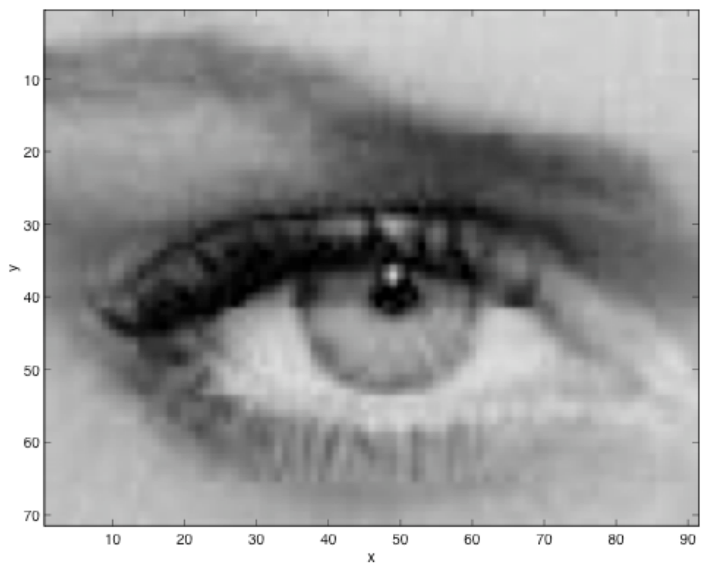
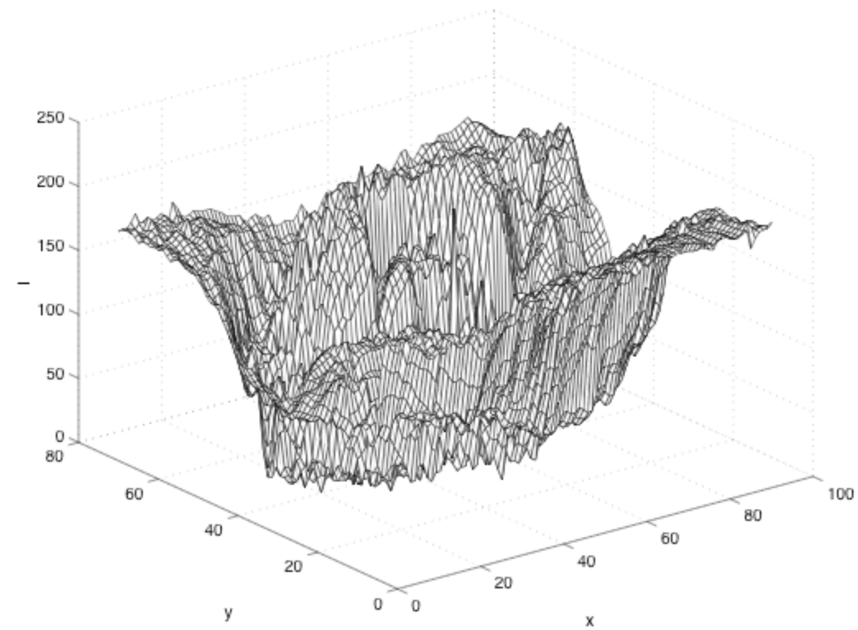
Image Primitives and Correspondence

Image



Brightness values



I(x,y)

Image Features

Local, meaningful, detectable parts of the image.

- Edge detection
- Line detection
- Corner detection

Motivation

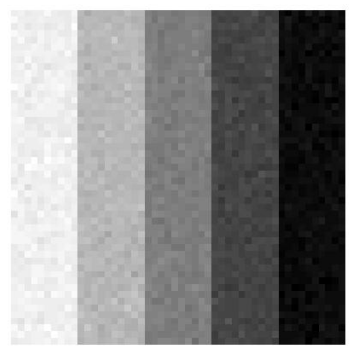
- Information content high
- Invariant to change of view point, illumination
- Reduces computational burden
- Uniqueness
- Can be tuned to a task at hand

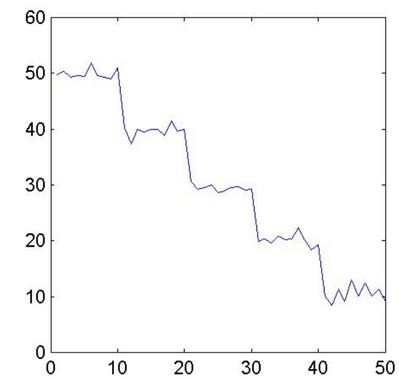
Filetring and Image Features

Given a noisy image

How do we reduce noise ?
How do we find useful features ?

Today:
- Filtering
- Point-wise operations
- Edge detection





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?

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

"box filter"

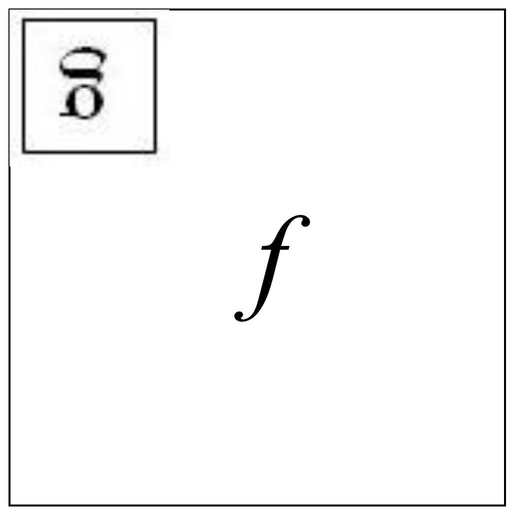
Source: D. Lowe

Defining convolution

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

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

Convention: kernel is "flipped"

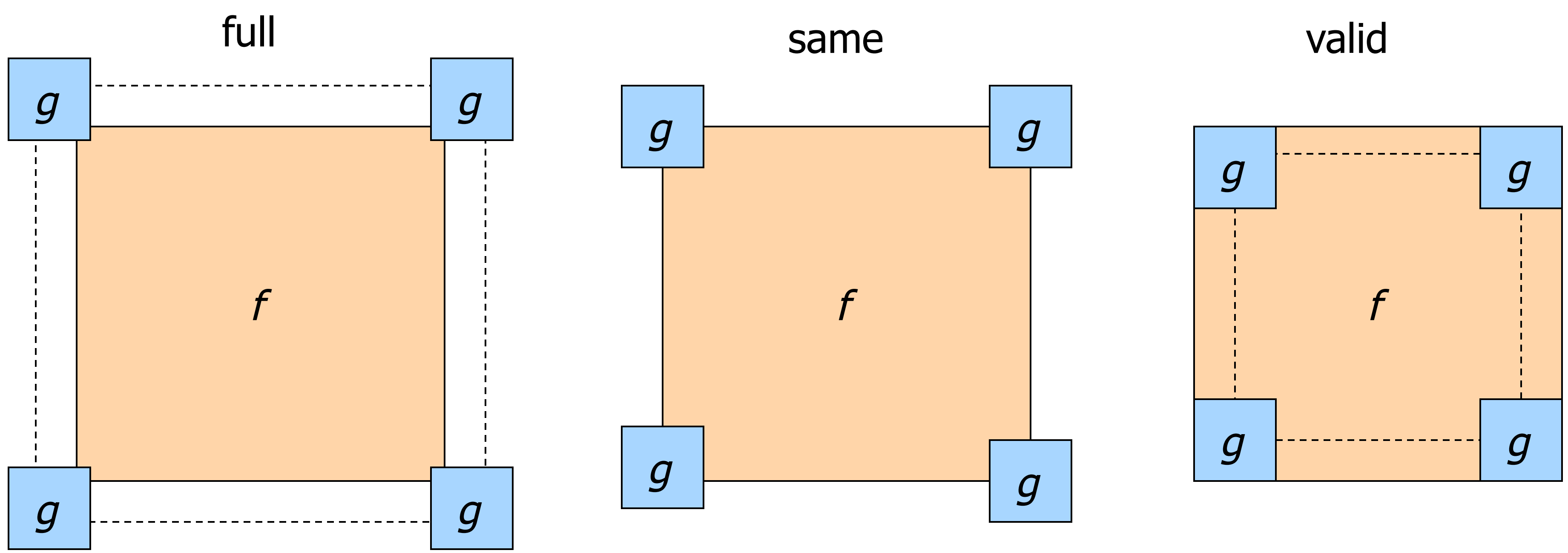


- MATLAB functions: conv2, filter2, imfilter

Source: F. Durand

Details

- What is the size of the output?
- MATLAB: filter2(g, f, *shape*)
  - *shape* = 'full': output size is sum of sizes of f and g
  - *shape* = 'same': output size is same as f
  - *shape* = 'valid': output size is difference of sizes of f and g



