SAIR

Spatial AI & Robotics Lab

CSE 473/573-A

W3: FILTERING & EDGE DETECTION

Chen Wang

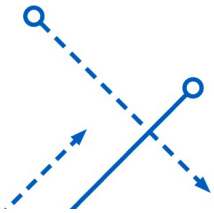
Spatial AI & Robotics Lab

Department of Computer Science and Engineering

University at Buffalo The State University of New York

Content

- Filtering
  - Linear filters, Correlation, Equivariance, Invariance
  - Gaussian Filter, Smoothing, Median filter, Sobel operator
- Template Matching
  - Cross-correlation, kernel cross-correlation
- Edge Detection
  - Image differentiation and gradient
  - Derivative theorem of convolution
  - Derivative of Gaussian filter, Laplacian of Gaussian
  - 2D edge detection filters
  - Canny edge detector, Hysteresis thresholding



SAIR

Spatial AI & Robotics Lab

IMAGE PROCESSING

Filtering

Recap

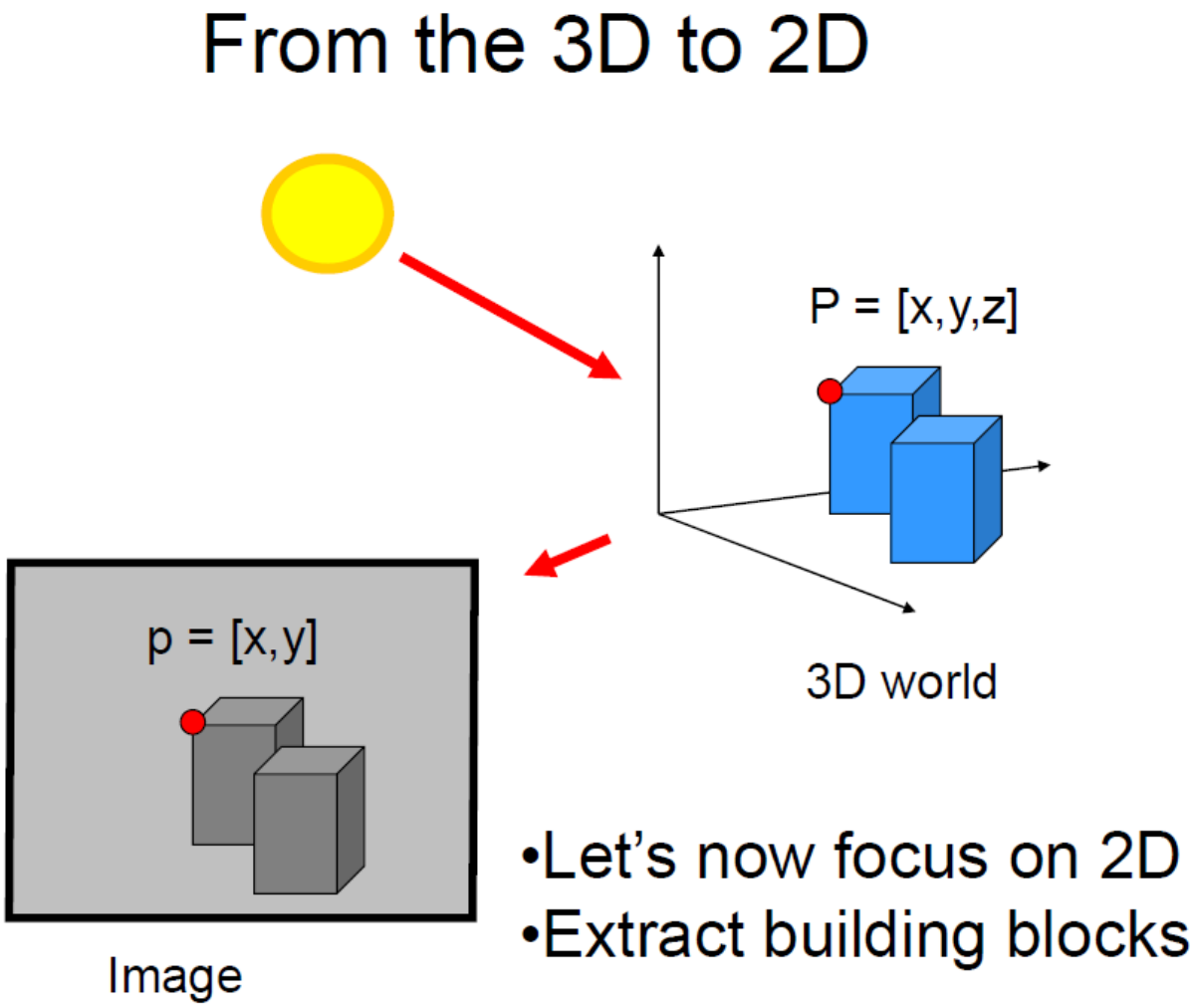
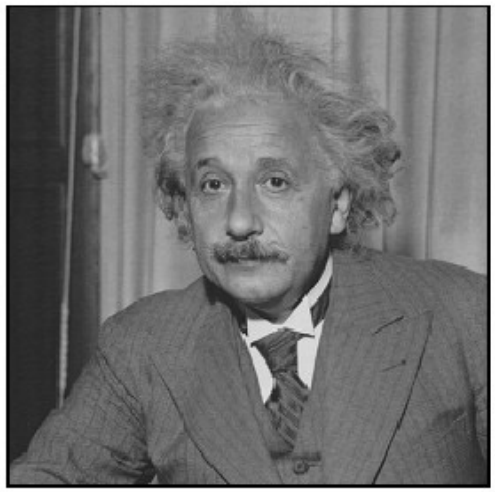
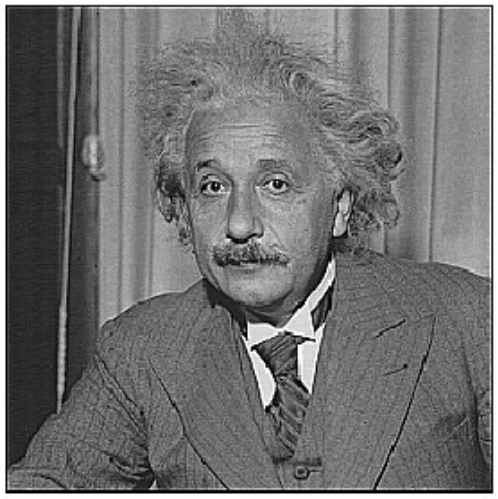

From the 3D to 2D
P = [x,y,z]
3D world
p = [x,y]
Image
•Let's now focus on 2D
•Extract building blocks

Image Filtering





Smooth/Sharpen Images...

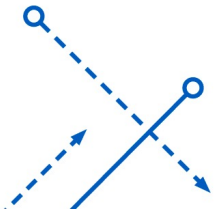
Find edges...

Find Waldo...

Image filtering

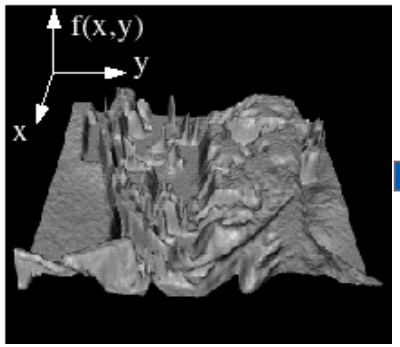
- Image filtering: compute a function of the local neighborhood at each position

- Really important!
  - Enhance images
    - Denoising, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
    - Deep Convolutional Networks



Digital images (Recap)

- In computer vision, we operate on **digital** (**discrete**) images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.

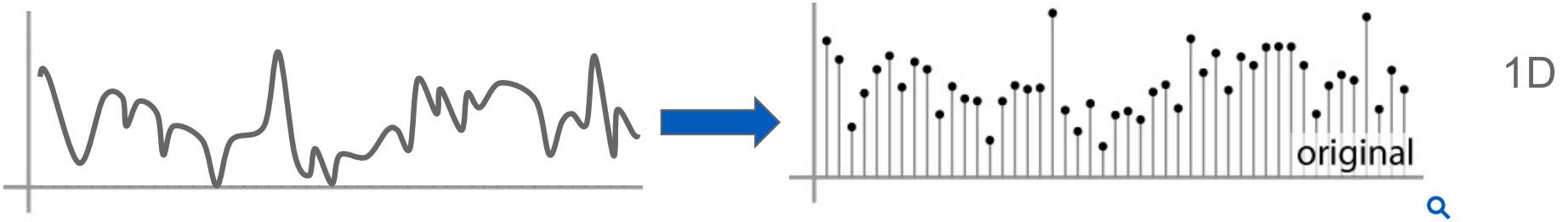


j

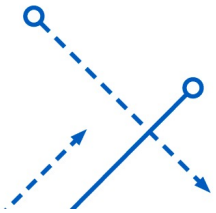
i

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

2D



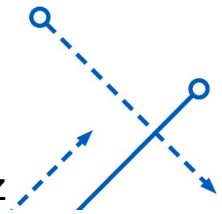
1D



Images as functions (Recap)

- Take an image as a function, f, from R^2 to R:
  - $f(x, y)$ gives the intensity at position (x, y)
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - f: $[a, b]$ x $[c, d]$ → $[0, 255]$
    - Important: we often convert [0, 255] to **[0,1.0]**.
- A color image is three functions pasted together, a "vector-valued" function

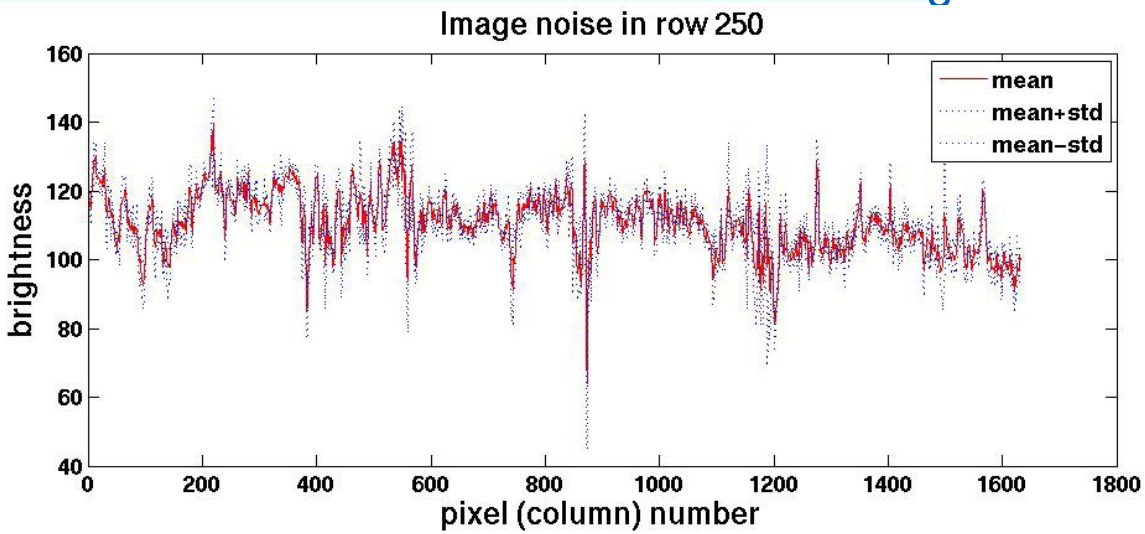
$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

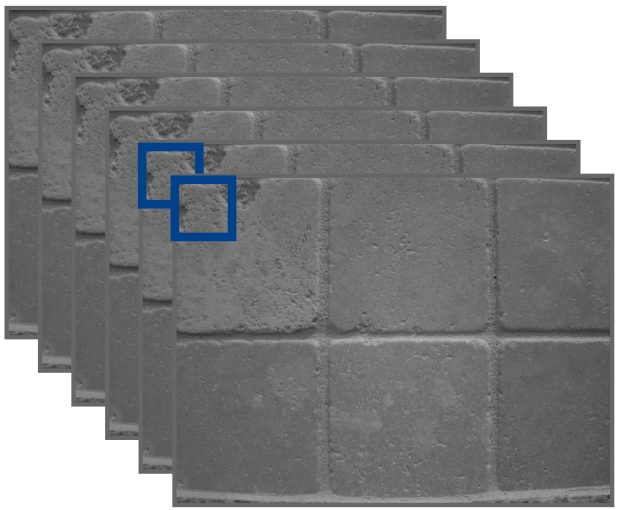


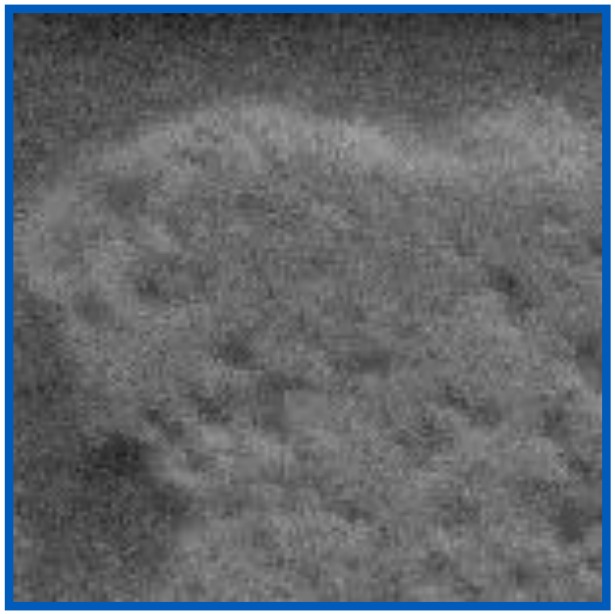
Source: S. Seitz

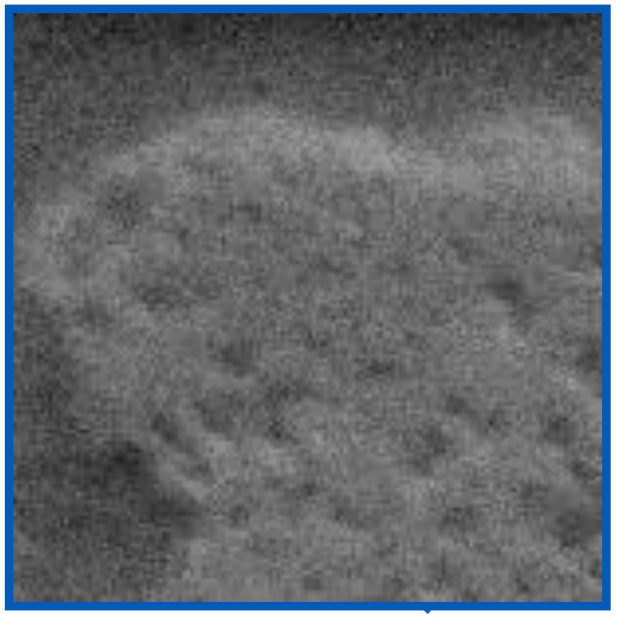
How can we do noise reduction?

- We can measure **noise** in multiple images of the same static scene.
- How could we reduce the noise, i.e., give an estimate of the true intensities?



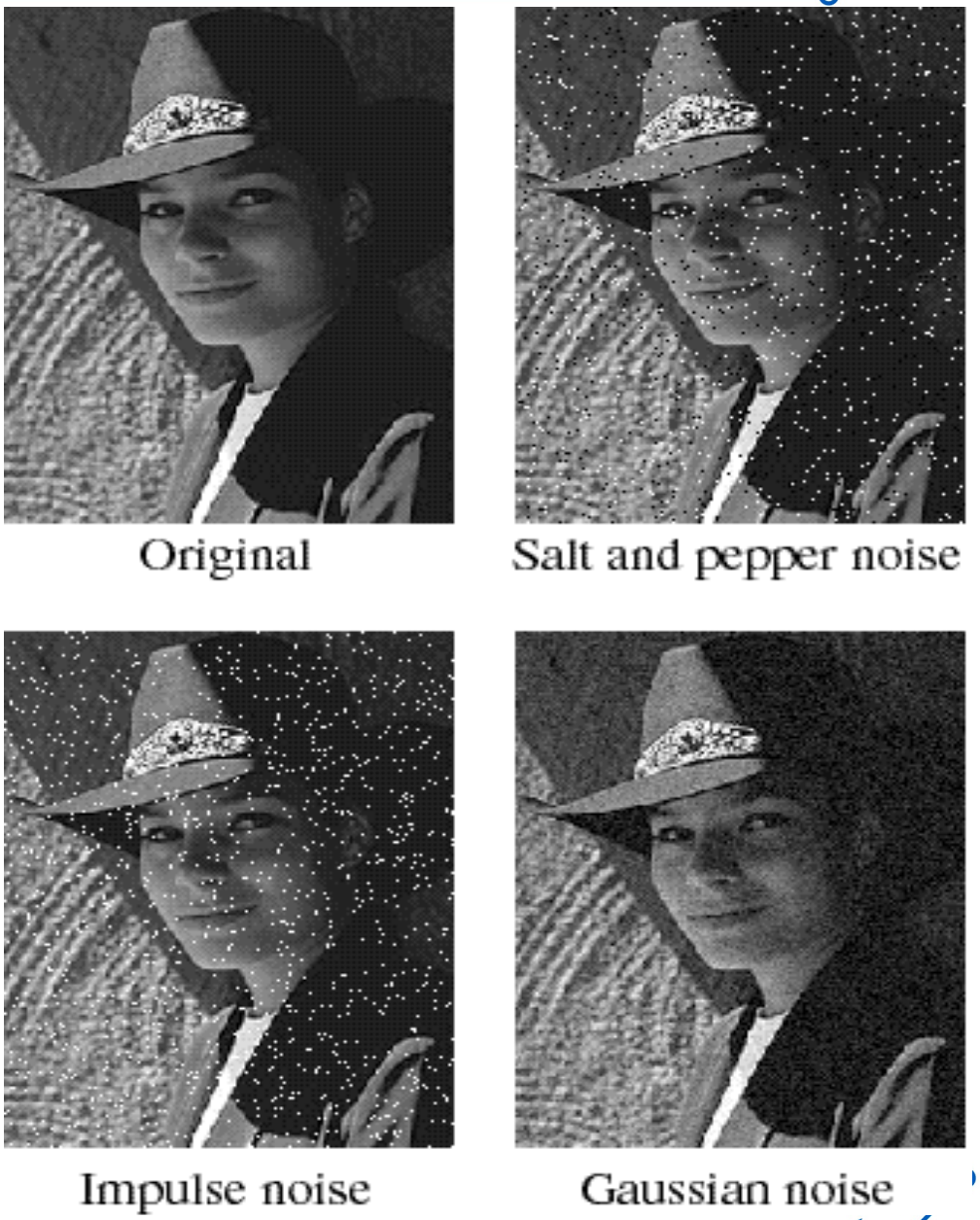






Common types of noise

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



Source: S. Seitz

Gaussian noise

- Additive Noise

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

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

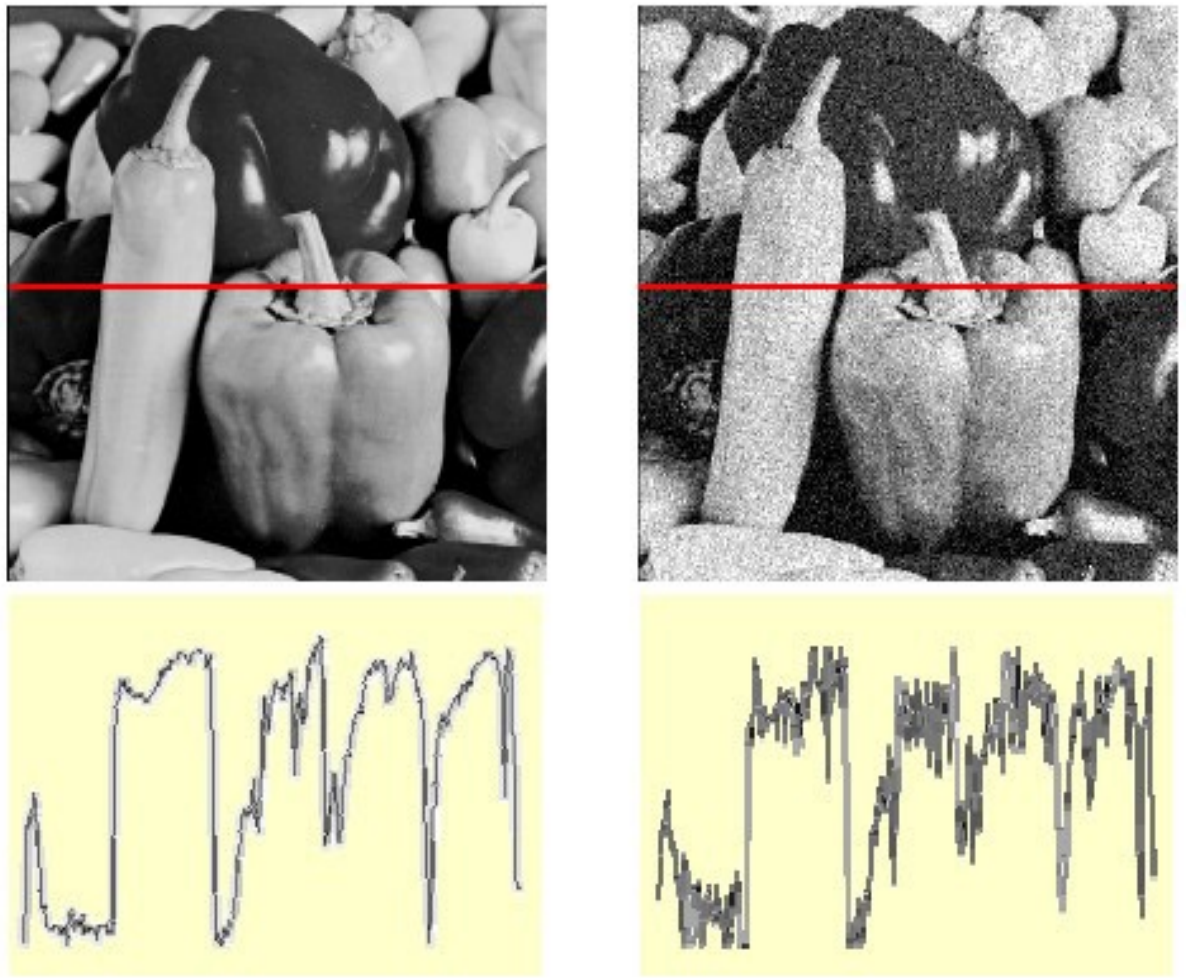
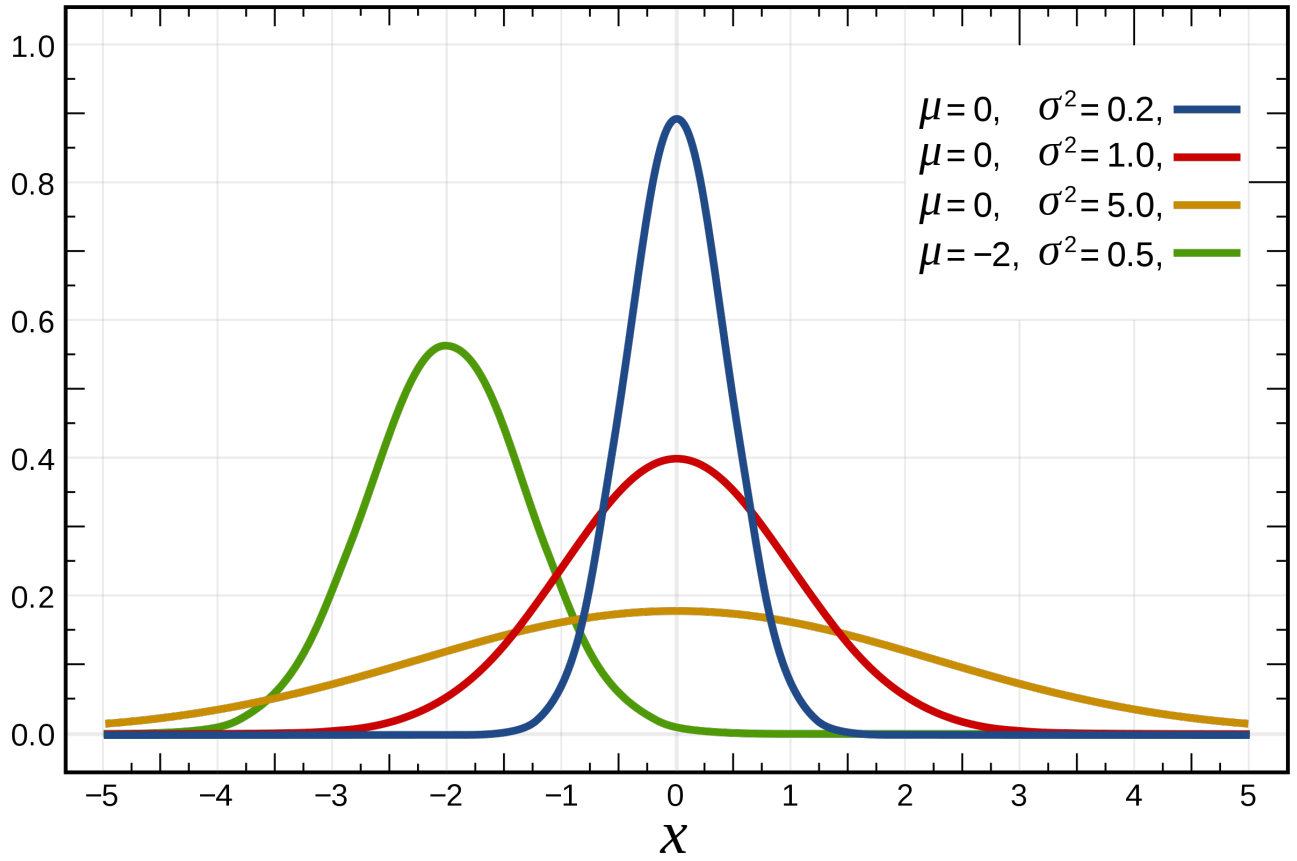


