CSCI 4830 / 5722
Computer Vision

University of Colorado **Boulder**

CSCI 4830 / 5722
Computer Vision

Dr. Ioana Fleming
Spring 2019
Lecture 11

Reminders

Submissions:

- Homework 2: due Wed 2/13 at 11 pm

Readings:

- Szeliski:
  - chapter 3 (filters, changing resoluti
    pyramids, warping)
  - chapter 4.1 (points) and 4.2 (edge detection)
- P&F Ch. 4,5

Today

- Gaussian filter – blurring
- Canny edge detector

Linear filters

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

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

Box/Mean

Prewitt

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Roberts

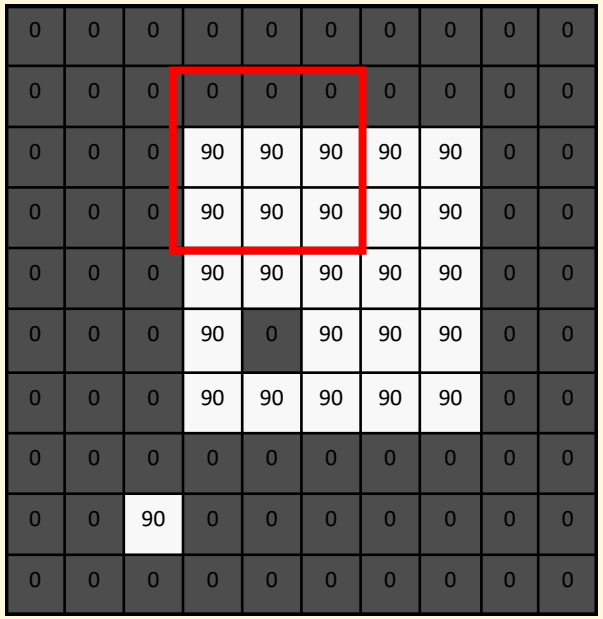
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel

Convolution

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

- Each pixel in output image is
  - weighted average of window of pixels in input image
  - weights stay the same
  - window centered on pixel
  - window weights form the convolution **kernel**



Key properties

- **Linearity:** filter($f_1 + f_2$) = filter(f_1) + filter(f_2)
- **Shift invariance:** same behavior regardless of pixel location: filter(shift(f)) = shift(filter(f))
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

Properties in more detail

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another:

    $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter:

    a $* (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k\,(a * b)$
- Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Convolution - Example I

- Example: replace each pixel with an average of neighbors (mean filter or box filter)
  - (2k+1) by (2k+1) window centered at pixel

$$\mathcal{R}_{ij} = \frac{1}{(2k+1)^2} \sum_{u=i-k}^{u=i+k} \sum_{v=j-k}^{v=j+k} \mathcal{F}_{uv}.$$

i, j' th pixel in output image

u, v' th pixel in input image

If k = 1, then Kernel_size = ?

If k = 2, then Kernel_size = ?

Forming a weighted average

How does our formula change when we don't have equal weights?

- Place window over image at pixel, sum weighted pixels

Weights

$$R_{ij} = \sum_{u,v} H_{i-u,j-v} F_{u,v}$$

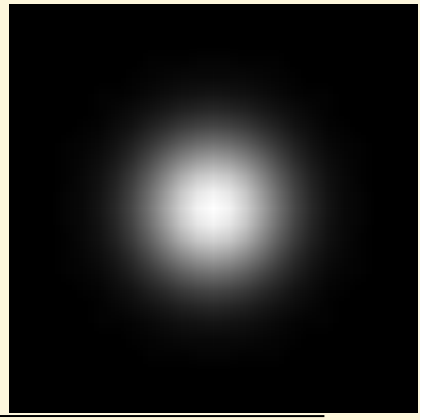
i, j' th pixel in output image

u, v' th pixel in input image

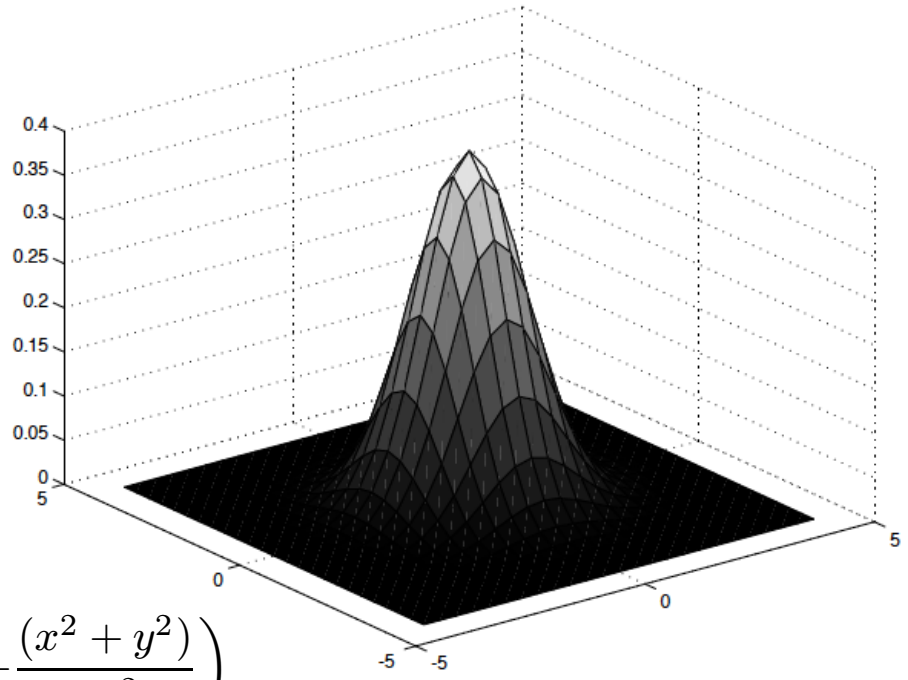
Convolution - Example II

- Averaging neighbors yields poor smoothing
  - look at edges - ringing effects
  - distant neighbors have the same effect as nearby neighbors
- Idea:
  - distant neighbors have small weights, nearby have large
  - weights from Gaussian