Averaging filter 1-D example

$$g[x] = \sum_{k=-\infty}^{\infty} f[k]h[x-k]$$

$$f[x] = [...0, 0, 2, -2, 2, 0, 0, ...] \quad h[x] = \frac{1}{3}[1, 1, 1]$$

$h[-1] = \frac{1}{3}, h[0] = \frac{1}{3}, h[1] = \frac{1}{3}$ and 0 everywhere else

$$f[-1] = -2, f[0] = 2, f[1] = -2$$

Box filter $\quad g[x] = \sum_{k=-1}^{1} f[k]h[x-k]$

Ex. cont.

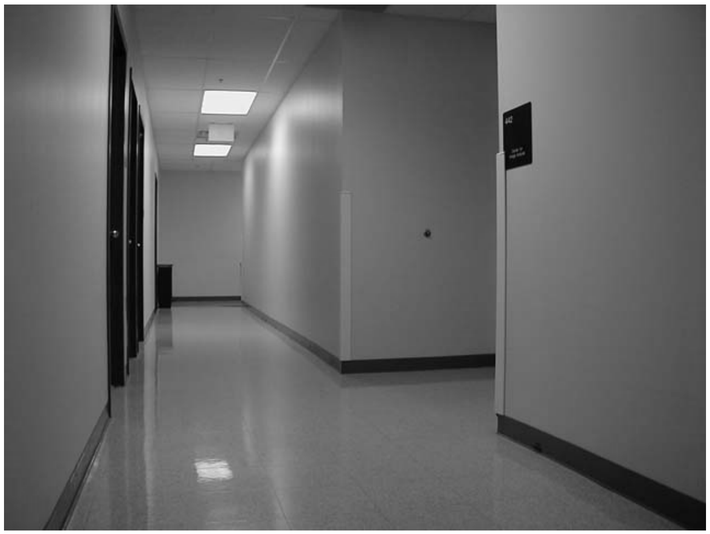
$$g[-1] = f[-1]h[-1-1] + f[0]h[-1] + f[1]h[0]$$

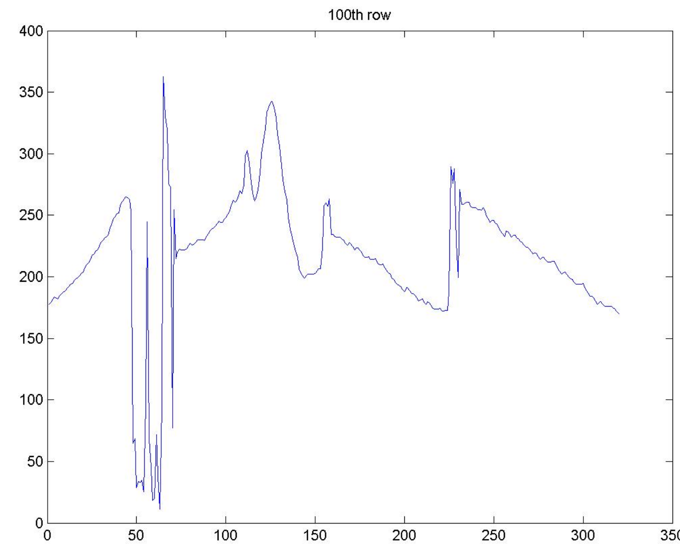
$$g[0] = f[-1]h[-1] + f[0]h[0] + f[1]h[1]$$

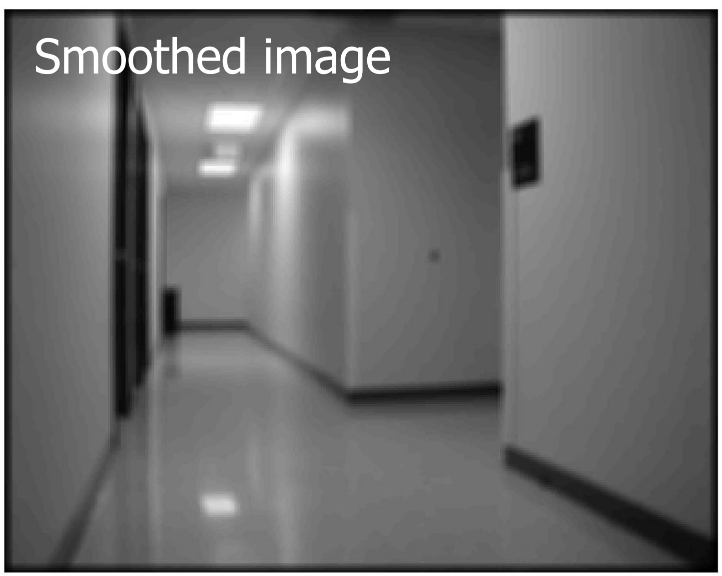
Averaging filter center pixel weighted more

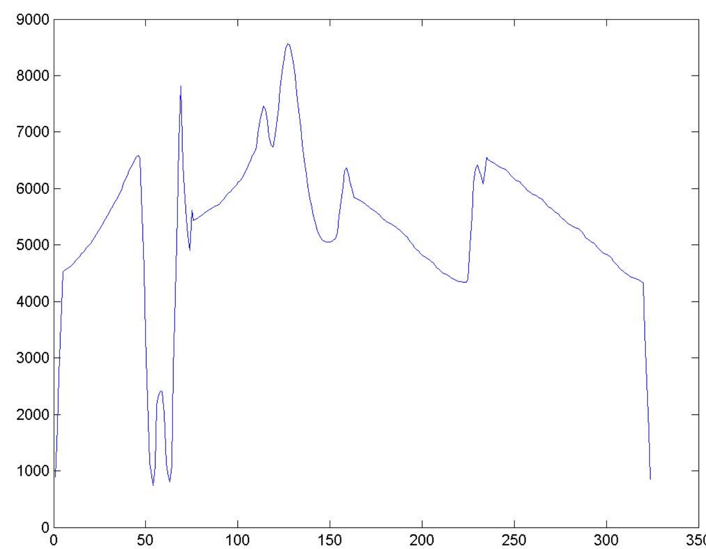
$$h[x] = [0.25, 0.5, 0.25]$$

Averaging filter









Convolution in 2D

$$g[x,y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} f[k,l]h[x-k,y-l]$$

f

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

h

1/9

1	1	1
1	1	1
1	1	1

g

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

1/9.(10x1 + 11x1 + 10x1 + 9x1 + 10x1 + 11x1 + 10x1 + 9x1 + 10x1) = 1/9.(90) = 10

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

O

X	X	X	X	X	X
X	10	7	4	1	X
X					X
X					X
X					X
X	X	X	X	X	X

F

1/9

1	1	1
1	1	1
1	1	1

1/9.(10x1 + 0x1 + 0x1 + 11x1 + 1x1 + 0x1 + 10x1 + 0x1 + 2x1) =
1/9.(34) = 3.7778

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	**99**	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10	7	4	1	X
X					X
X					X
X				**20**	X
X	X	X	X	X	X

1/9.(10x1 + 9x1 + 11x1 + 9x1 + 99x1 + 11x1 + 11x1 + 10x1 + 10x1) =

1/9.(180) = 20

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	$\boxed{10}$	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

