Gradients & Edges

CS 4243 Computer Vision & Pattern Recognition

Angela Yao

Recap & Outline

Last Week

- Linear filtering, cross-correlation vs. convolution operations
- Denoising, sharpening, template matching
- Gaussian & Laplacian image pyramids
- Non-linear filtering, e.g. median filters

Today's Lecture

- Motivation for studying image edges
- Derivative filters to extract gradients
- Need for smoothing
- Canny edge detector

Why Study Edges?

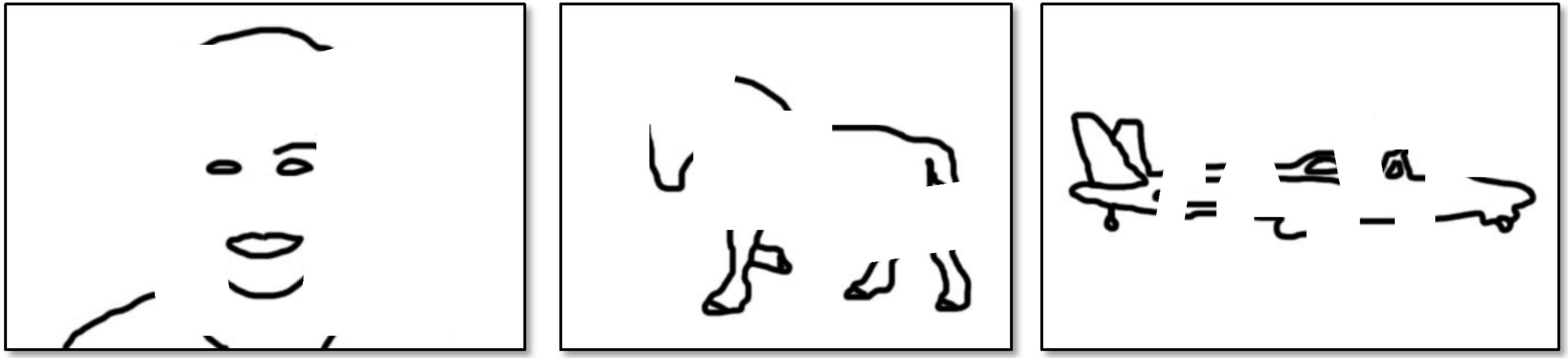


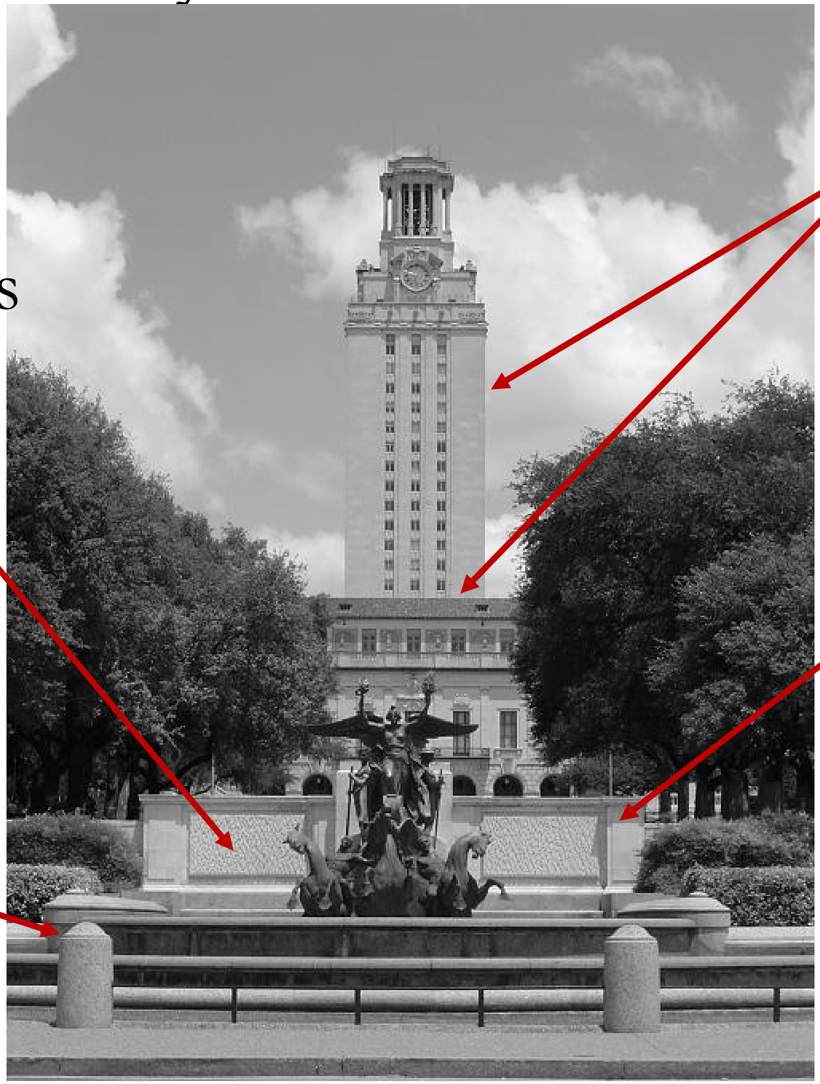
Figure from J. Shotton et al., PAMI 2007

- Human visual system is sensitive to edges
- Edges compactly conveys salient information

Image Edges Convey Information

reflectance changes, textures & appearances

change in surface orientations & shape



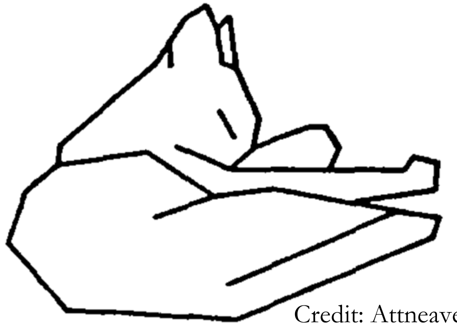
depth discontinuities & object boundaries

shadows

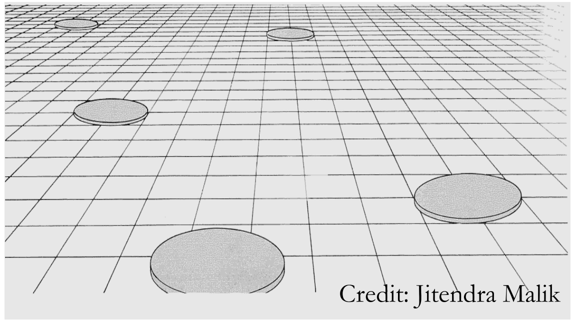
Adapted from K. Grauman

Why Extract Edges?