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



The Gaussian Smoothing Kernel

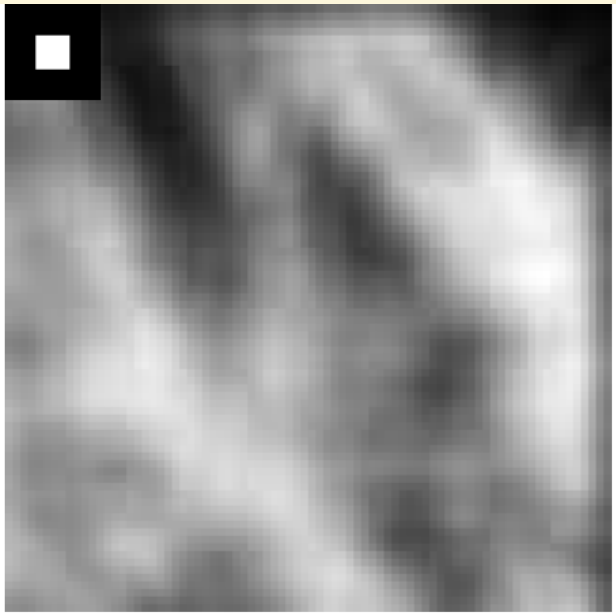


$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right)$$

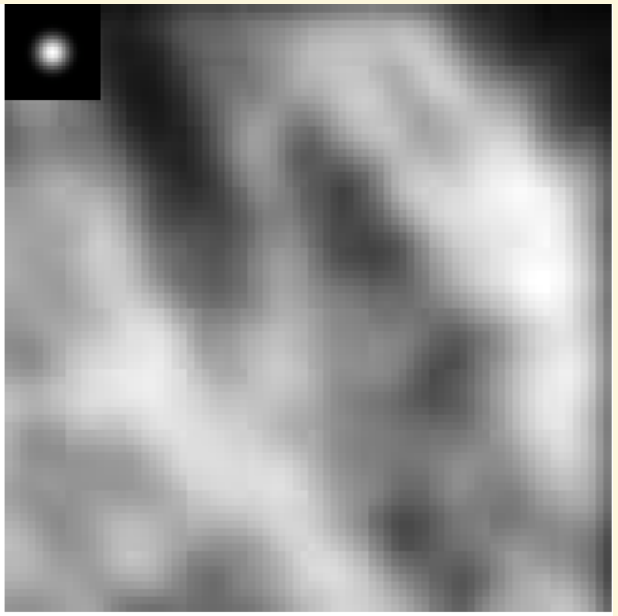
FIGURE 4.2: The symmetric Gaussian kernel in 2D. This view shows a kernel scaled so that its sum is equal to one; this scaling is quite often omitted. The kernel shown has $\sigma = 1$. Convolution with this kernel forms a weighted average that stresses the point at the center of the convolution window and incorporates little contribution from those at the boundary. Notice how the Gaussian is qualitatively similar to our description of the point spread function of image blur: it is circularly symmetric, has strongest response in the center, and dies away near the boundaries.

Smoothing with a Gaussian

Input image



Output image, average



Output image, gaussian weights

Figure 7.1, Ponce & Forsyth

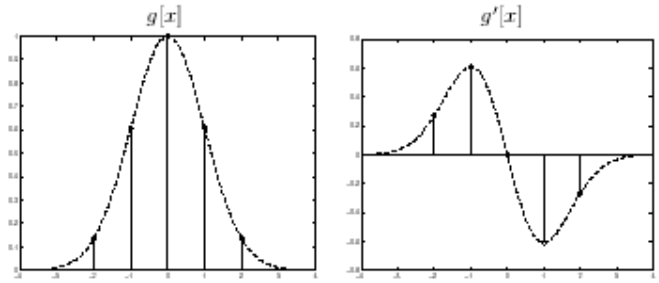
Mean filter vs Gaussian filter?

1. Reducing noise:
   - Mean filter is optimal for reducing random noise in spatial domain (image space).
2. Separating frequencies:
   - However Mean filter is the worst filter for frequency domain, with little ability to separate one band of frequencies from another. Gaussian filter has better performance in frequency domain.
3. Time
   - Mean filter is faster, can be implemented without convolution. Gaussian filter is slower, even if separable

How big should the mask be?

The std. dev of the Gaussian σ determines the amount of smoothing.

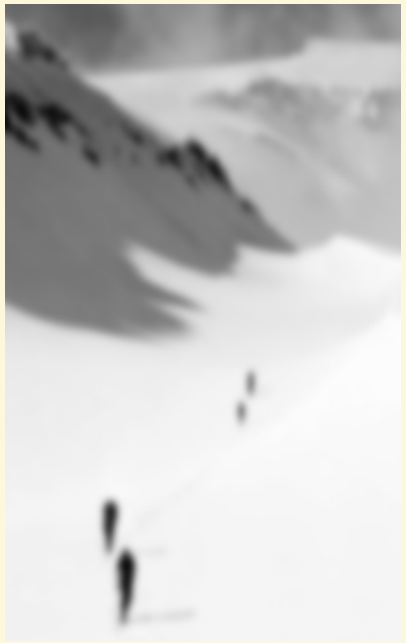
- if standard deviation of the Gaussian is very small — say smaller than one pixel — the smoothing will have very little effect, because the weights for all pixels off the center will be very small;
- for a larger standard deviation, the neighboring pixels will have larger weights in the weighted average, which means in turn that the average will be strongly biased toward a consensus of the neighbors — this will be a good estimate of a pixel's value, and the noise will largely disappear, at the cost of some blurring;
- a kernel that has very large standard deviation will cause much of the image detail to disappear along with the noise.



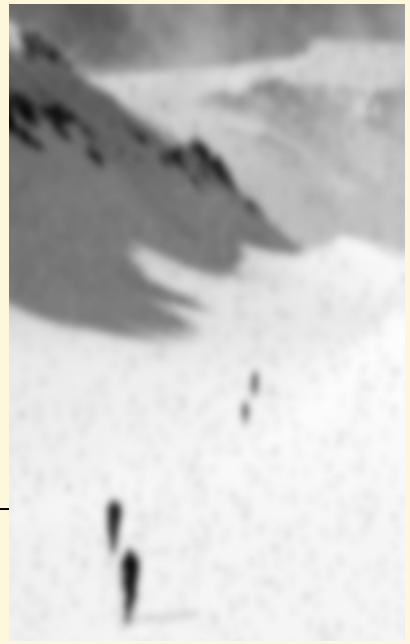
Rule of thumb: kernel size = 3 * σ in every direction

