

# The Power of Filtering: Edges, Blobs, and More

Princeton AI4ALL

# Image processing



90	92	92	93	93	94	94	95	95	96
94	95	96	96	97	98	98	99	99	99
98	99	99	100	101	101	102	102	102	103
103	103	104	104	105	107	106	106	111	121
108	108	109	110	112	111	112	119	123	117
113	113	110	111	113	112	122	120	117	106
118	118	109	96	106	113	112	108	117	114
116	132	120	111	109	106	101	106	117	118
111	142	112	111	101	106	104	109	113	110
114	139	109	108	103	106	107	108	108	108
115	139	117	114	101	104	103	105	114	110
115	129	103	114	101	97	109	116	117	118
120	130	104	111	116	104	107	109	110	99
125	130	103	109	108	98	104	109	119	105
119	128	123	138	140	133	139	120	137	145
164	138	143	163	155	133	145	125	133	155
174	126	123	122	102	106	108	62	62	114
169	134	133	127	92	102	94	47	52	118
125	132	117	122	102	103	98	51	53	120
109	99	113	116	111	98	104	82	99	116

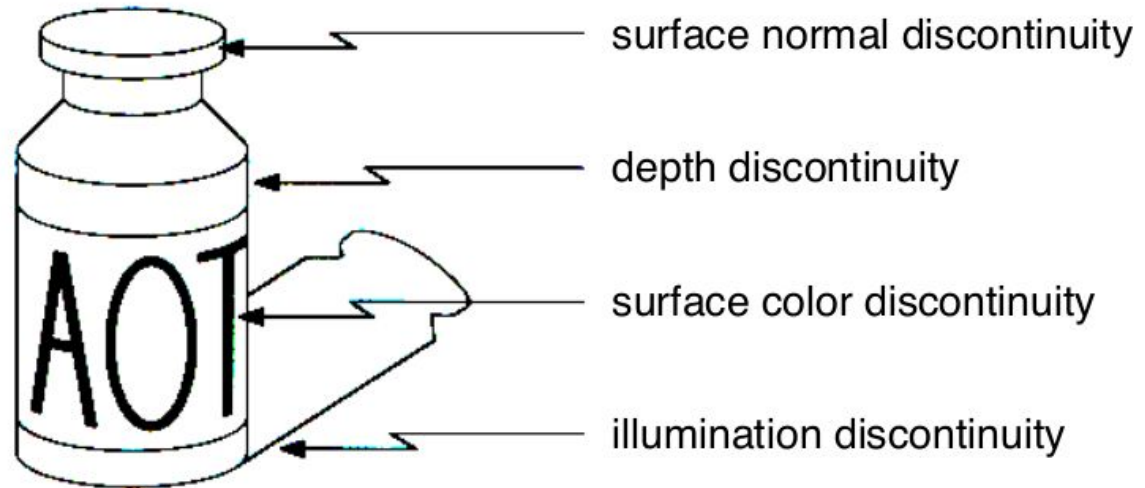
What's the basic structure we may want to detect?



# Edges

# Origin of Edges

- Edges are caused by a variety of factors:

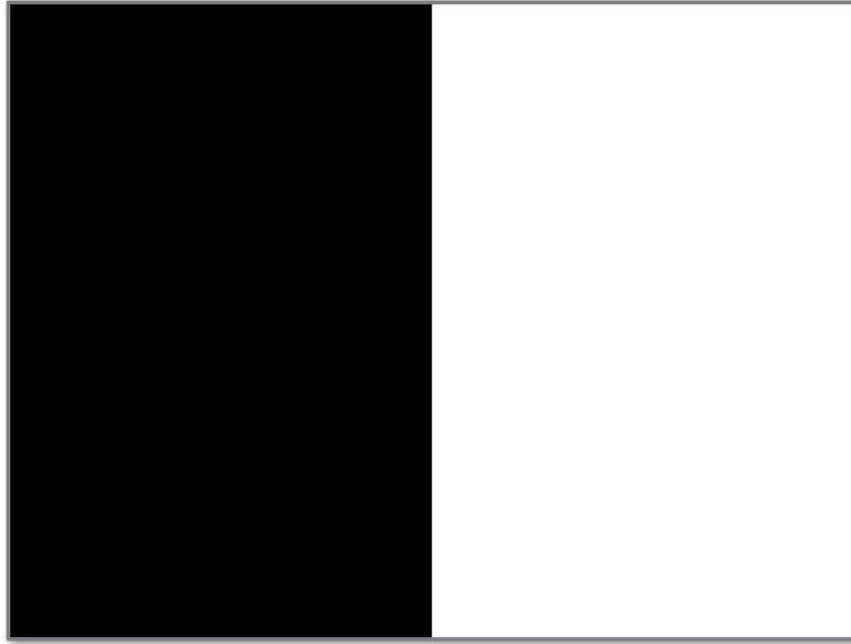


# Edge Detection

- Intuitively, much of semantic and shape information is available in the edges
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)
- But what, **mathematically**, is an edge?



