edges

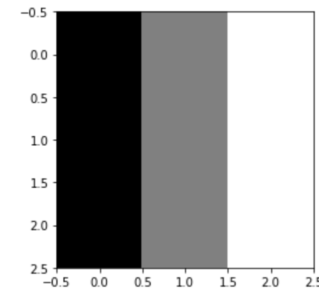
```
array([[ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.]])
```



F

*

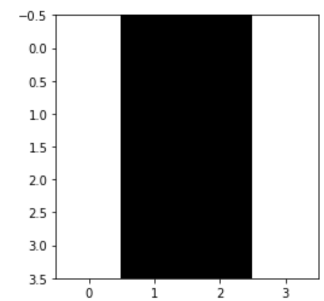
```
array([[-1.,  0.,  1.],
       [-1.,  0.,  1.],
       [-1.,  0.,  1.]])
```



H

=

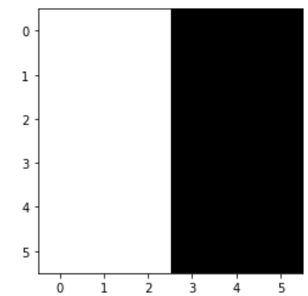
```
[[    0. -765. -765.     0.]
 [    0. -765. -765.     0.]
 [    0. -765. -765.     0.]
 [    0. -765. -765.     0.]]
```



G

Vertical Edges

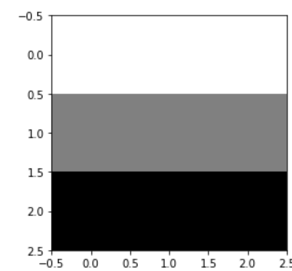
```
array([[ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.]])
```



F

*

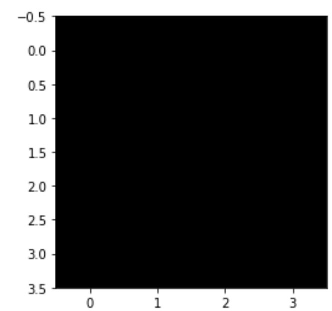
```
array([[ 1.,  1.,  1.],
       [ 0.,  0.,  0.],
       [-1., -1., -1.]])
```



H

=

```
[[ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]]
```



G

Gradient operators

Δ_1			Δ_2	
0	1		1	0
-1	0		0	-1

(a)

Δ_1				Δ_2		
-1	0	1		1	1	1
-1	0	1		0	0	0
-1	0	1		-1	-1	-1

(b)

Δ_1				Δ_2		
-1	0	1		1	2	1
-2	0	2		0	0	0
-1	0	1		-1	-2	-1

(c)

Δ_1					Δ_2			
-3	-1	1	3		3	3	3	3
-3	-1	1	3		1	1	1	1
-3	-1	1	3		-1	-1	-1	-1
-3	-1	1	3		-3	-3	-3	-3

(d)

(a): Roberts' cross operator (b): 3x3 Prewitt operator
(c): Sobel operator (d) 4x4 Prewitt operator