Original

Smoothed at σ=1

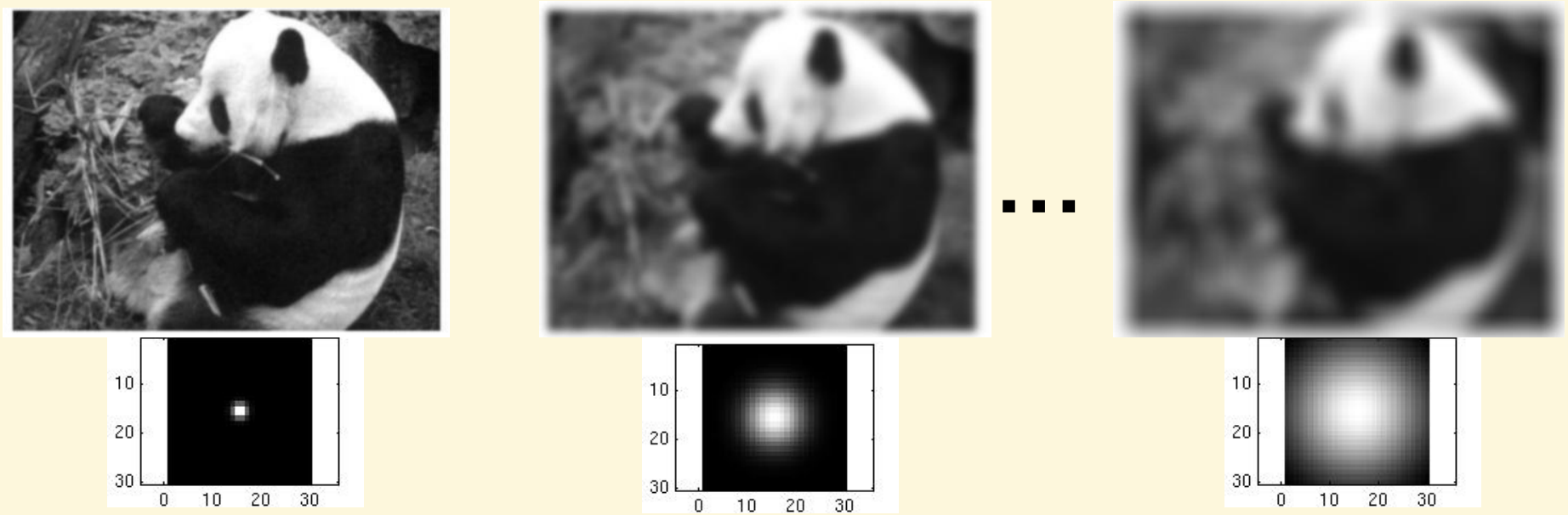


Smoothed at σ=2



Smoothing with a Gaussian

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



Slide credit: Kristen Grauman

Effect of σ on derivatives

σ = 1 pixel

σ = 3 pixels

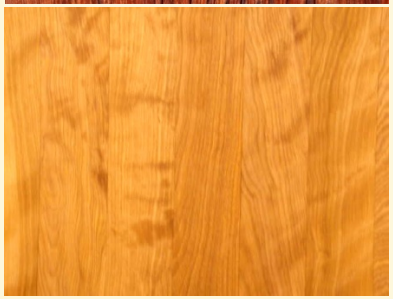
The apparent structures differ depending on Gaussian's scale parameter.

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

Slide credit: Kristen Grauman

So, what scale to choose?

It depends what we're looking for.



Slide credit: Kristen Grauman

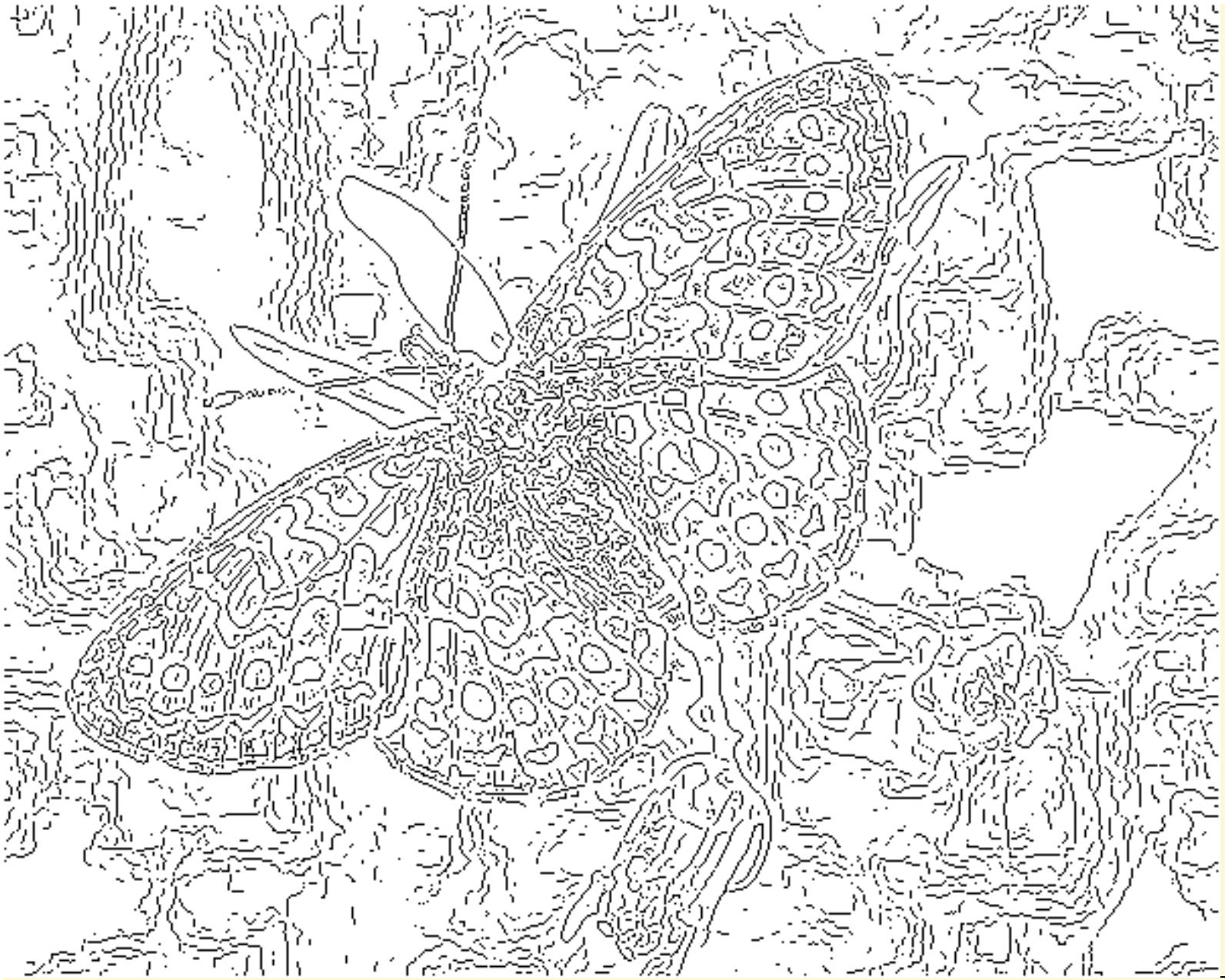
Reminder: thresholding

- Choose a threshold value a
- Set any pixels less than a to zero
- Set any pixels greater than or equal to a to one

$$I^{'}(x,y) = \begin{cases} 1 & \text{if} \quad I(x,y) \geq a \\ 0 & \text{else} \end{cases}$$

University of Colorado **Boulder**

magnitude of the gradient



fine scale
high threshold

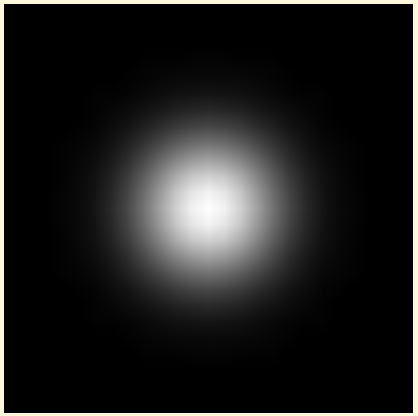
coarse scale,
high threshold

coarse scale
low threshold

Convolution - Example II

- Gaussian filter:
  - distant neighbors have small weights, nearby have large
  - weights from Gaussian

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



- Is the Gaussian filter separable?

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
  - Convolving two times with Gaussian kernel of width d is same as convolving once with kernel of width $\sqrt{d}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians
  - Useful: can convolve all rows, then all columns
  - How does this change the computational complexity?