Resilient to lighting and color → useful for recognition



Credit: Attneave



Credit: Jitendra Malik

Cue for shape and geometry
→ useful for recognition,
3D understanding

Adapted from Bharath Hariharan

Image Gradients

Definition, magnitude & orientation, Sobel filtering

What are Image Edges or Gradients?

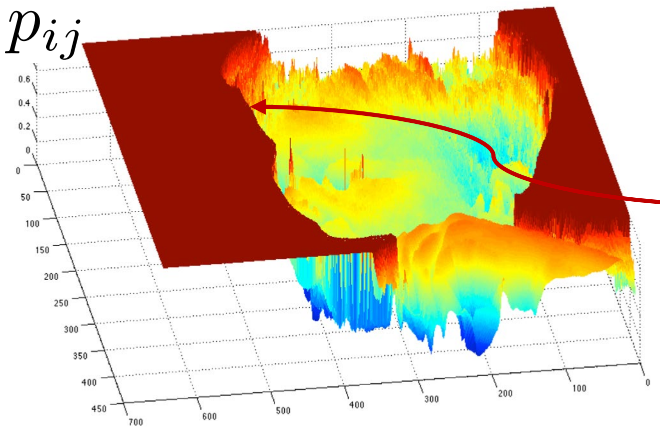
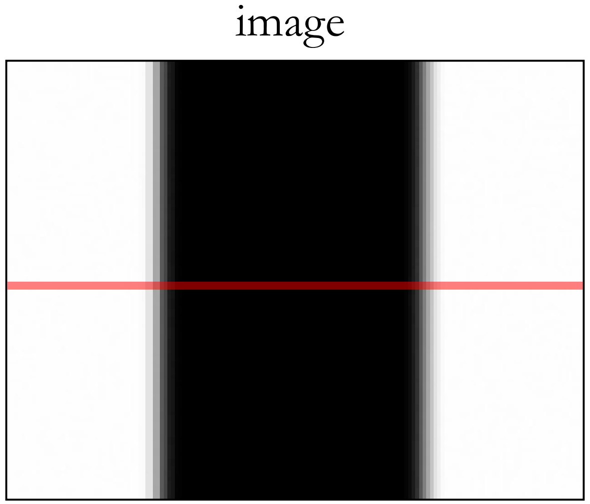
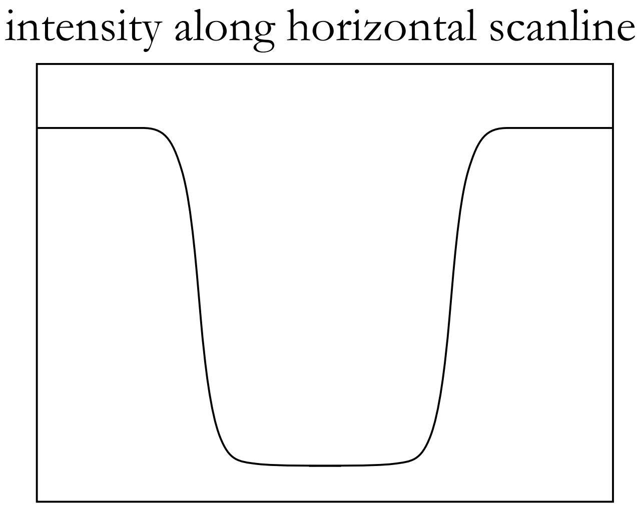
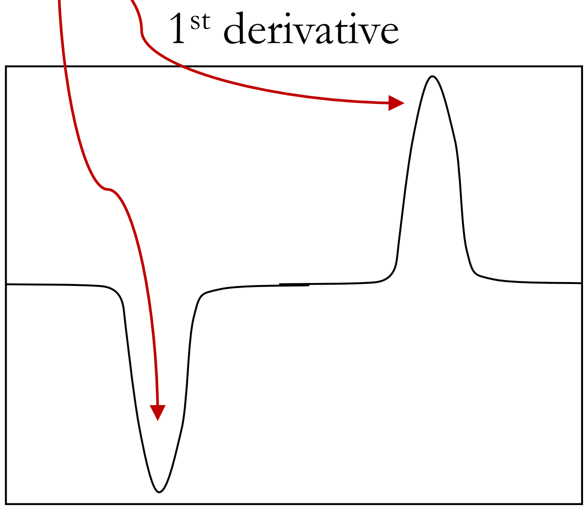


Image edges are sharp discontinuities in the intensity.

Derivatives are large at discontinuities. Edges correspond to extrema of derivative.





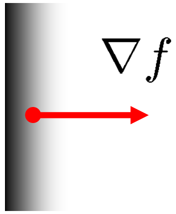


Adapted from I. Gkioulekas, S. Lazebnik

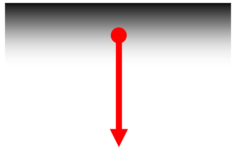
Understanding Image Gradients

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

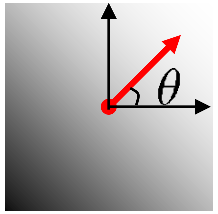
The gradient is a vector that points in the direction of most rapid change in intensity.



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



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

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

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

Images are discrete. How to take (partial) derivatives of discrete signals?

