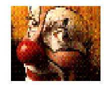
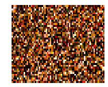
Lecture 4: Linear filters

Tuesday, Sept 11

Many slides by (or adapted from) D. Forsyth, Y. Boykov, L. Davis, W. Freeman, M. Hebert, D. Kreigman, P. Duygulu

Image neighborhoods

- Q: What happens if we reshuffle all pixels within the image?

- A: Its histogram won't change.
  Point-wise processing unaffected.
- Filters reflect spatial information

Image filtering

Modify the pixels in an image based on some function of a local neighborhood of the pixels

10	5	3
4	5	1
1	1	7

Some function →

	7	

Linear filtering

- Replace each pixel with a linear combination of its neighbors.
- Convolution kernel: prescription for the linear combination

10	5	3
4	5	1
1	1	7

$*$

0	0	0
0	0.5	0
0	1.0	0.5

kernel

$=$

	7	

Why filter images?

- Noise reduction
- Image enhancement
- Feature extraction

Convolution

- 1D Formula:

kernel signal

$$h(x) * f(x) \stackrel{C}{=} \int_u h(x-u) f(u)\, du$$

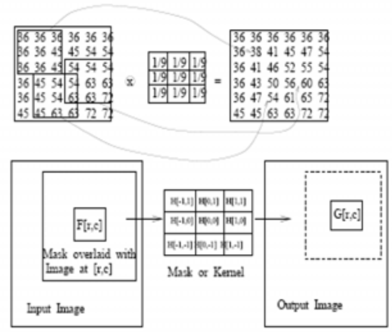
$$\stackrel{D}{=} \sum_i h[x-i] f[i]$$

- 2D Formula:

$$h(x,y) * f(x,y) \stackrel{C}{=} \int_v \int_u h(x-u, y-v) f(u,v)\, du\, dv$$

$$\stackrel{D}{=} \sum_i \sum_j h[x-i, y-j] f[i,j]$$

Convolution



Shapiro & Stockman

Convolution example

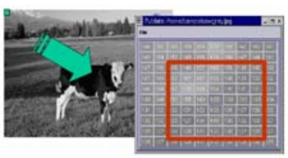
145	148	157	160	151	139	140
157	167	167	155	139	129	133
167	176	169	150	135	131	131
152	155	149	139	133	133	133
132	131	132	133	131	127	130
129	127	134	141	134	122	125
126	128	131	132	130	127	129

→

			141			

(169 + 150 + 135 + 149 + 139 + 133 + 132 + 133 + 131) / 9 = 141.2 = 141

Convolution example

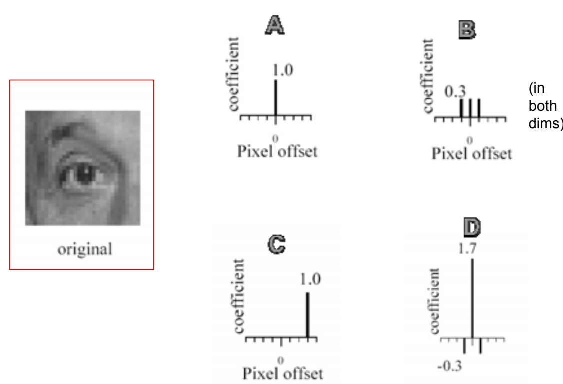


145	148	157	160	151	139	140
157	167	167	155	139	129	133
167	176	169	150	135	131	131
152	155	149	139	133	133	133
132	131	132	133	131	127	130
129	127	134	141	134	122	125
126	128	131	132	130	127	129

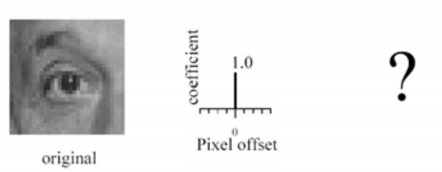
→

			141	137		

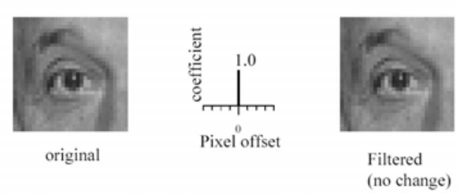
(150+135+131+139+133+133+133+131+127) / 9 = 134 + 6 = 137