Source: K. Grauman

Separability of the Gaussian filter

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

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

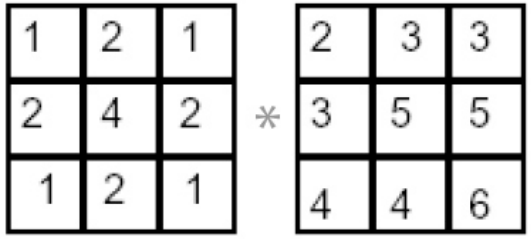
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

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

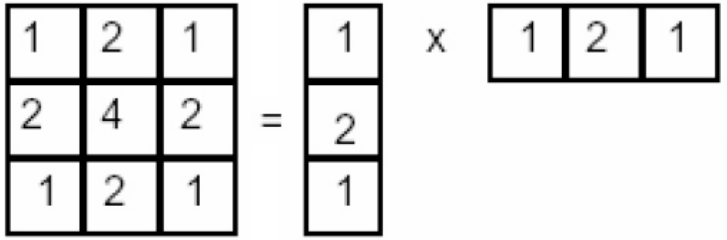
Source: D. Lowe

Separability example

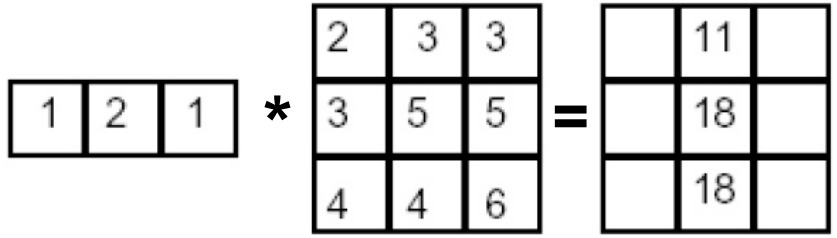
2D convolution
(center location only)



The filter factors
into a product of 1D
filters:



Perform convolution
along rows:



Followed by convolution
along the remaining column:

Source: K. Grauman

How to determine whether a given filter is separable

If a matrix is an outer product of two vectors, its rank is 1

Let's test this:

```
>> averaging = ones(5,5) / 25;
>> rank(averaging)
ans =
      1

>> sobel = [-1 0 1; -2 0 2; -1 0 1];
>> rank(sobel)
ans =
      1
```

How to determine whether a given filter is separable

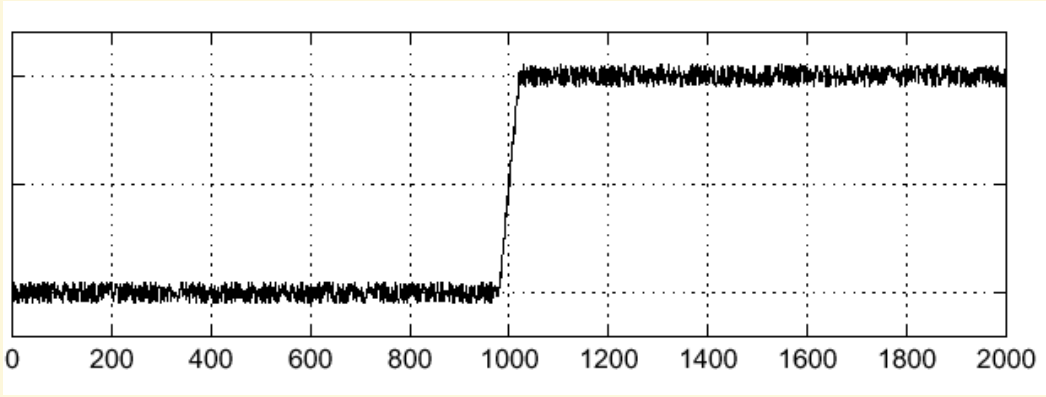
The two-dimensional Gaussian is the only radially symmetric function that is also separable:

```
>> gaussian = fspecial('gaussian');
>> rank(gaussian)
ans =
     1
```

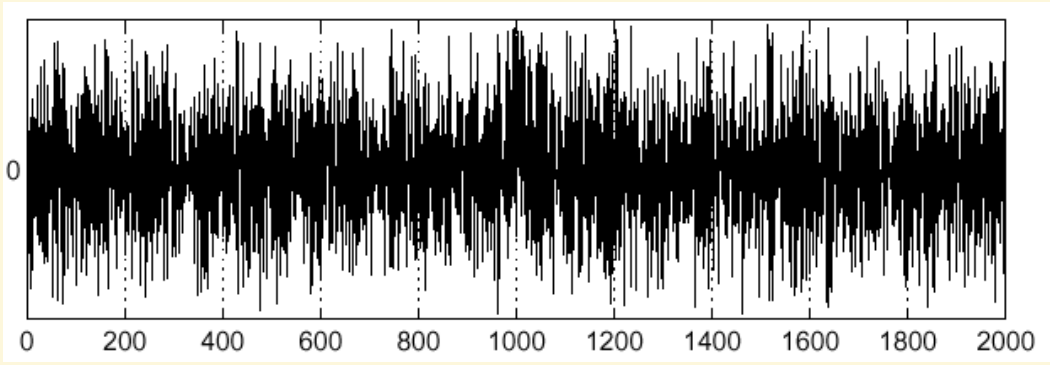
Effects of noise

- 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?