Take finite differences.

Slide Credit: S. Seitz

Finite differences

Recall: definition of a derivative (with a forward difference)

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative with central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

Discrete version: remove limit, set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

What filter kernel does this correspond to?

1D derivative filter

correlation	-1	0	1
convolution	1	0	-1

Adapted from I. Gkioulekas

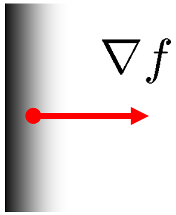
The Sobel filter

Horizontal Sobel filter: Horizontal gradients → vertical lines

$$\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

1D derivative filter

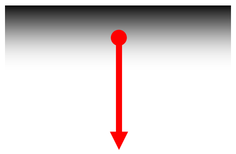
What filter is this? blurring



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

Vertical Sobel filter: Vertical gradients → horizontal lines

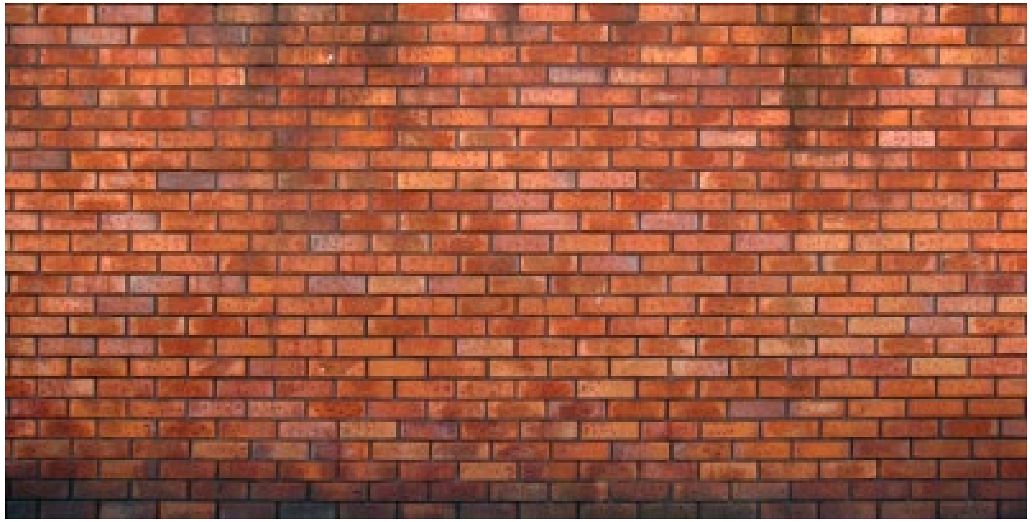
$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array}$$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$

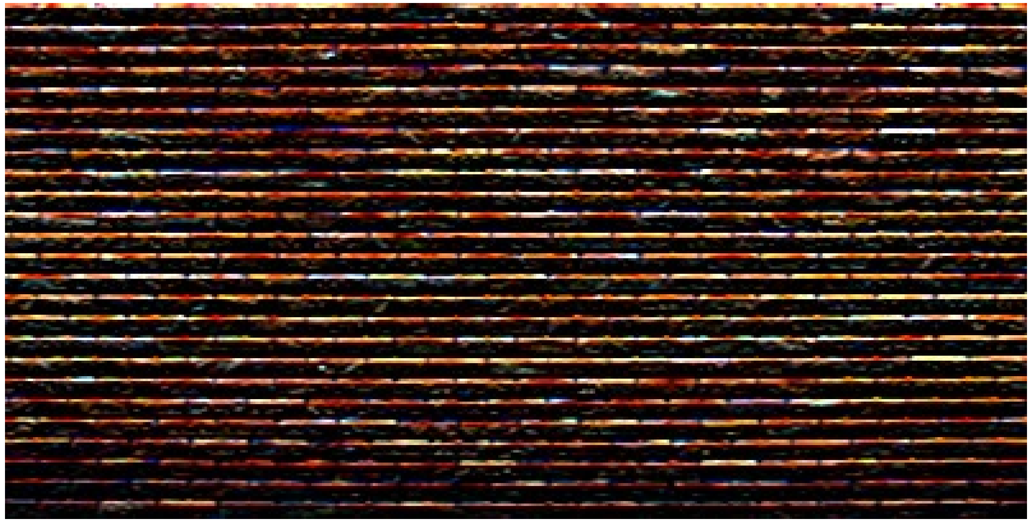
Adapted from I. Gkioulekas

Sobel filter example



original

horizontal Sobel filter



vertical Sobel filter

Slide Credit: I. Gkioulekas

Comparison of Gradient Operators

Most well-known, good tradeoff between simplicity and efficiency.

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

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

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

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

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

0	1
-1	0

1	0
0	-1

https://en.wikipedia.org/wiki/Sobel_operator

Computing Image Gradients

1. Select your favorite gradient filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image **f** to compute image gradient components.