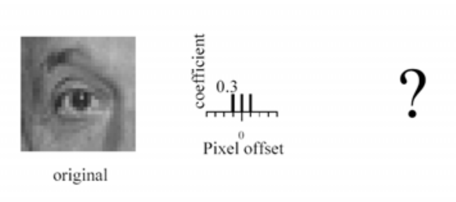
Filtering examples



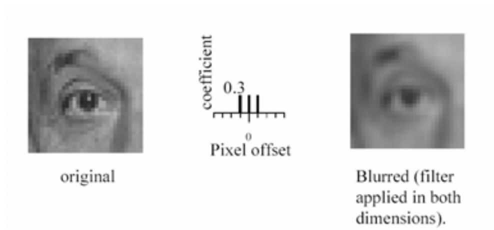
Filtering examples: Identity



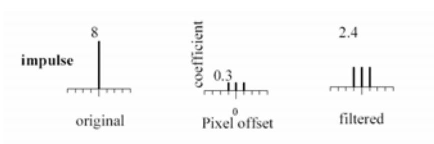
Filtering examples



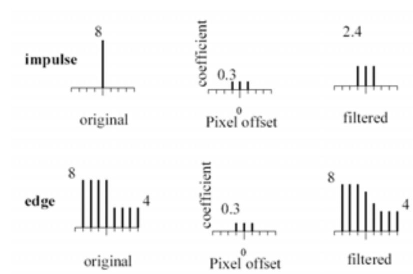
Filtering examples: Blur



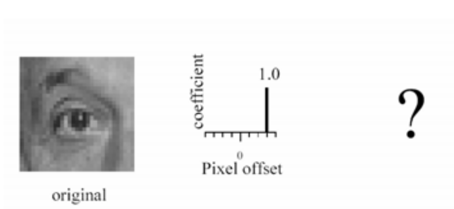
Filtering examples: Blur



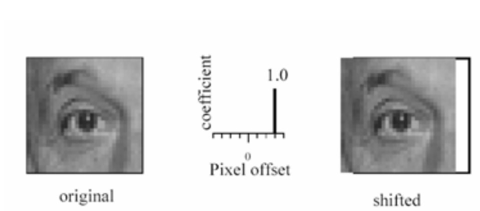
Filtering examples: Blur



Filtering examples

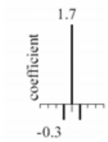


Filtering examples: Shift



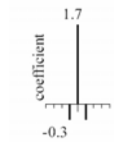
Filtering examples

original



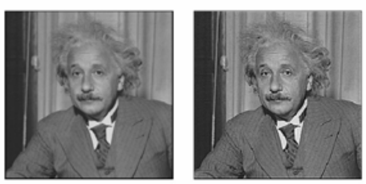
Filtering examples: sharpening

original



Sharpened original

Filtering examples: sharpening



before after

Properties

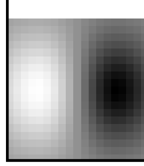
- Shift invariant
  - G(Shift(f(x))=Shift(G(f(x)))
- Linear
  - G(k f(x))=kG(f(x))
  - G(f+g) = G(f) + G(g)

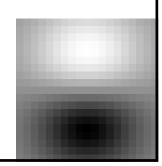
Properties

- Associative: (f * g) * h = f * (g * h)
- Differentiation rule: $\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$

Filters as templates

- Applying filter = taking a dot-product between 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



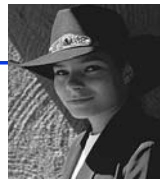


Noise

Filtering is useful for noise reduction...