Fig: M. Hebert

PDF of Gaussian distribution

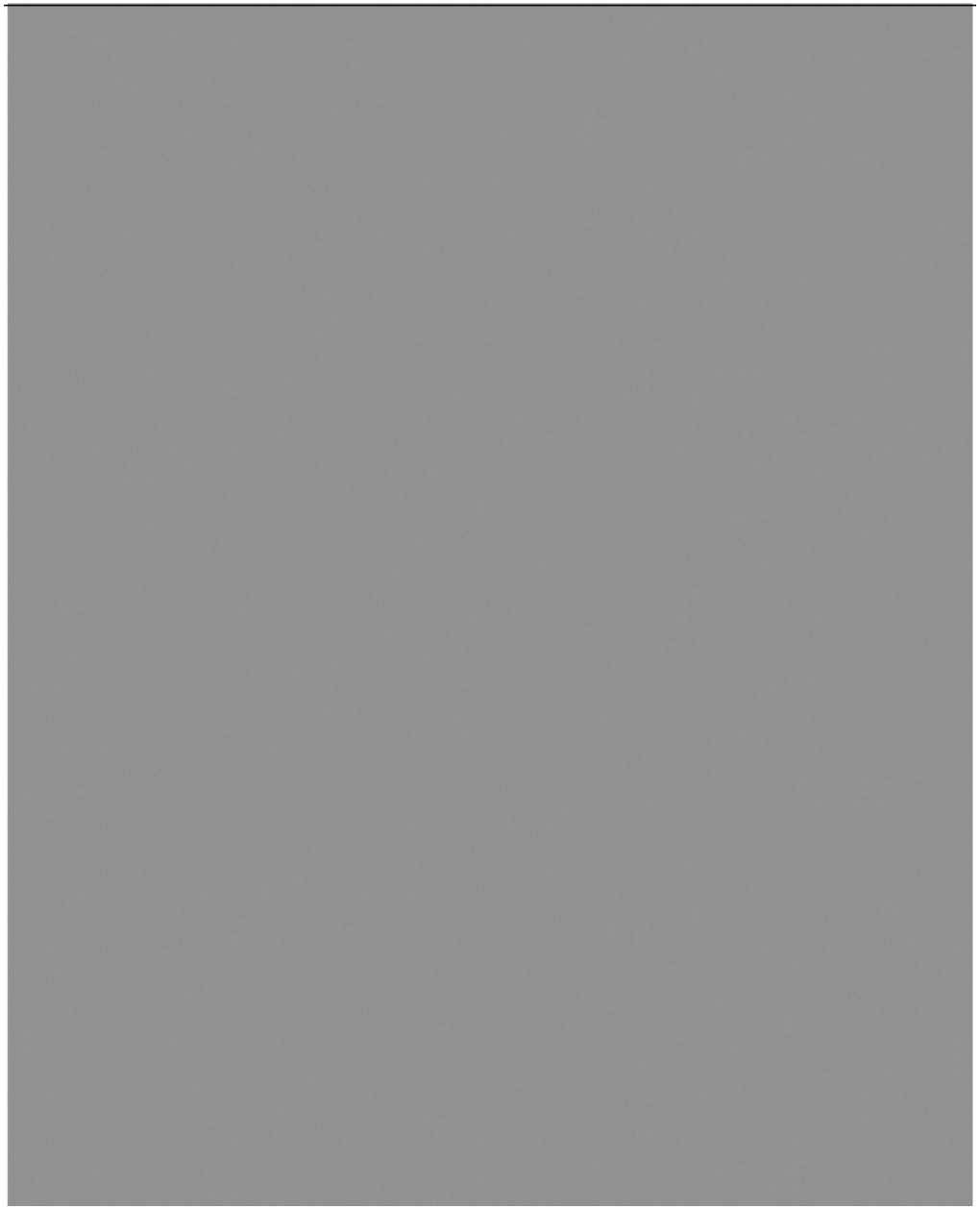
- **Probability density function**



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Gaussian noise

sigma=1

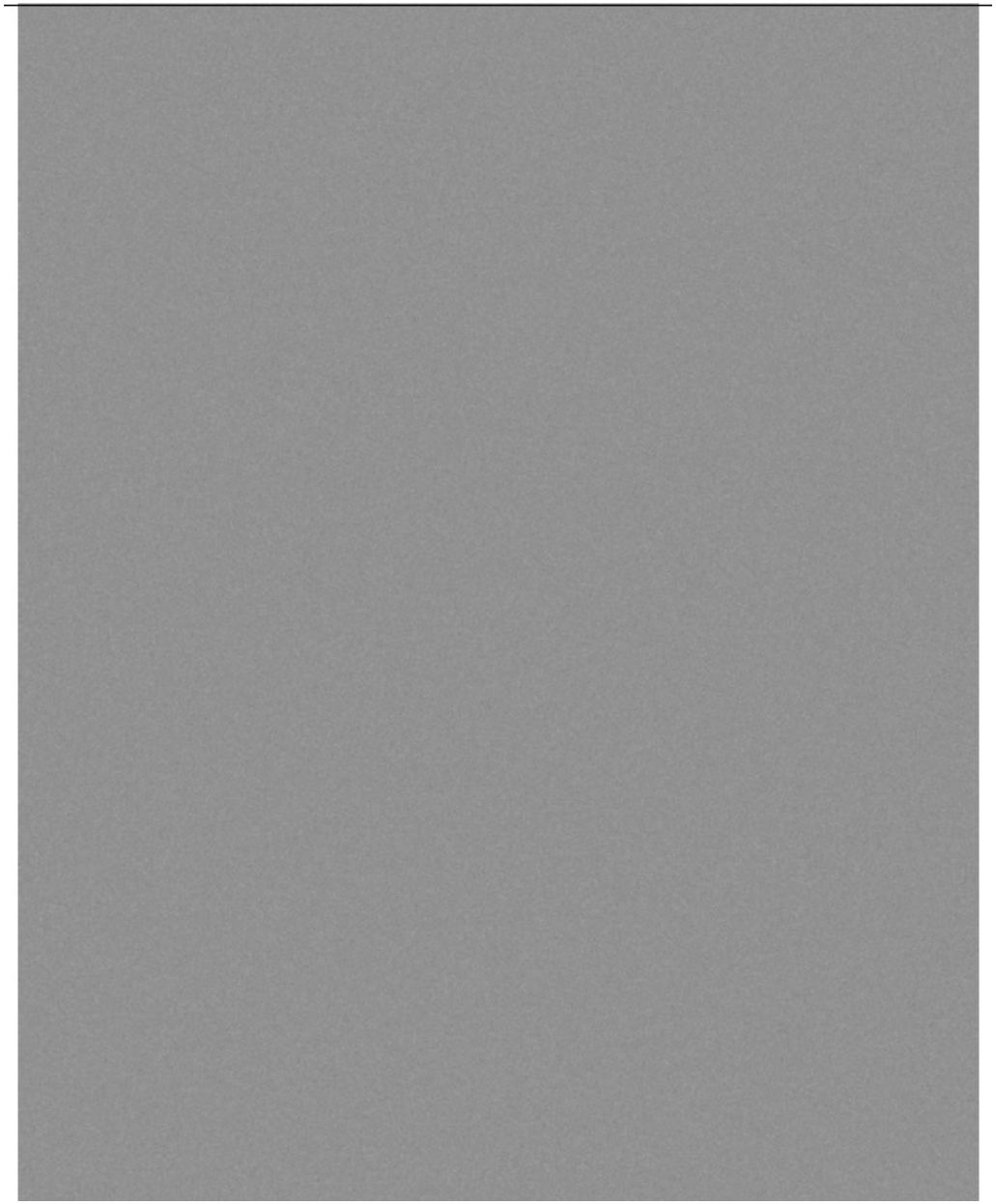


Effect of sigma on Gaussian noise:

Image shows the noise values themselves.

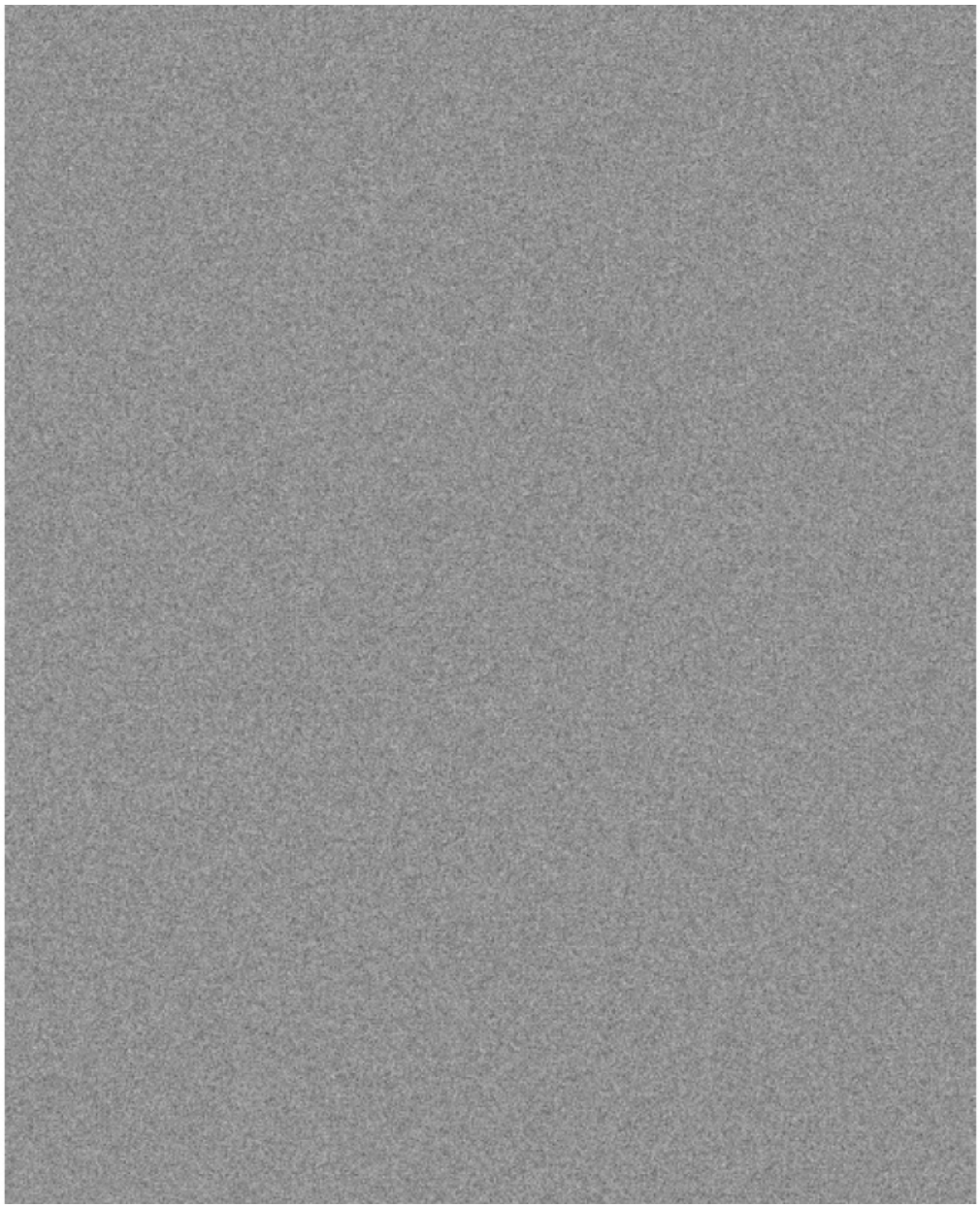
Gaussian noise

sigma=4



Gaussian noise

sigma=16



Gaussian noise

sigma=1

Effect of sigma on Gaussian noise:

This shows the noise values added to the raw intensities of an image.

Gaussian noise

sigma=16

Pixel neighborhoods are important.

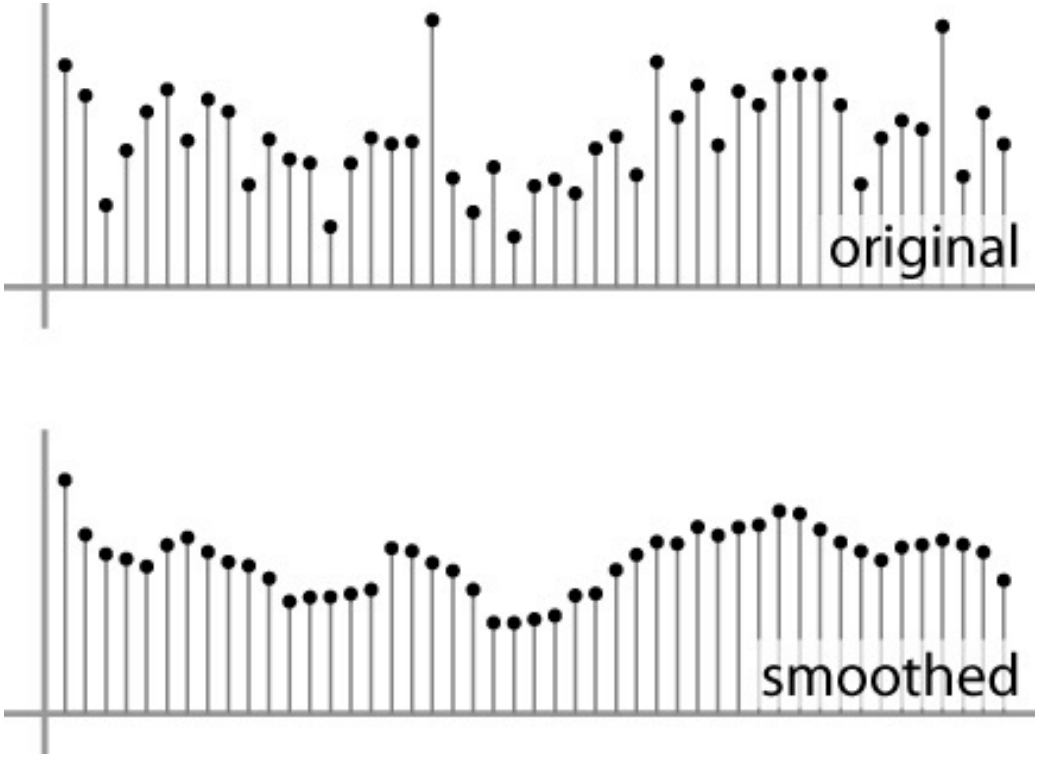
- Q: What happens if we reshuffle all pixels?

- A: Its histogram won't change.
  Point-wise processing unaffected.

- Can we use neighborhoods to remove image noise?

First attempt at a solution

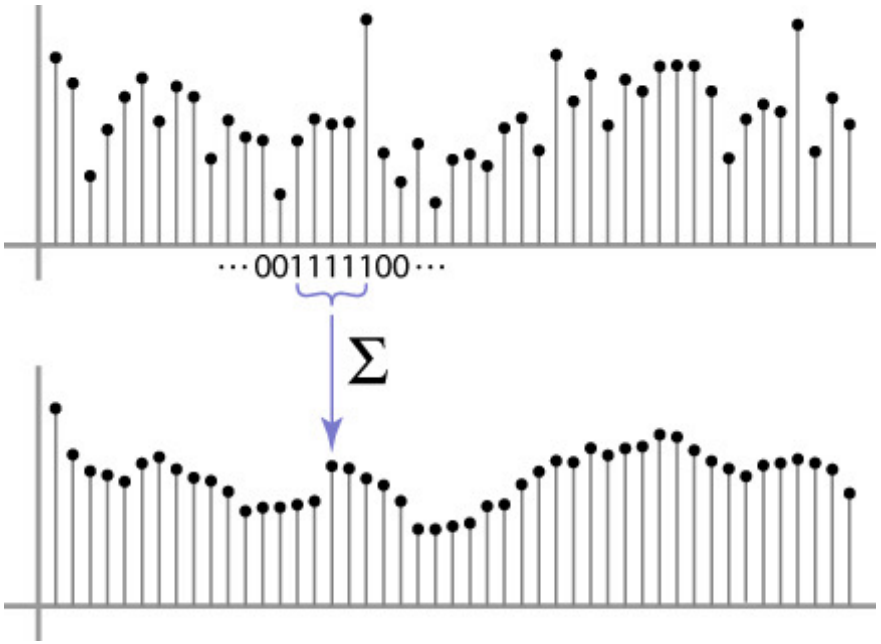
- Let's replace each pixel with an average of all the values in its neighborhood
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel.
  - Moving average in 1D:



Source: S. Marschner

Weighted Moving Average

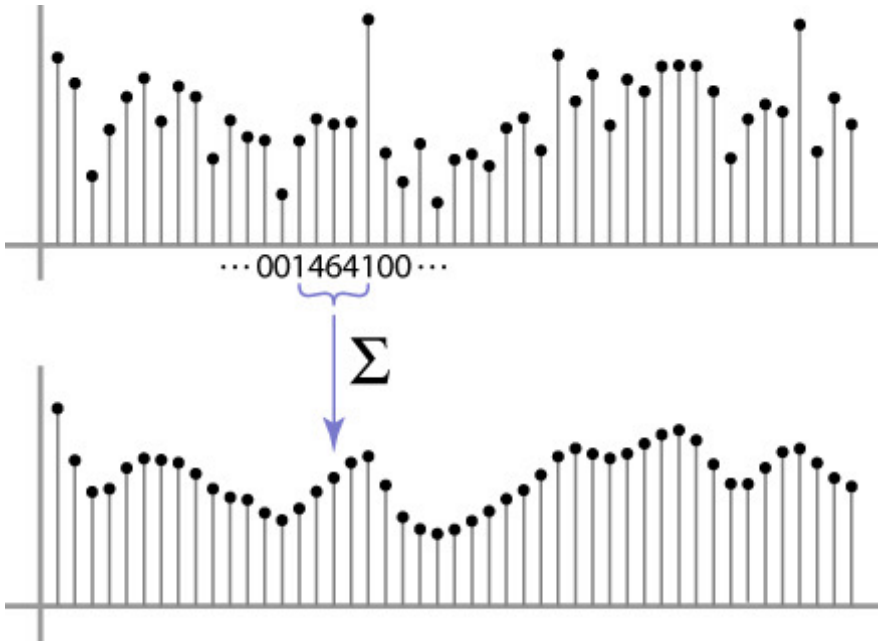
- We can add weights to moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Source: S. Marschner

Weighted Moving Average

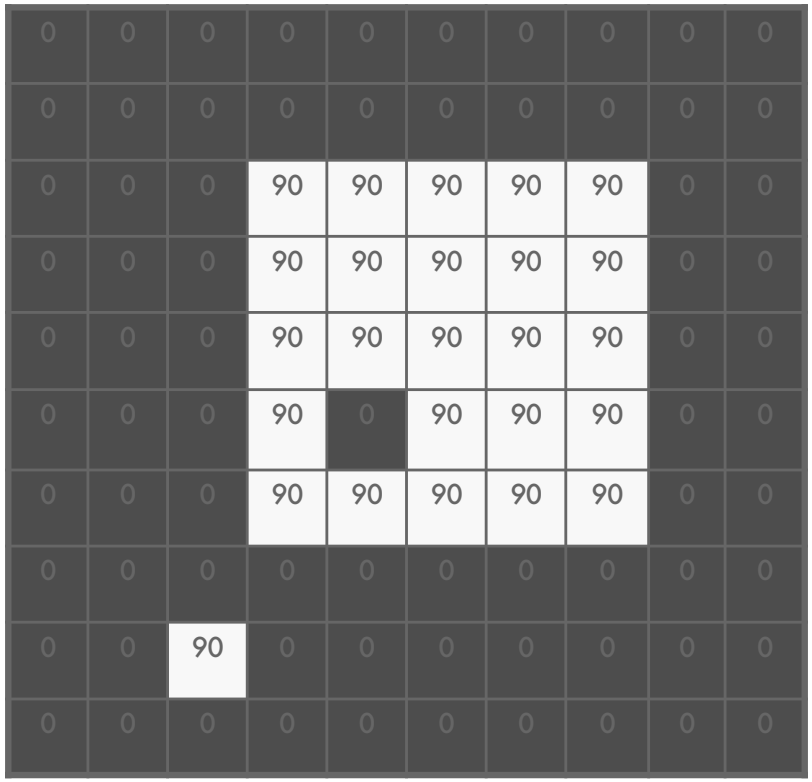
- Non-uniform weights [1, 4, 6, 4, 1] / 16



Source: S. Marschner

Example: Box Filter

$f[.,.]$



$h[\cdot\,,\cdot\,]$

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

Slide credit: David Lowe (UBC)