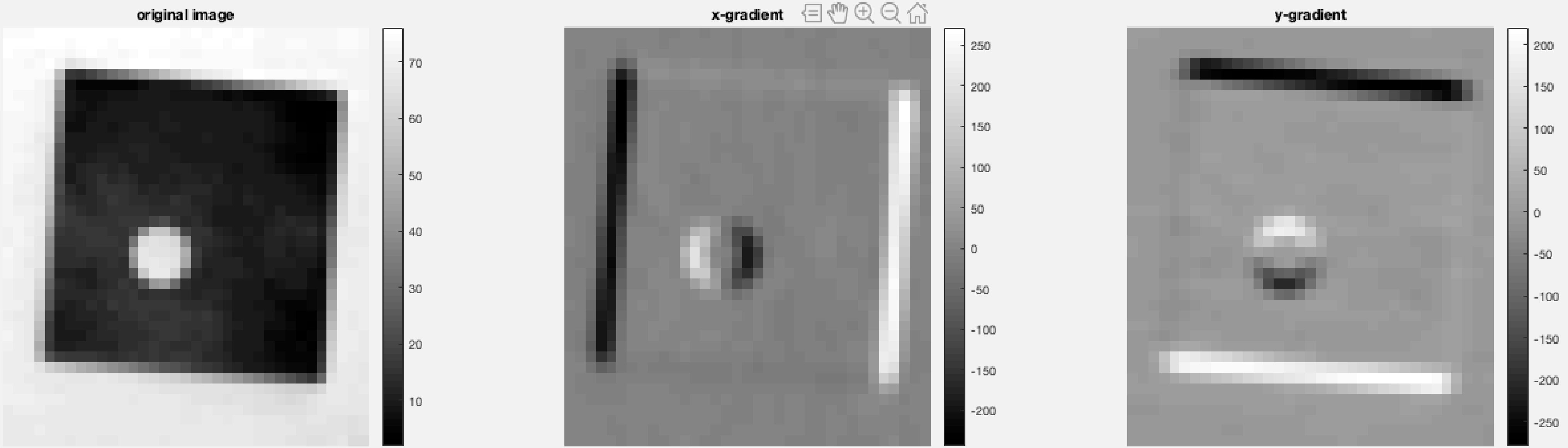
$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x * \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y * \boldsymbol{f}$$

3. Compute gradient orientation and magnitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient orientation magnitude

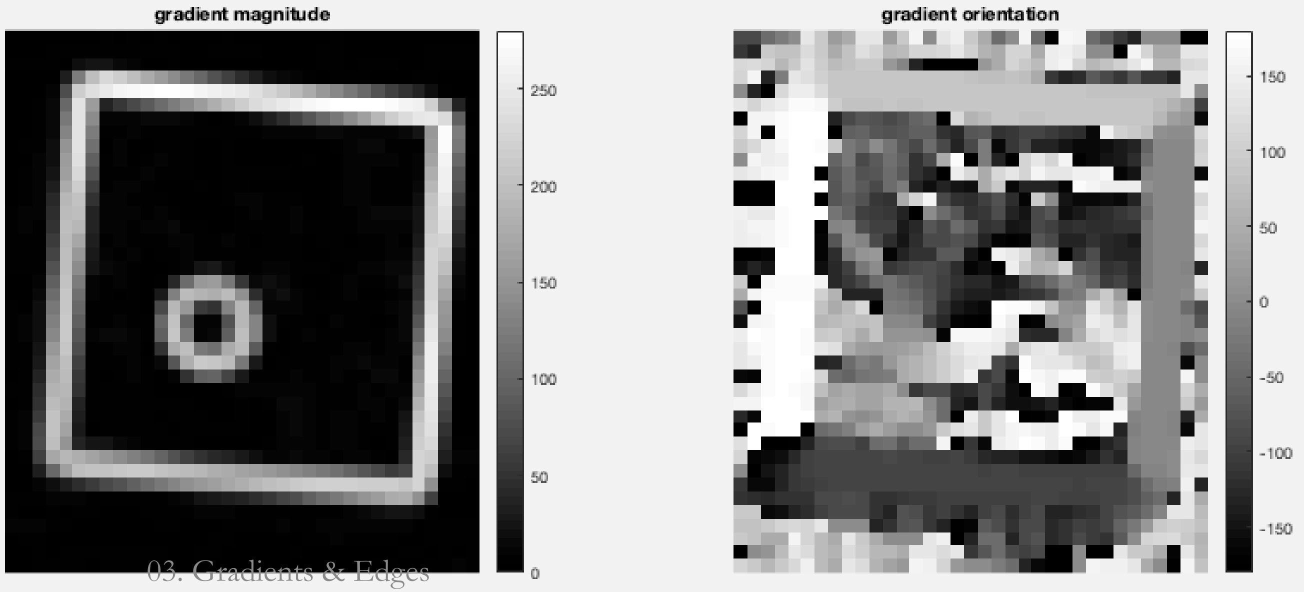
Slide Credit: I. Gkioulekas



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

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

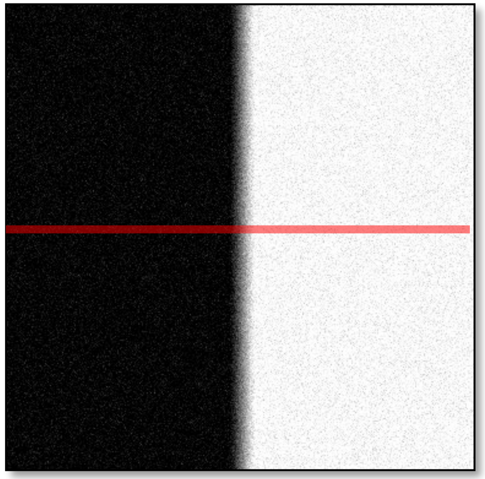
orientation $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$



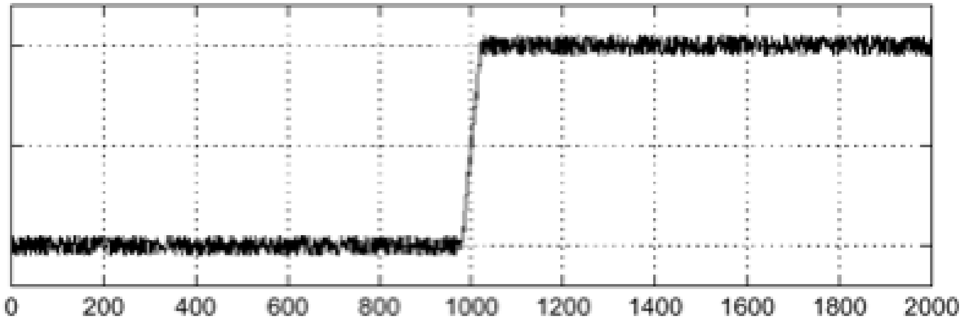
DoG & Laplace Filters

Derivative of Gaussians, Laplacian of Gaussians

How do you find the edge in this signal?