# What is an Edge?



Edge easy to find

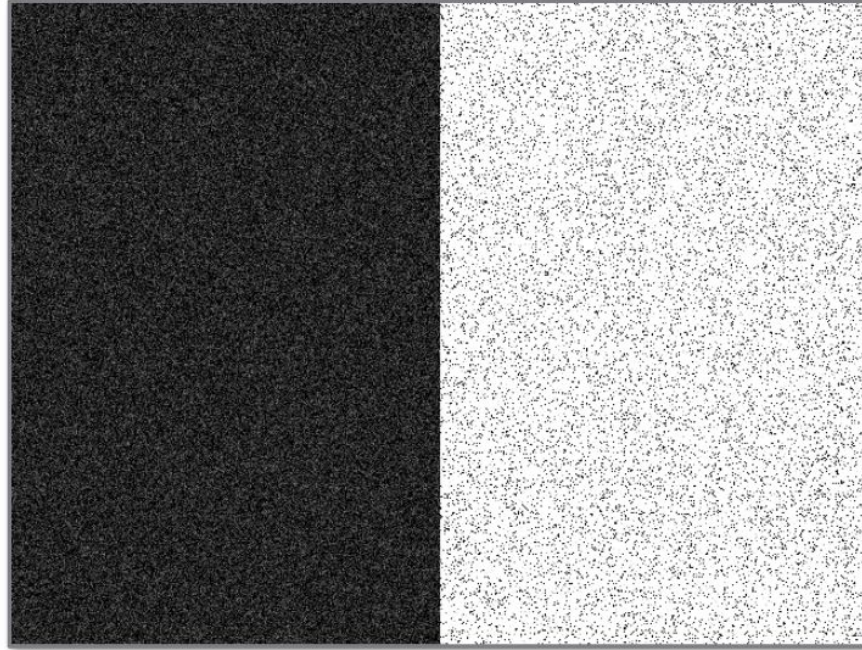


# What is an Edge?



Where is edge? Single pixel wide or multiple pixels?

# What is an Edge?



Noise: have to distinguish noise from actual edge

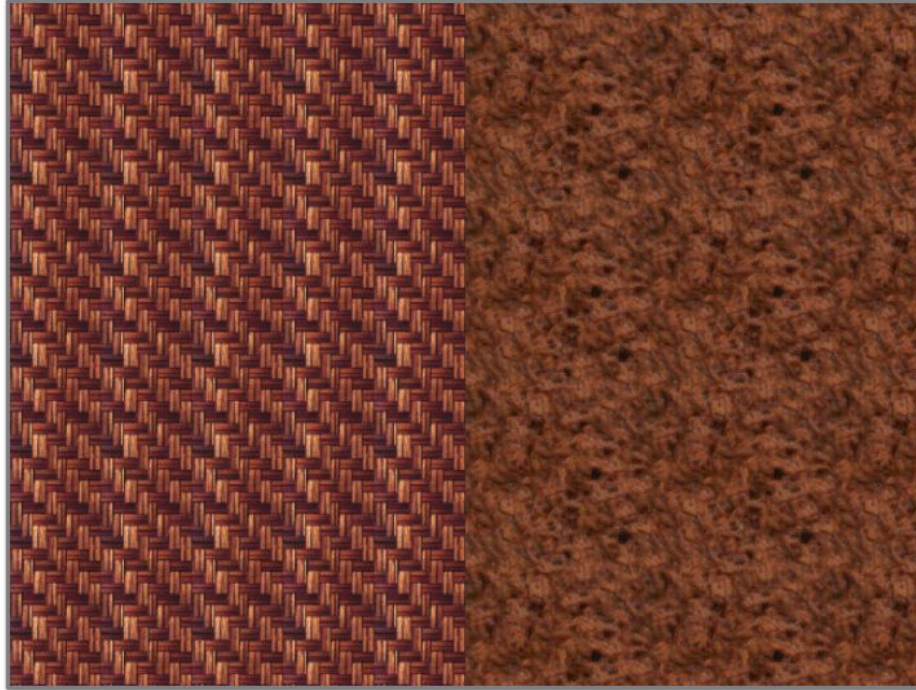


# What is an Edge?



Is this one edge or two?