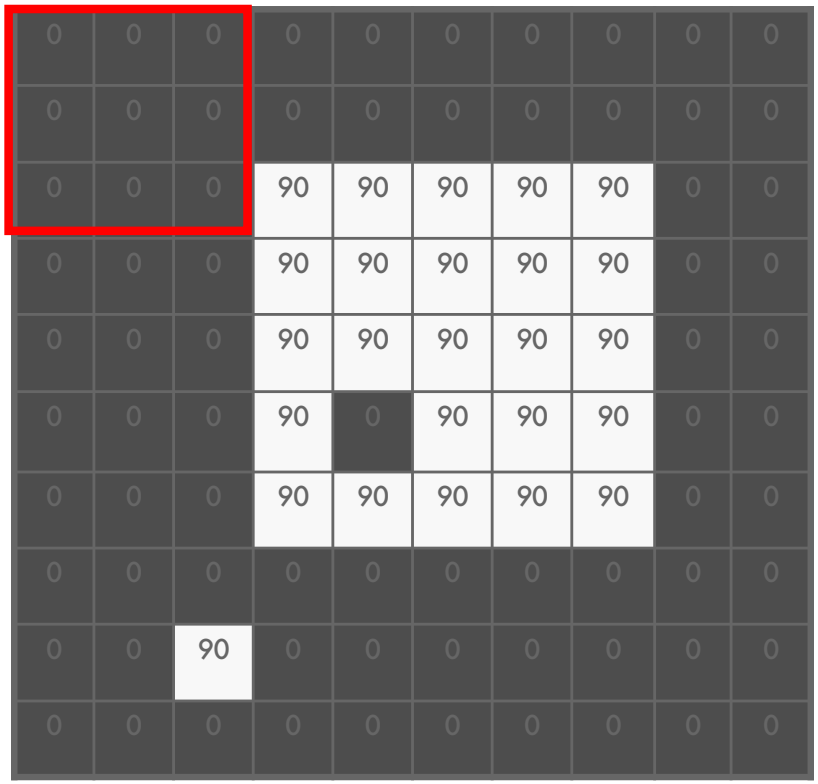
Box Filter

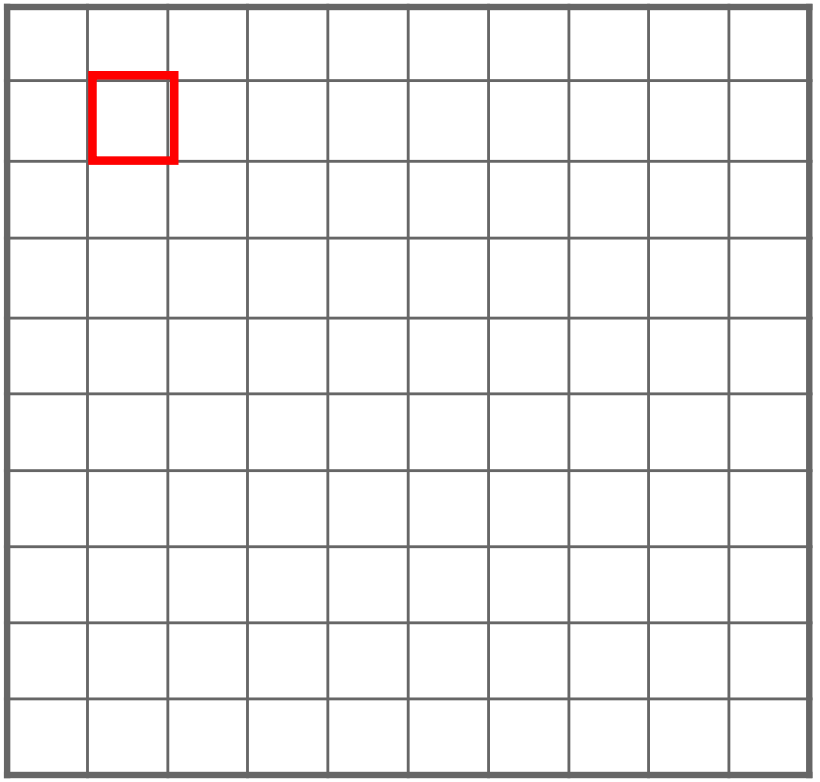
$f[.,.]$

$g[.,.]$

$h[\cdot,\cdot]$

1	1	1
1	1	1
1	1	1



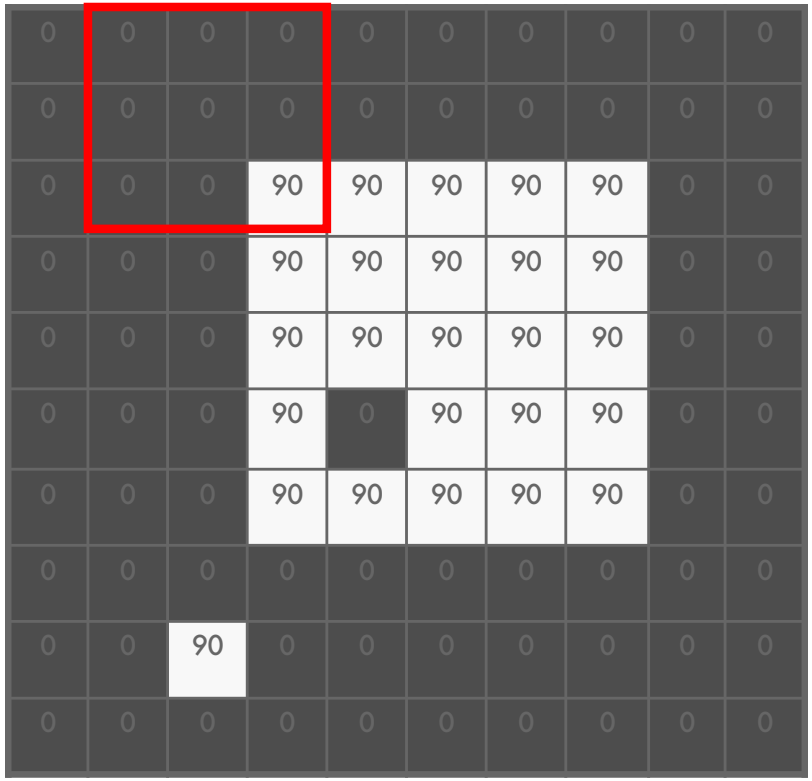


$$g[m,n] = \sum_{k,l} h[k,l]\ f[m+k, n+l]$$

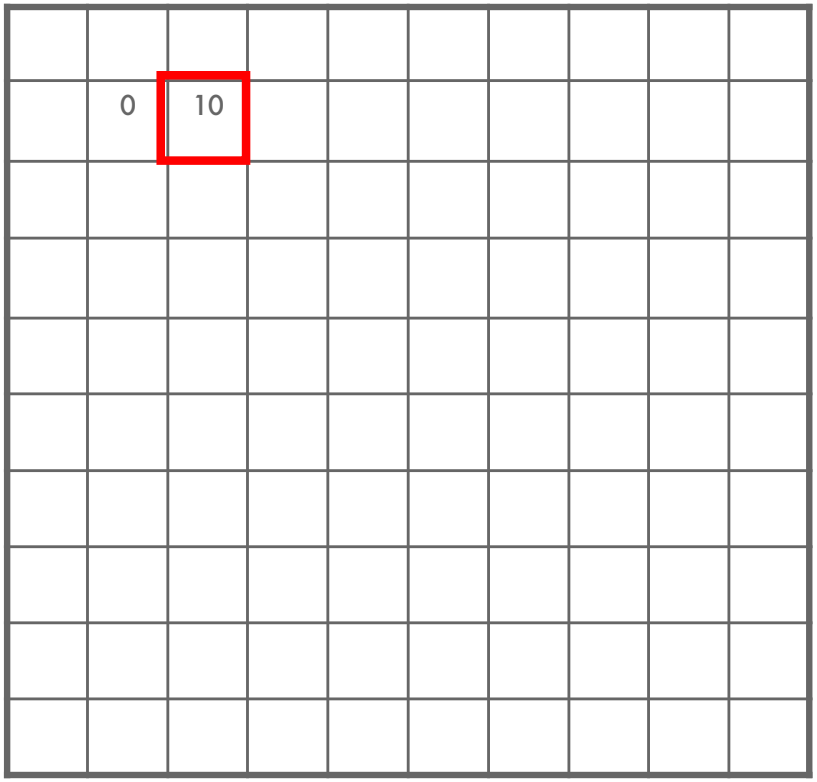
Credit: S. Seitz

Box Filter

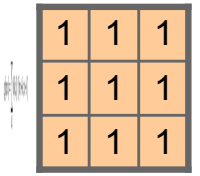
$f[.,.]$



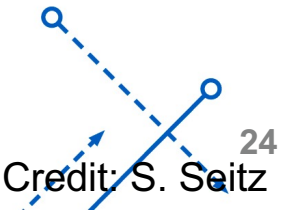
$g[.,.]$



$h[\cdot,\cdot]$



$$g[m,n] = \sum_{k,l} h[k,l]\ f[m+k, n+l]$$



Credit: S. Seitz

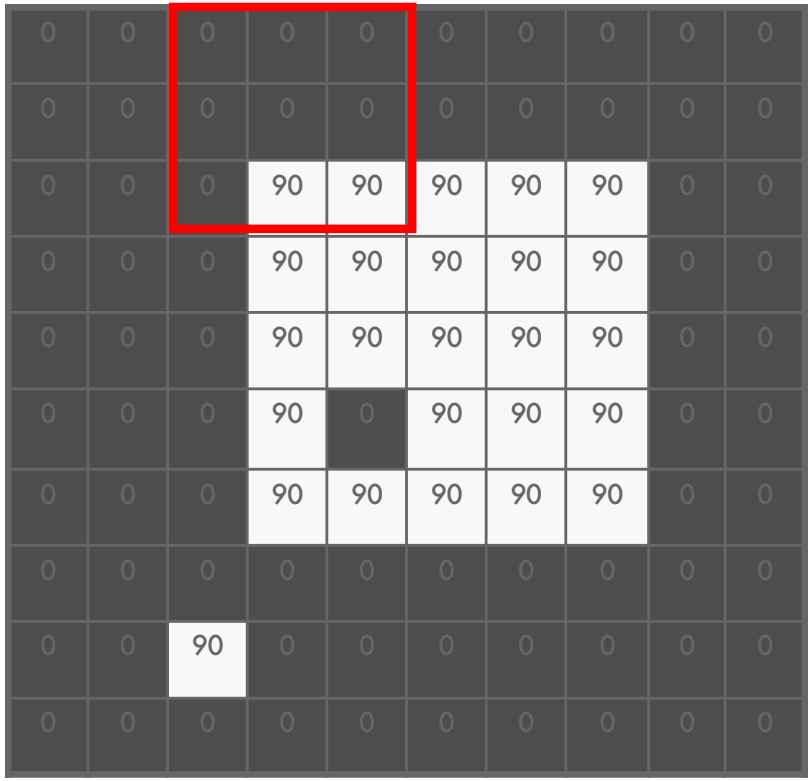
Box Filter

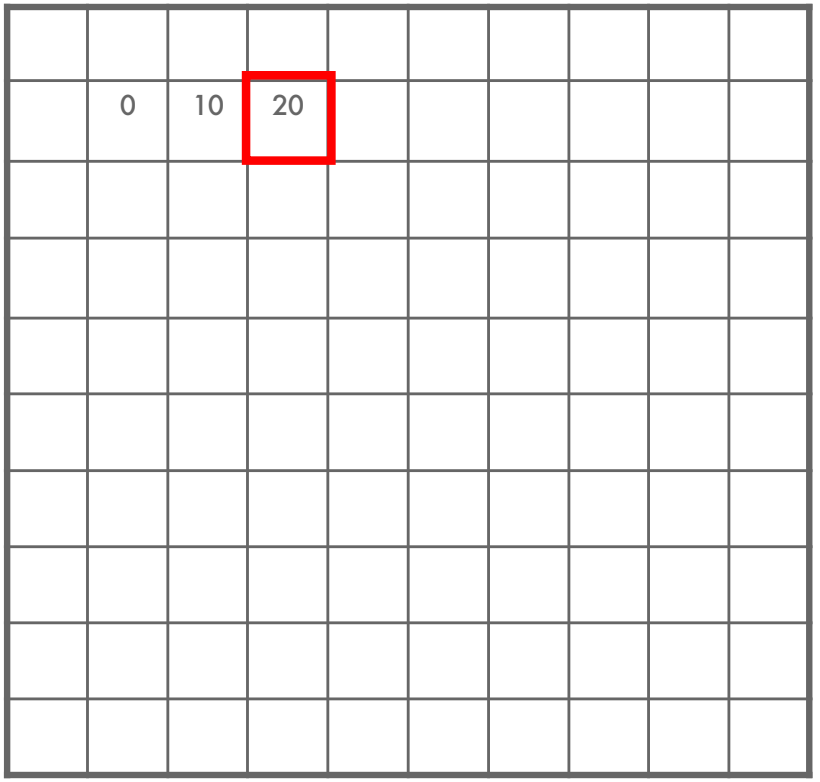
$f[.,.]$

$g[.,.]$

$h[\cdot,\cdot]$

1	1	1
1	1	1
1	1	1



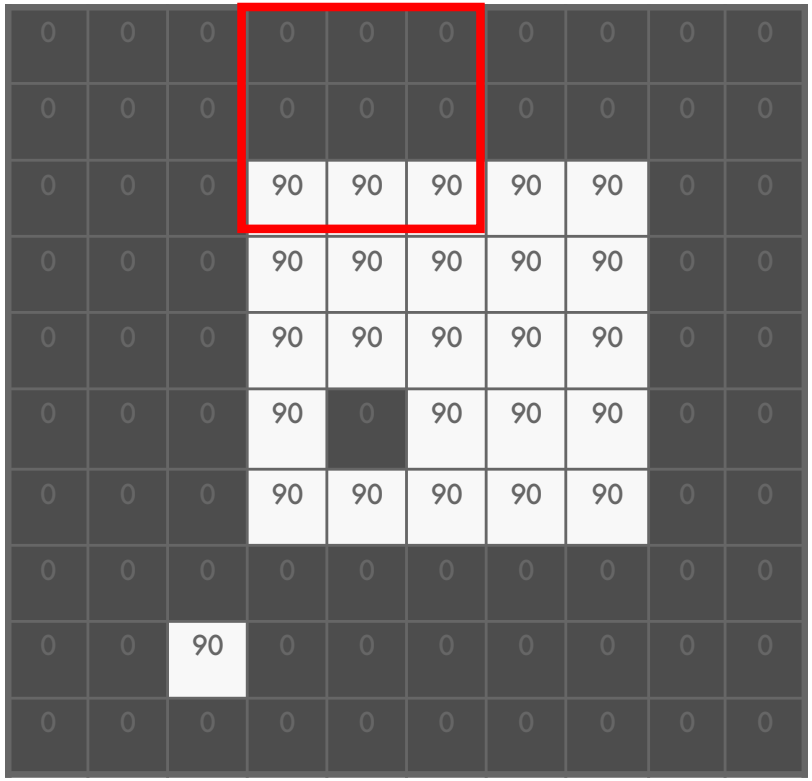


$$g[m,n] = \sum_{k,l} h[k,l]\ f[m+k, n+l]$$

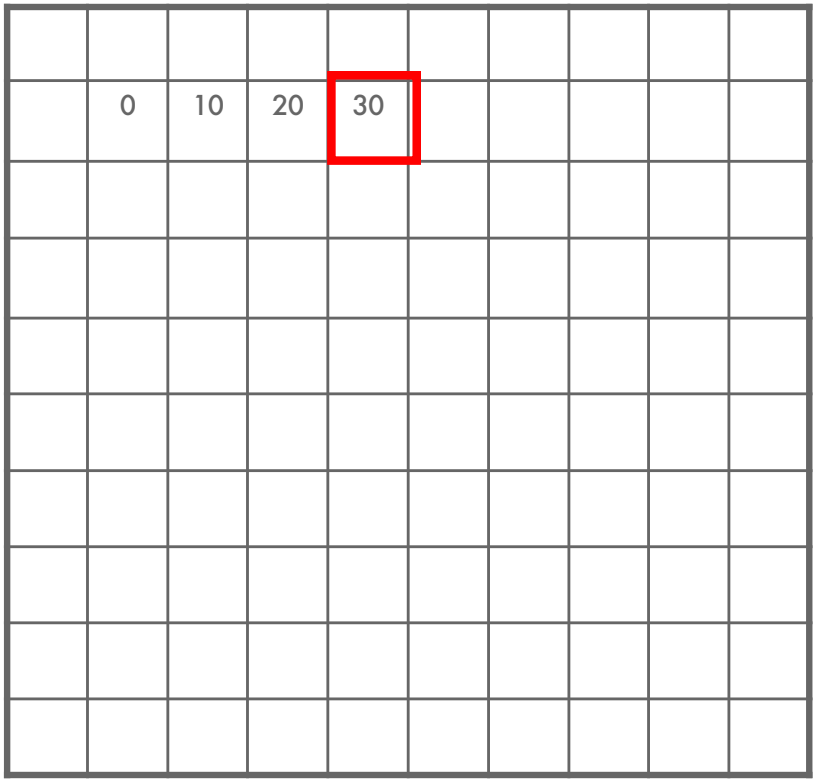
Credit: S. Seitz

Box Filter

$f[.,.]$



$g[.,.]$



$h[\cdot,\cdot]$

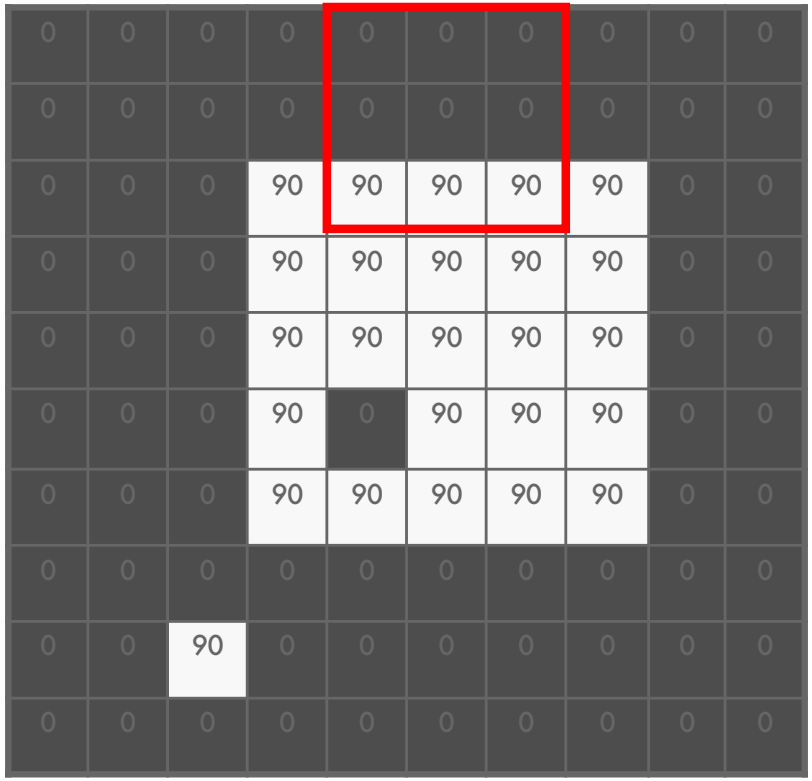
1	1	1
1	1	1
1	1	1

$$g[m,n] = \sum_{k,l} h[k,l]\ f[m+k, n+l]$$

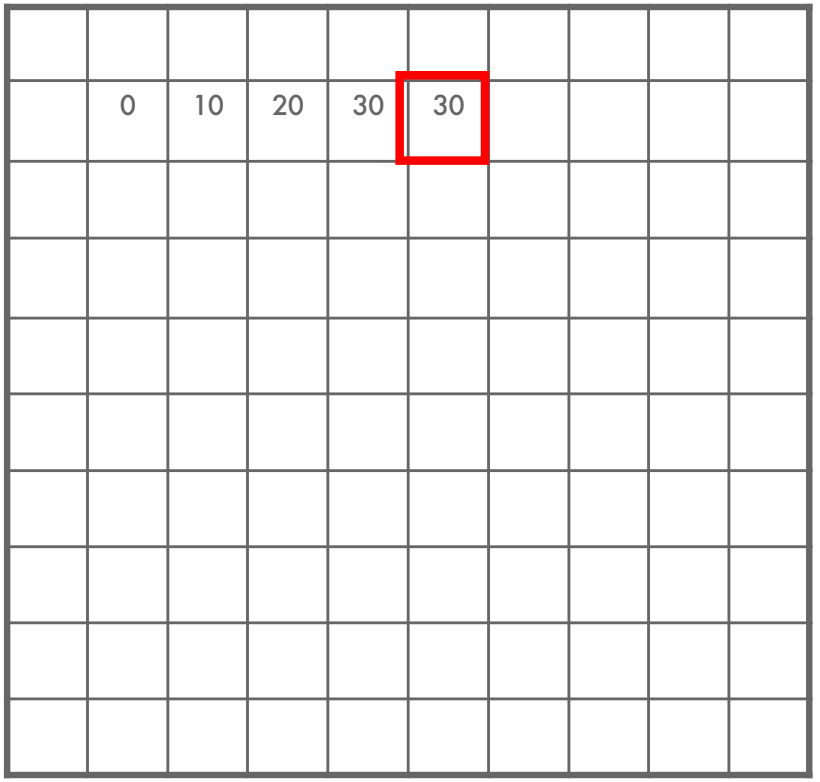
Credit: S. Seitz

Box Filter

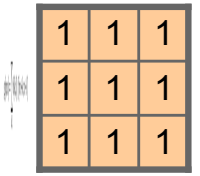
$f[.,.]$



$g[.,.]$



$h[\cdot,\cdot]$



$$g[m,n] = \sum_{k,l} h[k,l]\ f[m+k, n+l]$$

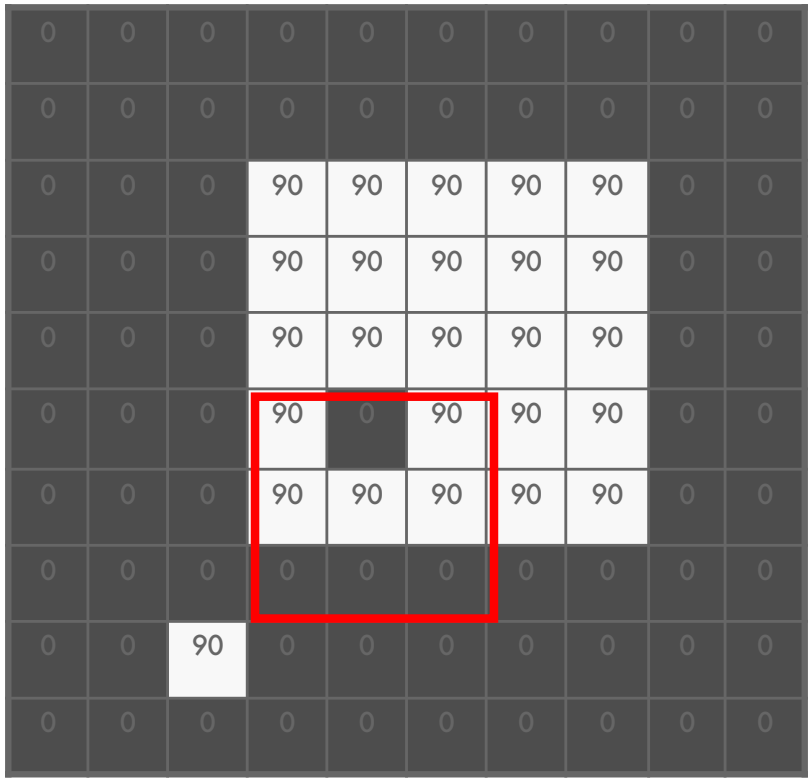
Box Filter

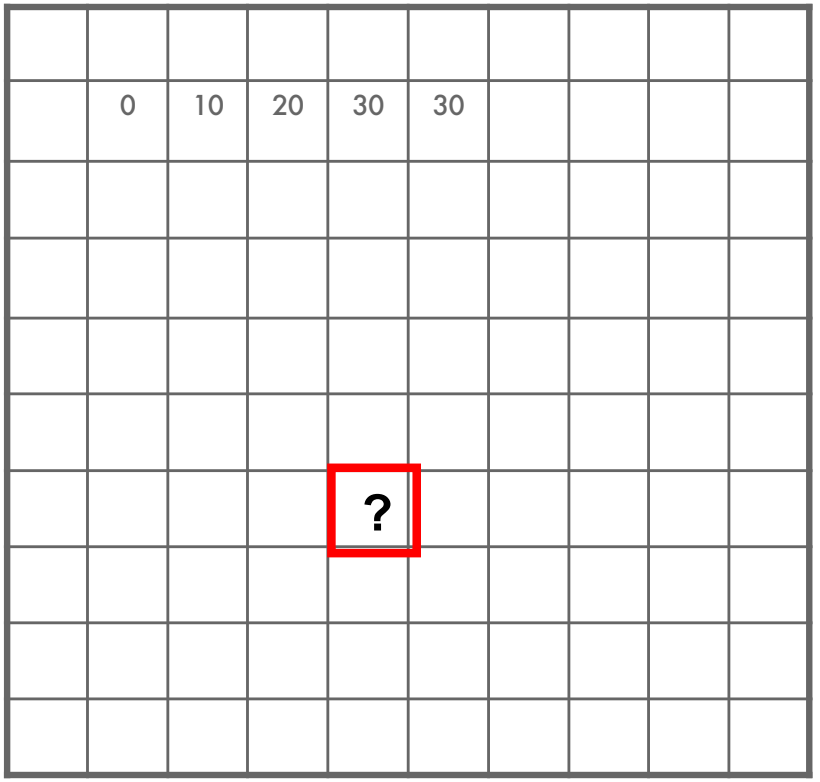
$f[.,.]$

$g[.,.]$

$h[\cdot,\cdot]$

1	1	1
1	1	1
1	1	1





$$g[m,n] = \sum_{k,l} h[k,l]\ f[m+k, n+l]$$

Credit: S. Seitz

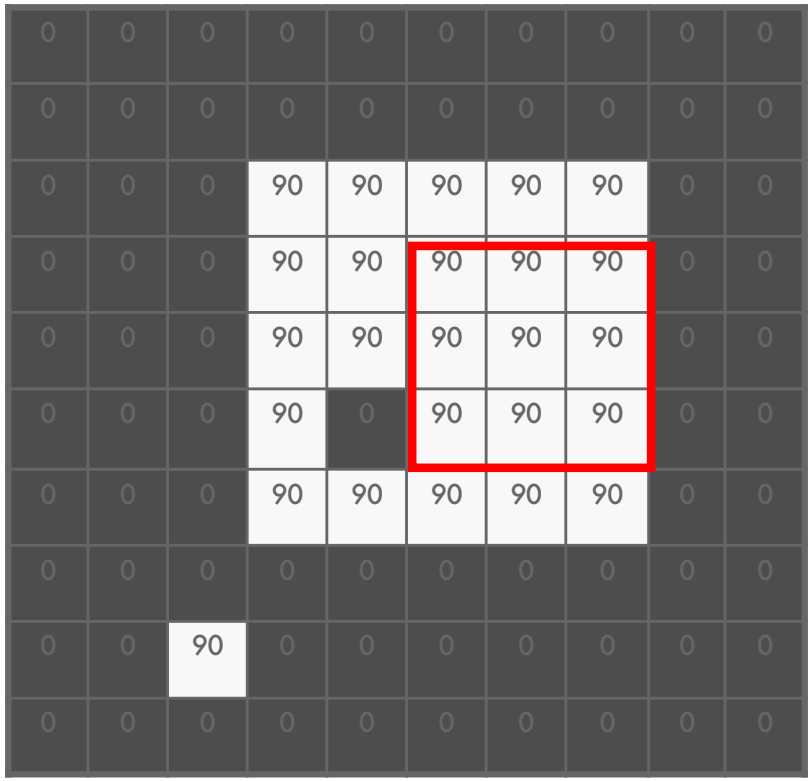
Box Filter

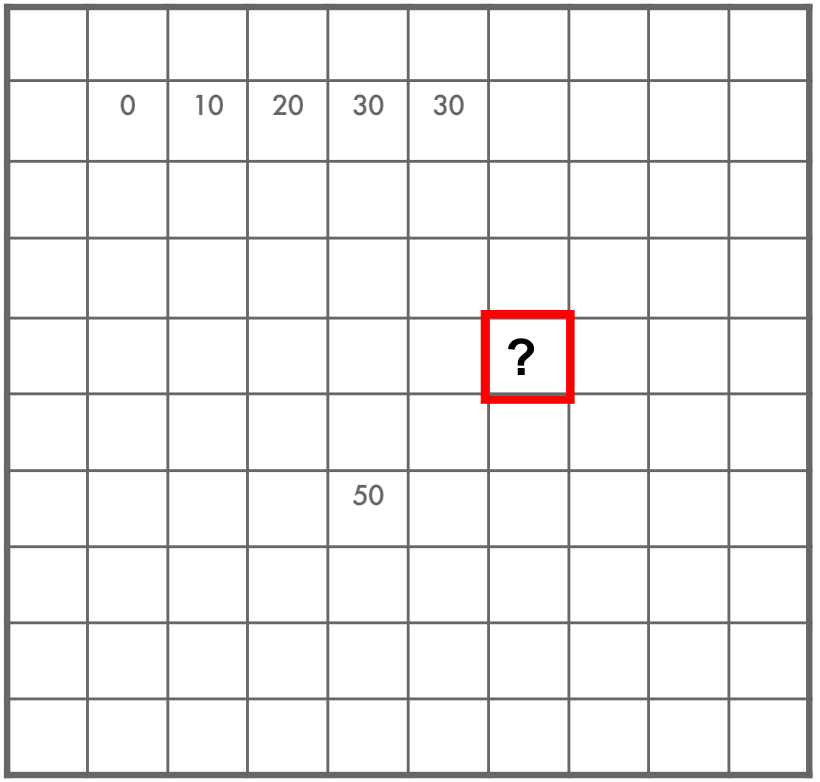
$f[.,.]$

$g[.,.]$

$h[\cdot,\cdot]$

1	1	1
1	1	1
1	1	1





$$g[m,n] = \sum_{k,l} h[k,l]\ f[m+k, n+l]$$

Credit: S. Seitz

Box Filter

$f[.,.]$

$g[.,.]$

$h[\cdot,\cdot]$

1	1	1
1	1	1
1	1	1

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

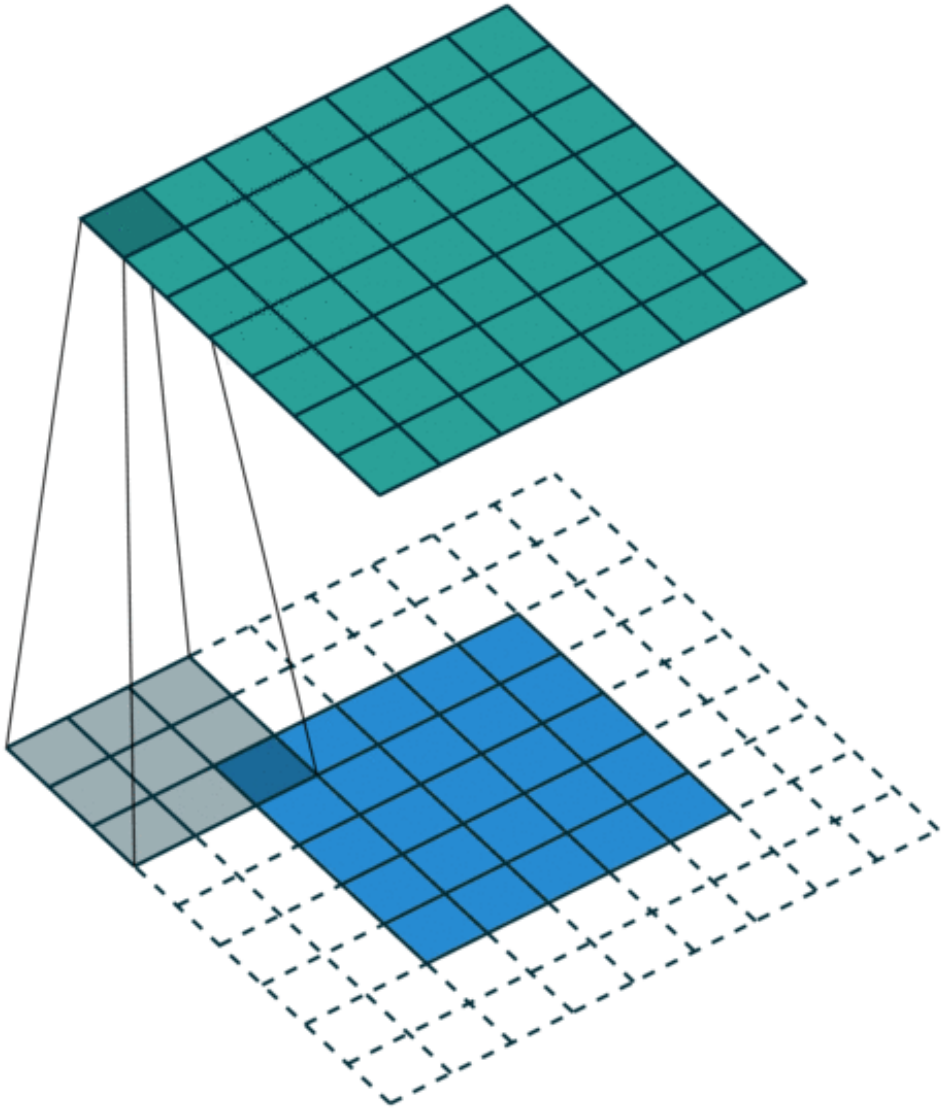
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$g[m,n] = \sum_{k,l} h[k,l]\ f[m+k, n+l]$$

Credit: S. Seitz

Correlation

- Correlation is the process of moving a filter mask over an image
- At each point in the image, one computes the sum of products at each location
- Filter is often referred to as the **Kernel or Mask.**
- Correlation is a function of the displacement



Box Filter

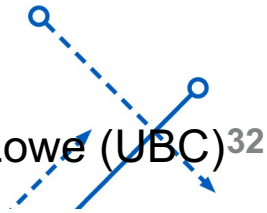
What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