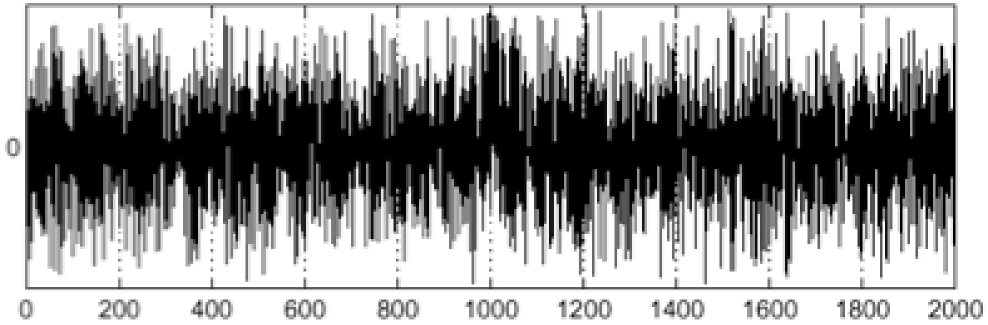


intensity plot



Using a derivative filter:

Where is the extrema here?



Noise is high-frequency.
Differentiation accentuates noise

$$\frac{d \sin \omega x}{dx} = \omega \cos \omega x$$

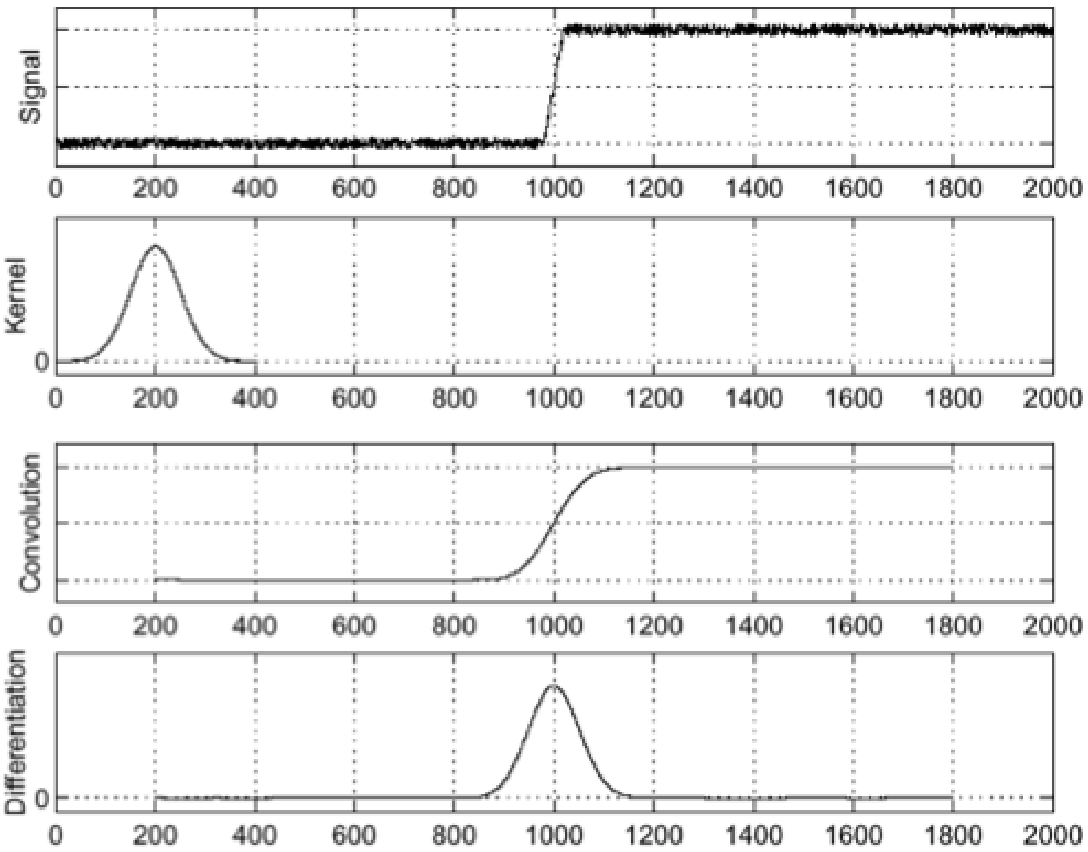
Multiplicative factor proportional to frequency!

Adapted from S. Seitz

Differentiation is very sensitive to noise

When using derivative filters, it is critical to blur first!

f	input
h	Gaussian
$h * f$	blurred
$\frac{\partial}{\partial x}(h * f)$	derivative of blurred



Where is the edge?