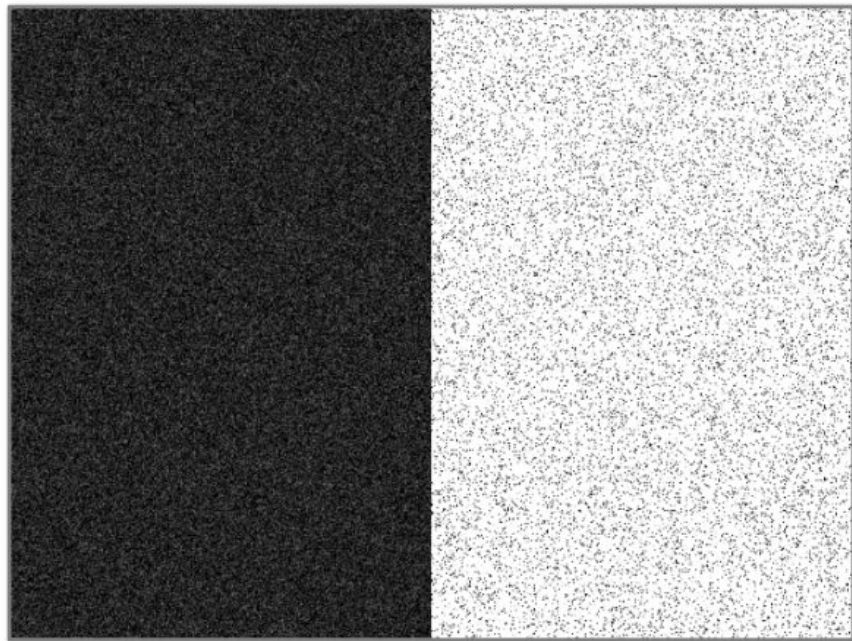
# What is an Edge?



Texture discontinuity

# Linear filtering

Let's think about the noise problem...



Q: What's one approach to reducing the amount of noise in an image?



# Idea #1: Moving average

- Let's replace each pixel with a *weighted* average of its neighborhood
- The weights are called the *filter kernel*

1	1	1	1
—	1	1	1
9	1	1	1

“box filter”

90	92	92	93	93	94	94	95	95	96
94	95	96	96	97	98	98	99	99	99
98	99	99	100	101	101	102	102	102	103
103	103	104	104	105	107	106	106	111	121
108	108	109	110	112	111	112	119	123	117
113	113	110	111	113	112	122	120	117	106
118	118	109	96	106	113	112	108	117	114
116	132	120	111	109	106	101	106	117	118
111	142	112	111	101	106	104	109	113	110
114	139	109	108	103	106	107	108	108	108
115	139	117	114	101	104	103	105	114	110
115	129	103	114	101	97	109	116	117	118
120	130	104	111	116	104	107	109	110	99
125	130	103	109	108	98	104	109	119	105
119	128	123	138	140	133	139	120	137	145
164	138	143	163	155	133	145	125	133	155

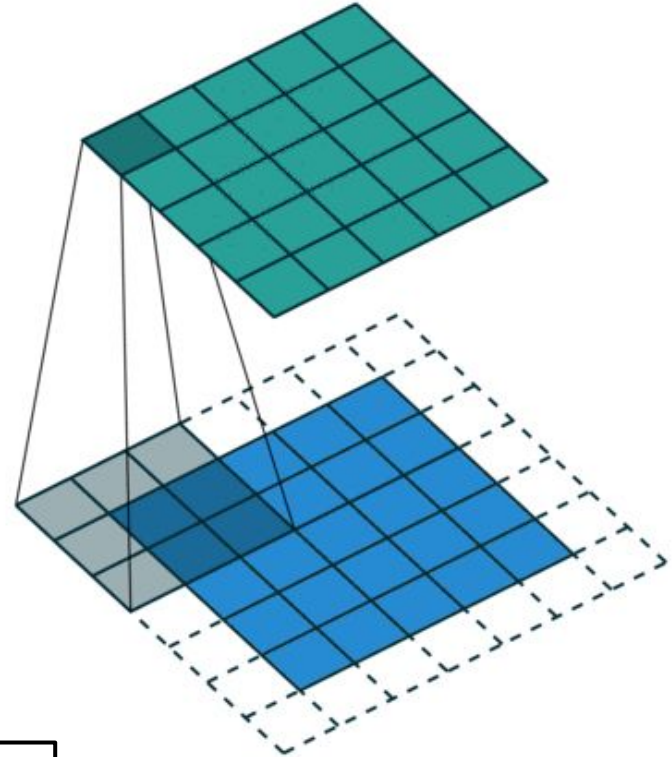
# Applying the filter

The filter:

$\frac{1}{9}$	1	1	1
	1	1	1
	1	1	1

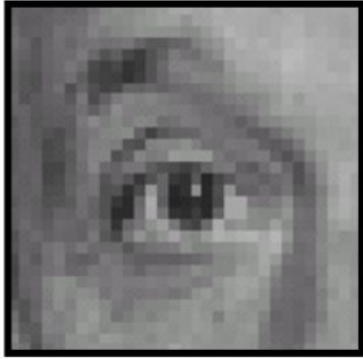
Output image

Input image



This special way of applying the filter is called “convolution”.