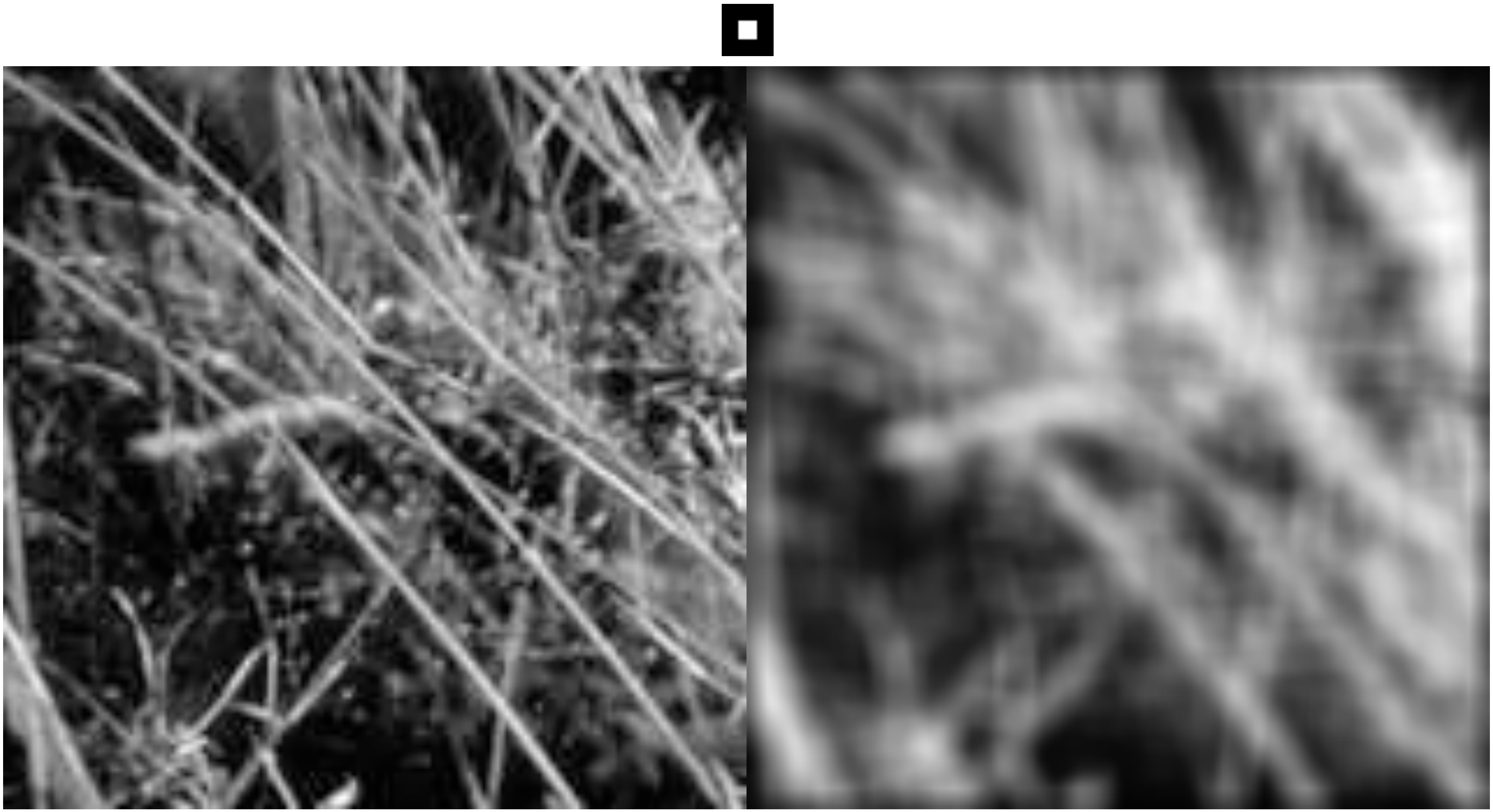
X	X	X	X	X	X
X	10	7	4	1	X
X					X
X			$\boxed{18}$		X
X				20	X
X	X	X	X	X	X

1/9.(10x1 + 0x1 + 2x1 + 9x1 + 10x1 + 9x1 + 11x1 + 9x1 + 99x1) =
1/9.(159) = 17.6667

How big should the mask be?

- The bigger the mask,
  - more neighbors contribute.
  - smaller noise variance of the output.
  - bigger noise spread.
  - more blurring.
  - more expensive to compute.
  - In Matlab function **conv, conv2**

Example: Smoothing by Averaging



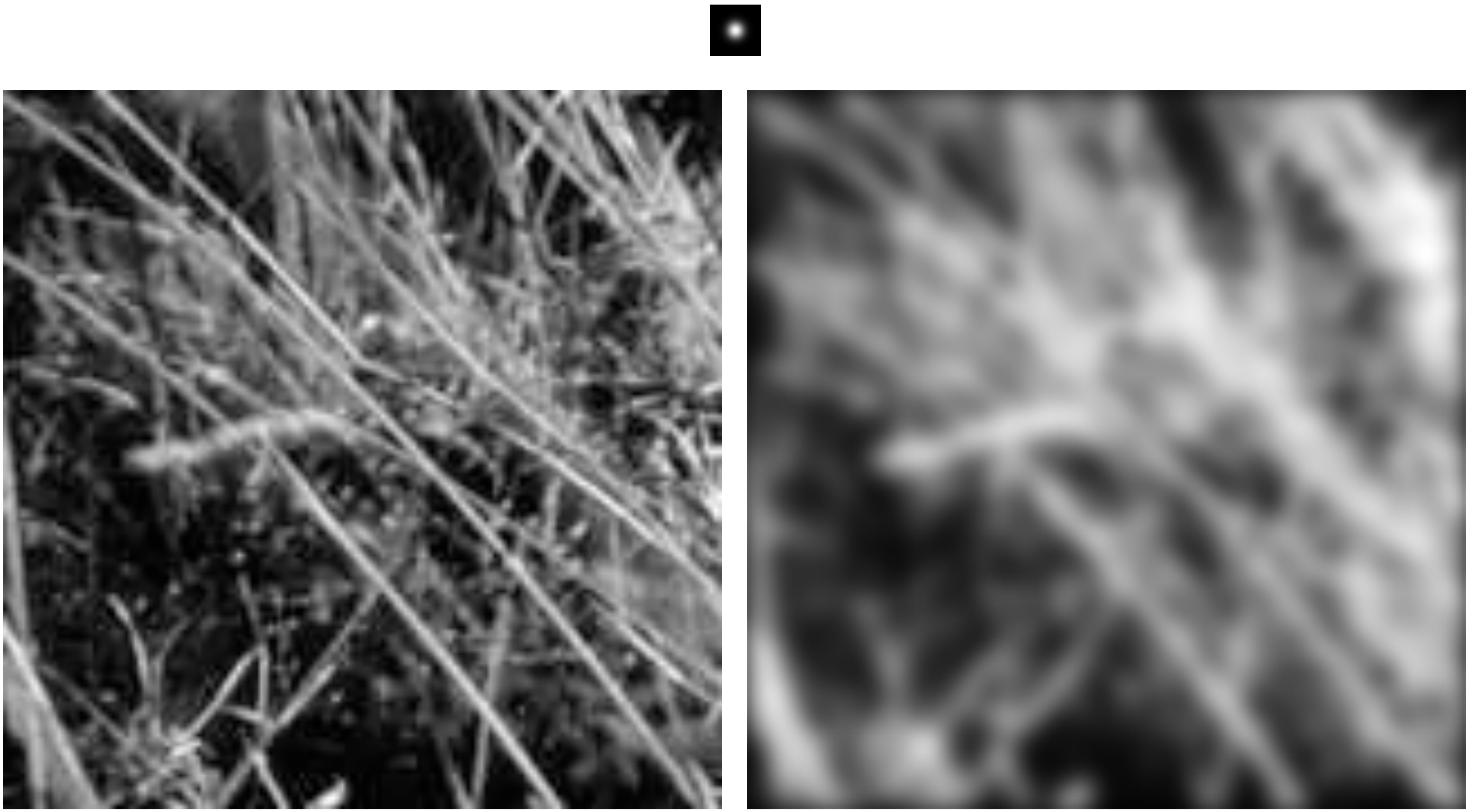
Gaussian Filter

- A particular case of averaging
  - The coefficients are samples of a 1D Gaussian.
  - Gives more weight at the central pixel and less weights to the neighbors.
  - The further away the neighbors, the smaller the weight.