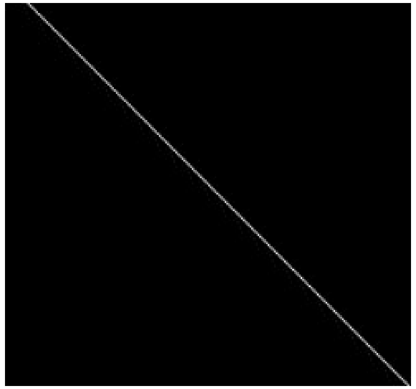
Look for peaks in

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

Adapted from S. Seitz

Trade Off Noise vs. Edge Localization

Image with Edge



Edge Location

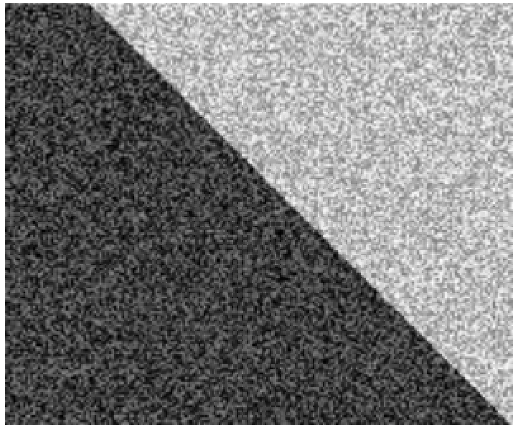
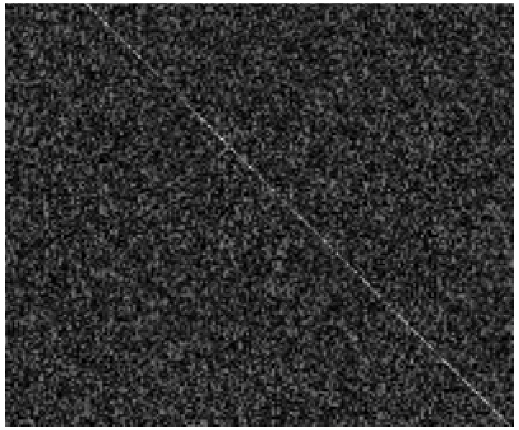
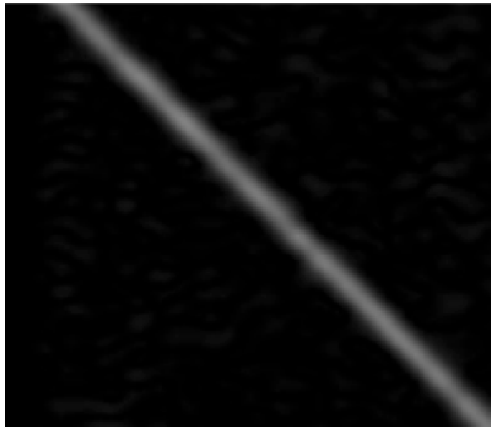


Image + Noise



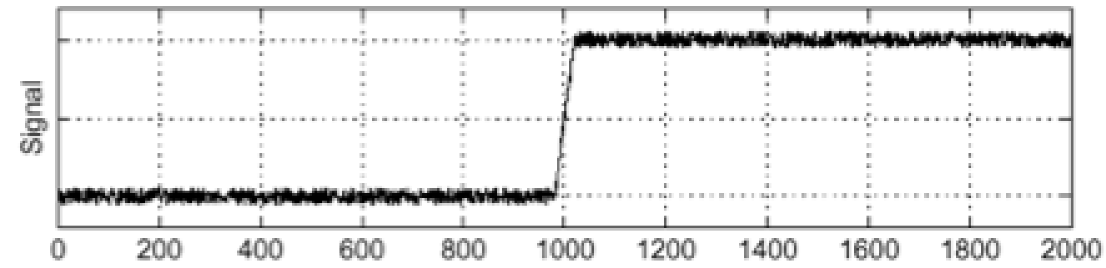
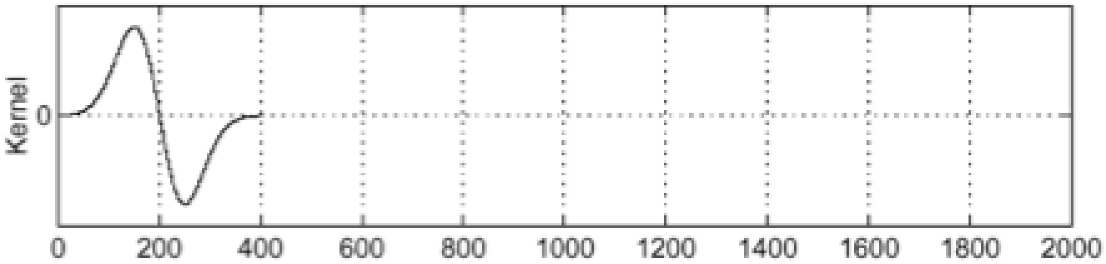
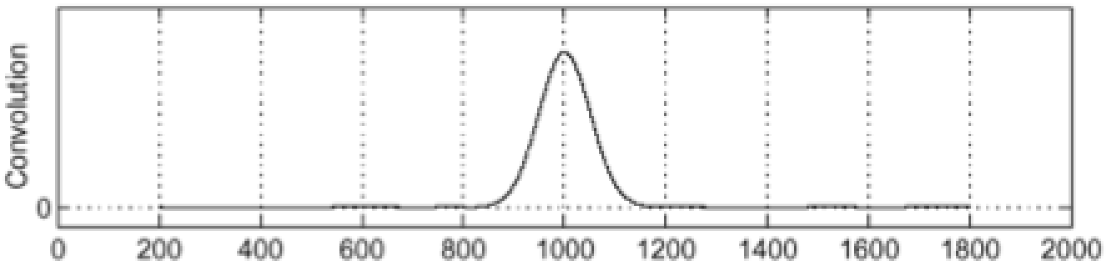
W/out smoothing, edge and noise is picked up by derivative



Smoothing removes the noise, but also blurs the edge

Derivative of Gaussian Filter

Differentiation Property of Convolution: $\frac{\partial}{\partial x}(h * f) = (\frac{\partial}{\partial x}h) * f$

f	input	
$\frac{\partial}{\partial x}h$	derivative of Gaussian	
$(\frac{\partial}{\partial x}h) * f$	output (same as before)	

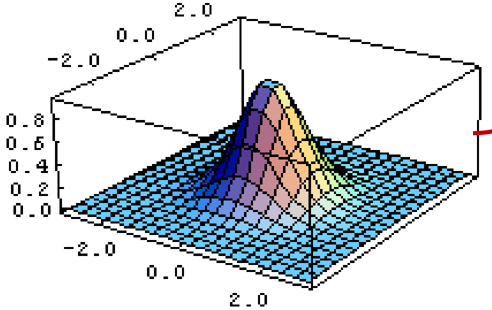
Adapted from S. Seitz

Derivative of Gaussian Filter

Differentiation is a convolution operation.