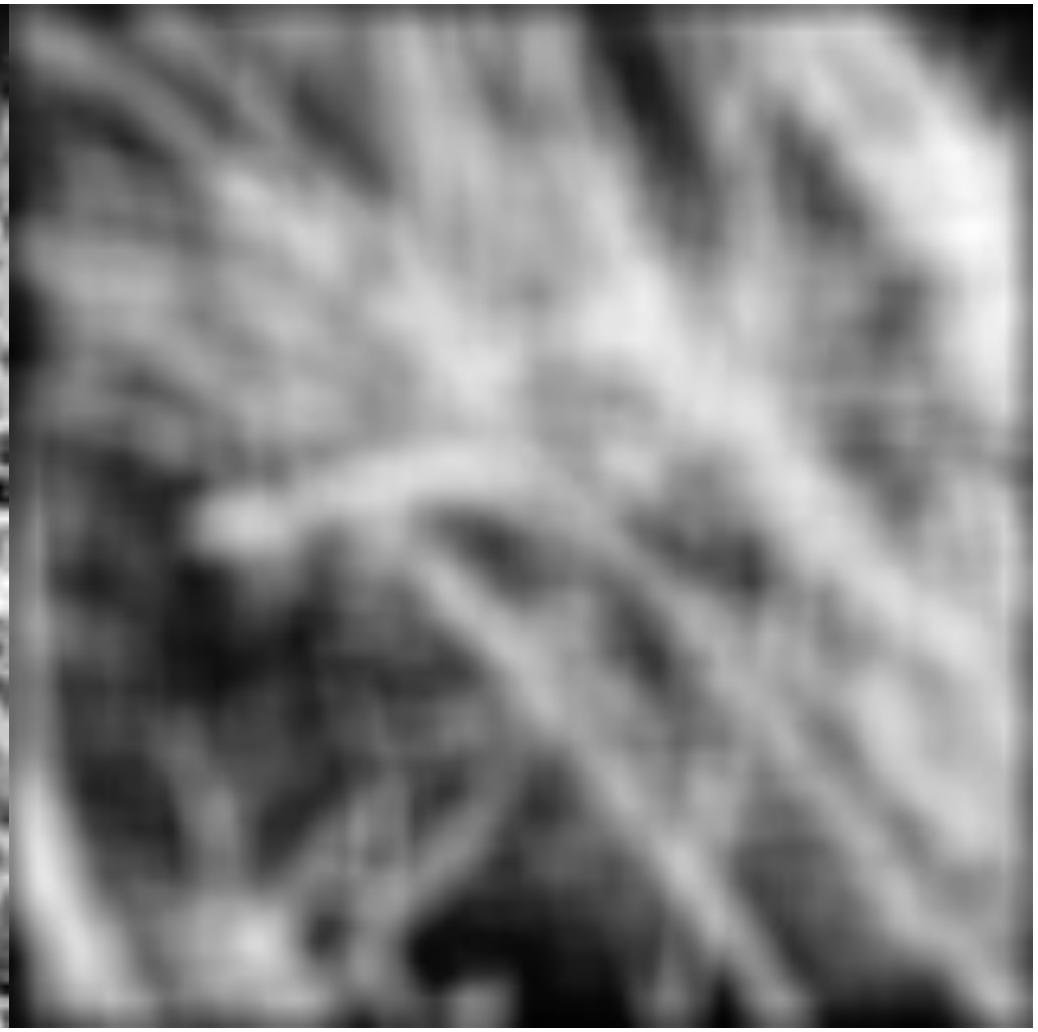
$h[\cdot\,,\cdot\,]$

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

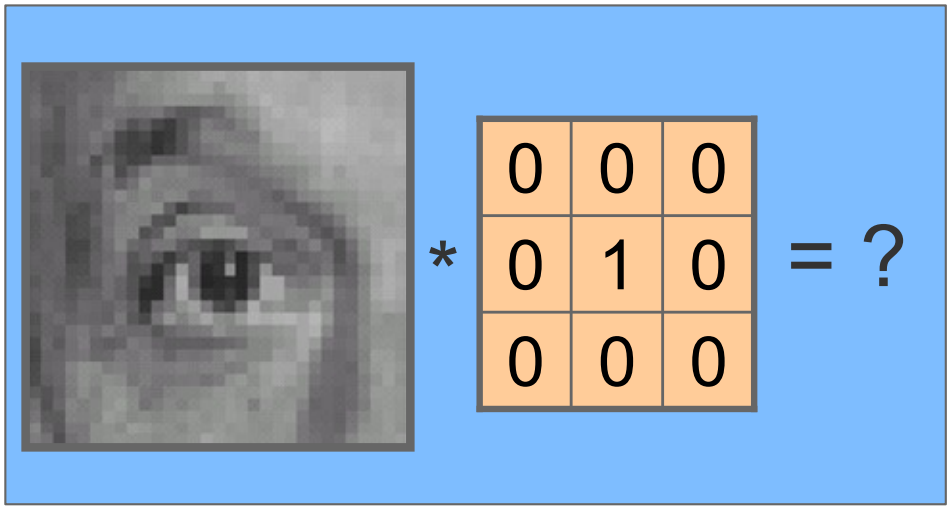


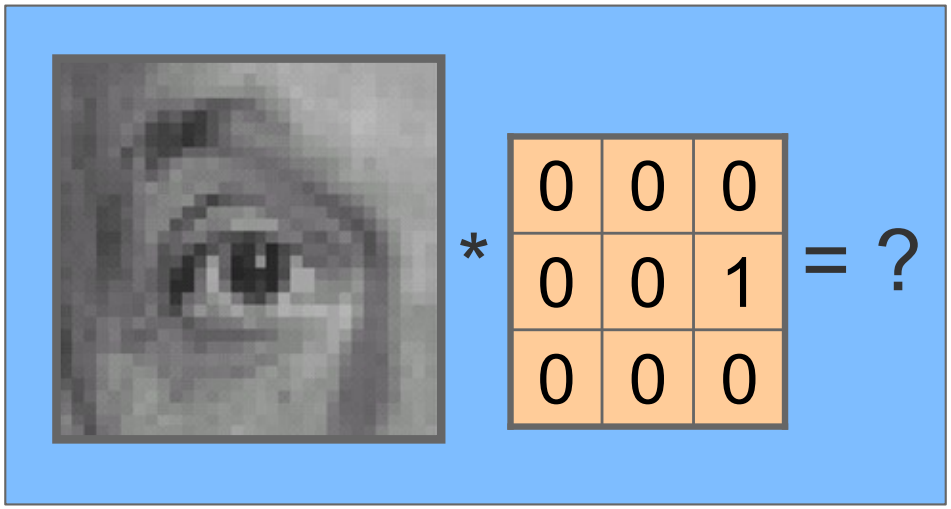
Slide credit: David Lowe (UBC)

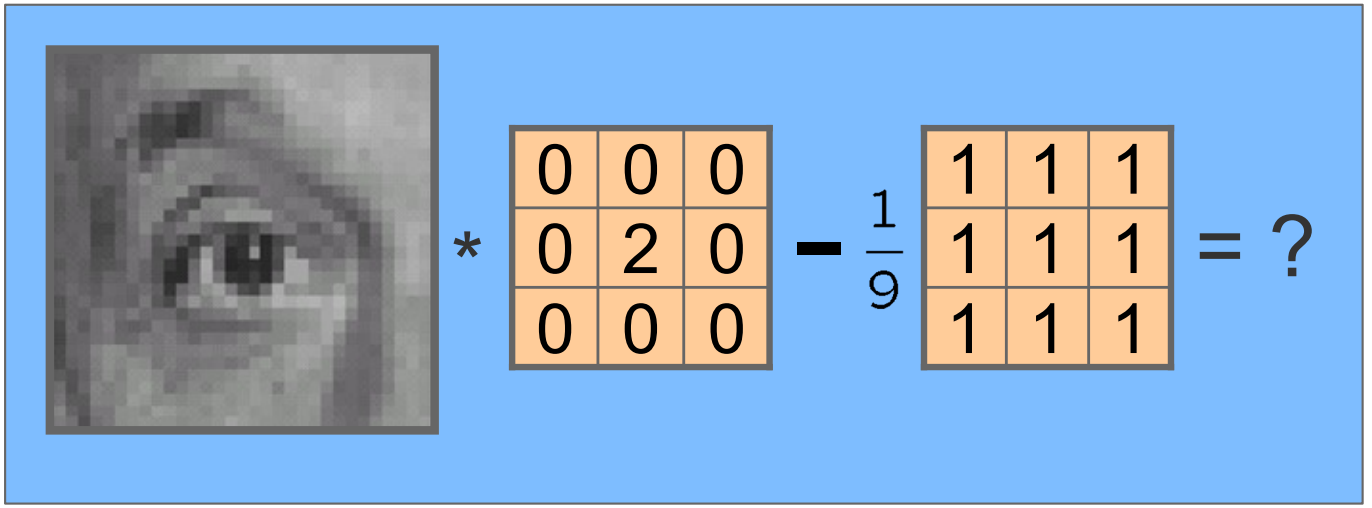
Smoothing with box filter



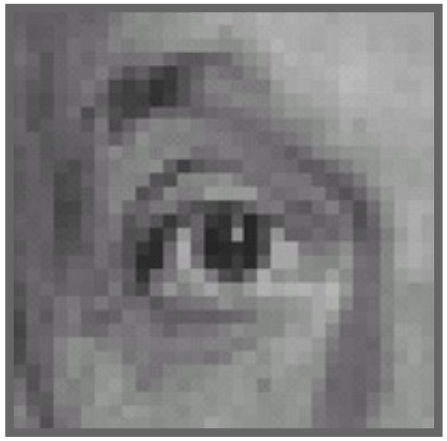
Predict the filtered outputs







Practice with linear filters



Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

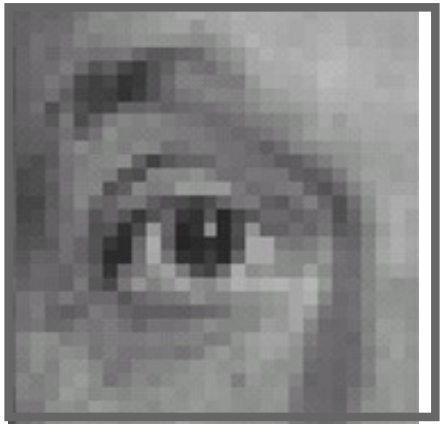
?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

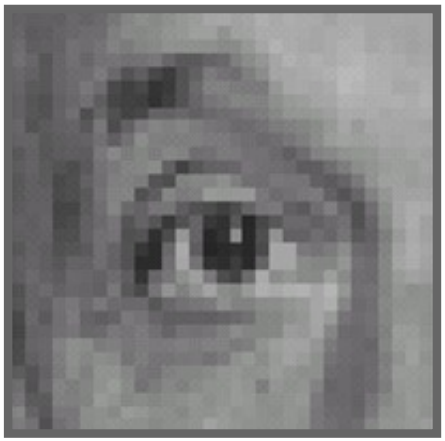
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

(Note that filter sums to 1)

Source: D. Lowe

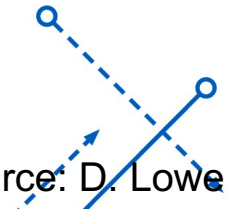
Practice with linear filters



Original

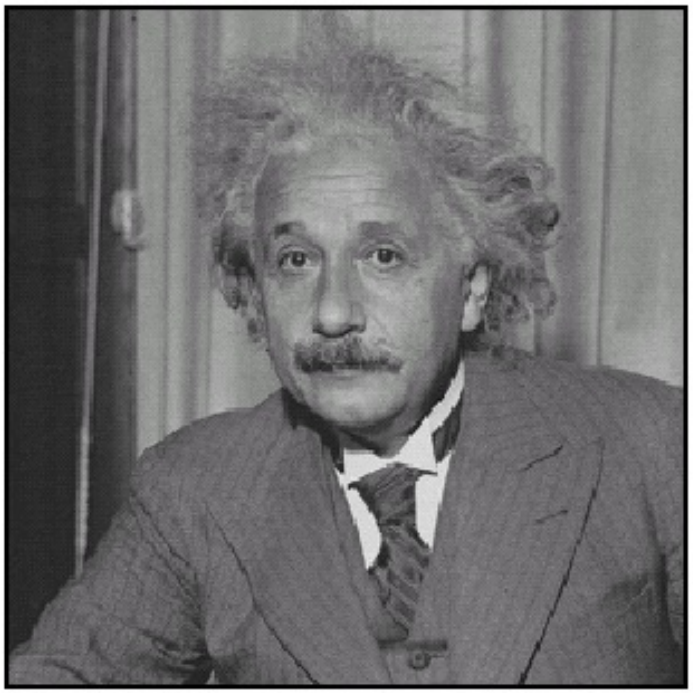
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter
Accentuates differences with local average

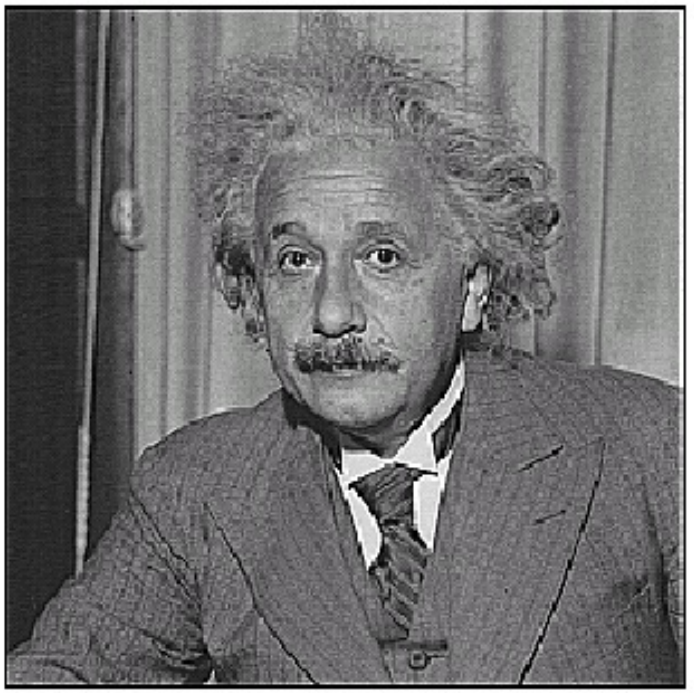


Source: D. Lowe

Sharpening



before



after

Source: D. Lowe

Correlation (another name for filtering)

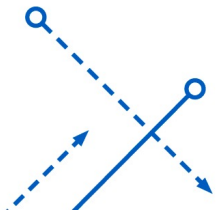
H=Filter F=Signal

$$G[m,n] = \sum_{k,l} H[k,l]\ F[m+k, n+l]$$

- 2D Convolution
  - PyTorch implements convolution as correlation
  - `import torch.nn.functional as F`
  - `G = F.conv2d(I, f)`

I=image f=filter

$$G[m,n] = \sum_{k,l} H[k,l]\ F[m-k, n-l]$$



Correlation vs. Convolution

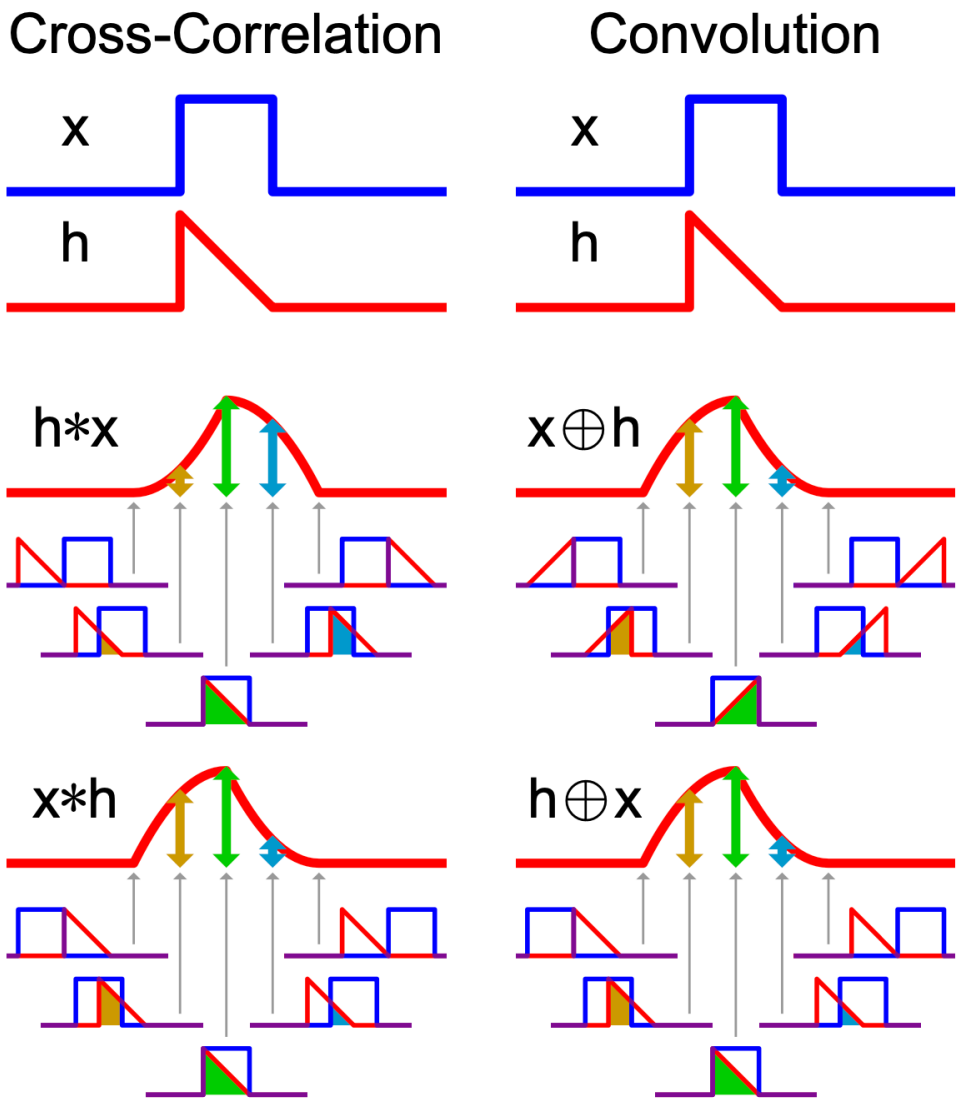
- 2D Correlation

$$h[m,n] = \sum_{k,l} f[k,l]\ \ I[m+k, n+l]$$

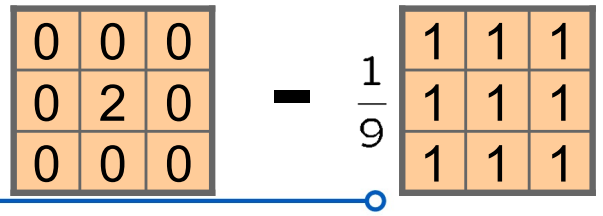
- 2D Convolution

$$h[m,n] = \sum_{k,l} f[k,l]\ \ I[m-k, n-l]$$

- 1D Case



Correlation filtering



Say the averaging window size is (2k+1) x (2k+1):

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood around image pixel F[i,j]}}$$

Now generalize to allow different weights depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Correlation filtering

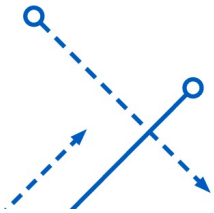
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

More formally, it is called cross-correlation, i.e.,

$$G = H \otimes F$$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter is also called “kernel”, “mask”, or a “window”.



More Formal/Strict Terminology

- Cross-correlation:
  - Often referred as **correlation** in computer vision.

$$G[m,n] = \sum_{k,l} H[k,l] \;\; F[m+k, n+l]$$

- Correlation (in statistics):

$$\operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

- Reading more:
  - Chen Wang, et al. "Kernel Cross-correlator." In Proceedings of the AAAI Conference on Artificial Intelligence, 2018.
  - Convolution vs Cross-correlation
    - Section 2.2 in “Chen Wang, Kernel learning for visual perception”. PhD thesis.

Properties

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

- Commutative:

$$\boldsymbol{x} \otimes \boldsymbol{y} = \boldsymbol{y} \otimes \boldsymbol{x}$$

  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality

- Associative:

$$\boldsymbol{x} \otimes (\boldsymbol{y} \otimes \boldsymbol{z}) = (\boldsymbol{x} \otimes \boldsymbol{y}) \otimes \boldsymbol{z}$$

  - Often apply several filters one after another: $\boldsymbol{x} \otimes \boldsymbol{y_1} \otimes \boldsymbol{y_2} \otimes \boldsymbol{y_3}$
  - This is equivalent to applying one filter: $\boldsymbol{x} \otimes (\boldsymbol{y_1} \otimes \boldsymbol{y_2} \otimes \boldsymbol{y_3})$

- Distributes over addition:

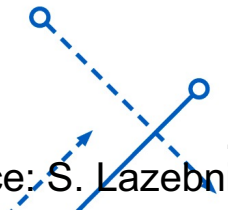
$$\boldsymbol{x} \otimes (\boldsymbol{y} + \boldsymbol{z}) = (\boldsymbol{x} \otimes \boldsymbol{y}) + (\boldsymbol{x} \otimes \boldsymbol{z})$$

- Scalars factor out:

$$a\boldsymbol{x} \otimes \boldsymbol{y} = \boldsymbol{x} \otimes a\boldsymbol{y} = a(\boldsymbol{x} \otimes \boldsymbol{y})$$

- Identity (unit impulse), e.g., *e* = [0, 0, 1, 0, 0],

$$\boldsymbol{x} \otimes \boldsymbol{e} = \boldsymbol{x}$$



Source: S. Lazebnik

Key properties of Linear Filters

- Assume $f(\boldsymbol{x})$ is image filtering: $f(\boldsymbol{x}) = \boldsymbol{x} \otimes \boldsymbol{w}$.
- Linearity (superposition property):

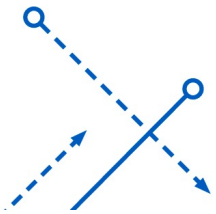
$$f(a \cdot \boldsymbol{x} + b \cdot \boldsymbol{y}) = af(\boldsymbol{x}) + bf(\boldsymbol{y})$$

- Linear filter is **equivariant** (not **invariant)** to translation.
- Assume $T(\boldsymbol{x})$ is image shift (translation):
  - **Equivariance**:

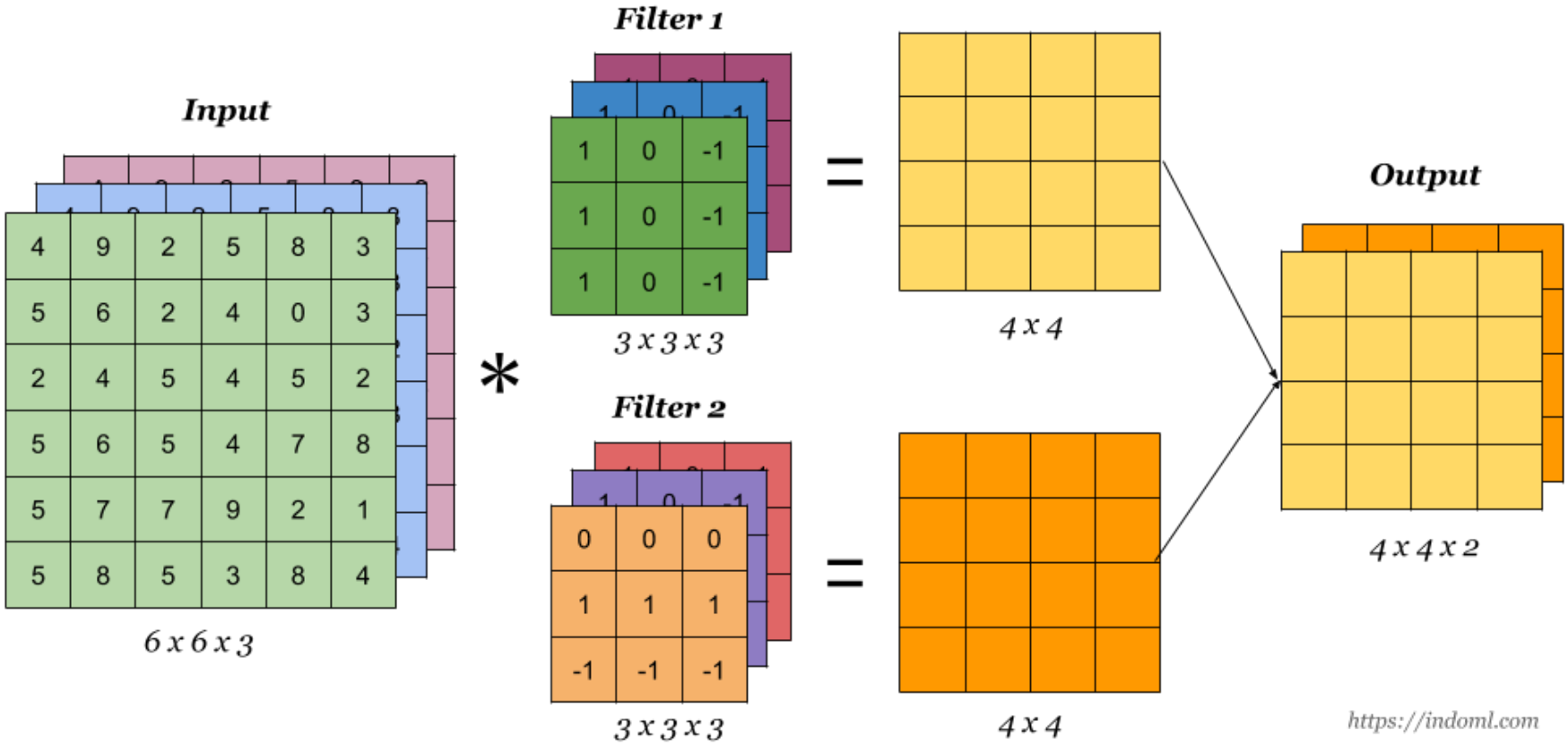
$$f(T(X)) = T\big(f(x)\big)$$

  - **Invariance**:

$$f(T(X)) = f(x)$$



Multi-channel correlation



Examples:

```
>>> # With square kernels and equal stride
>>> filters = torch.randn(8, 4, 3, 3)
>>> inputs = torch.randn(1, 4, 5, 5)
>>> F.conv2d(inputs, filters, padding=1)
```

What is shape of the output in the left example?