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}},$$

Sample from the continuous Gaussian

Smoothing with a Gaussian



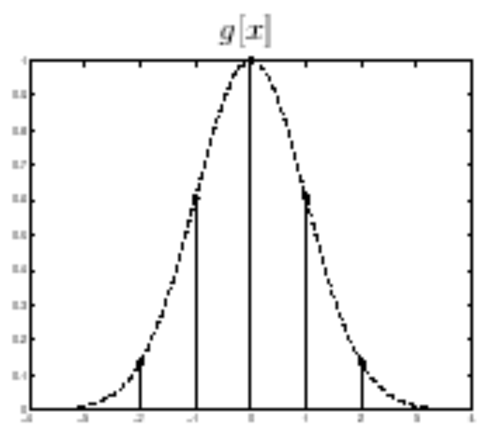
How big should the mask be?

- The std. dev of the Gaussian σ determines the amount of smoothing.
- The samples should adequately represent a Gaussian
- For a 98.76% of the area, we need

$$m = 5\sigma$$

$$5.(1/\sigma) \leq 2\pi \Rightarrow \sigma \geq 0.796,\ m \geq 5$$

5-tap filter



g[x] = [0.136, 0.6065, 1.00, 0.606, 0.136]

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

Source: K. Grauman

Separability of the Gaussian filter

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

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

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

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

Source: D. Lowe

Separability example

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

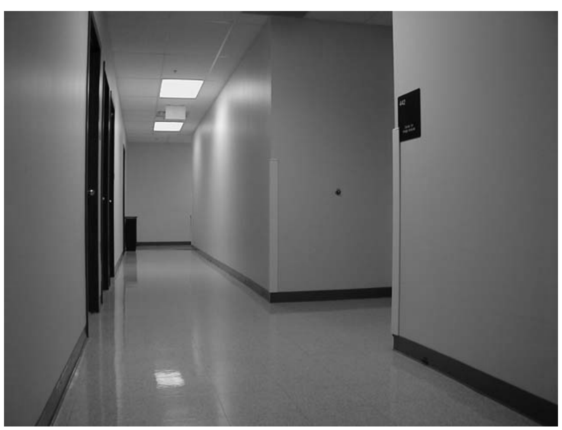
Image Smoothing

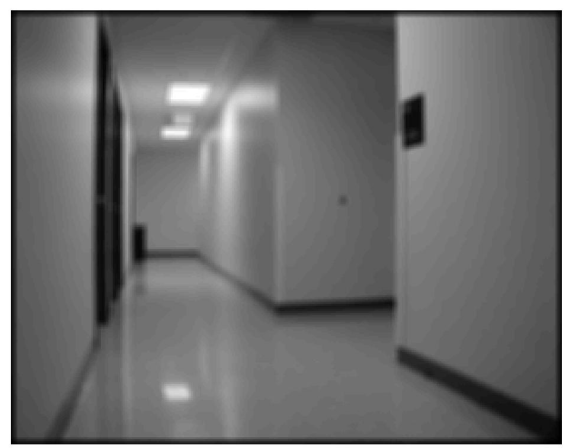
- Convolution with a 2D Gaussian filter

$$\tilde{I}(x,y) = I(x,y) * g(x,y) = I(x,y) * g(x) * g(y)$$

- Gaussian filter is separable, convolution can be accomplished as two 1-D convolutions

$$\tilde{I}[x,y] = I[x,y] * g[x,y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} I[k,l]g[x-k]g[y-l]$$



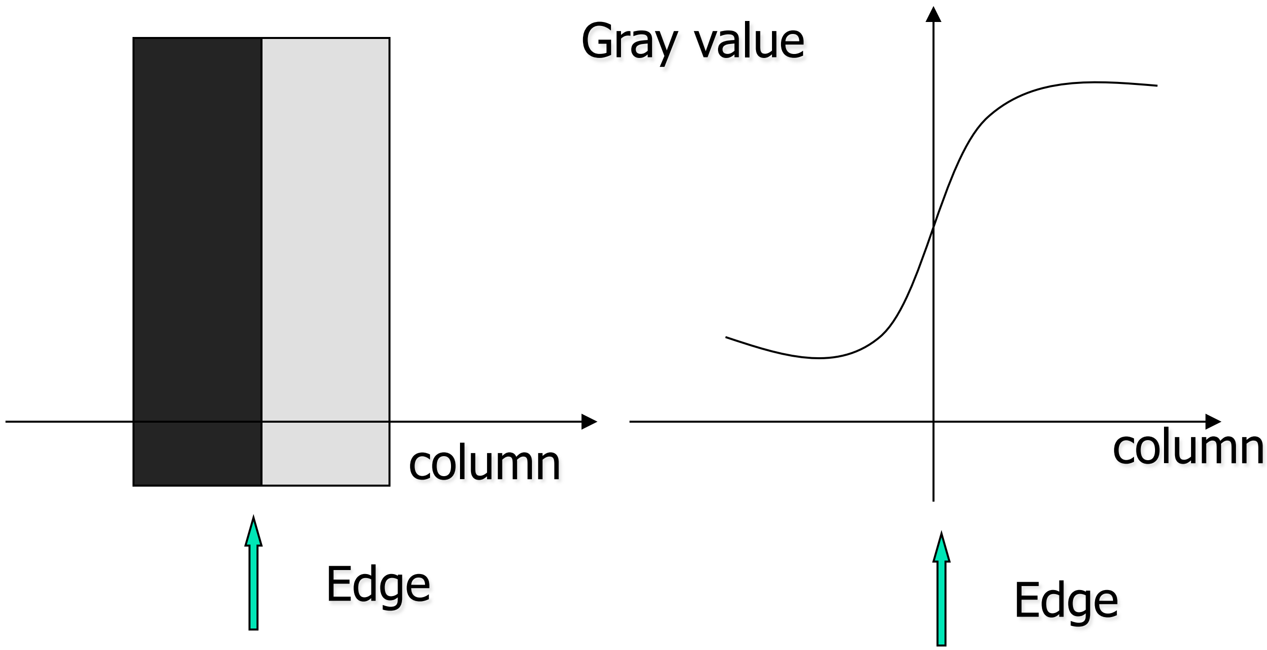


How big should the mask be?