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

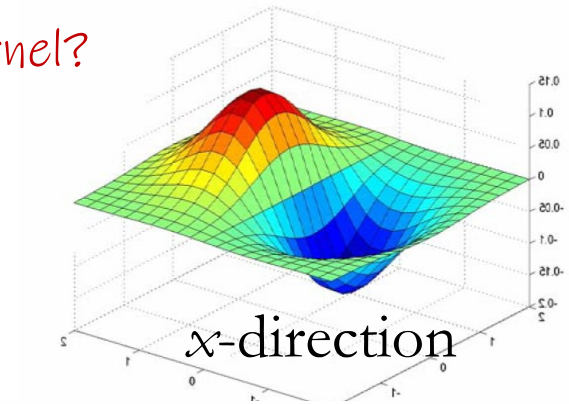
Which kernel?



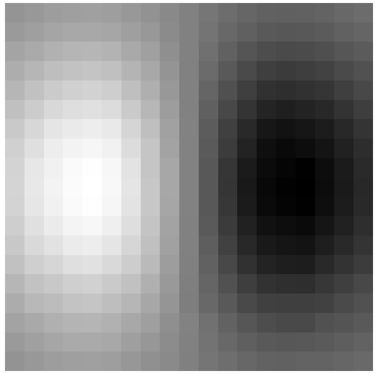
$*$

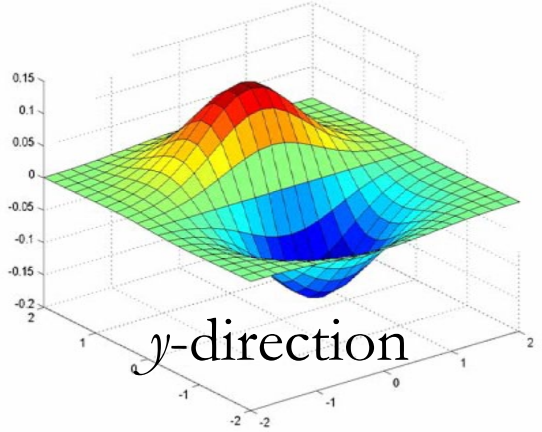
1	0	-1

$=$

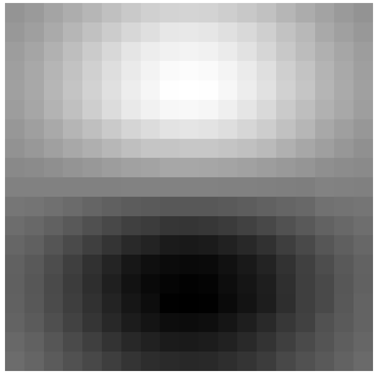


x-direction





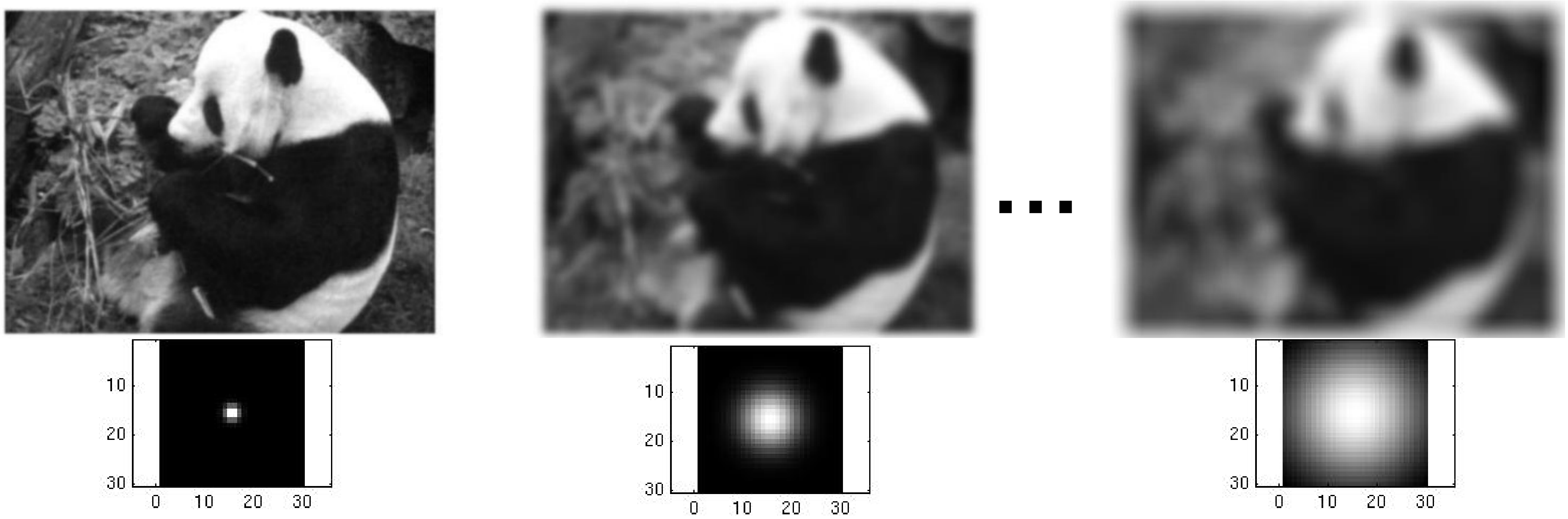
y-direction



What should we set as σ for the Gaussian?

Smoothing with a Gaussian

σ is the “scale” of the Gaussian kernel and determines the extent of smoothing.



Adapted from K. Grauman

Effect of σ on Derivatives

Depending on the applied σ, different structures appear.

Larger σ detects larger-scale edges, smaller σ detects finer edges.