(1, 8, 5, 5)

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

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

$F[x, y]$

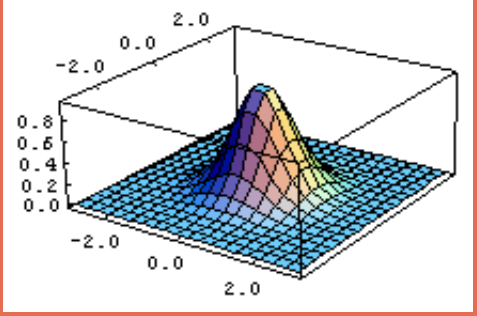
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$H[u, v]$

This kernel is an approximation of a Gaussian function:

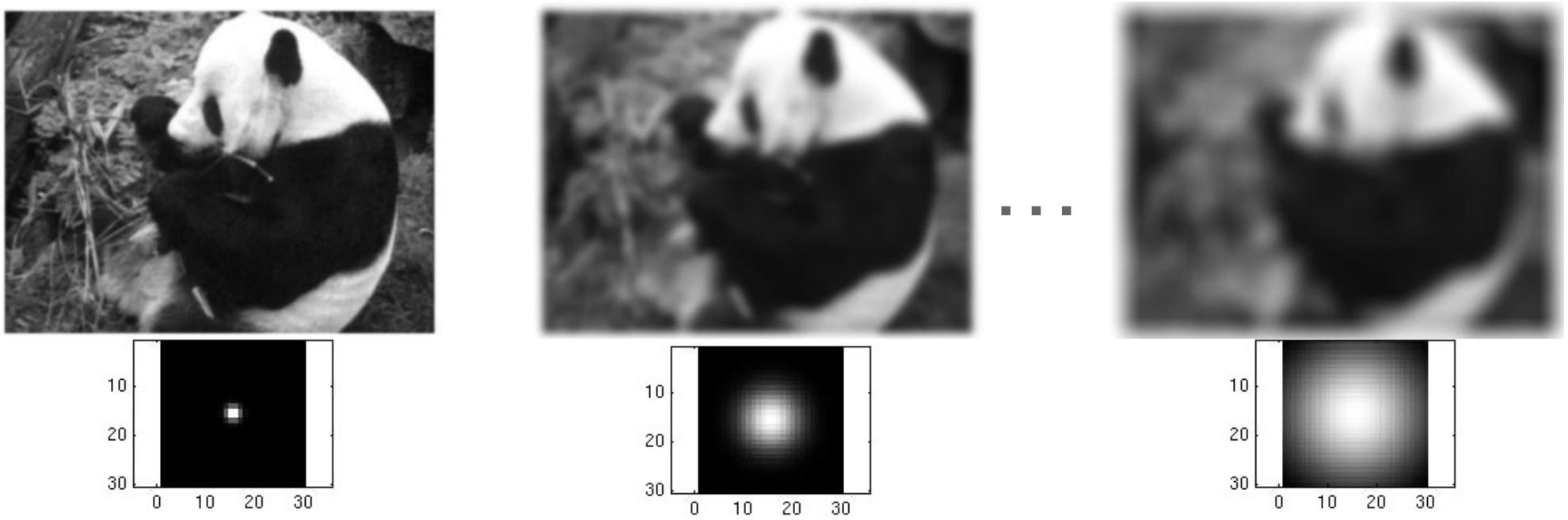
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Source: S. Seitz

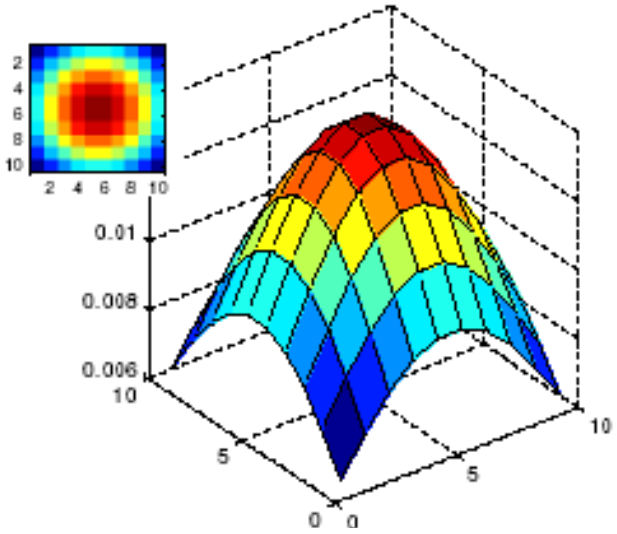
Smoothing with a Gaussian

- Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

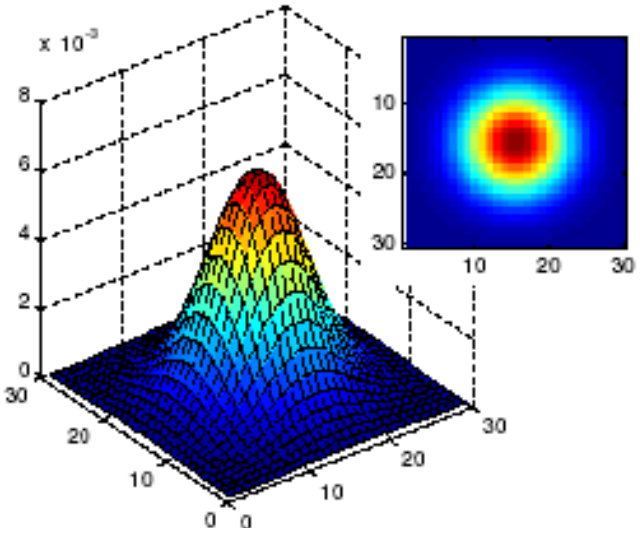


Gaussian filters

- What parameters matter here?
- **Size** of kernel / mask
  - Gaussian function has infinite support, but discrete filters use finite kernels



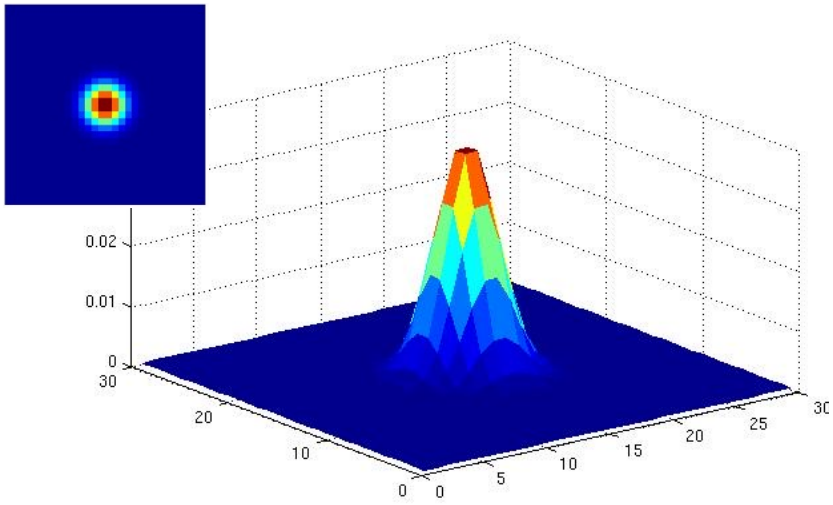
σ = 5 with 10 x 10 kernel



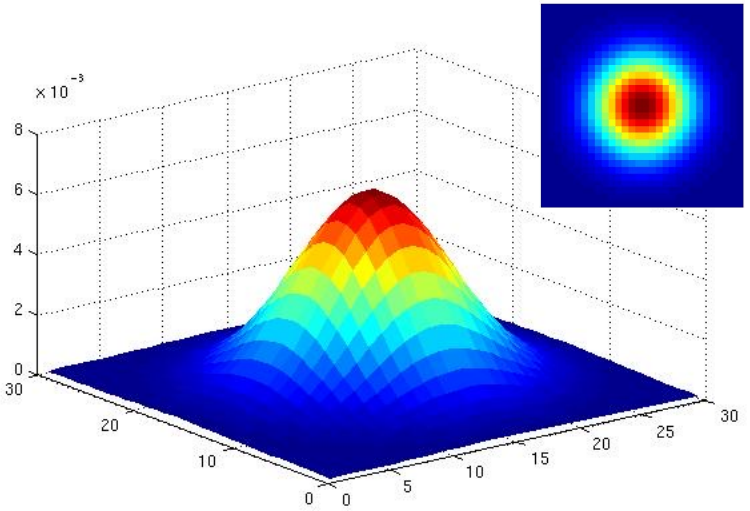
σ = 5 with 30 x 30 kernel

Gaussian filters

- **Variance**: determines extent of smoothing



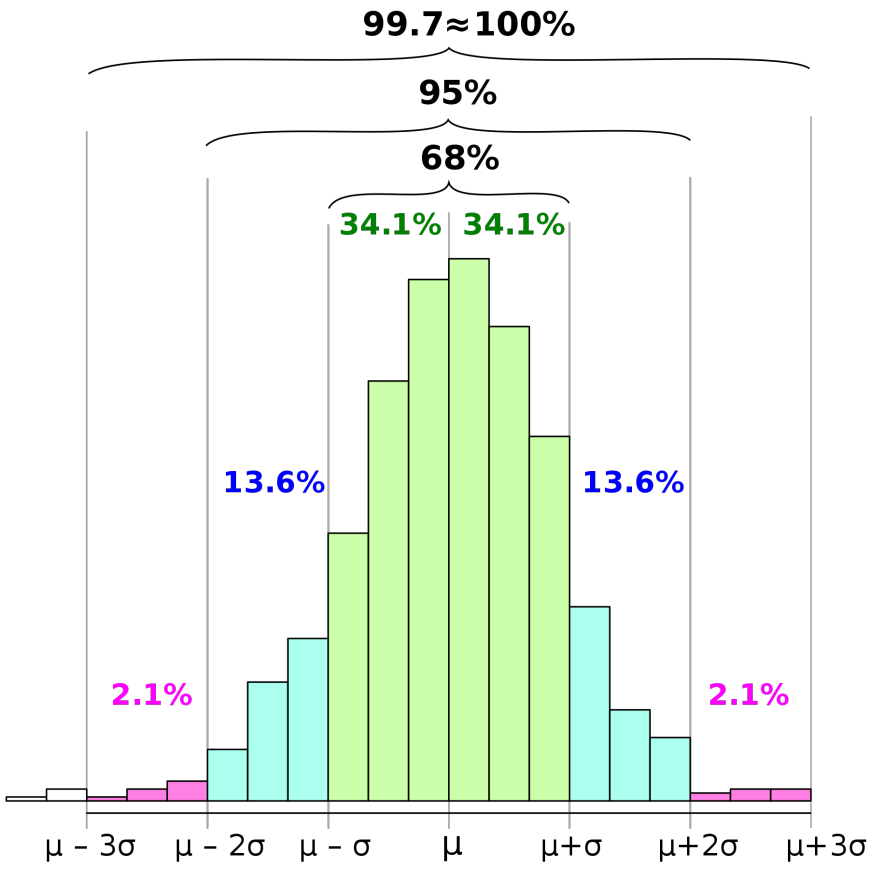
σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

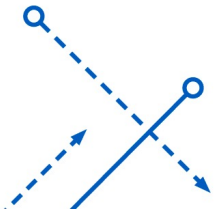
How big should the filter be?

- Values at edges should be near zero
- Rule of thumb for Gaussian filter:
  - set filter half-width to about 3σ

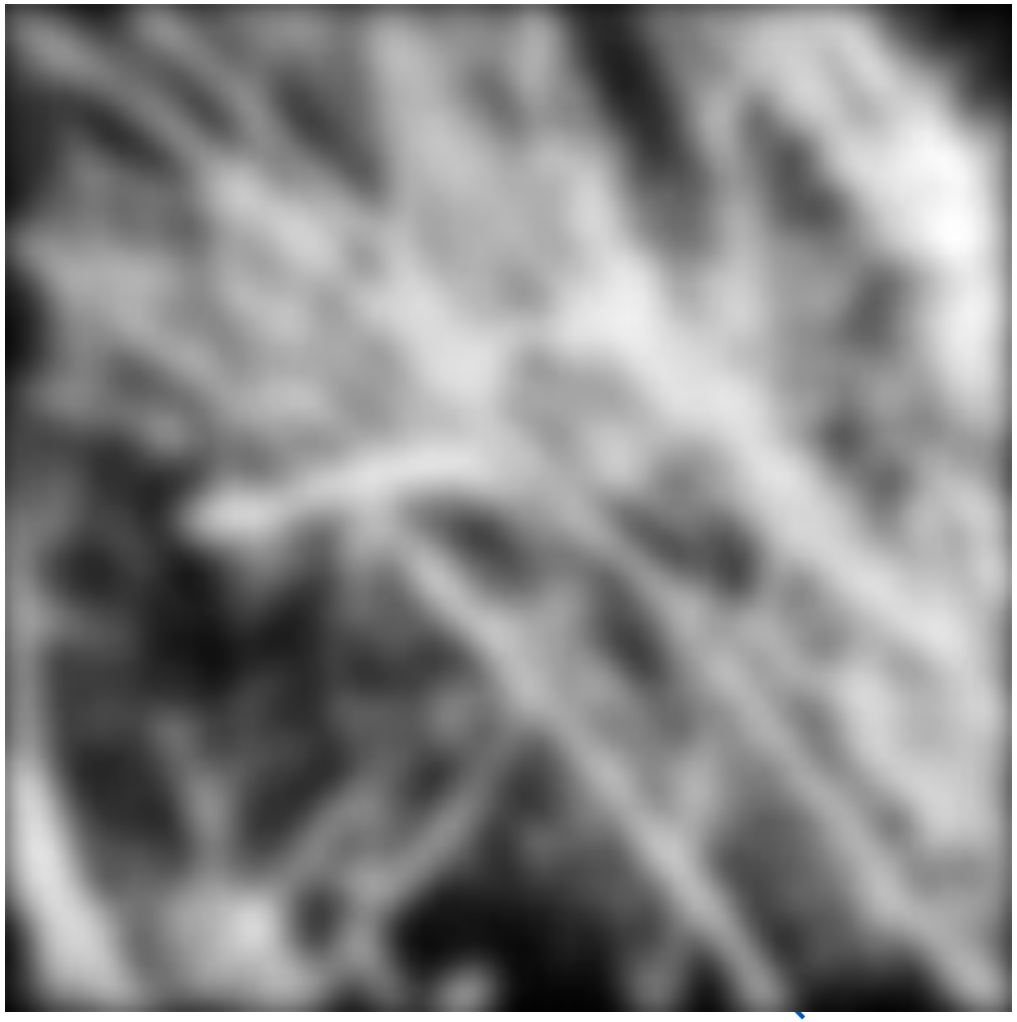


Gaussian filters

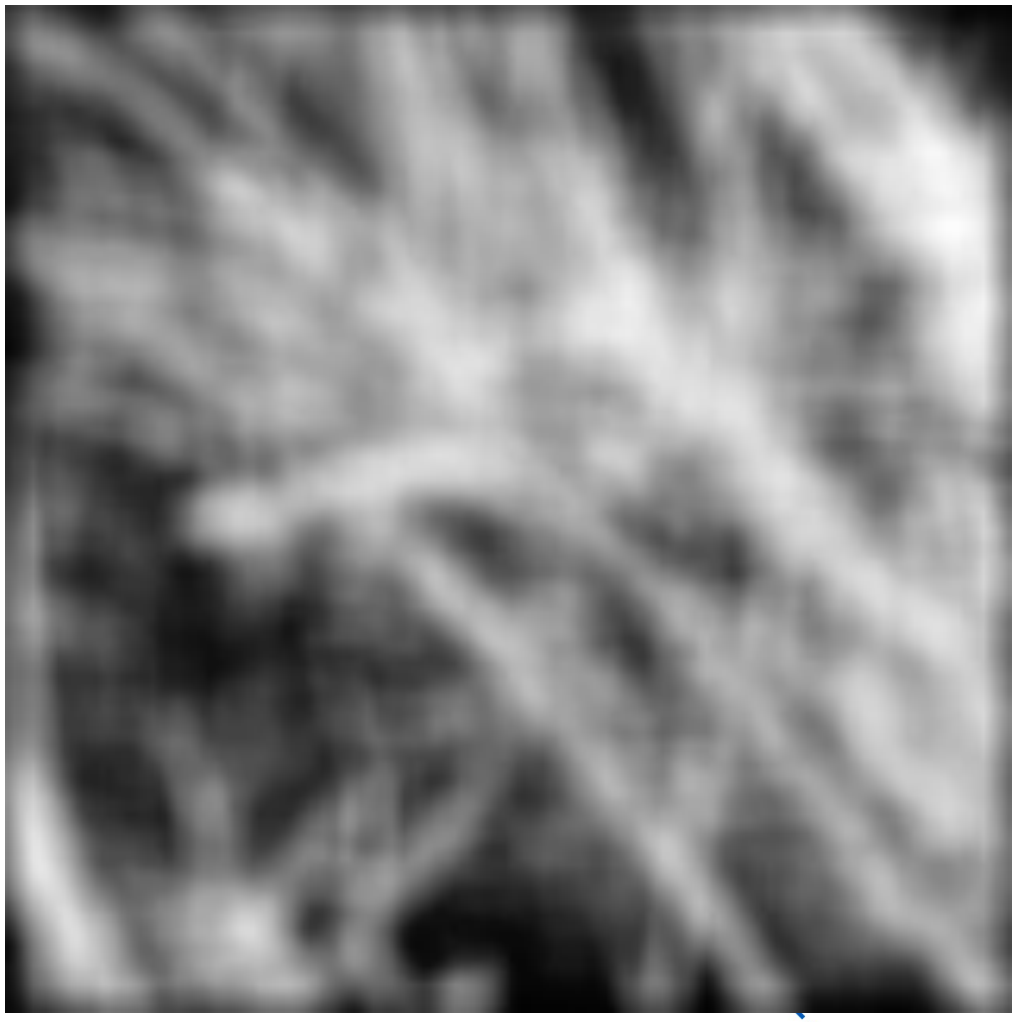
- Remove “high-frequency” components from image
  - Low-pass filter: Images become smoother.
- Recap of frequency in a signal (image)
  - High-frequency components
    - Fine details and edges
  - Low-frequency components
    - Large-scale structures and smooth regions



Smoothing with Gaussian filter

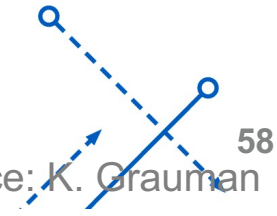


Smoothing with box filter



Gaussian filters

- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians



Source: K. Grauman

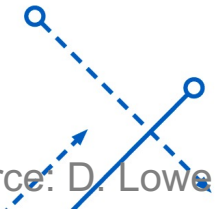
Separability of the Gaussian filter

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

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

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

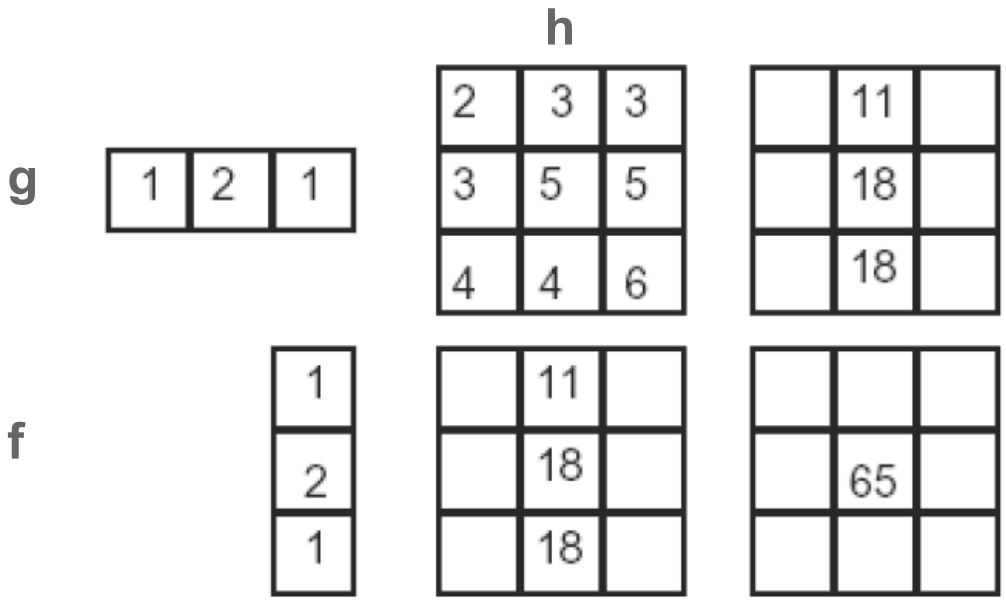


Source: D. Lowe

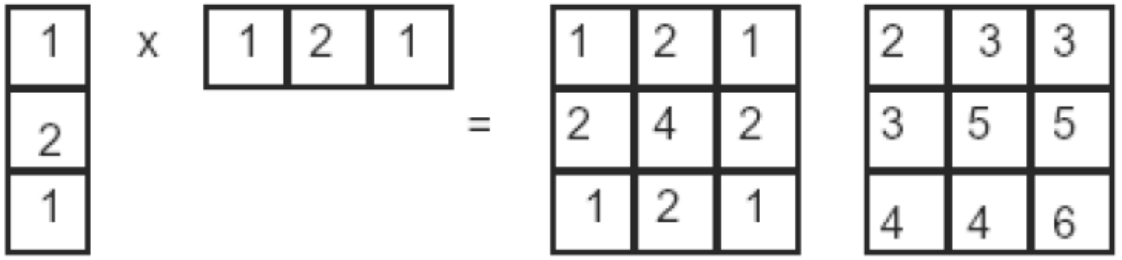
Separability

- In some cases, filter is separable, (factor into two steps):

$$\boldsymbol{f} \otimes (\boldsymbol{g} \otimes \boldsymbol{h}) = (\boldsymbol{f} \otimes \boldsymbol{g}) \otimes \boldsymbol{h}$$

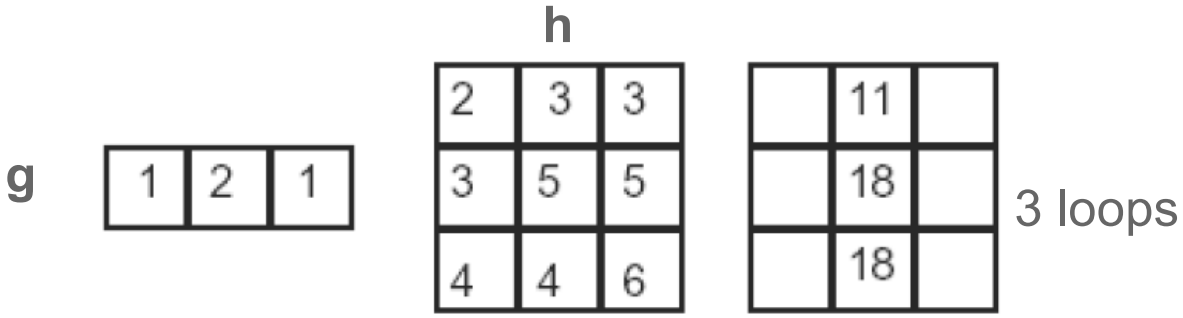


What is the computational complexity advantage for a separable filter of size k x k, in terms of **number of operations per output pixel**?



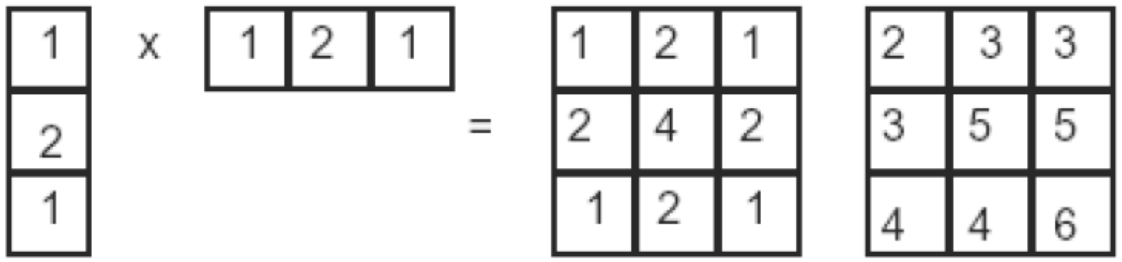
Separability

- **Advantage:** much reduced computational cost.
- **Disadvantage:** requires an extra ram **memory** to store the intermediate image, problematic in certain applications.

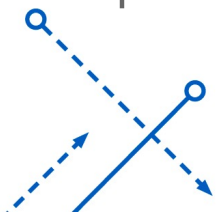


1 loop is roughly 3 multiplication

Total 4 loops

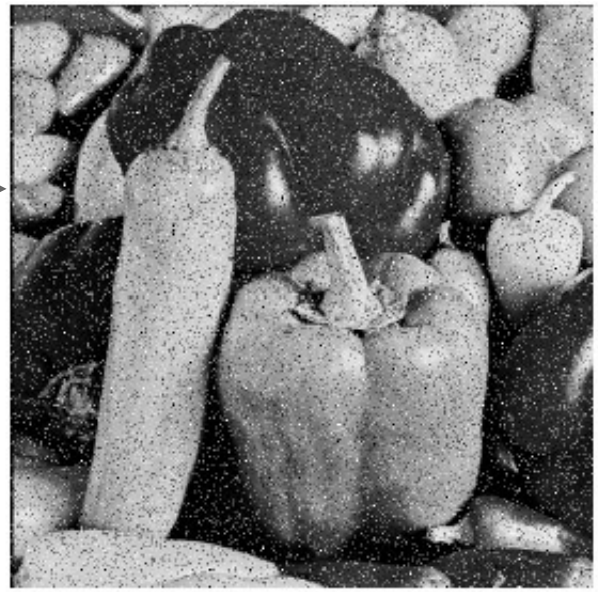


Total 6 loops

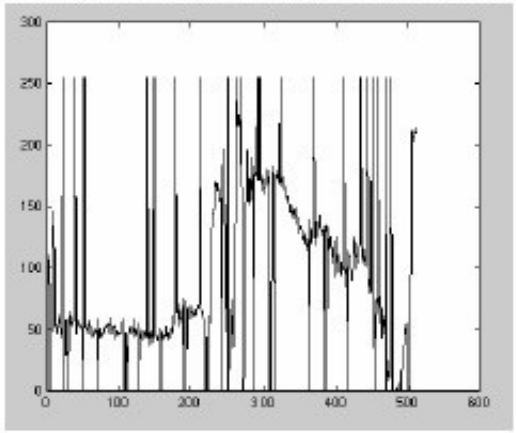


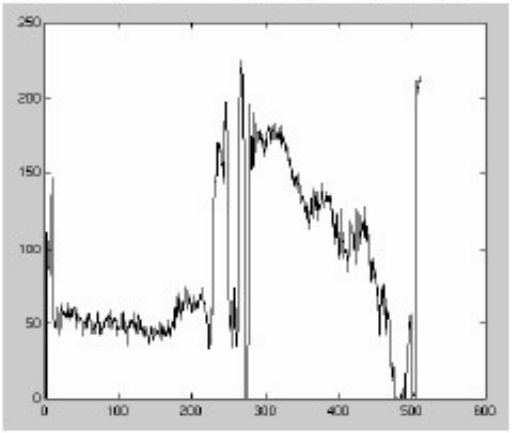
Median filter

Salt and pepper noise



Median filtered



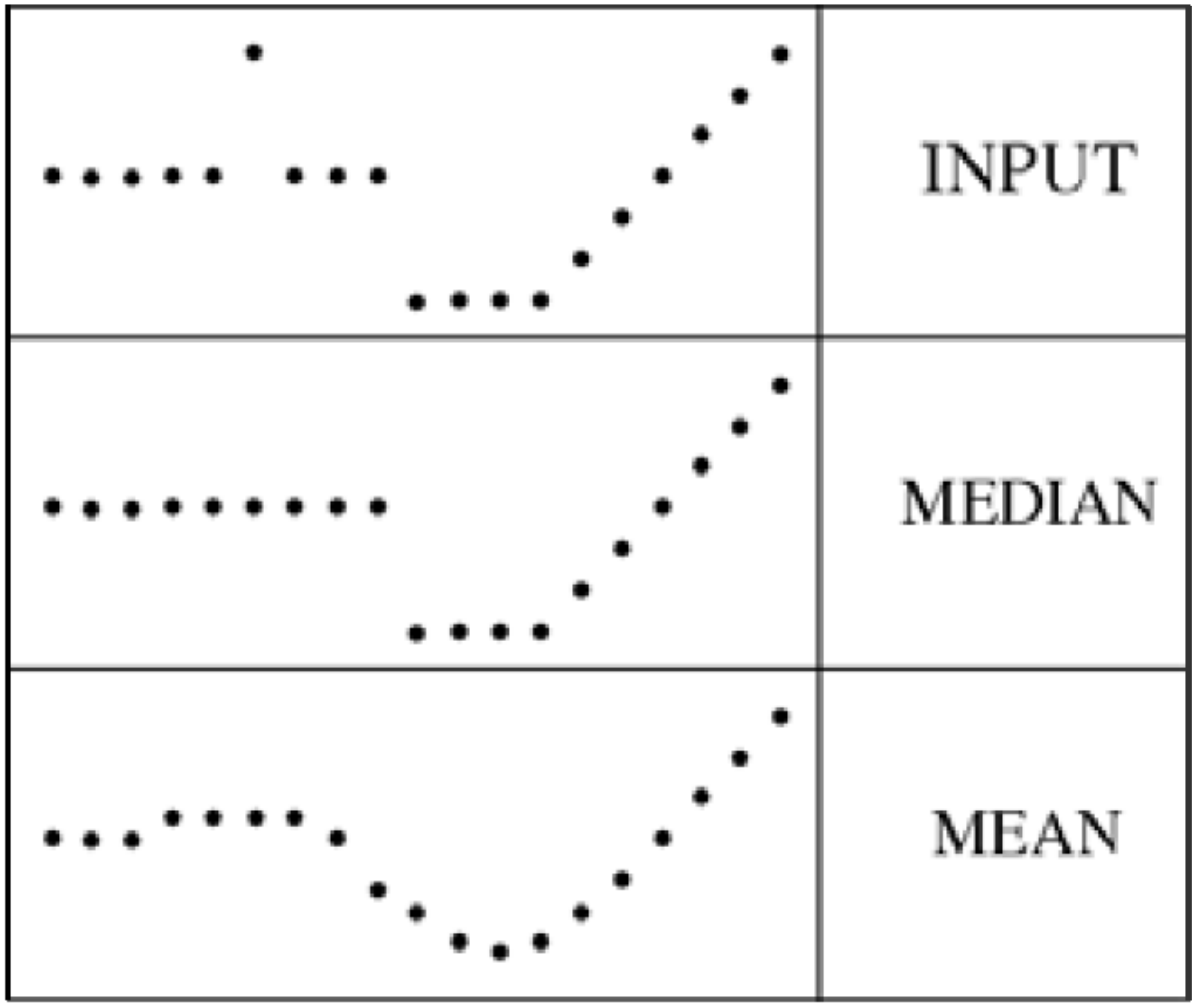


Plots of a row of the image

Source: M. Hebert

Median filter

- Median filter is edge preserving
- It doesn't introduce new intensities, which is often expected.