Common types of noise:

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



Original

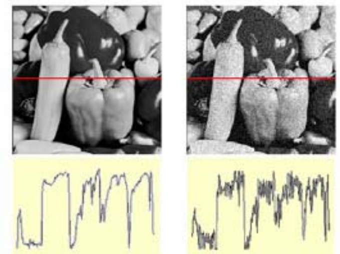
Salt and pepper noise

Impulse noise

Gaussian noise

Gaussian noise

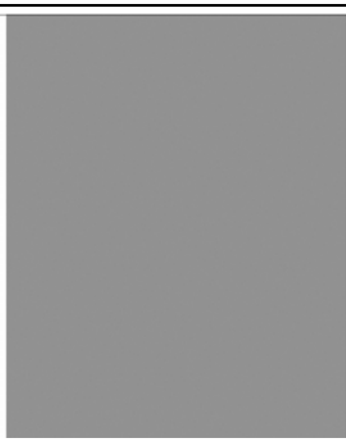
Image
Noise



$$f(x,y) = \overbrace{\hat{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

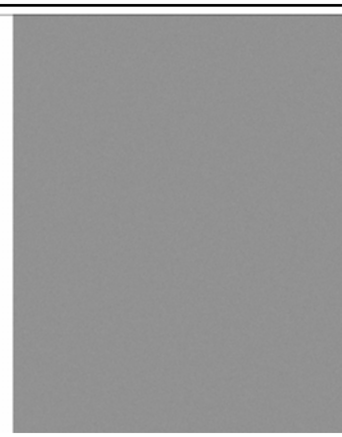
Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

sigma=1



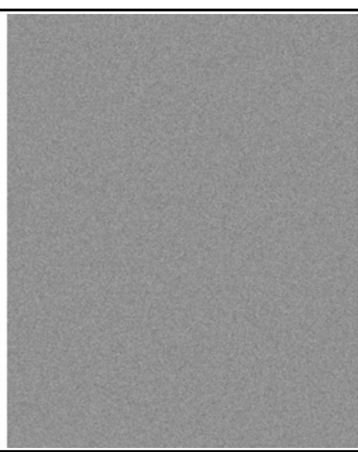
Effect of sigma on Gaussian noise

sigma=4



Effect of sigma on Gaussian noise

sigma=
16



Effect of sigma on Gaussian noise

sigma=1

Effect of sigma on Gaussian noise

sigma=16

Effect of sigma on Gaussian noise

Gaussian noise

- Issues
  - allows noise values greater than maximum or less than zero
  - good model for small standard deviations
  - independence may not be justified
  - does not model other sources of “noise”

Smoothing and noise

- Expect pixels to “be like” their neighbors
- Expect noise processes to be independent from pixel to pixel

⇩

- Smoothing suppresses noise, for appropriate noise models
- Impact of scale: more pixels involved in the image, more noise suppressed, but also more blurring

Mean filtering

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

$F[x, y]$

$G[x, y]$

Mean filtering

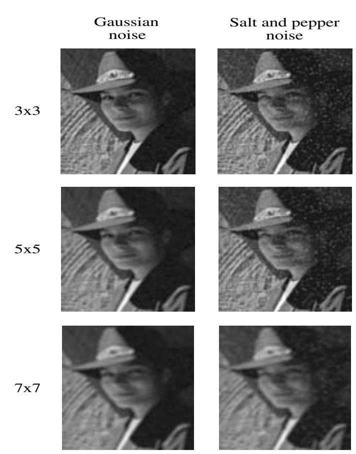
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

	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	

$G[x, y]$

Effect of mean filters



Mean kernel

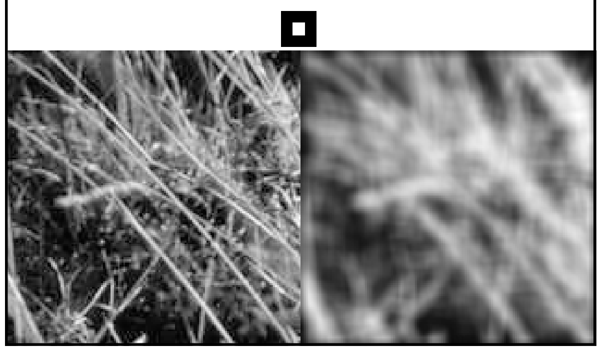
- What's the kernel for a 3x3 mean filter?