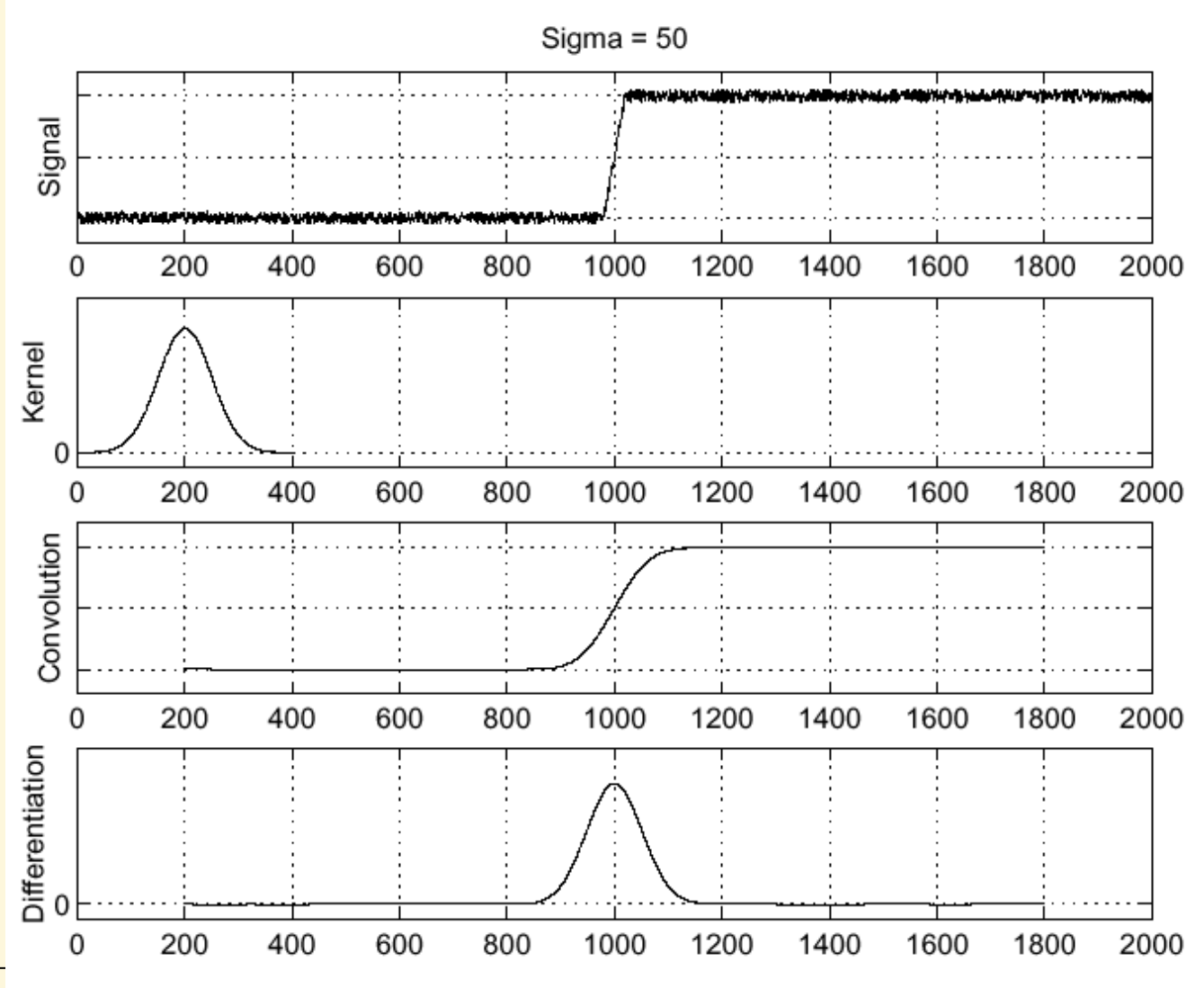
Solution: smooth first

f

h

$h \star f$

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



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Properties of Convolution

- Commutative: f * g = g * f
- Associative : (f * g) * h = f * (g * h)
- Differentiation:

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

Derivative theorem of convolution

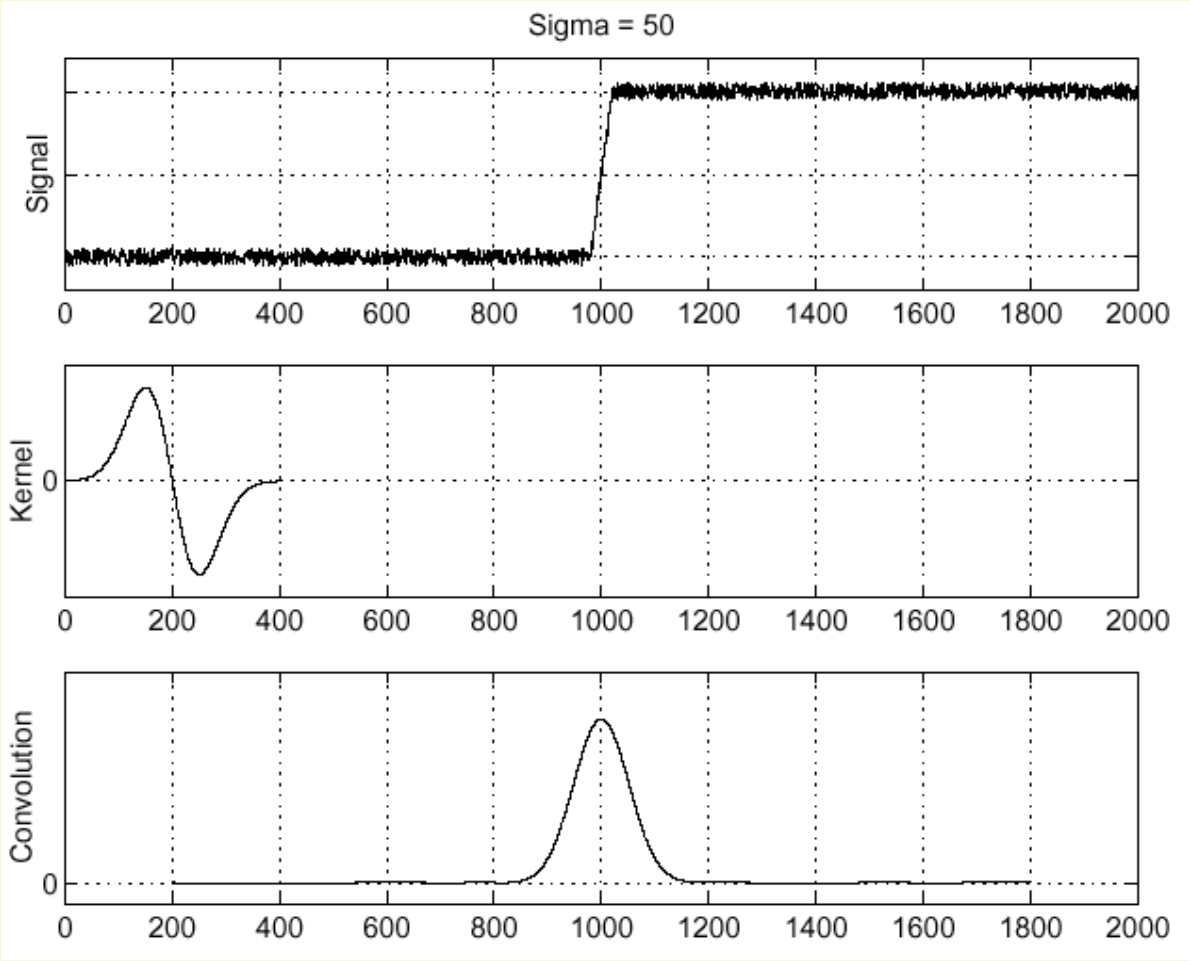
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