SAIR

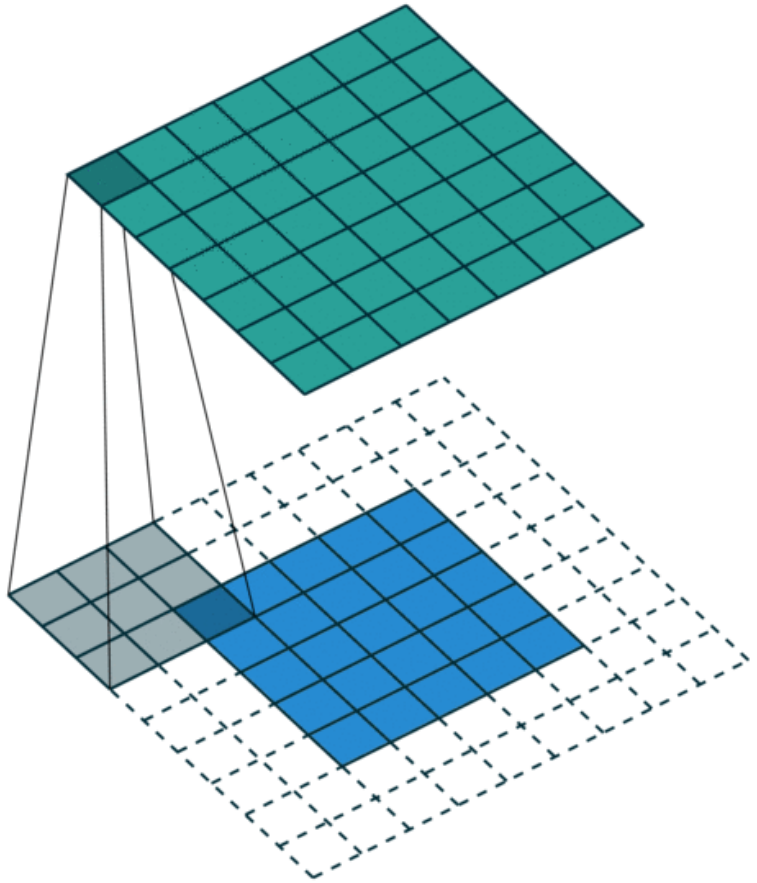
Spatial AI & Robotics Lab

IMAGE PROCESSING

Template Matching

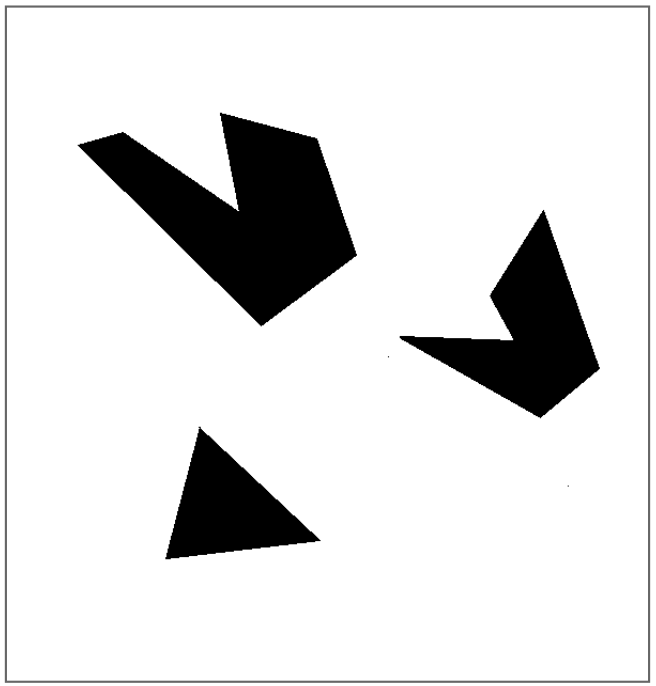
Similarity/Distance of Signals

- L1-norm / Manhattan distance
  - $|\boldsymbol{x} - \boldsymbol{w}|_1$
- L2-norm / Euclidean distance
  - $||\boldsymbol{x} - \boldsymbol{w}||_2$
- Inner Product
  - $\boldsymbol{x} \cdot \boldsymbol{w}$
- Cosine Similarity
  - $\frac{\boldsymbol{x} \cdot \boldsymbol{w}}{||\boldsymbol{x}||_2 ||\boldsymbol{w}||_2}$
- Filtering gives us a kind of similarity measurement, i.e., inner product.

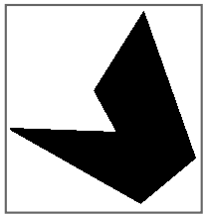


Template matching

- Each element of the output is a similarity measure of a specific pattern, i.e., a filter or a template.
- Each similarity measure is also called a ``response''.
- This process is called template matching.



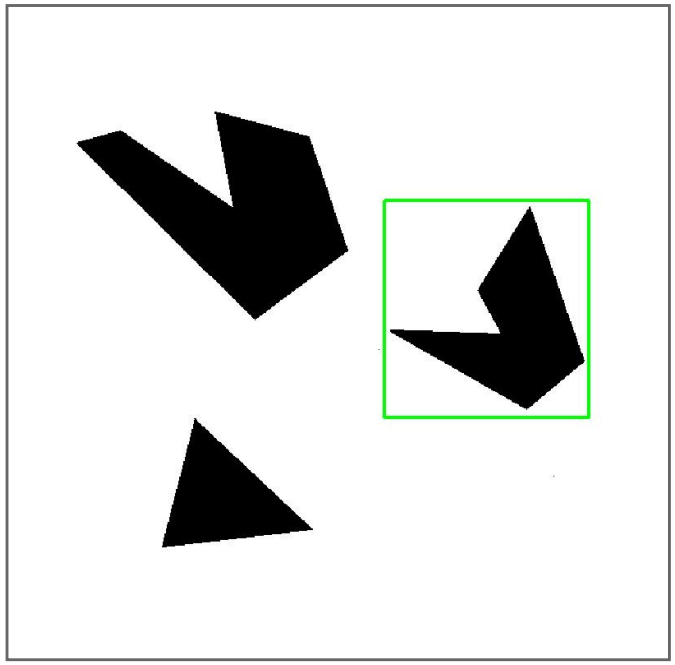
A toy example



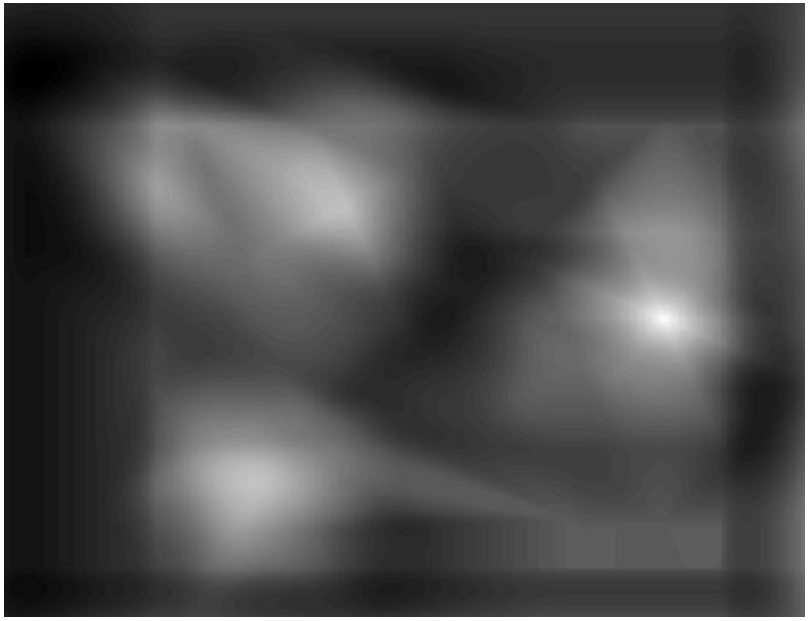
Template (mask)

Is there only one match?
What if the pattern is not exact?

Correlation of Template and Imgae



Detected template



Correlation map

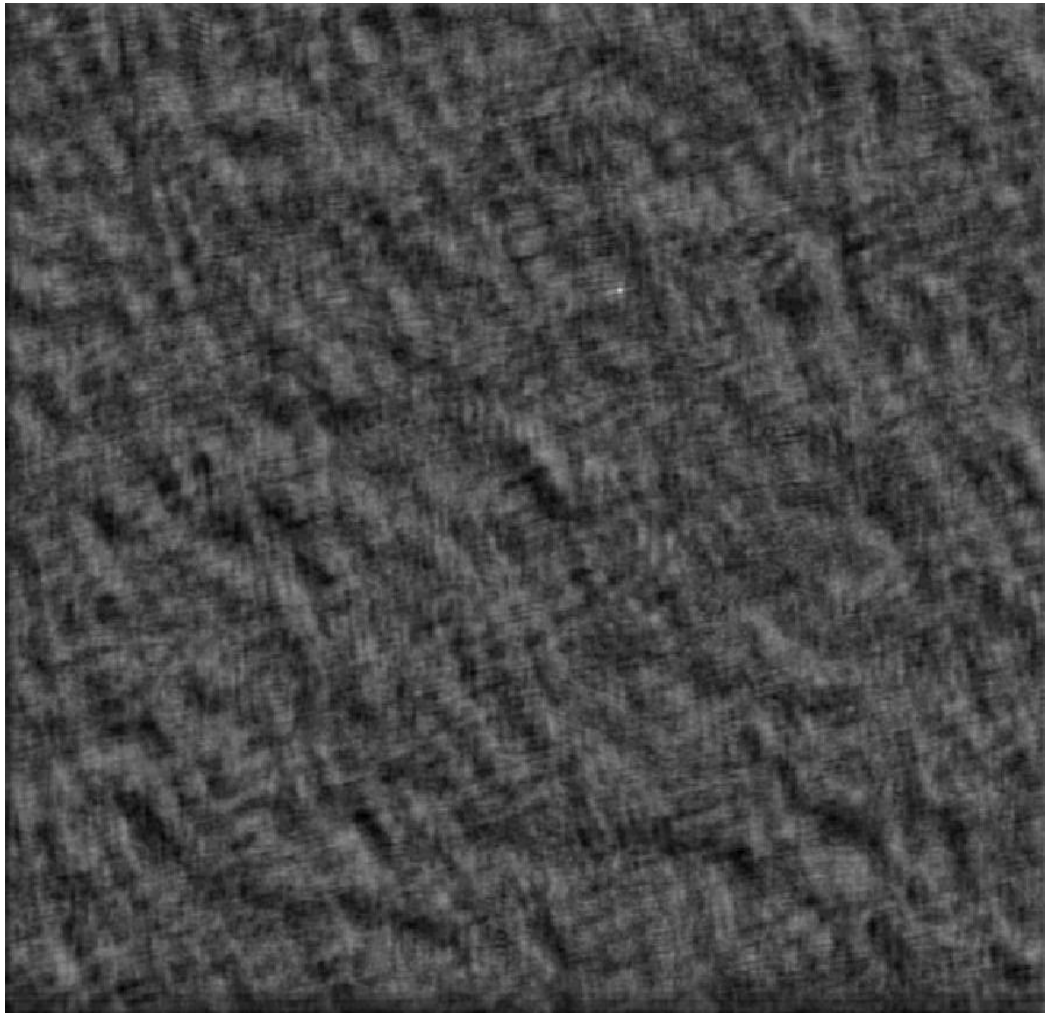
Is there only one match?
What if the pattern is not exact?

Where's Waldo?

Scene

Template

Where's Waldo?

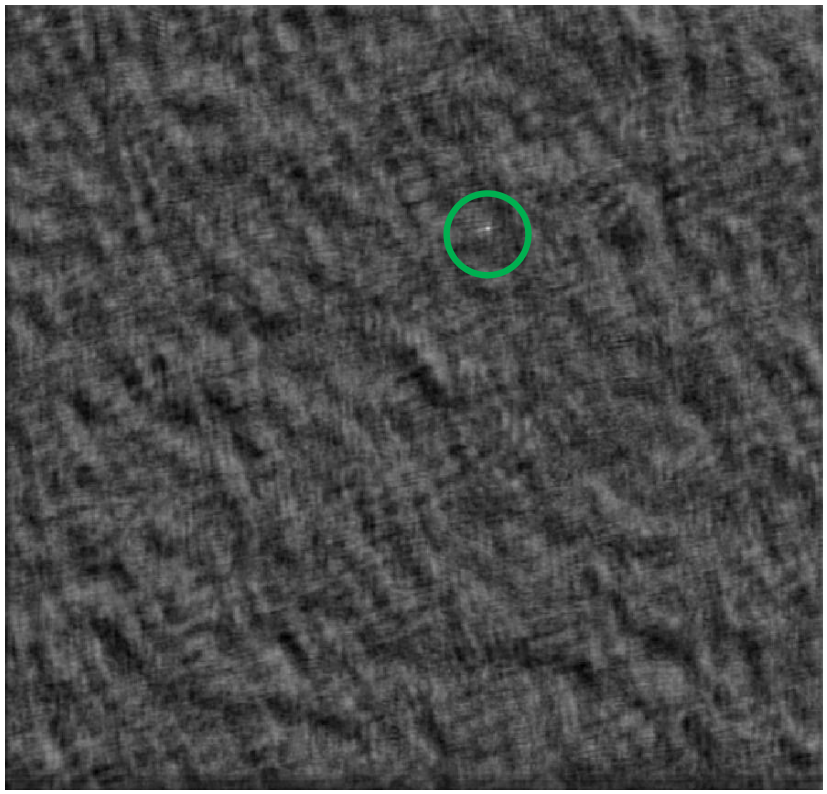


Scene

Template

Where's Waldo?

Detected template

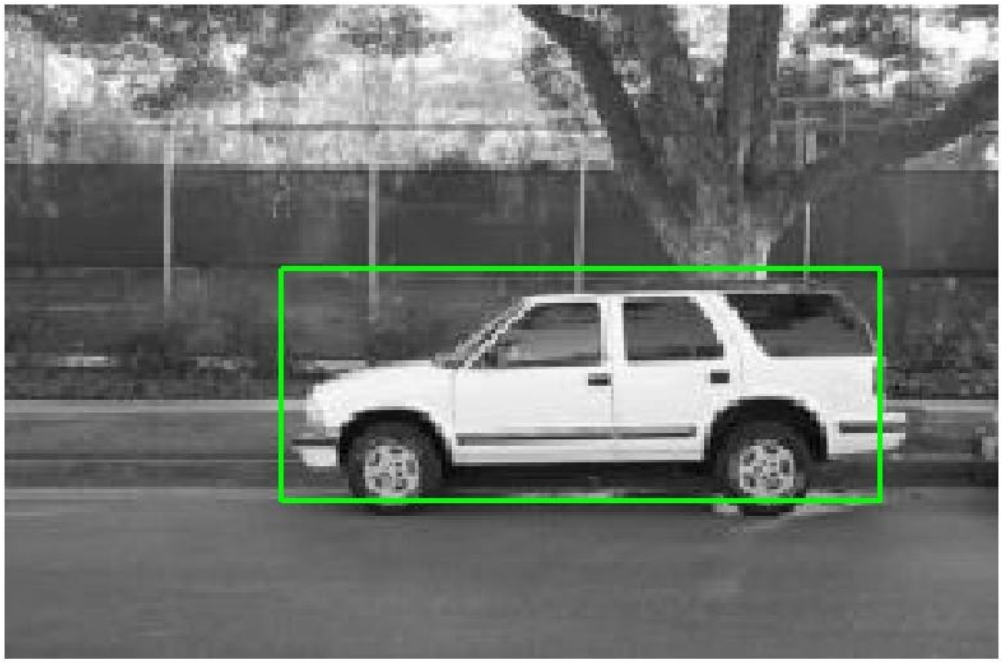


Correlation map

- Use normalized cross-correlation score to find a given pattern (template) in the image (Szeliski Eq. 8.11 in textbook).
- Normalization needed to control for relative brightness.

Template matching

- Match can be meaningful, if scale, orientation, and general appearance is right.



Detected template

Template

Template matching

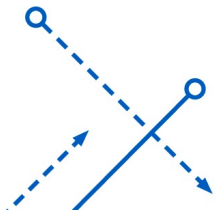
- What if the template is not identical to some sub image in the scene?

Scene

Template

Template matching

- We need more flexible, powerful and forgiving representations.
  - Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. Visual object tracking using adaptive correlation filters. CVPR, 2010.
    - Computational complexity: $\mathbb{O}(N \log N)$
    - Equivariant to translation, robust to small appearance variance.
  - Wang, C., Zhang, L., Xie, L., & Yuan, J. (2018, April). Kernel cross-correlator. *AAAI, 2018*.
    - Nonlinear cross-correlation with the kernel trick.
    - Equivariant to any transforms:
      - Translation, Scale, Rotation, Affine, etc.
    - Same computational complexity with linear filter: $\mathbb{O}(N \log N)$



SAIR

Spatial AI & Robotics Lab

IMAGE PROCESSING

Edge Detection

Filters for features

- Previously, filtering is a way to
  - Remove or reduce **noise.**
  - **Template matching**
- Filters also allows us to abstract higher-level "**features**".
  - Map raw pixels to intermediate representations used for **subsequent processing**.
  - Reduce amount of data, discard redundancy, preserve useful information.

Edge detection

- **Goal**: map image from 2D array of pixels to a set of **curves** or **line segments**, or **contours**.
- **Why?**

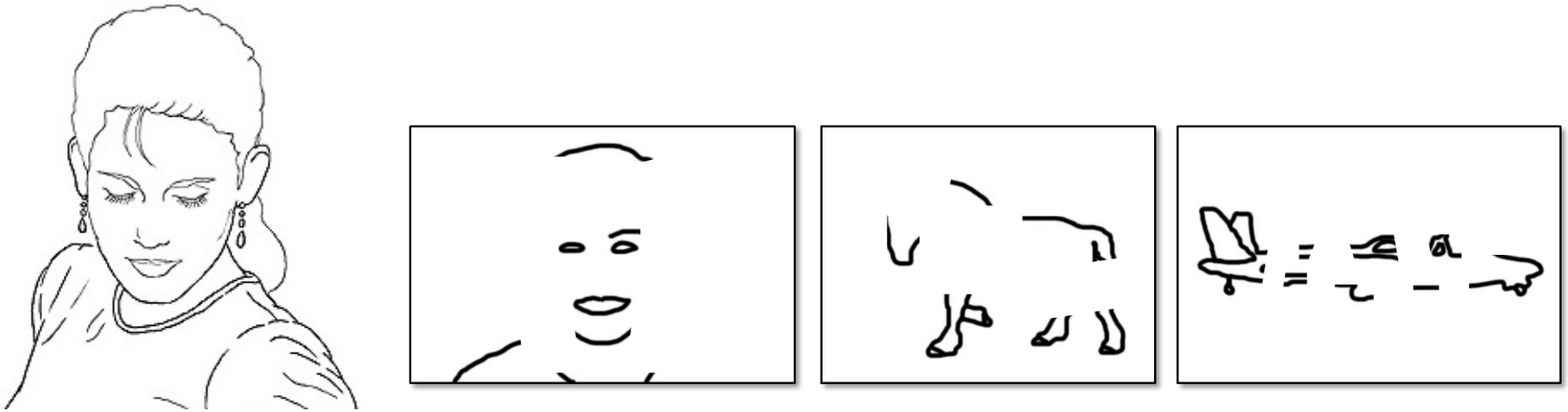
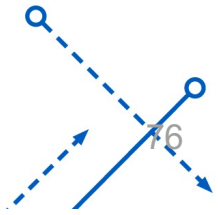
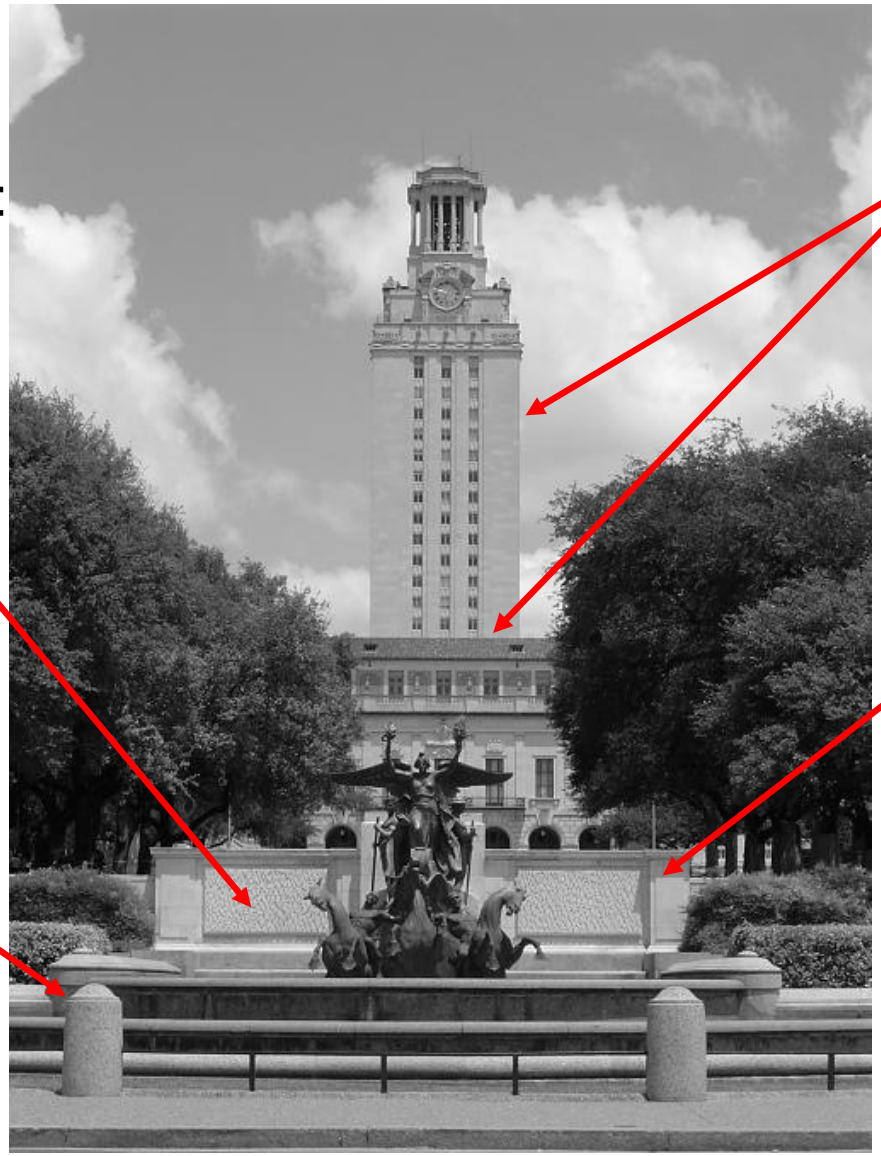


Figure from J. Shotton et al., PAMI 2007

- **Even simple sketch of the objects are quite meaningful.**
- **Main idea**: look for strong gradients, post-process.



What can cause an edge?