# Convolution with linear filters



Original

 $\frac{1}{9}$ 

1	1	1
1	1	1
1	1	1

?



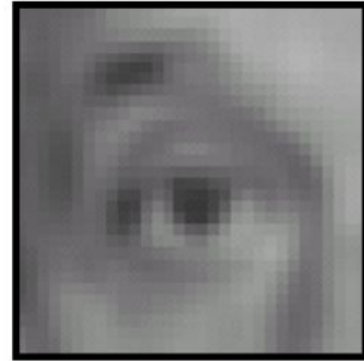
# Convolution with linear filters



Original

 $\frac{1}{9}$ 

1	1	1
1	1	1
1	1	1



Blur (with a  
box filter)



# Convolution with linear filters



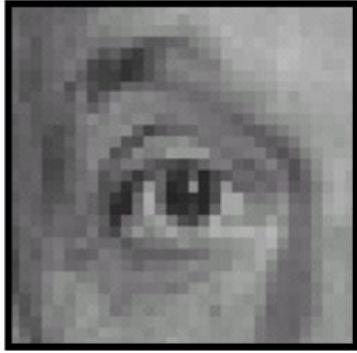
Original

0	0	0
0	1	0
0	0	0

?

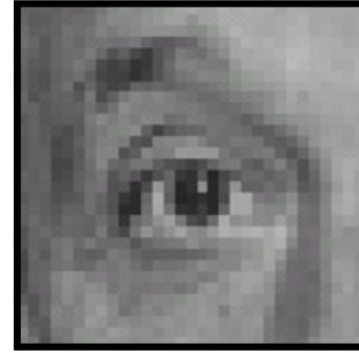


# Convolution with linear filters



Original

0	0	0
0	1	0
0	0	0



Filtered  
(no change)



# Convolution with linear filters



Original

0	0	0
1	0	0
0	0	0

?



# Convolution with linear filters



Original

0	0	0
1	0	0
0	0	0

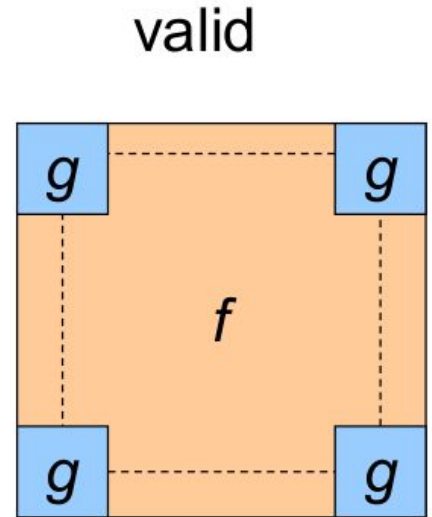
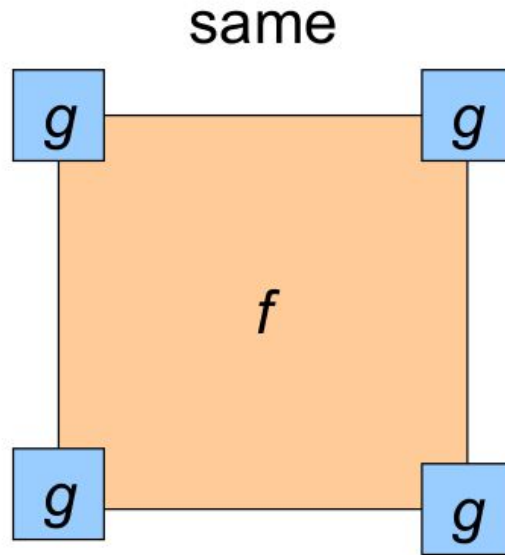
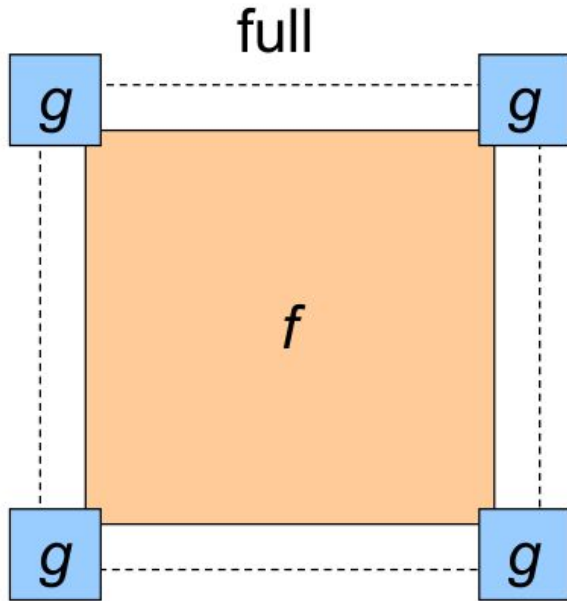


Shifted *left*  
By 1 pixel



# Annoying Detail

What is the size of the output?