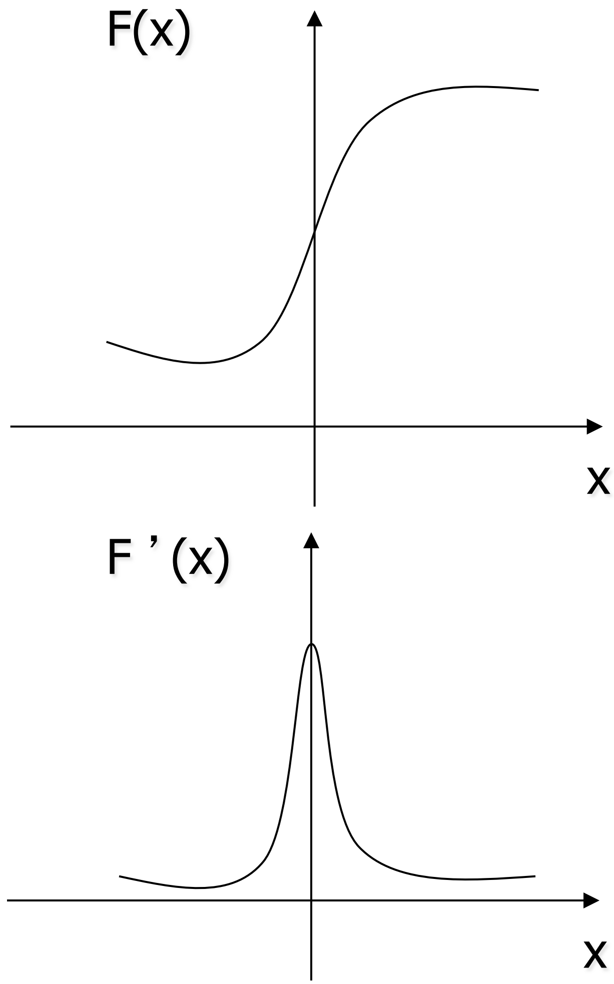
- The bigger the mask,
  - more neighbors contribute.
  - smaller noise variance of the output.
  - bigger noise spread.
  - more blurring.
  - more expensive to compute.

Edges

- They happen at places where the image values exhibit sharp variation



Edge detection (1D)

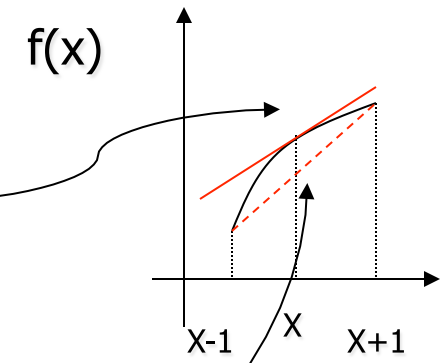


Edge= sharp variation

Large first derivative

Digital Approximation of 1st derivatives

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$



$$\frac{df(x)}{dx} \cong \frac{f(x+1) - f(x-1)}{2}$$

Convolve with:

-1	0	1

Edge Detection (2D)

Vertical Edges:

Convolve with:

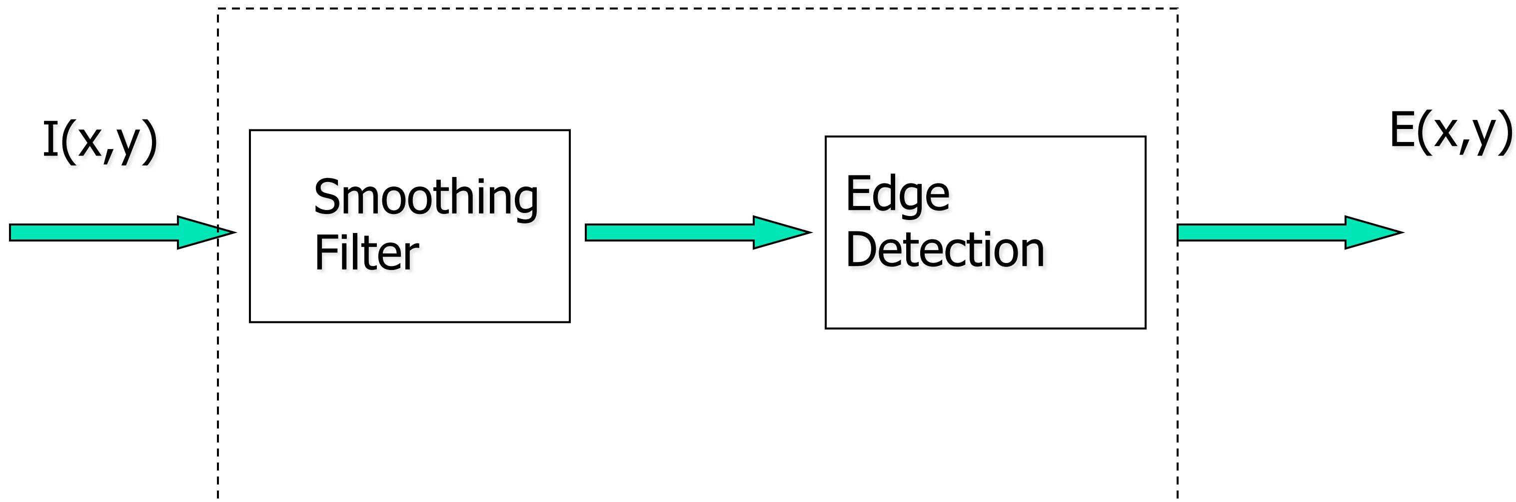
-1	0	1

Horizontal Edges:

Convolve with:

-1
0
1

Noise cleaning and Edge Detection



- we need to also deal with noise
- Combine Linear Filters

- Instead of smoothing, followed by derivative computation
- Convolve with derivative of the smoothing filter

Noise Smoothing & Edge Detection

Convolve with:

-1	0	1
-1	0	1
-1	0	1

Noise Smoothing

Vertical Edge Detection

This mask is called the (vertical) Prewitt Edge Detector

Outer product of box filter $[1\ 1\ 1]^T$ and $[-1\ 0\ 1]$

Noise Smoothing & Edge Detection

Convolve with:

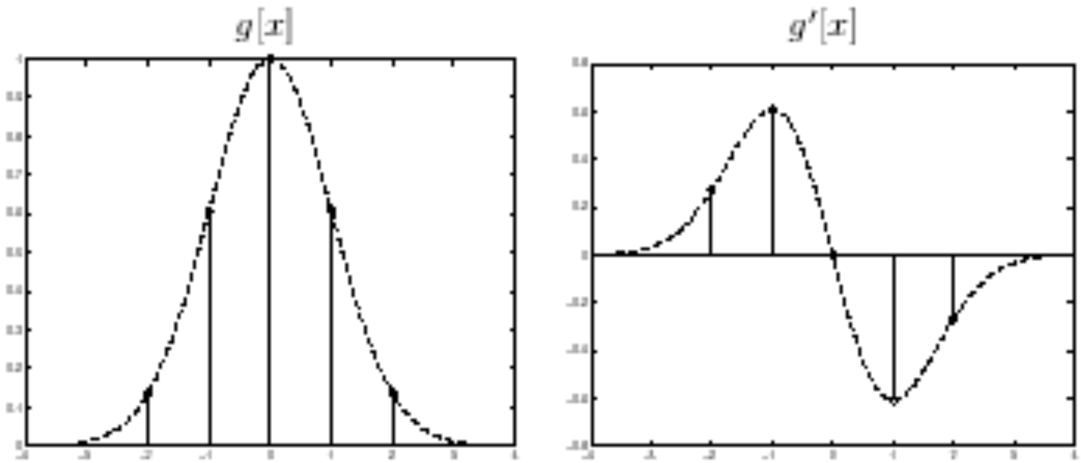
-1	-1	-1
0	0	0
1	1	1

Noise Smoothing

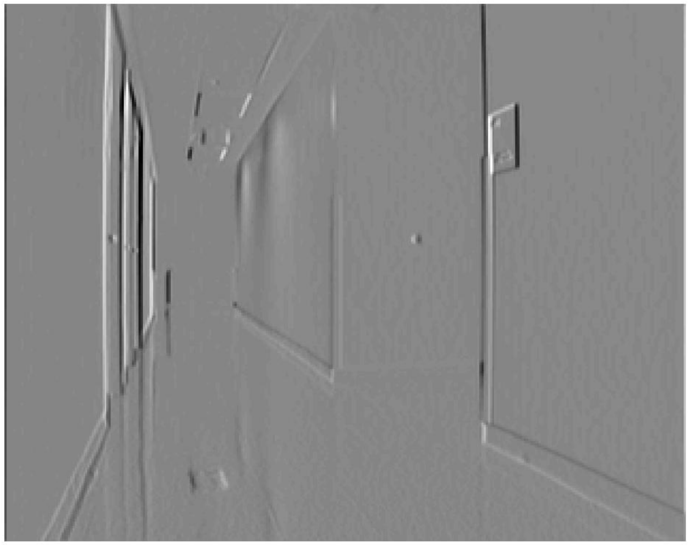
Horizontal Edge Detection

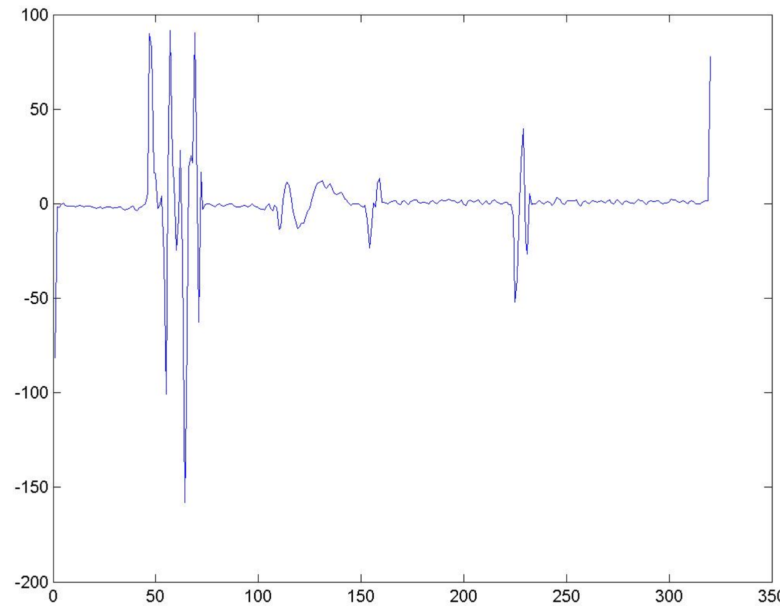
This mask is called the (horizontal) Prewitt Edge Detector

Gaussian and its derivative



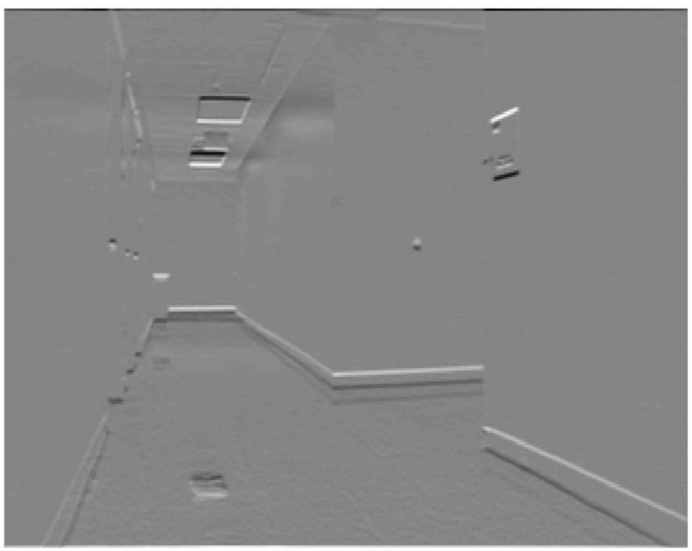
$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}, \quad g'(x) = -\frac{x}{\sigma^2\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}.$$





Vertical edges $I_x(x, y) = \frac{\partial I}{\partial x}$

First derivative - one column



$I_y(x, y) = \frac{\partial I}{\partial y}$

Horizontal edges

Gradient orientation

- Image Gradient $\nabla I = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]$

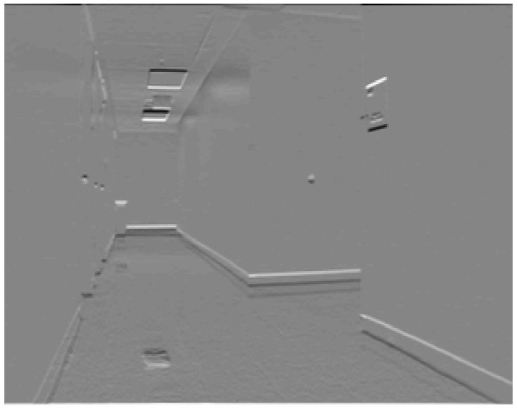
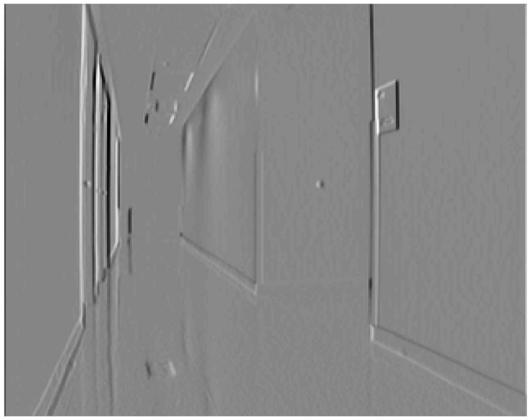
- Gradient Magnitude

$$m = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

- Gradient Orientation

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

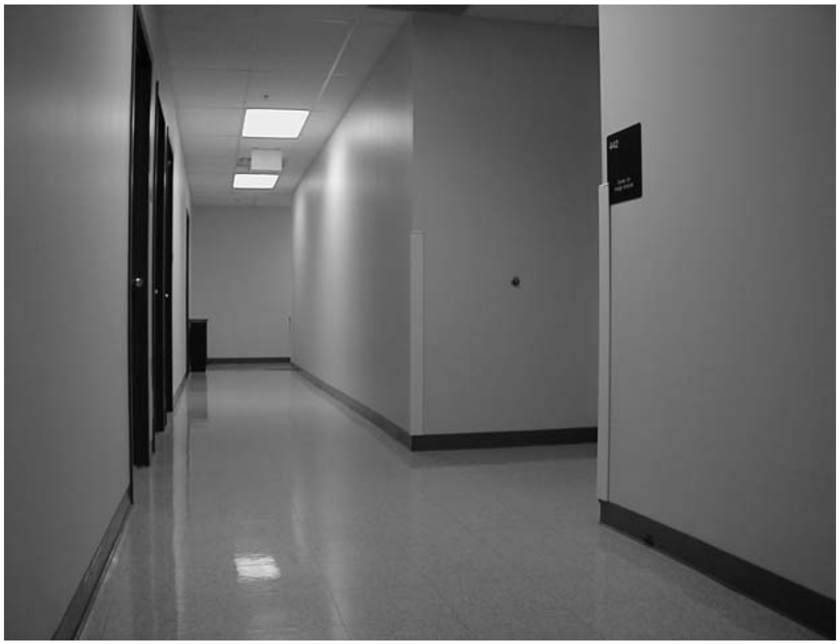
Canny Edge Detector

- Edge detection involves 3 steps:
  - Noise smoothing
  - Edge enhancement
  - Edge localization
- J. Canny formalized these steps to design an *optimal* edge detector
- How to go from derivatives to edges ?