Reflectance change: appearance information, texture

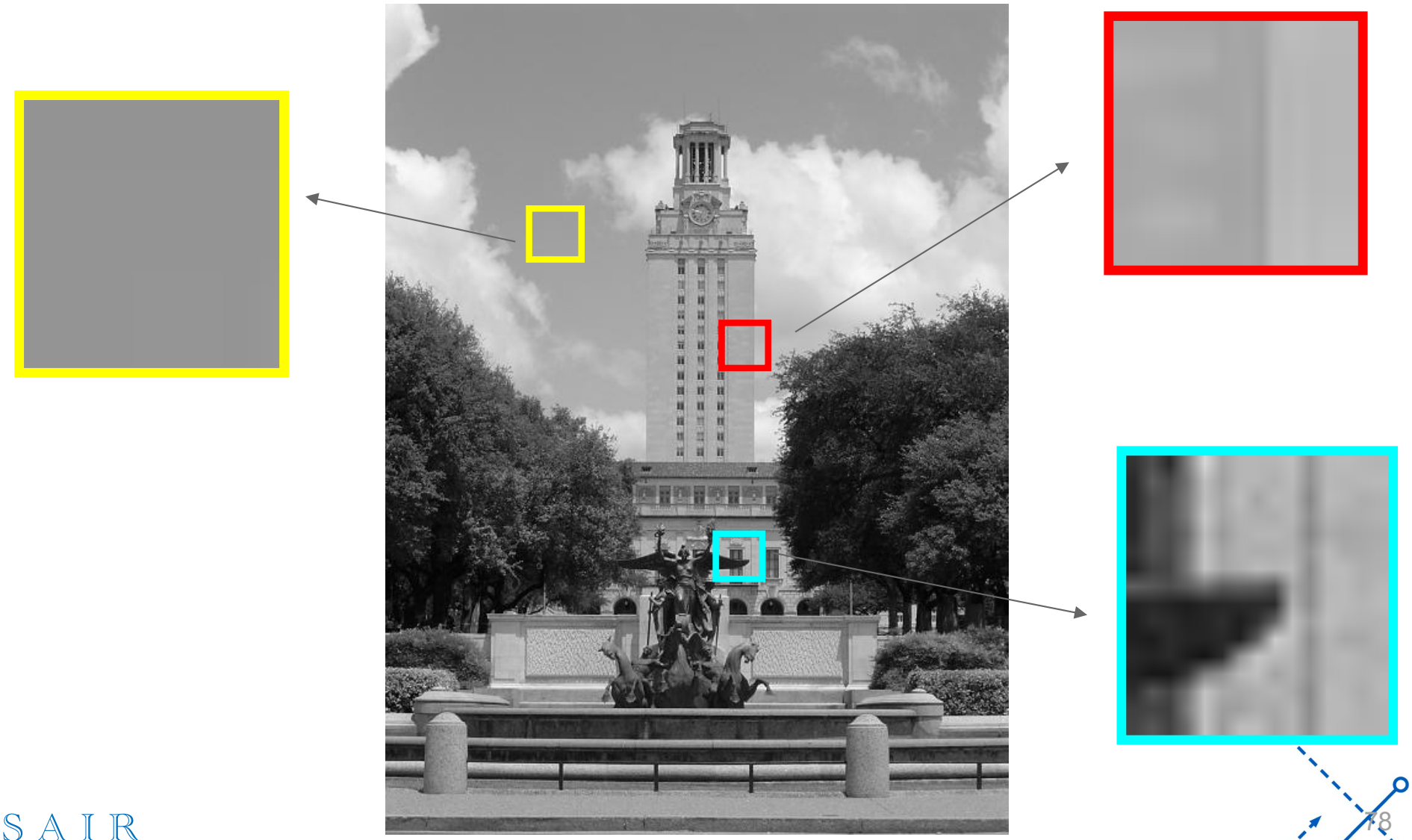
Depth discontinuity: object boundary

Cast shadows

Change in surface orientation: shape

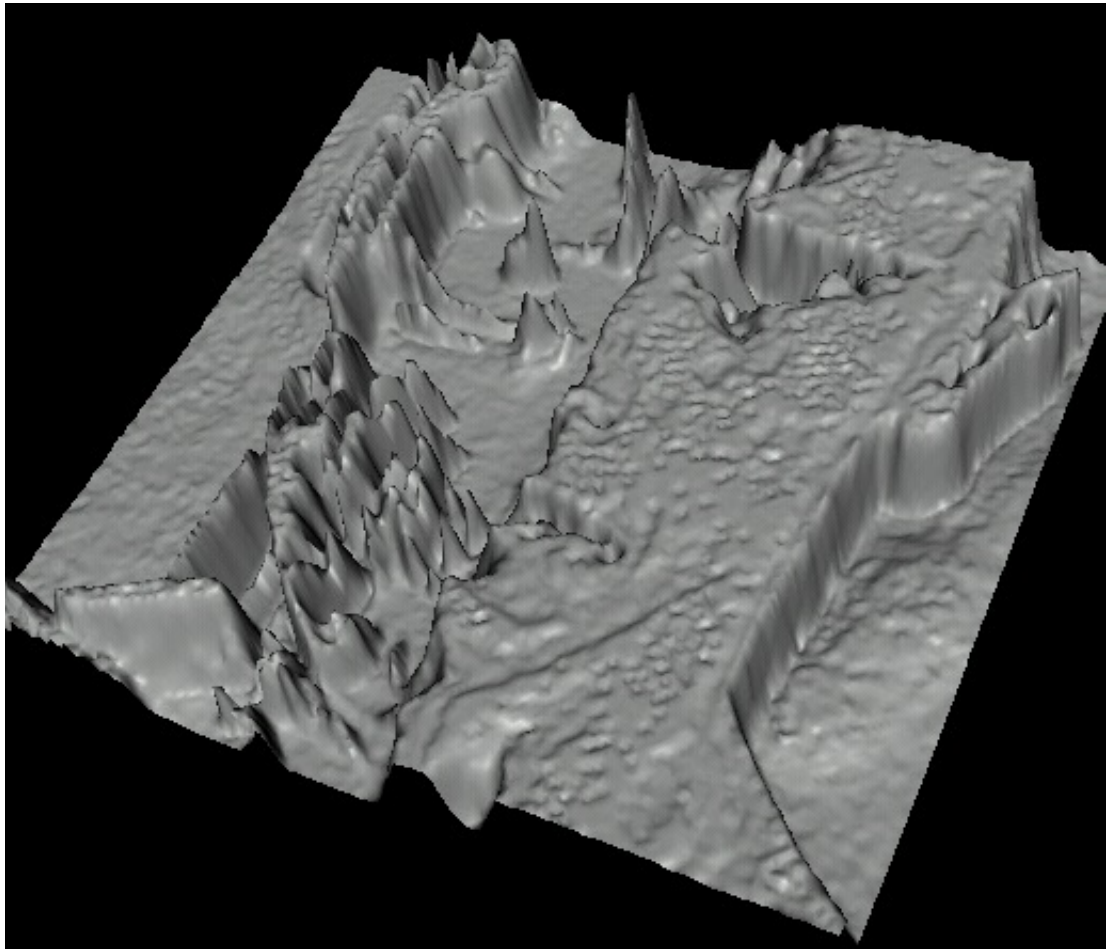
77

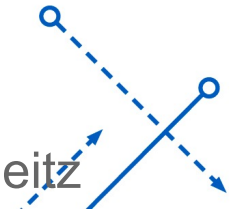
Contrast and invariance



78

Edges look like steep cliffs

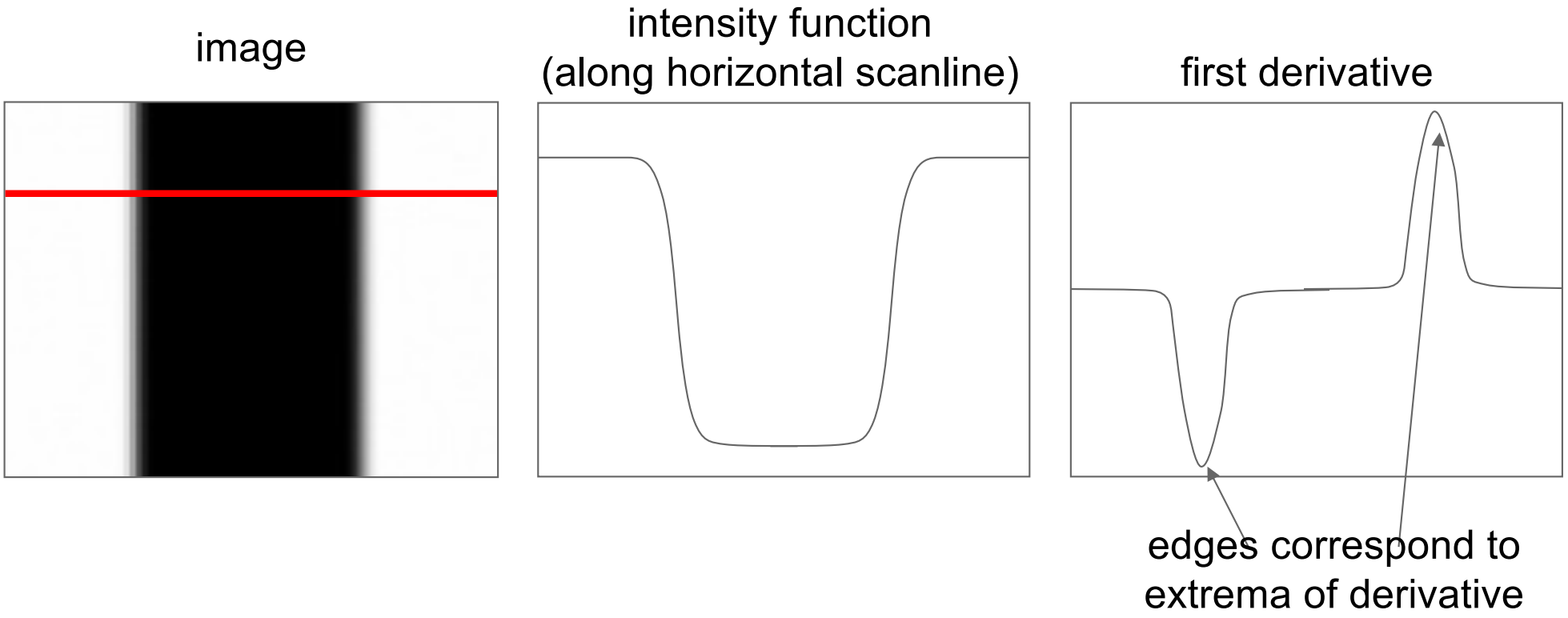




Source: S. Seitz

Derivatives and edges

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

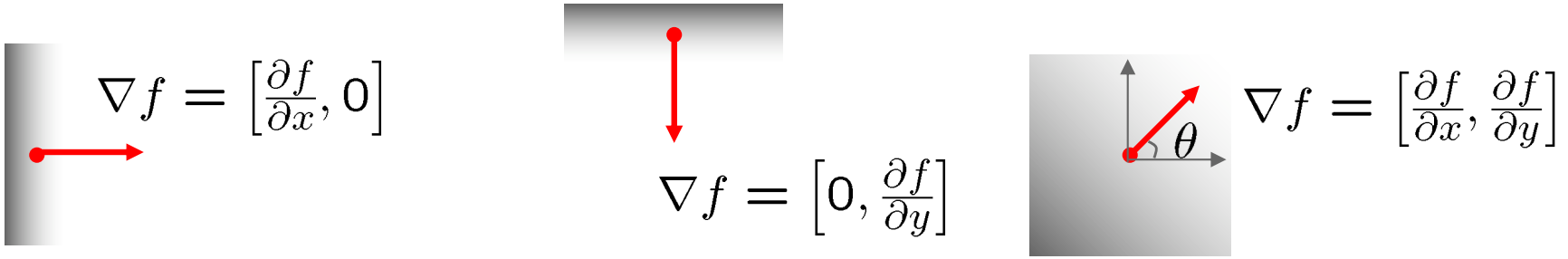


Source: L. Lazebnik

Image gradient

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

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



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

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

Differentiation and convolution

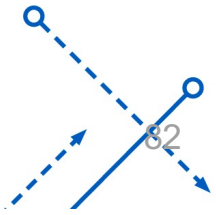
For 2D function, $f(x, y)$, the partial derivative is:

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

For discrete data, we can approximate using finite differences:

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

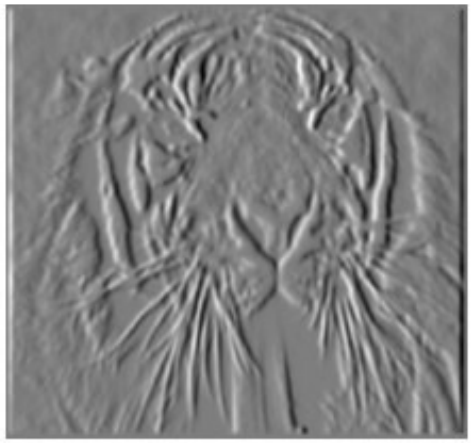
To implement above as correlation, what would be the filter?

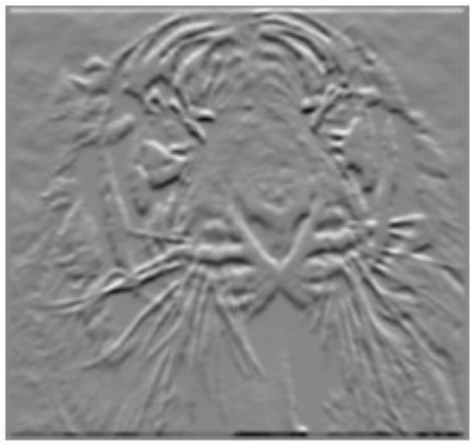


Partial derivatives of an image

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

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





-1	1

-1
1

? or

1
-1

Which shows changes with respect to x?

(showing flipped filters)

Prewitt operator

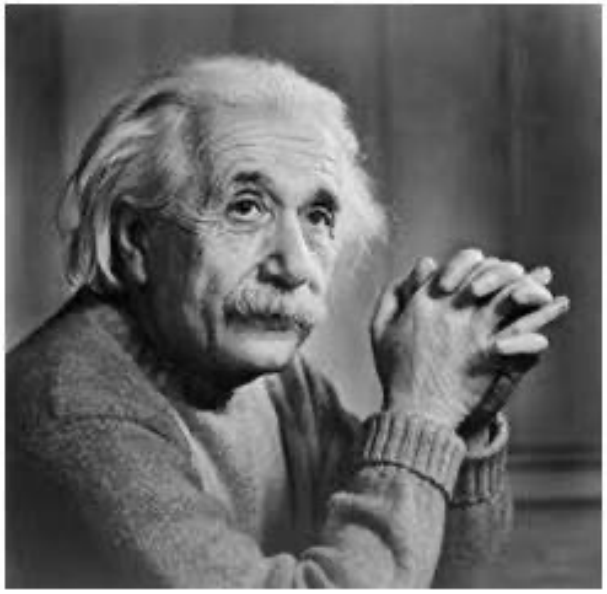
$G_x =$

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

$G_y =$

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

Fig. 1. The horizontal and vertical Prewitt edge detection masks.



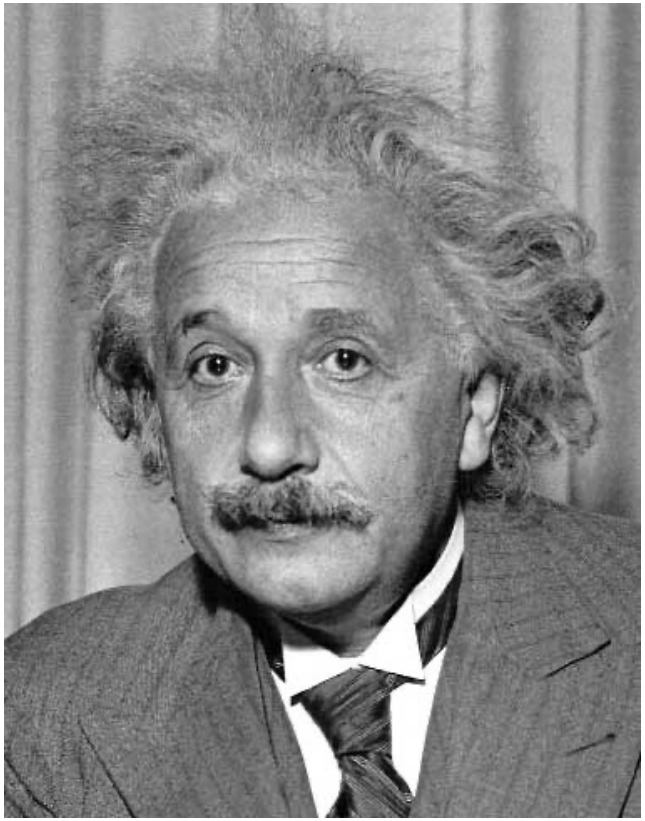
Sobel Operator

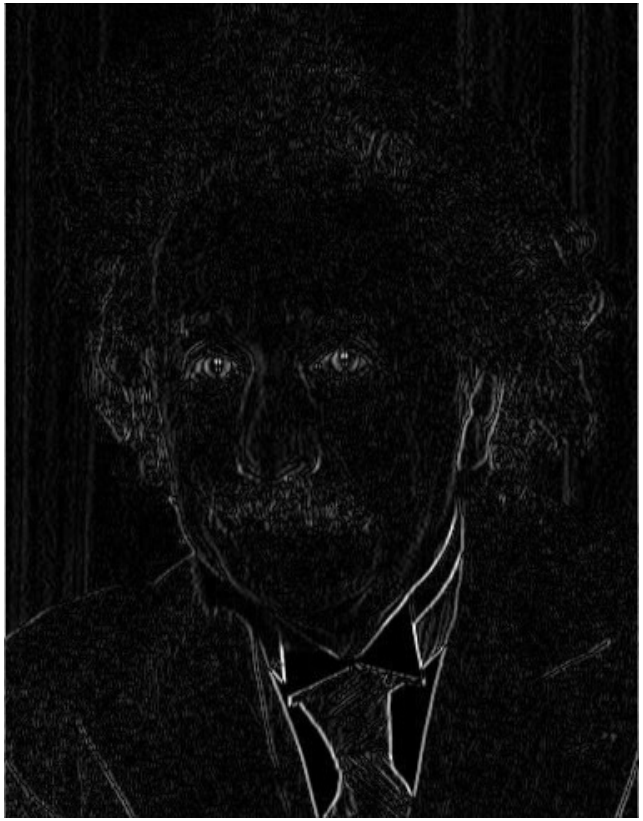
$G_x =$

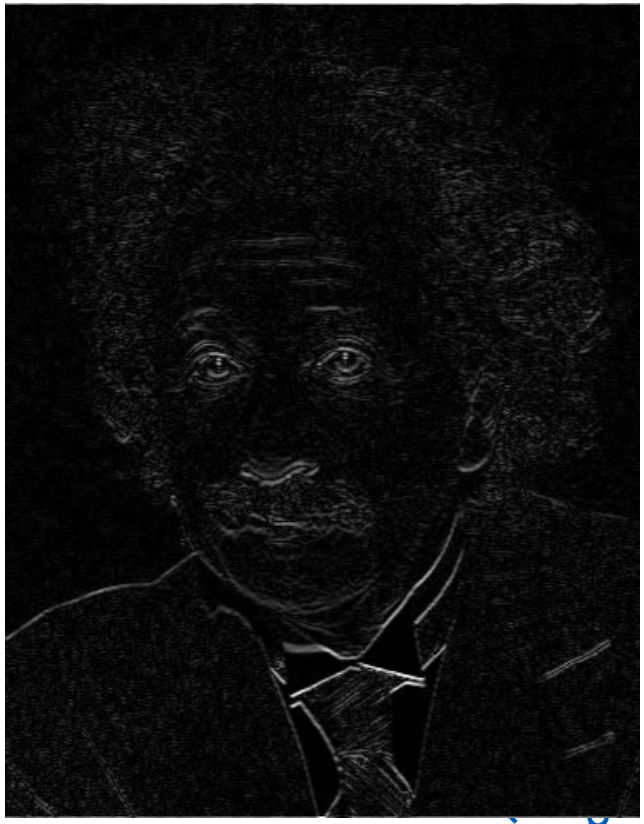
1	0	-1
2	0	-2
1	0	-1

$G_y =$

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







Roberts Operator

+1	0
0	-1

Gx

0	+1
-1	0

Gy

$$|G| = \sqrt{Gx^2 + Gy^2}$$

$$|G| = |Gx| + |Gy|$$

$$|G| = |P_1 - P_4| + |P_2 - P_3|$$

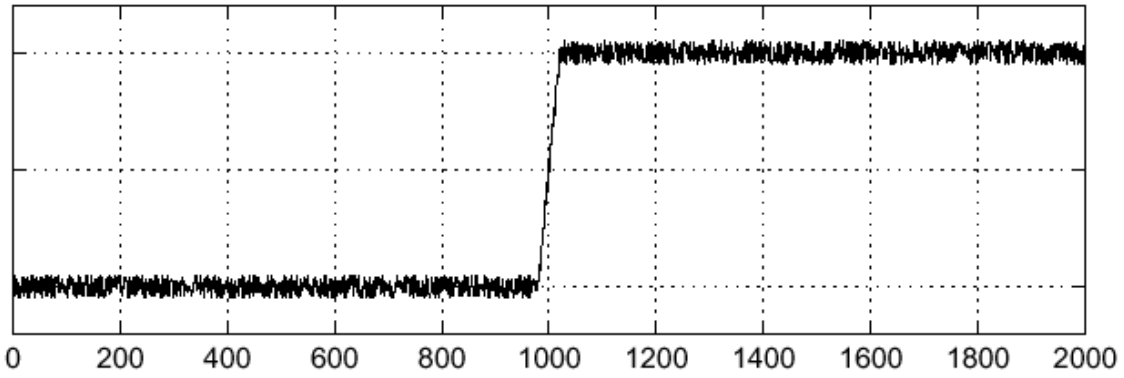
P_1	P_2
P_3	P_4

Effects of noise

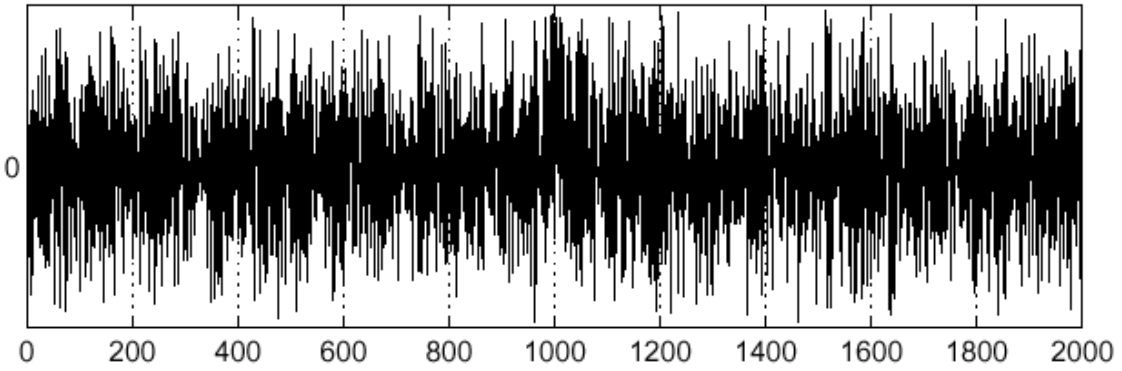
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$



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



Where is the edge?

87

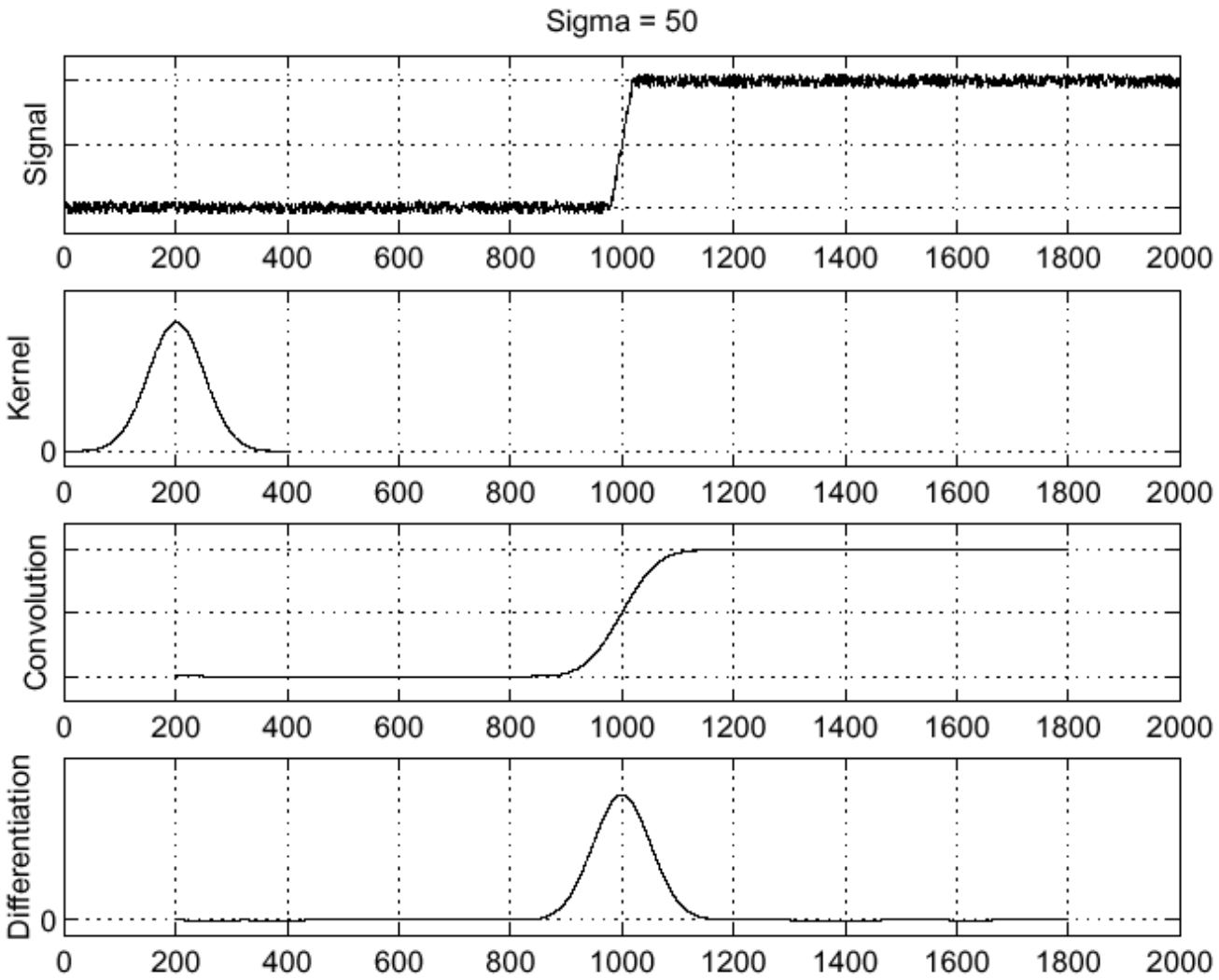
Solution: smooth first

f

h

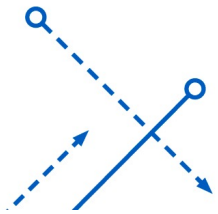
$h \star f$

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



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$



Derivative theorem of convolution

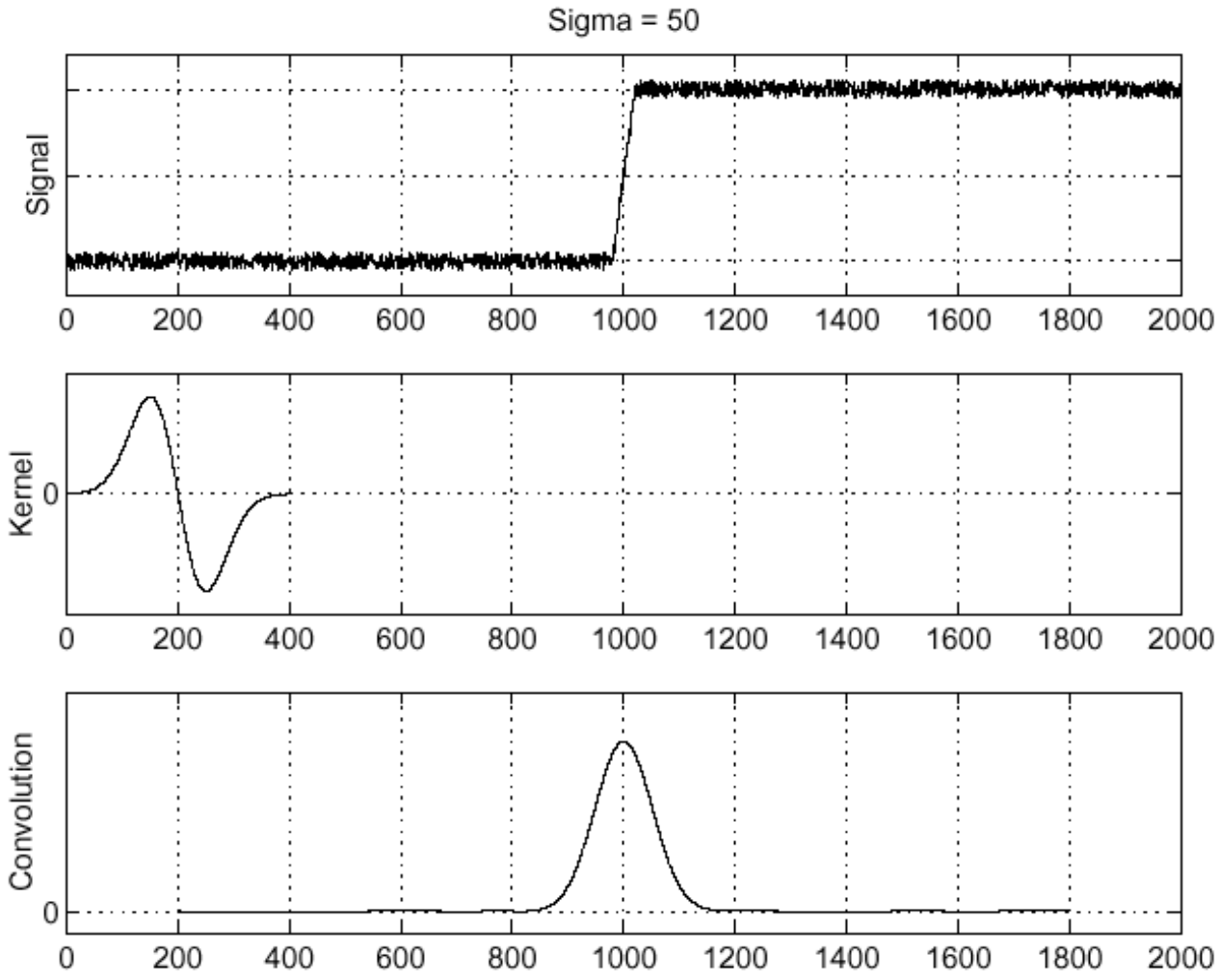
Differentiation property of convolution.