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

$F[x,y]$

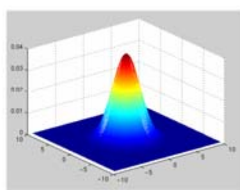
$H[u,v]$

Smoothing by Averaging



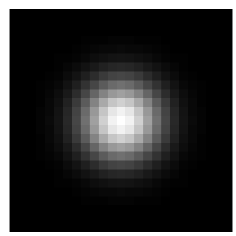
Smoothing with a Gaussian

- Averaging does not model defocused lens well
- Gaussian kernel weights pixels at its center much more strongly than its boundaries



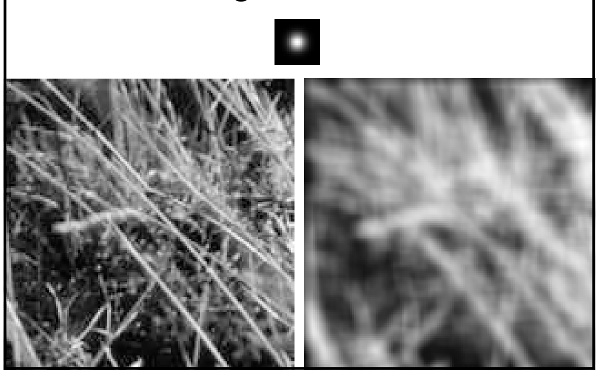
$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

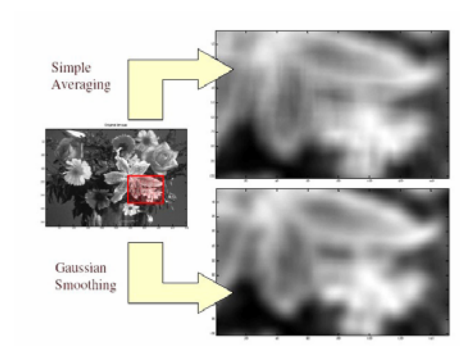
Isotropic Gaussian



Reasonable model of a circularly symmetric blob

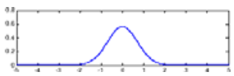
Smoothing with a Gaussian

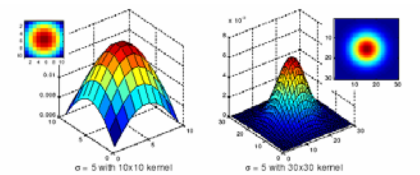




Gaussian filters

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

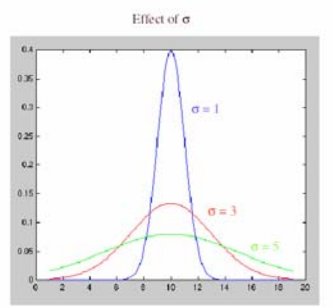




$$\frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

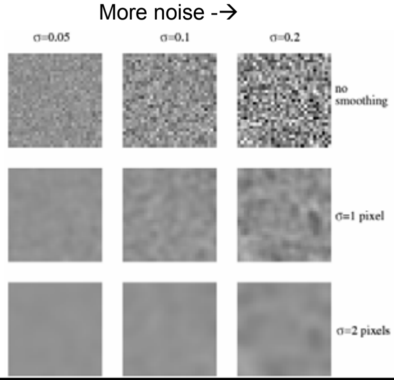
$H[u, v]$

Gaussian filters



Smoothing and noise

More noise -→



Wider kernel →

Gaussian filters

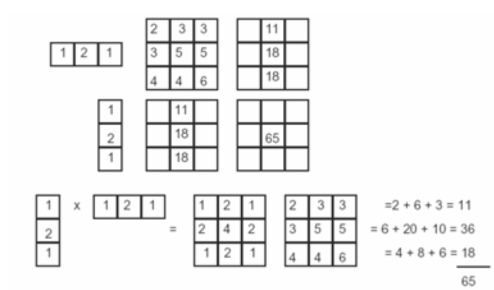
- Remove “high-frequency” components from the image → “low pass” filter
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - 2x with σ ⇔ 1x with √2σ
- *Separable* kernel

Separability

- Isotropic Gaussians factorable into product of two 1D Gaussians
- Useful: can convolve all rows, then all columns
- Linear vs. quadratic time in mask size