Horizontal edges

Edge Detection



original image

gradient magnitude

Canny edge detector

- Compute image derivatives
- if gradient magnitude > τ and the value is a local maximum along gradient direction – pixel is an edge candidate

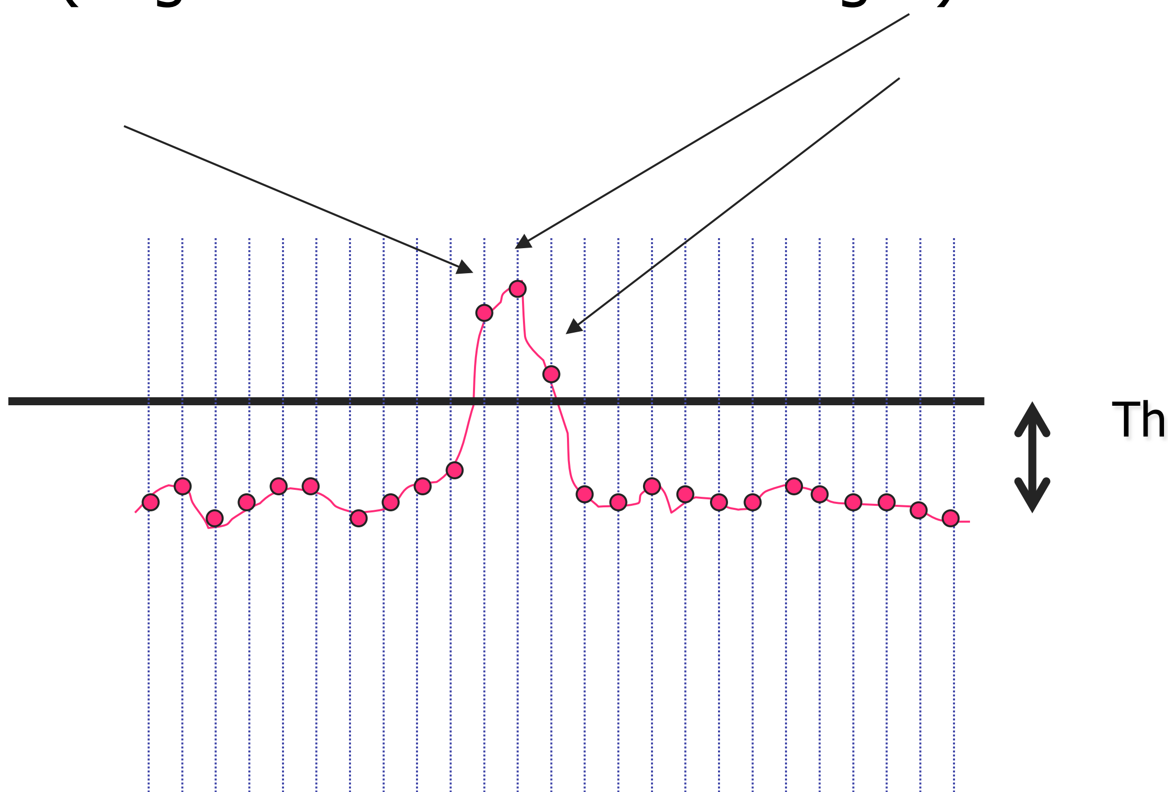
Algorithm Canny Edge detector

- The input is image I; G is a zero mean Gaussian filter (std = σ)

  1. $J = I * G$ (smoothing)
  2. For each pixel (i,j): (edge enhancement)
     - Compute the image gradient
       - $\nabla J(i,j) = (J_x(i,j), J_y(i,j))'$
     - Estimate edge strength
       - $e_s(i,j) = (J_x^2(i,j) + J_y^2(i,j))^{1/2}$
     - Estimate edge orientation
       - $e_o(i,j) = \arctan(J_x(i,j)/J_y(i,j))$
- The output are images E_s - Edge Strength - Magnitude
- and Edge Orientation E_o .

- E_s has large values at edges: Find local maxima

- … but it also may have wide ridges around the local maxima (large values *around* the edges)

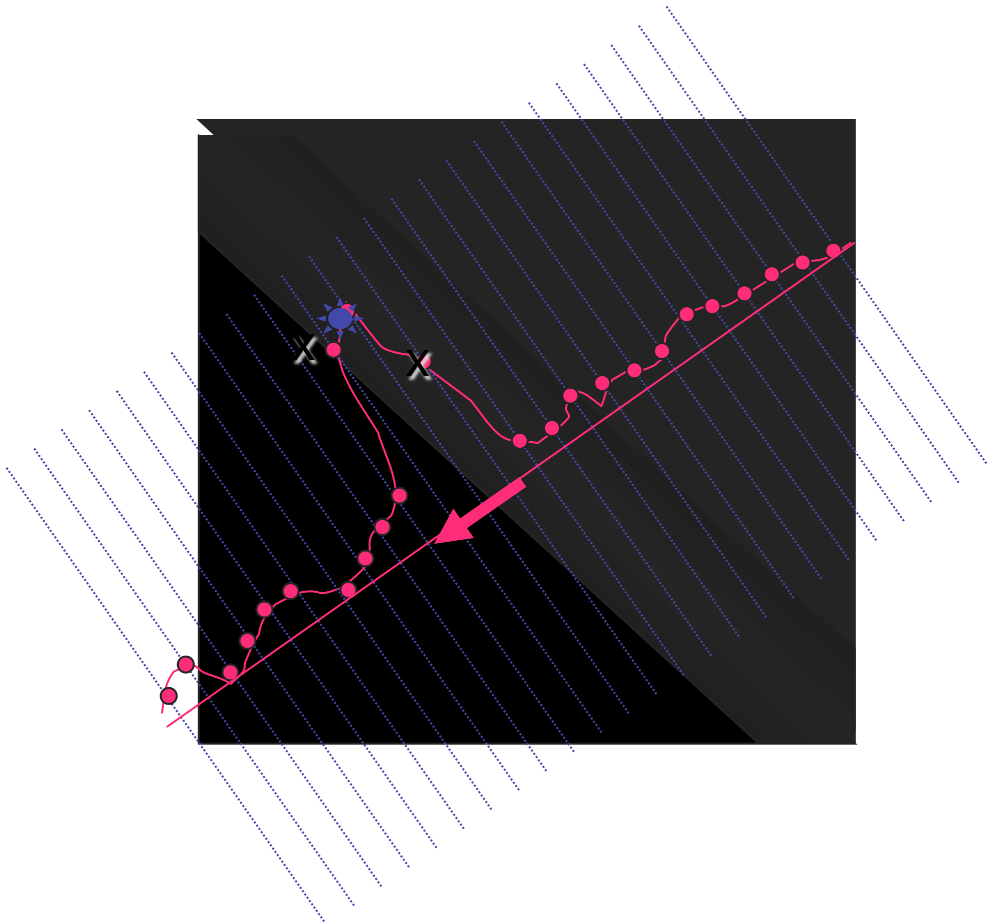


NONMAX_SUPRESSION

Edge orientation

- The inputs are E_s & E_o (outputs of CANNY_ENHANCER)
- Consider 4 directions $D=\{ 0,45,90,135\}$ wrt x
- For each pixel (i,j) do:
  1. Find the direction $d \in D$ s.t. $d \cong E_o(i,j)$ (normal to the edge)
  2. If {$E_s(i,j)$ is smaller than at least one of its neigh. along d}
     - $I_N(i,j)=0$
     - Otherwise, $I_N(i,j)= E_s(i,j)$
- The output is the thinned edge image I_N