f

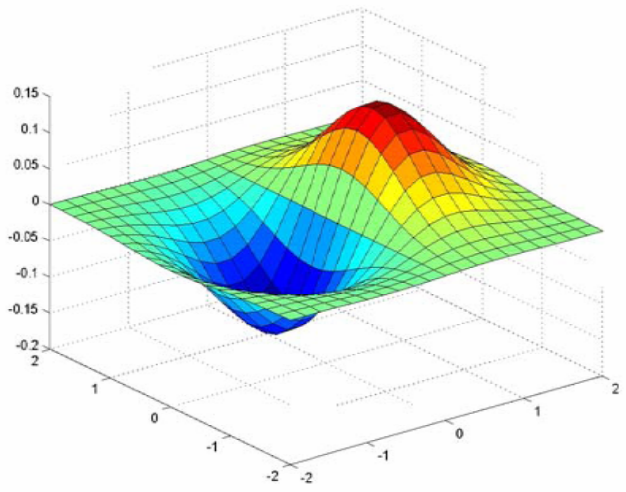
$\frac{\partial}{\partial x}h$

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

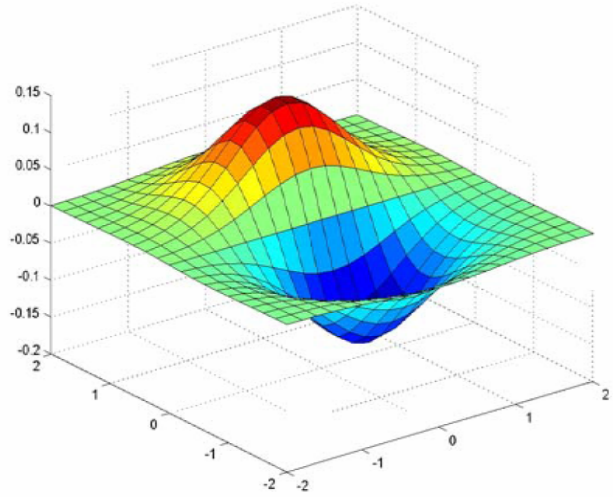


Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

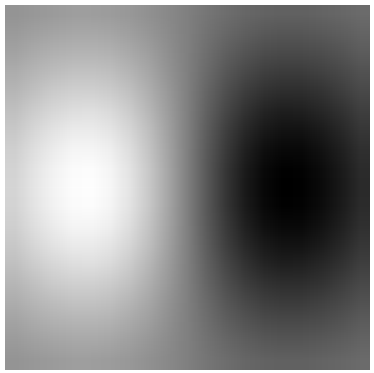
Derivative of Gaussian filters

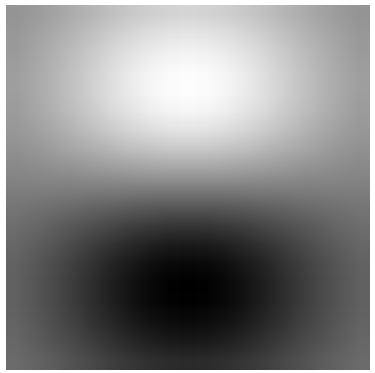


x-direction



y-direction



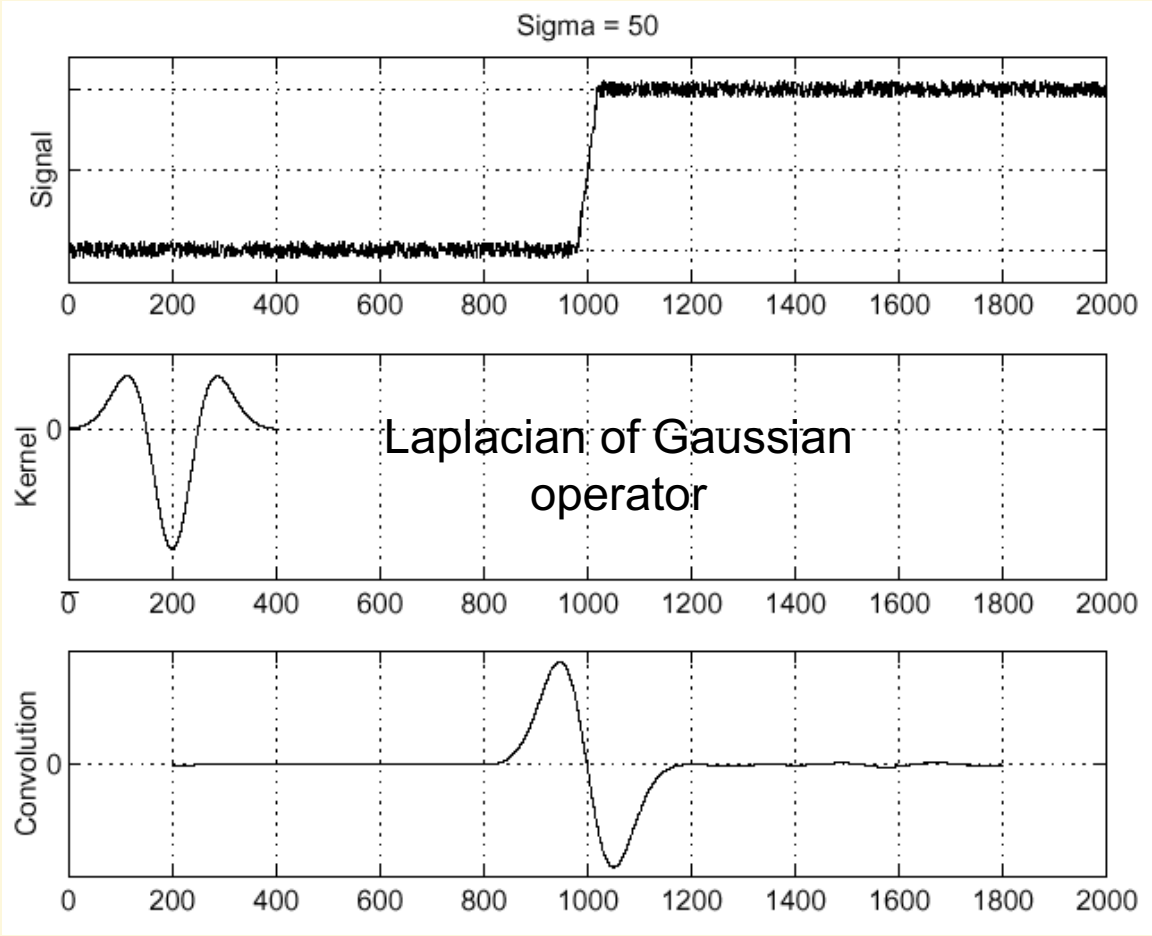


Source: L. Lazebnik

Laplacian of Gaussian

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

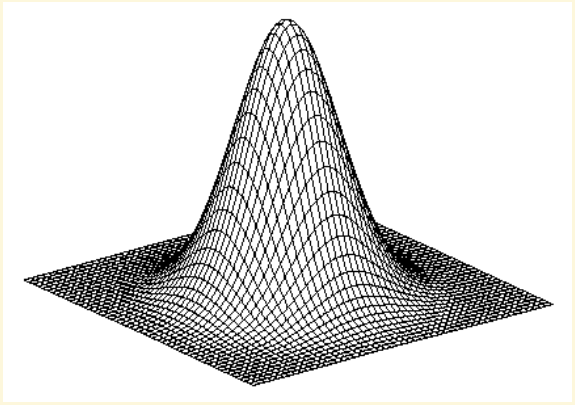
$$\left(\frac{\partial^2}{\partial x^2} h\right) \star f$$



Where is the edge?