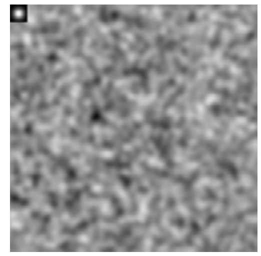
$$G_\sigma * f = g_{\sigma\rightarrow} * g_{\sigma\uparrow} * f$$

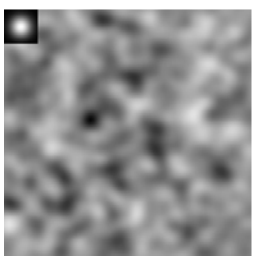
Separability

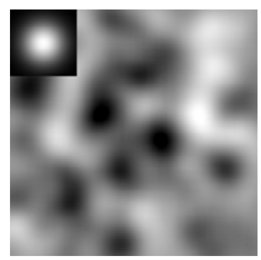


Correlation of filter responses

- Filter responses are correlated over scales similar to scale of filter
- *Filtered noise* is sometimes useful
  - looks like some natural textures

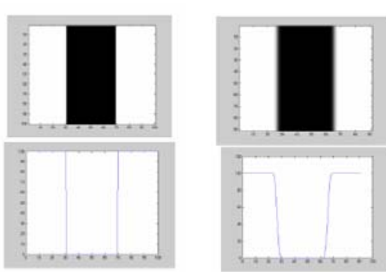




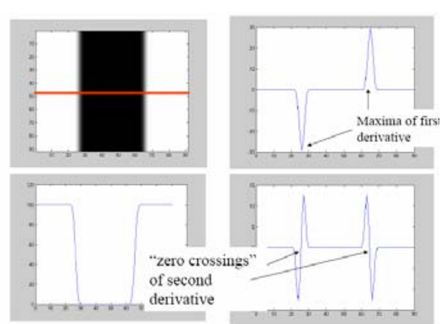


Edges and derivatives

- Edges correspond to fast changes



Edges and derivatives



Finite difference filters

Image derivatives can be approximated with convolution.

0	0	0
1	0	-1
0	0	0

$H[u, v]$

Finite differences

- M = [-1 0 1]

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	0	0	0	0	12	12	0	0	0	0

(a) S_1 is an upward step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	$\otimes$	M	0	0	0	0	-12	-12	0	0	0	0

(b) S_2 is a downward step edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	$\otimes$	M	0	0	0	3	6	6	6	3	0	0

(c) S_3 is an upward ramp

S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	0	0	0	12	0	-12	0	0	0	0

(d) S_4 is a bright impulse or "line"

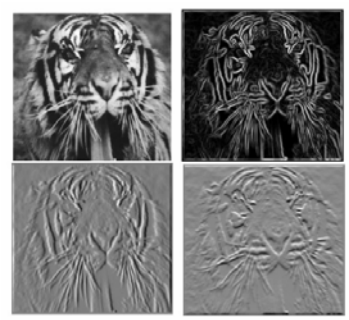
Finite difference filters

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

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

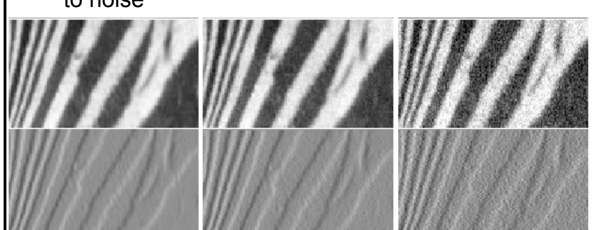
Finite differences



Which is derivative in the x direction?