Source: S. Lazebnik



# Activity:

## Applying filters in Python

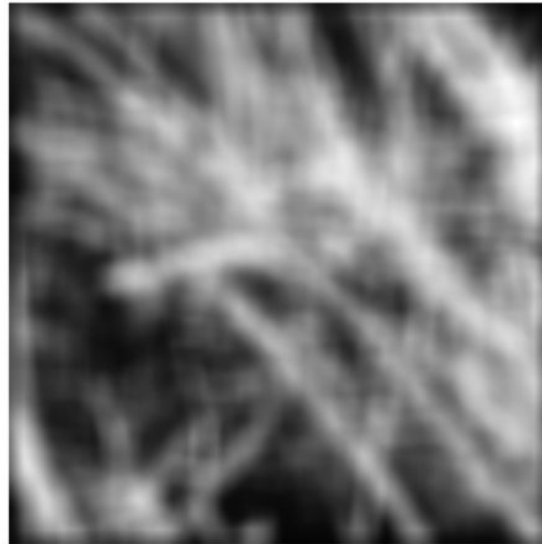
We learned how to *blur* an image with a filter. Now let's learn about...

# Sharpening and Gaussian filters



# Smoothing with box filter: Issues

- What's wrong with this picture?
- What's the solution?



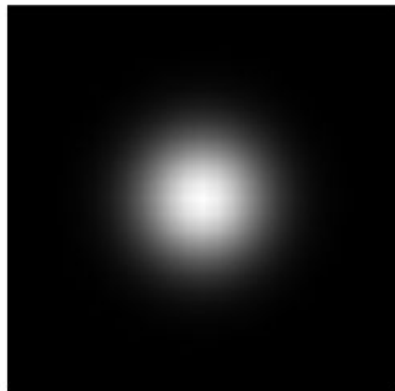
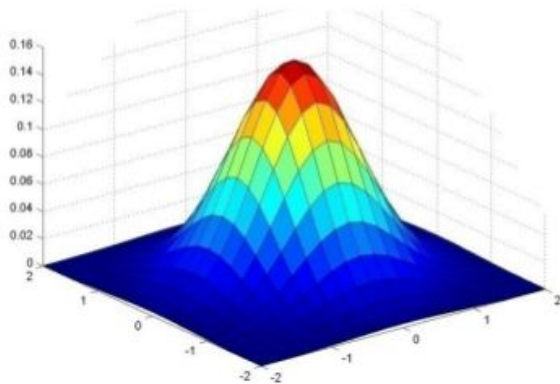
# Smoothing with box filter: Issues

- What's the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center



“fuzzy blob”

# Gaussian kernel



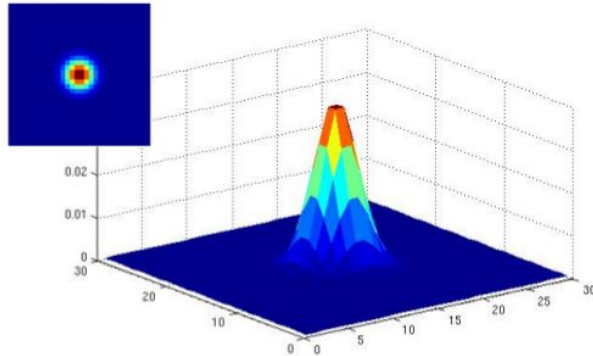
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