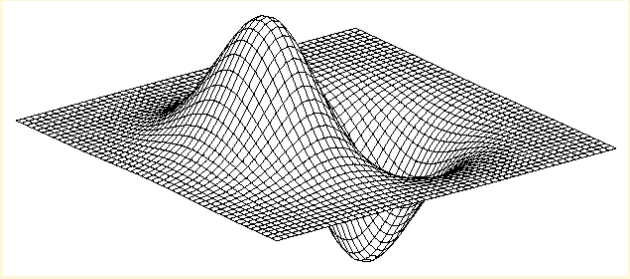
Zero-crossings of bottom graph

2D edge detection filters



Gaussian

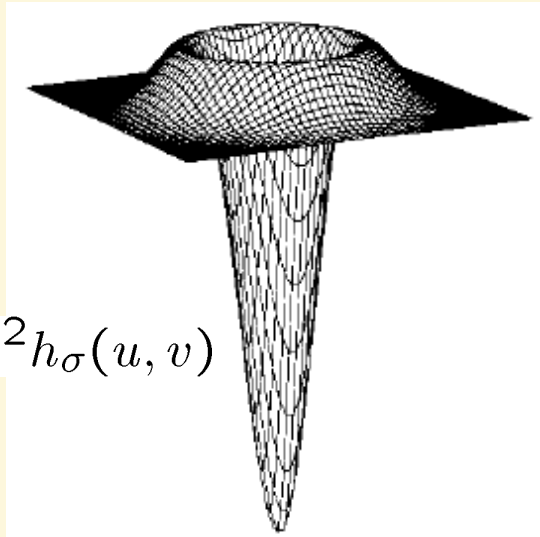
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



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:

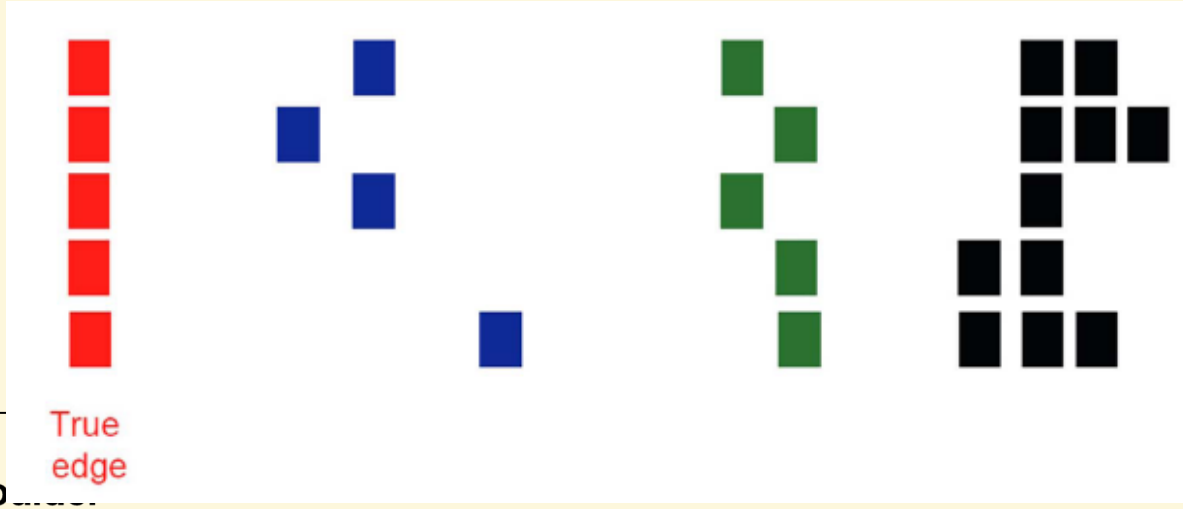
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Notes on Kernels

- Smoothing
  - Values positive
  - Sum to 1: constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- 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

Optimal Edge Detection

- Edge detector should have:
  - Good Detection. Filter responds to edge, not noise.
  - Good Localization: detected edge near true edge.
  - Single Response: one per edge.



Optimal Edge Detection: Canny

From gradients to edges

- Primary steps:
  - Smoothing: suppress noise
  - Edge enhancement: filter for contrast
  - Edge localization
  - Determine which local maxima from filter output are actually edges vs. noise
    - ***thresholding, thinning***

Canny edge detector

- Filter image with derivative of Gaussian (DoG)
- Find magnitude and orientation of gradient
- **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

- MATLAB: `edge(image, 'canny');`
- `>>help edge`

The Canny edge detector

original image (Lena)