Finite differences

Strong response to fast change → sensitive to noise



Increasing noise ->
(zero mean additive Gaussian noise)

Smoothed derivatives

- Smooth before differentiation: assume that "meaningful" changes won't be suppressed by smoothing, but noise will
- Two convolutions: smooth, then differentiate?

$$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$$

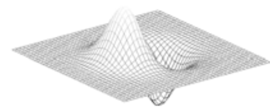
Smoothed derivatives

- Solution: First smooth the image by a Gaussian G_σ and then take derivatives:
$$\frac{\partial f}{\partial x} \approx \frac{\partial (G_\sigma * f)}{\partial x}$$
- Applying the differentiation property of the convolution:
$$\frac{\partial f}{\partial x} \approx \frac{\partial G_\sigma}{\partial x} * f$$

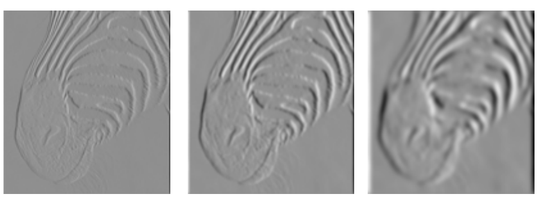
Derivative of Gaussian filter

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

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



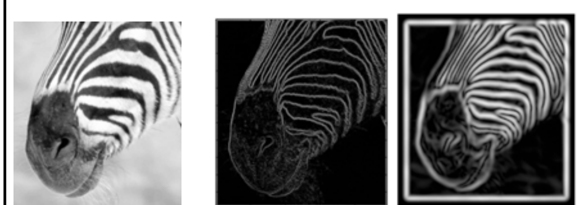
Derivative of Gaussian filter



σ = 1 pixel σ = 3 pixels σ = 7 pixels

Derivatives in the *x* direction

Derivative of Gaussian filter

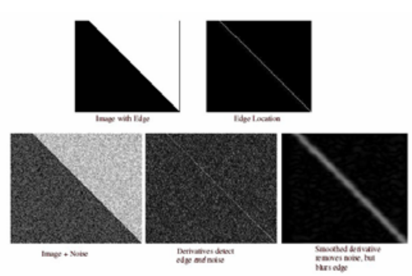


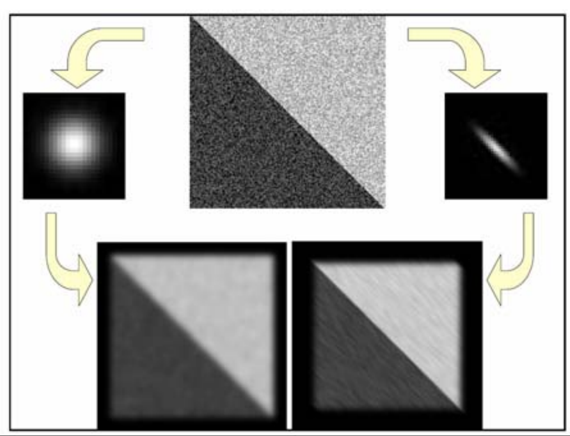
σ = 1 pixel σ = 2 pixels

The apparent structures differ depending on Gaussian's scale parameter.

Smoothed derivatives: caveat

- Tradeoff between localization and smoothing

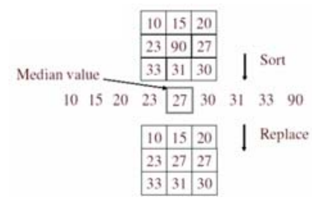




Typical mask properties

- 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
- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size

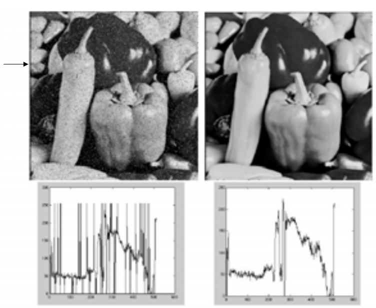
Median filter



- Non-linear
- No new pixel values
- Removes spikes

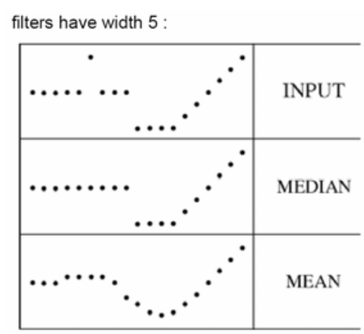
Median filter

Salt and pepper noise



Median filtered

Median filter



Median filter

Next

- More on edges, pyramids, and texture
- Pset 1 out tomorrow
- Reading: chapters 8 and 9