$5 \times 5, \sigma = 1$

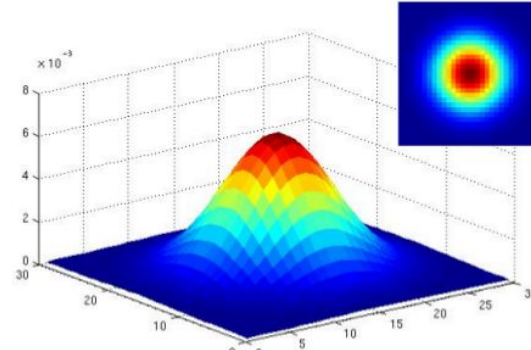
# The Gaussian kernel, formula

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

- Standard deviation  $\sigma$ : determines extent of smoothing

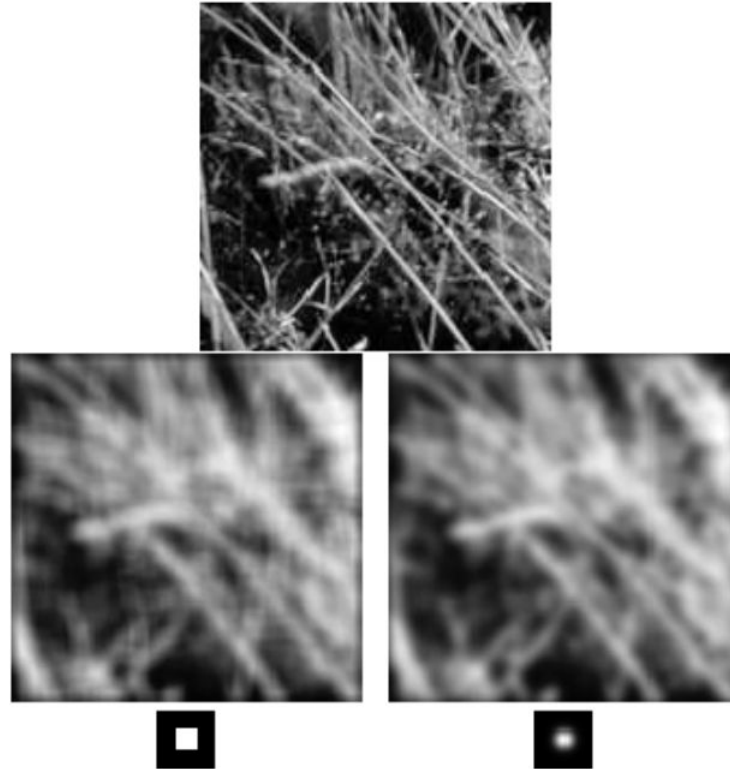


$\sigma = 2$  with  $30 \times 30$   
kernel



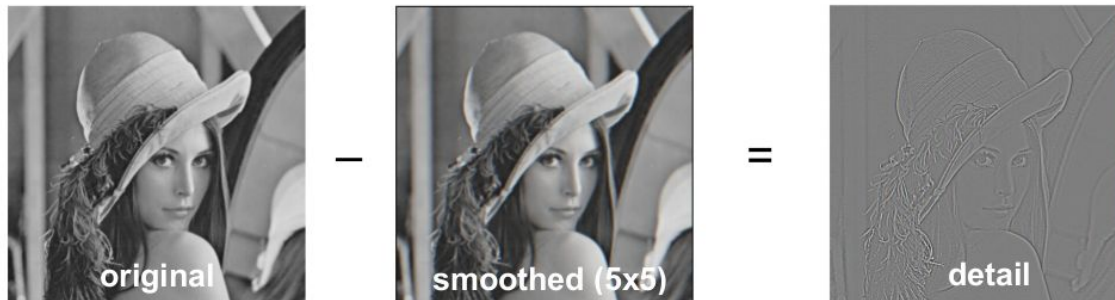
$\sigma = 5$  with  $30 \times 30$   
kernel

# Box filter versus Gaussian filter



# Sharpening an Image

What does blurring take away?



Let's add it back:



# Convolution with linear filters



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

?

(Note that filter sums to 1)