The Canny edge detector

magnitude of the gradient

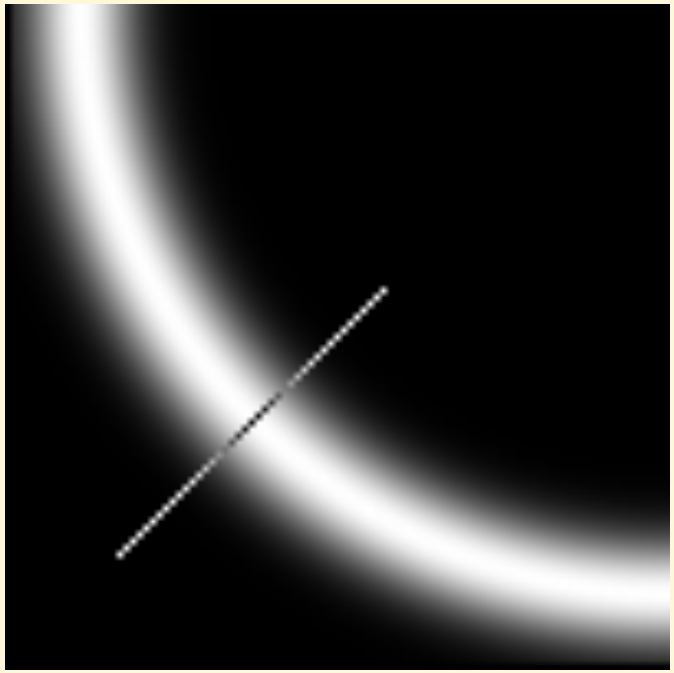
The Canny edge detector

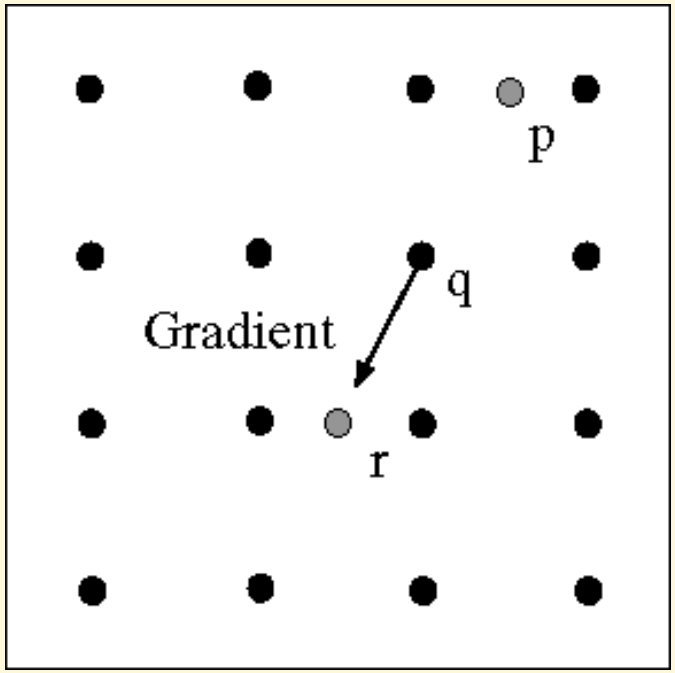
thresholding

The Canny edge detector

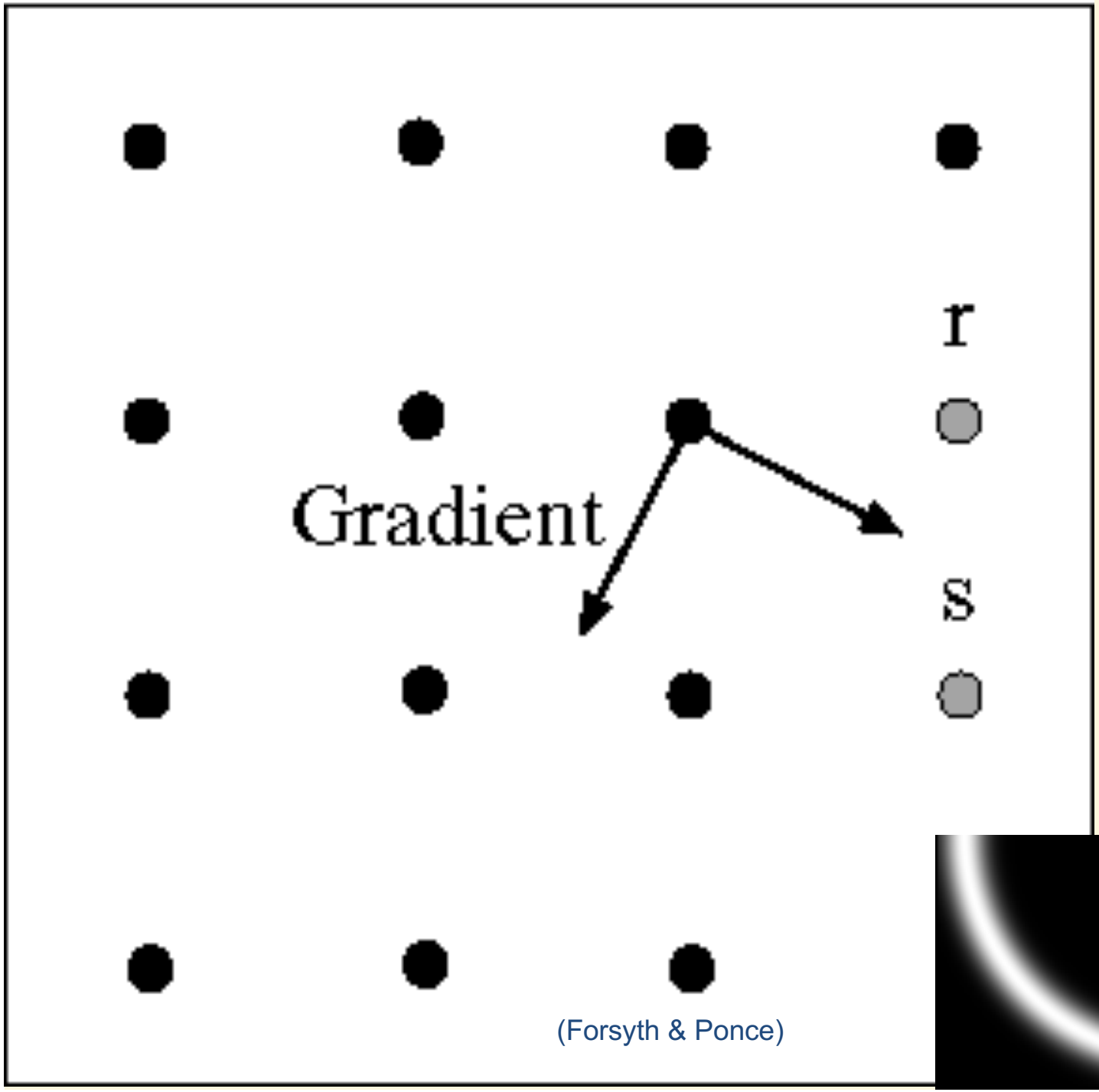
thinning (non-maximum suppression)

Non-maximum suppression





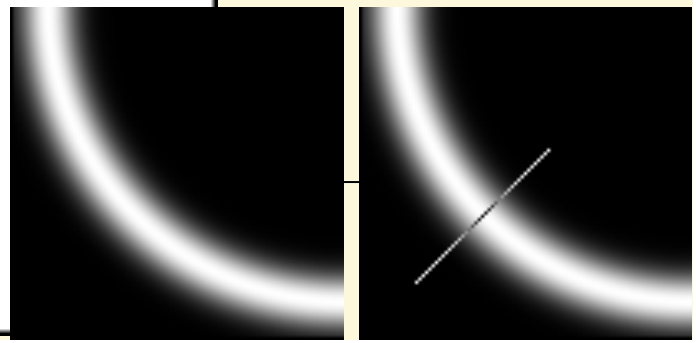
- Check if pixel is local maximum along gradient direction
  - requires checking interpolated pixels p and r



(Forsyth & Ponce)

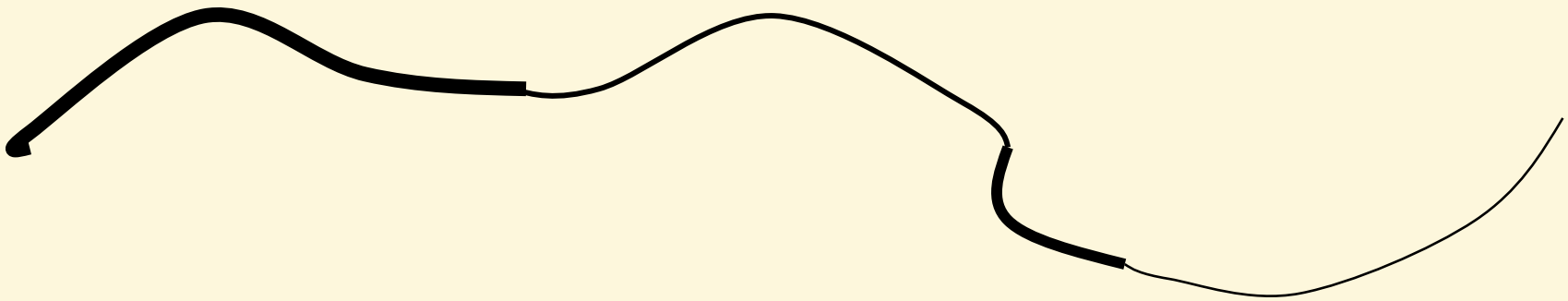
Predicting the next edge point

Assume the marked point is an edge point. We then construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Hysteresis

- 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.***



Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

A Quick Note

- Matlab's image processing toolbox provides *edge* function to find edges in an image.
- *edge* function supports six different edge-finding methods: Sobel, Prewitt, Roberts, Laplacian of Gaussian, Zero-cross, and Canny.