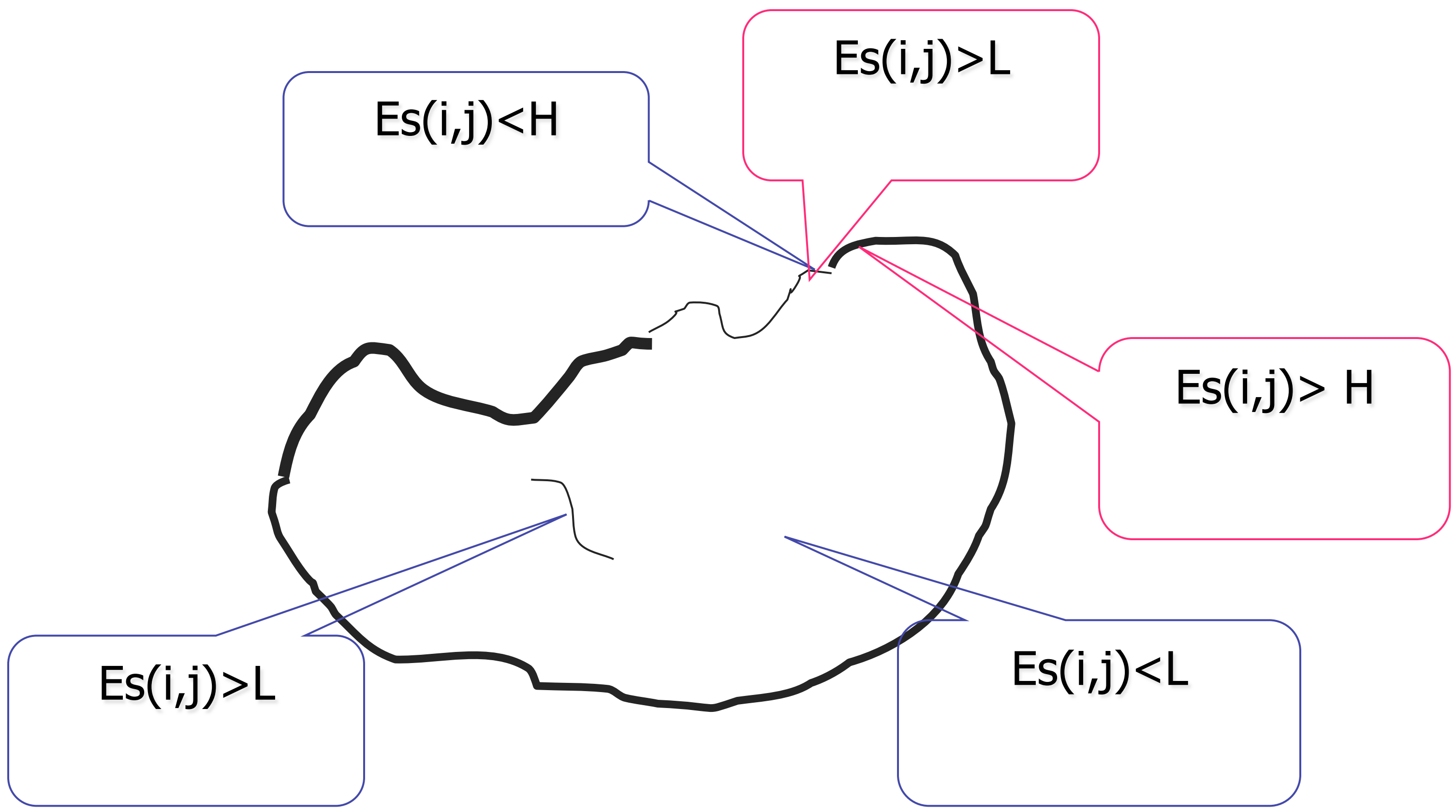
Graphical Interpretation



Thresholding

- Edges are found by thresholding the output of NONMAX_SUPRESSION
- If the threshold is too high:
  - Very few (none) edges
    - High MISDETECTIONS, many gaps
- If the threshold is too low:
  - Too many (all pixels) edges
    - High FALSE POSITIVES, many extra edges

SOLUTION: Hysteresis Thresholding



Canny Edge Detection (Example)

Original image

gap is gone

Strong + connected weak edges

Strong edges only

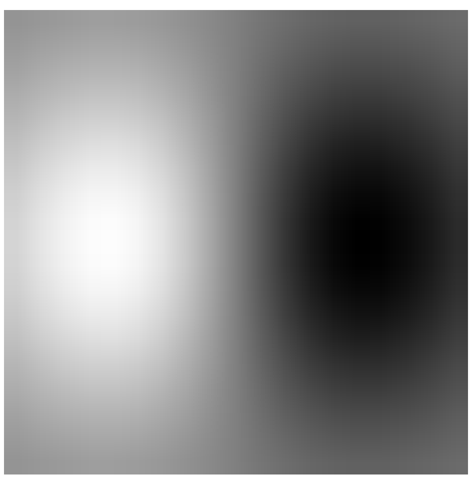
Weak edges

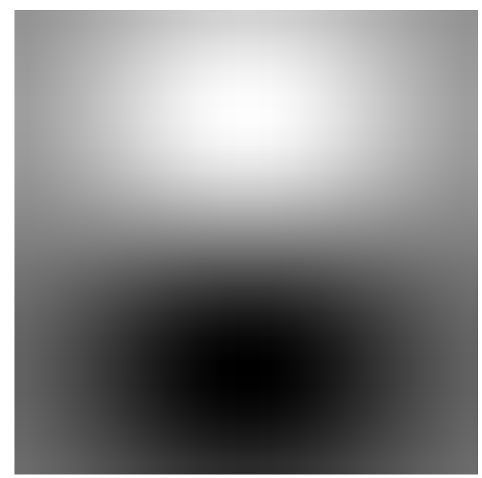
courtesy of G. Loy

Filters are templates

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector
- Filtering the image is a set of dot products

- Insight
  - filters look like the effects they are intended to find
  - filters find effects they look like





Robinson Compass Masks

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

→

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

↗

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

↑

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

↖

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

←

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

↙

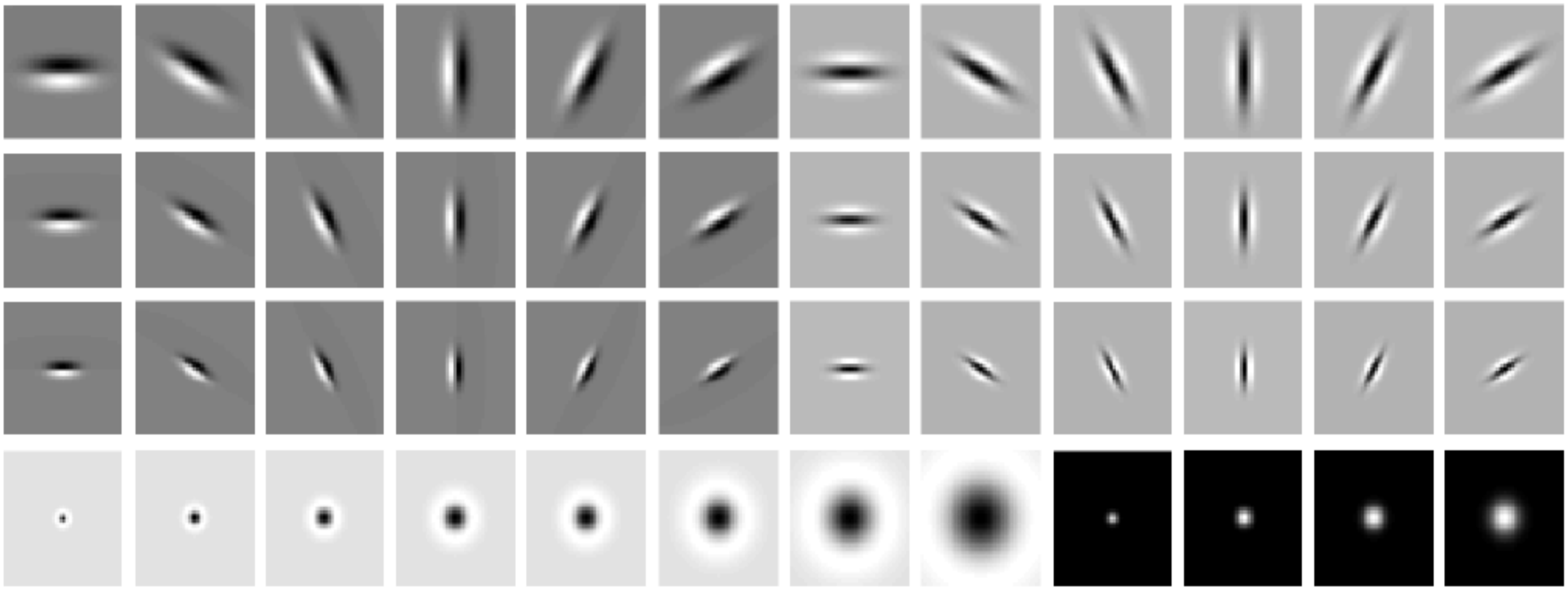
-1	-2	-1
0	0	0
1	2	1

↓

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

↘

Filter Bank



Leung & Malik, Representing and Recognizing the Visual Apperance using 3D Textons, IJCV 2001