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

f

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

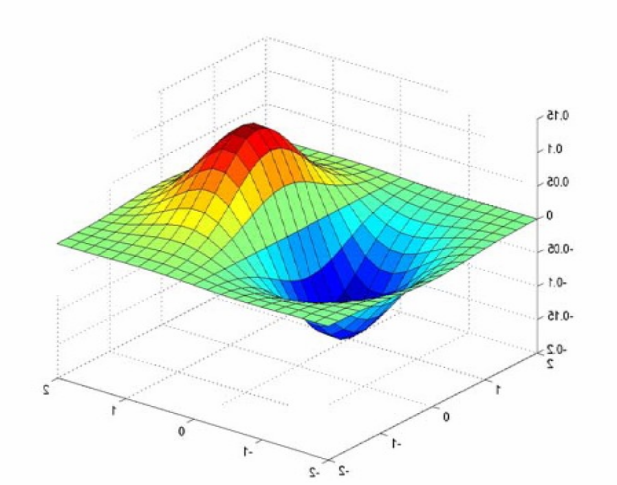
$(\frac{\partial}{\partial x}h) \star f$



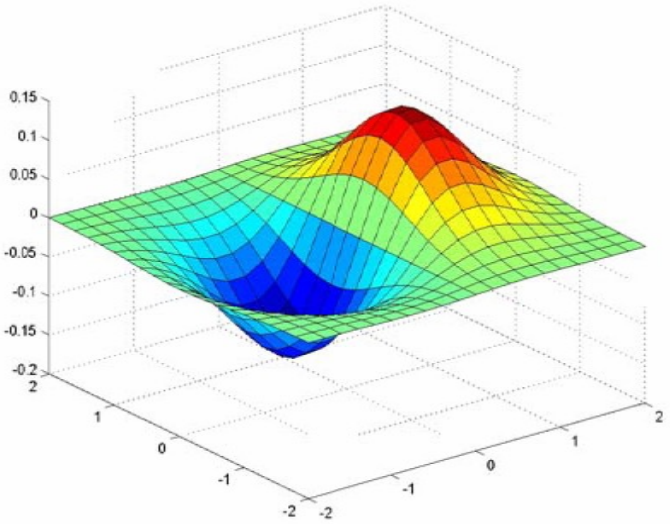
Derivative of Gaussian filter

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

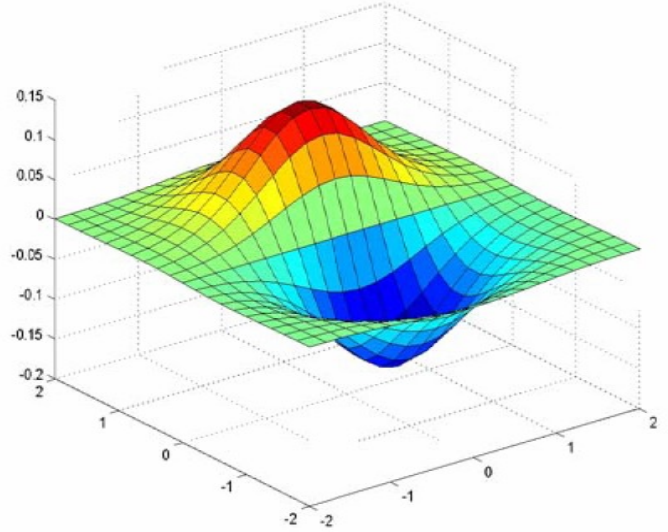
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$



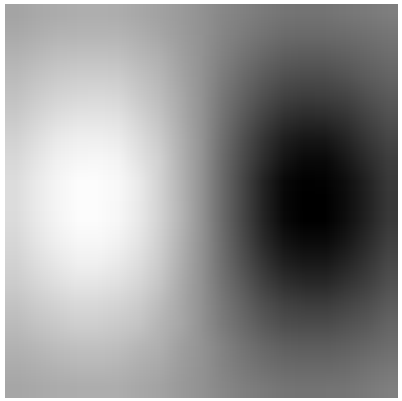
Derivative of Gaussian filters

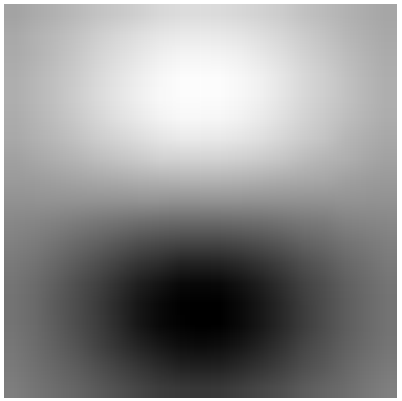


x-direction



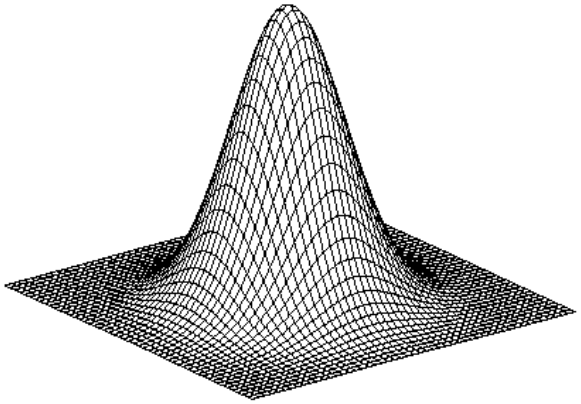
y-direction





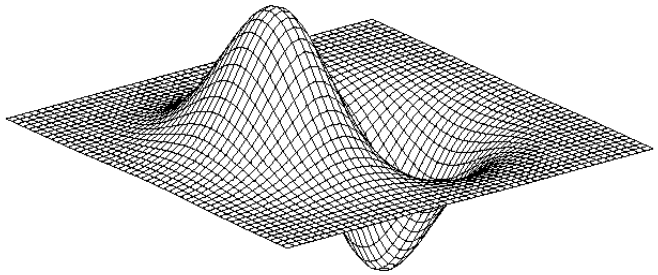
Source: L. Lazebnik

2D edge detection filters



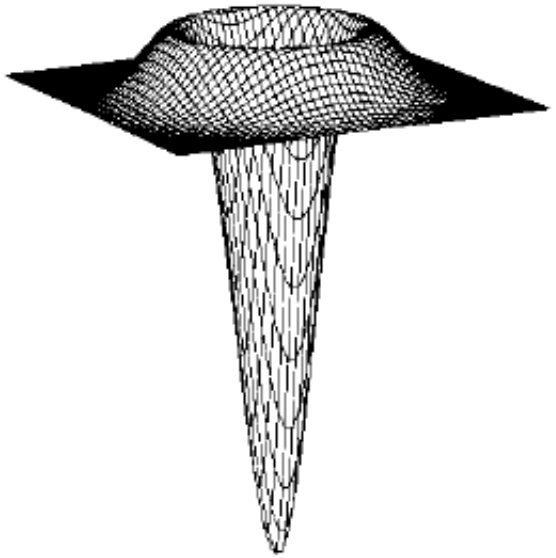
Gaussian

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



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



Laplacian of Gaussian

$$\nabla^2 h_\sigma(u,v)$$

- ∇^2 is the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

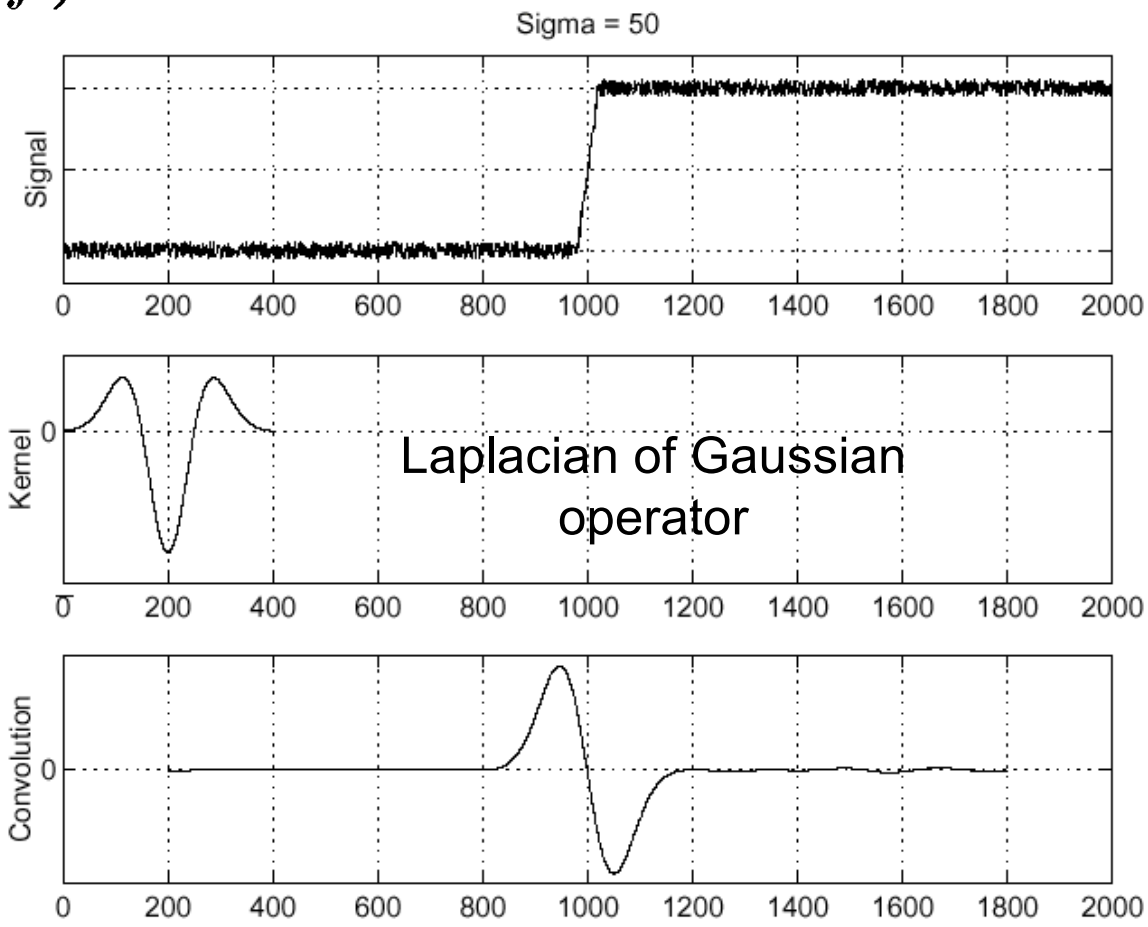
Laplacian of Gaussian

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

f

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

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

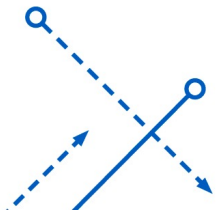


Where is the edge?

Zero-crossings of bottom graph

Mask/Filter/Kernel Properties

- Smoothing
  - Values positive
  - **Sum to 1:** constant regions → same as input (no change)
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- Template Matching
  - Dot product as correlation (inner product).
  - Highest response for regions “look the most like the filter”
- Derivatives
  - Opposite signs get high response in high contrast regions
  - **Sum to 0**: constant regions → no response
  - High contrast → high absolute values



Gradients -> edges

Primary edge detection steps:

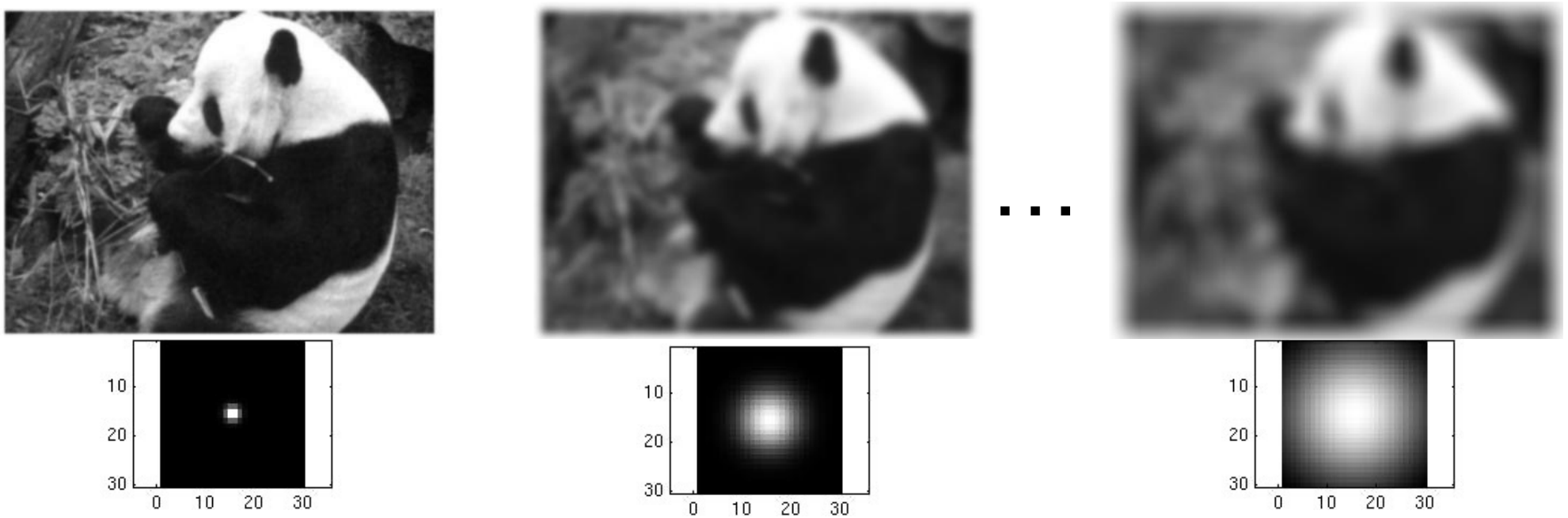
1. Smoothing: suppress noise
2. Edge enhancement: Filter for contrast
3. Edge localization

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

   - Threshold, Thin

Smoothing with a Gaussian

Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



Effect of σ on derivatives

σ = 1 pixel

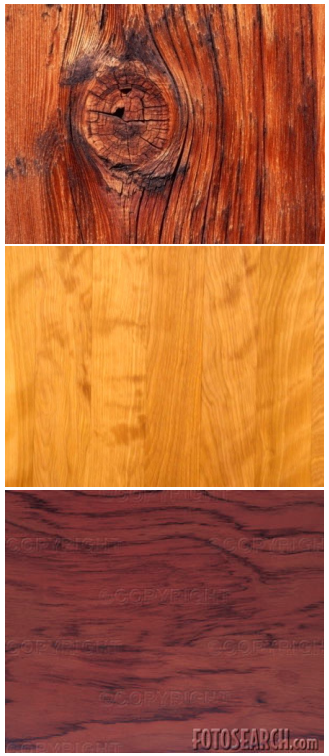
σ = 3 pixels

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

Larger values: larger scale edges detected
Smaller values: finer features detected

So, what scale to choose?

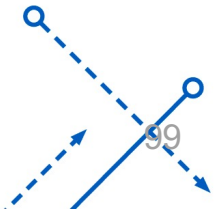
It depends what we're looking for.



Too small of a scale…can't see the forest for the trees.
Too big of a scale…can't tell the maple grain from the cherry.

Thresholding

- Choose a threshold value t.
- Set any pixels less than t to zero (off)
- Set any pixels greater than or equal to t to one (on)



Gradient magnitude image

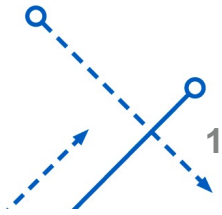
Thresholding gradient

lower threshold

higher threshold

Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Local non-maximum suppression**:
  - Thin multi-pixel wide “ridges” down to single pixel width
- Linking and thresholding (**hysteresis**):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them.



Source: D. Lowe, L. Fei-Fei

The Canny edge detector

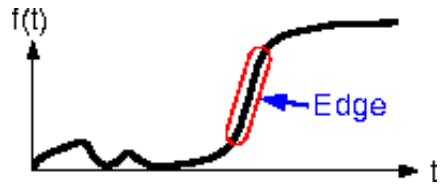
original image (Lena)

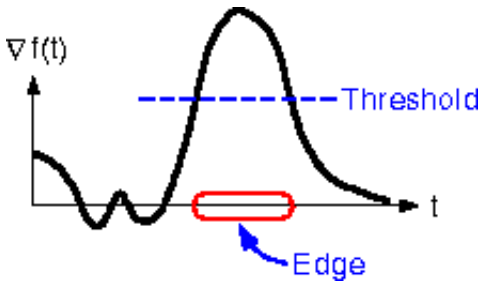
The Canny edge detector

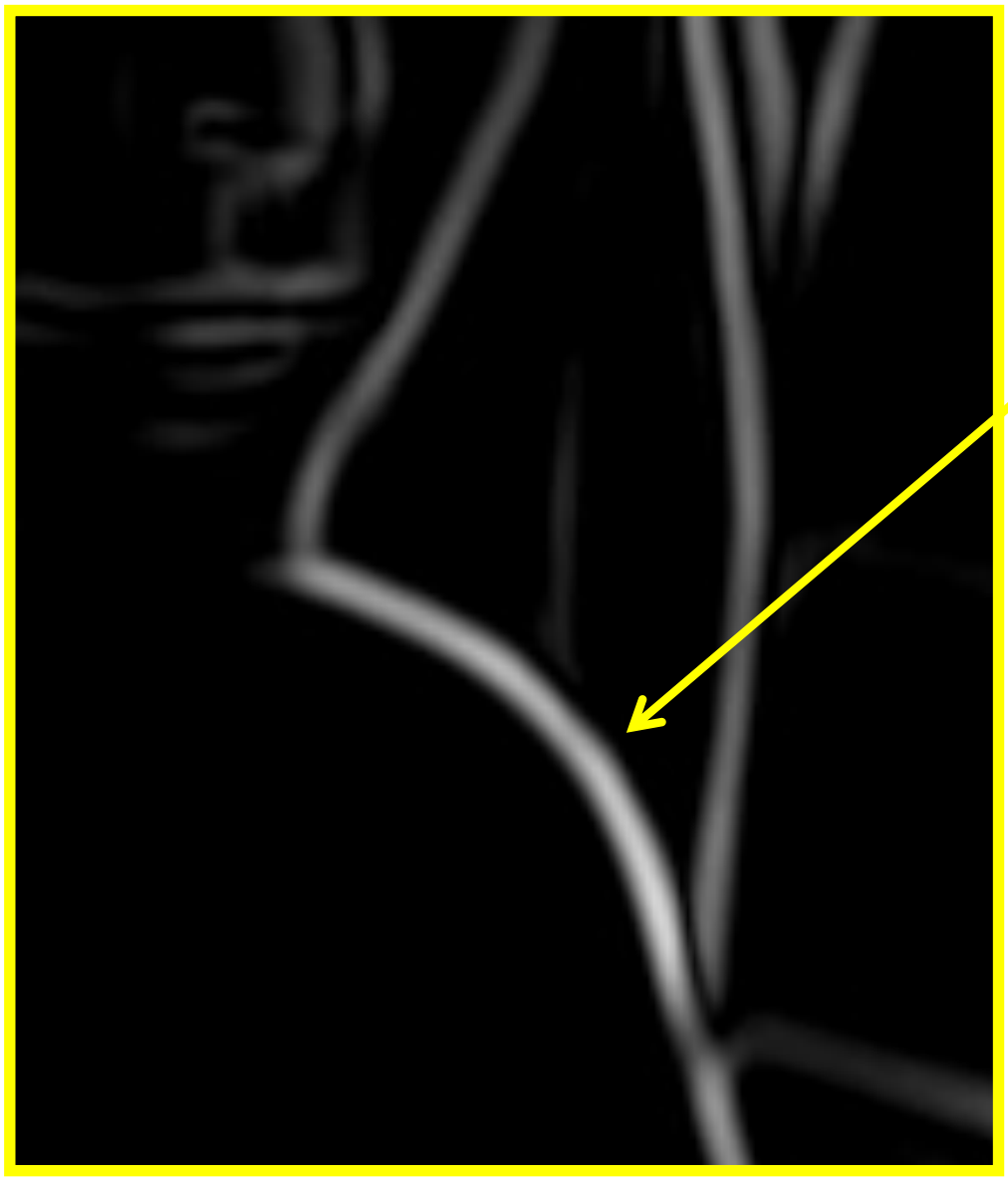
norm of the gradient

The Canny edge detector

The Canny edge detector







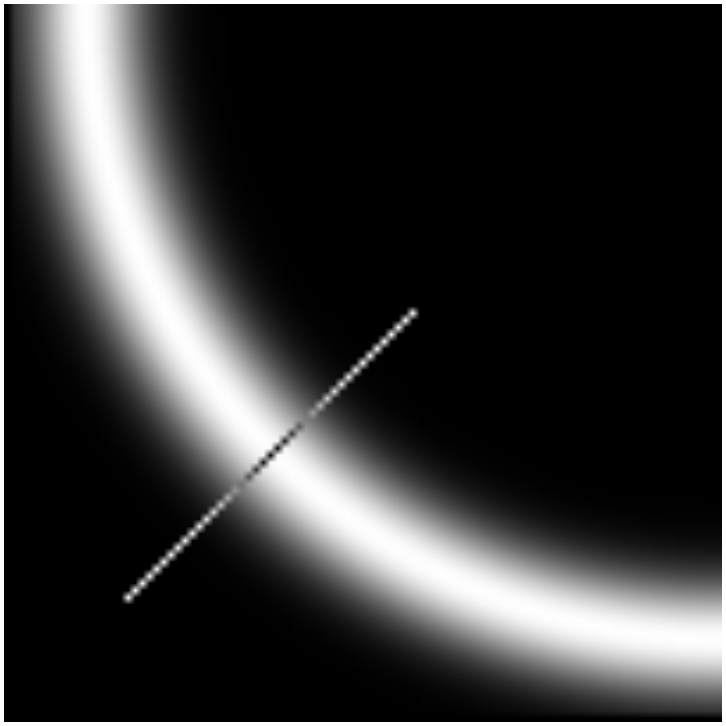
How to turn these thick regions of the gradient into curves?

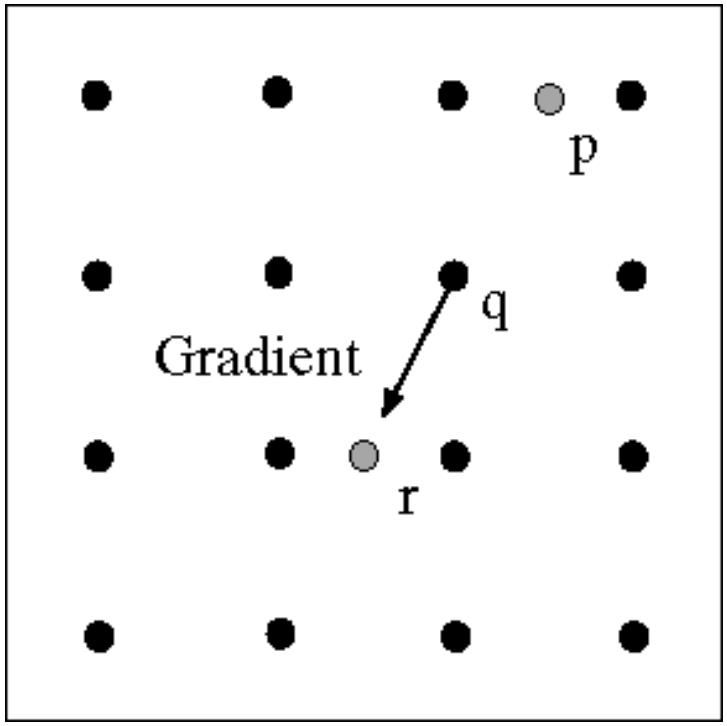
thresholding

Non-maximum suppression

Check if pixel is local maximum along gradient direction, select single max across width of the edge

- requires checking interpolated pixels p and r





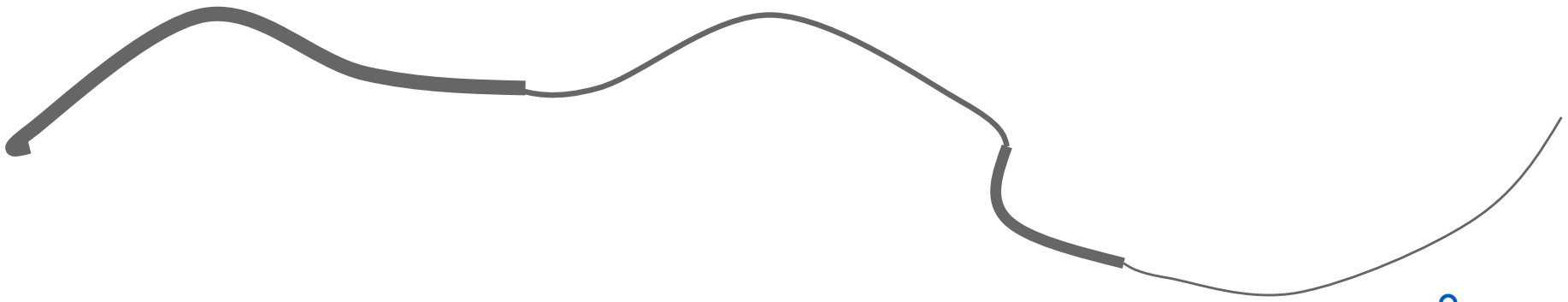
The Canny edge detector

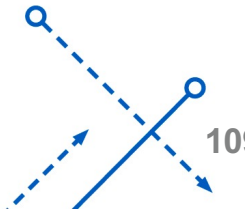
thinning
(non-maximum suppression)

Problem:
pixels along
this edge
didn't
survive the
thresholding

Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.





Source: S. Seitz

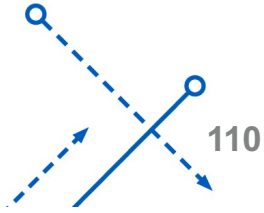
Hysteresis thresholding

original image

high threshold
(strong edges)

low threshold
(weak edges)

hysteresis threshold



Source: L. Fei-Fei

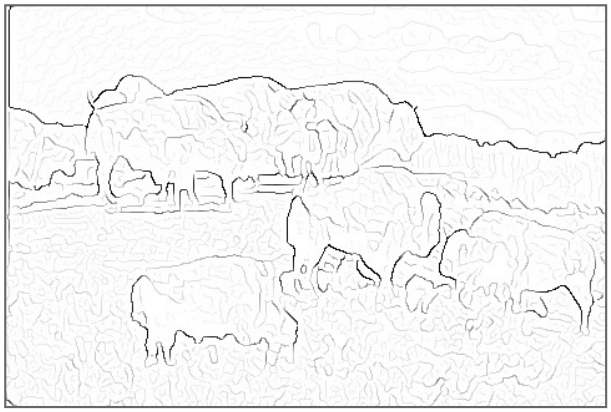
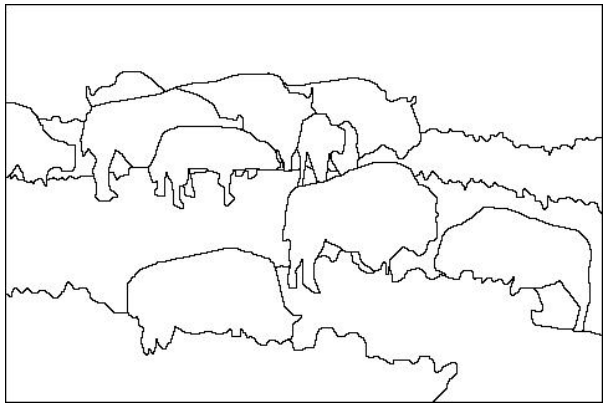
Object boundaries vs. edges

Background

Texture

Shadows

Edge detection is just the beginning…



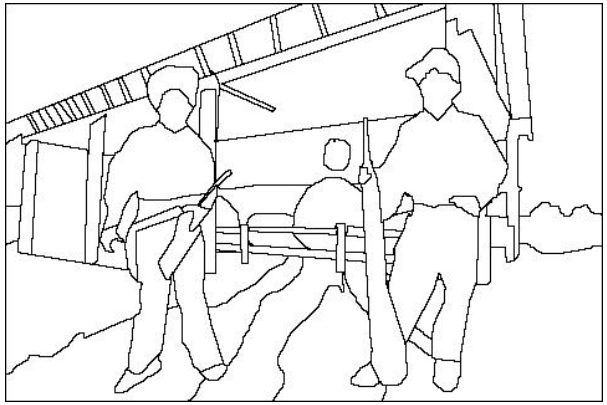


image | gradient magnitude | human segmentation

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

Source: L. Lazebnik

Important Concepts

- Filtering
  - Linear filters, Correlation, Equivariance, Invariance
  - Gaussian Filter, Smoothing, Median filter
- Template Matching
  - Cross-correlation
- Edge Detection
  - Image differentiation and gradient
  - Derivative theorem of convolution
  - 2D edge detection filters, Sobel operator
  - Canny edge detector, Hysteresis thresholding