# Convolution with linear filters



Original

0	0	0
0	2	0
0	0	0

−

$\frac{1}{9}$	1	1	1
	1	1	1
	1	1	1

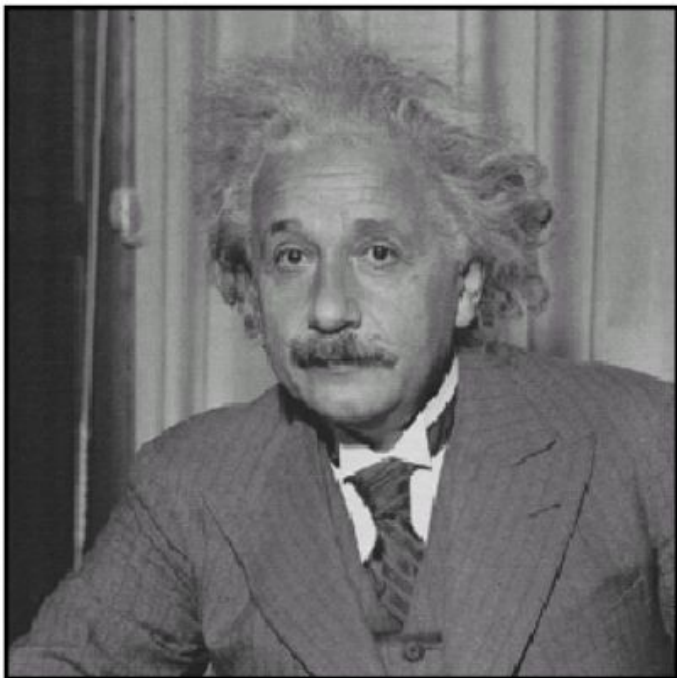


## Sharpening filter

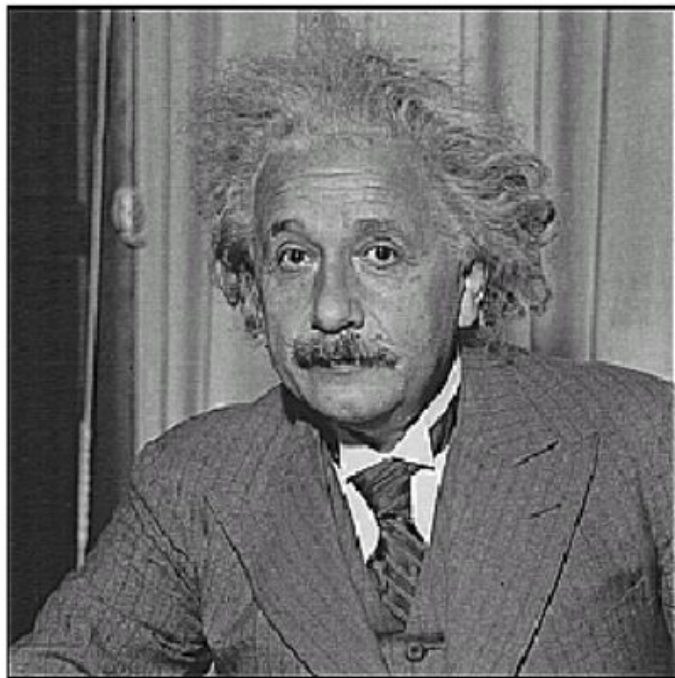
- Accentuates differences  
with local average



# Sharpening an image



**before**



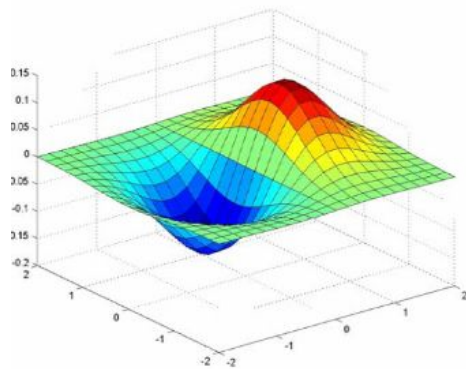
**after**

# Back to Edge Detection

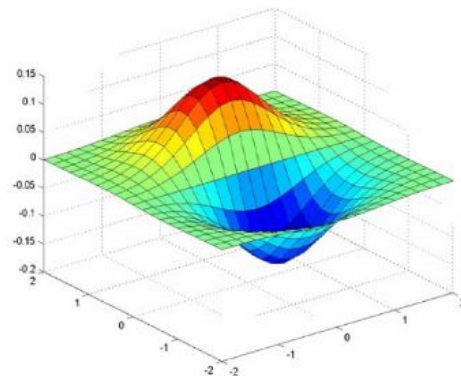
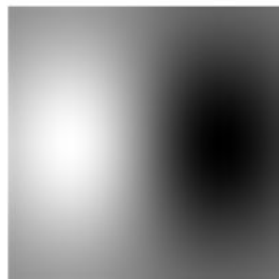
We won't go through all the gnarly details, but we'll check out the basic idea of how one algorithm works.



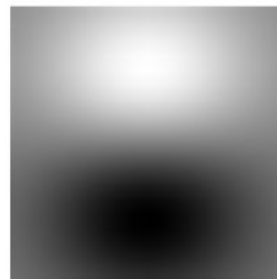
# Derivative of Gaussian filter:



x-direction

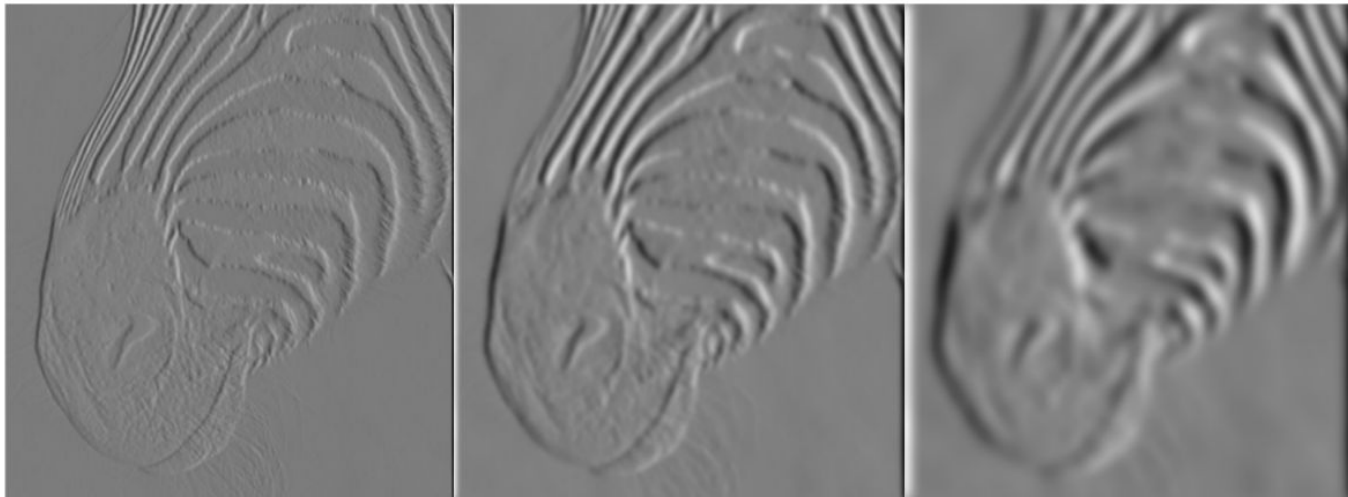


y-direction



## Derivative of Gaussian filter:

Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”



1 pixel

3 pixels

7 pixels

# The Canny Edge Detector:



original image

# The Canny Edge Detector:



magnitude of the gradient

# The Canny Edge Detector:



thresholding

# The Canny Edge Detector:

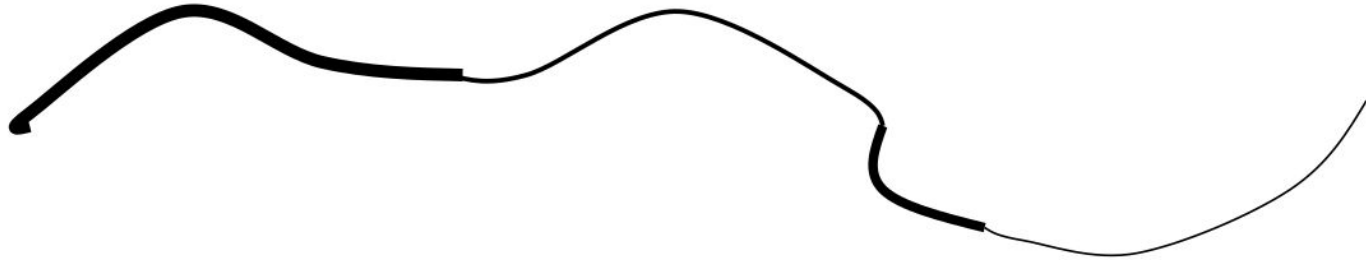


thinning



# Hysteresis Thresholding

Use a high threshold to start edge curves, and a low threshold to continue them.



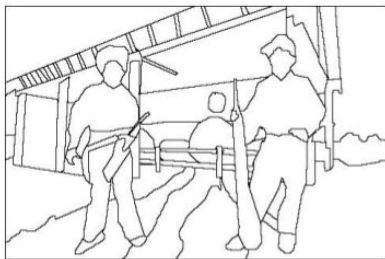
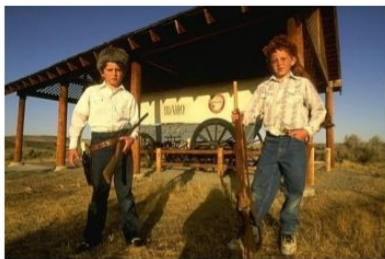
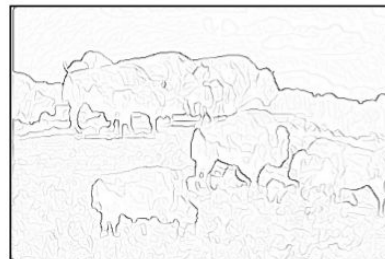
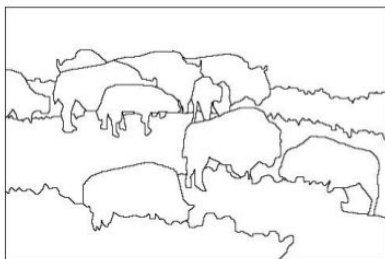
# The Canny Edge Detector:



# Edge detection for segmentation

- Berkeley segmentation database:

<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/>




image

human segmentation

gradient magnitude

# Edge detection for self-driving cars: Important!!





# Activity: Sharpening and edge detection