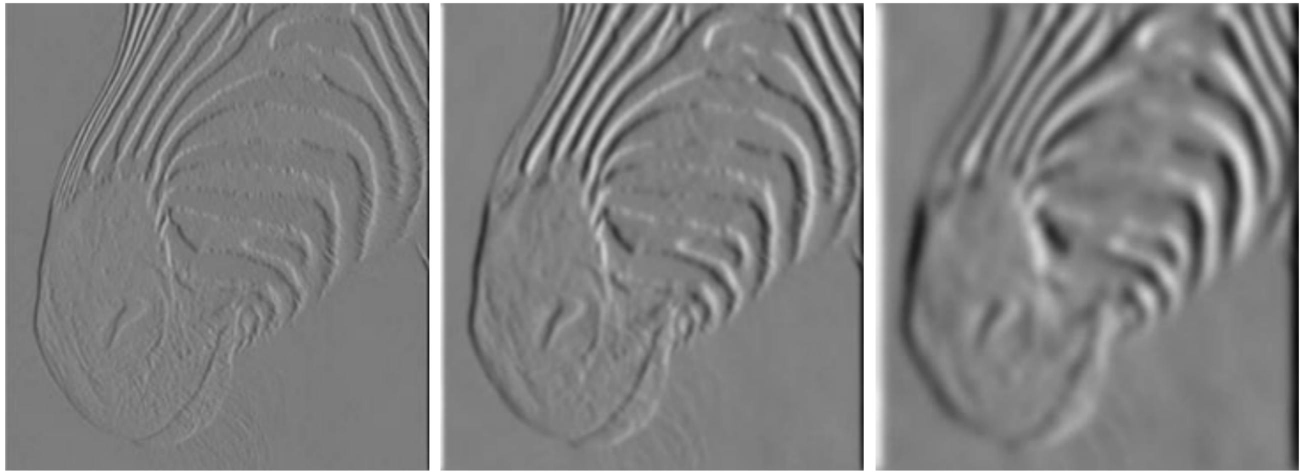


FIGURE 5.3: The scale (i.e., σ) of the Gaussian used in a derivative of Gaussian filter has significant effects on the results. The three images show estimates of the derivative in the x direction of an image of the head of a zebra obtained using a derivative of Gaussian filter with σ one pixel, three pixels, and seven pixels (**left** to **right**). Note how images at a finer scale show some hair, the animal's whiskers disappear at a medium scale, and the fine stripes at the top of the muzzle disappear at the coarser scale.

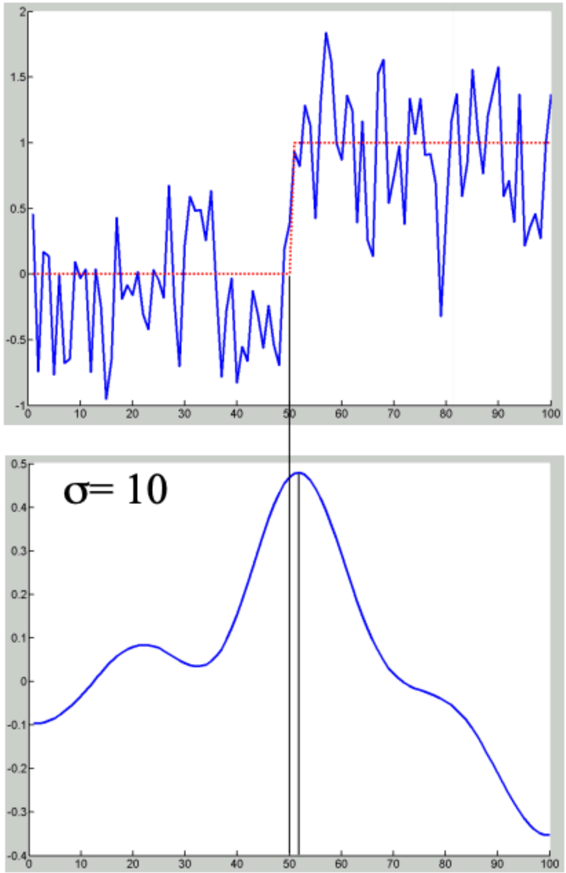
Adapted from K. Grauman

Which σ Should We Use?

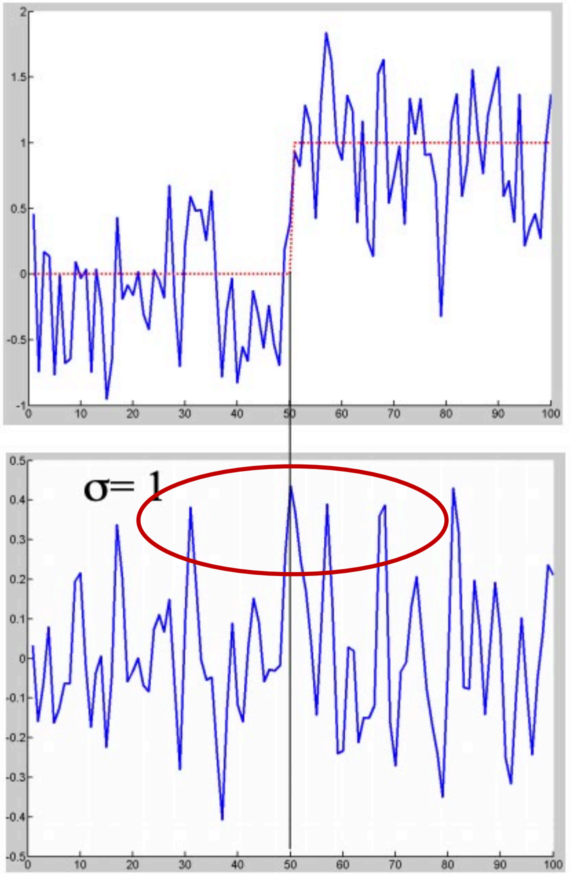
true signal

noisy observation

$$\frac{\partial}{\partial x}(h * f)$$ derivative of smoothed signal



Large σ → Good detection (high SNR)
Poor localization



Small σ → Poor detection (low SNR)
Good localization

Gets difficult to find the correct maximum corresponding to the edge

Laplace filter

A second derivative filter that can also be approximated with finite differences.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h} \longrightarrow$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} \longrightarrow$$

1D Laplace filter

1	-2	1

Discrete Laplacian of x,y

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \longrightarrow$$

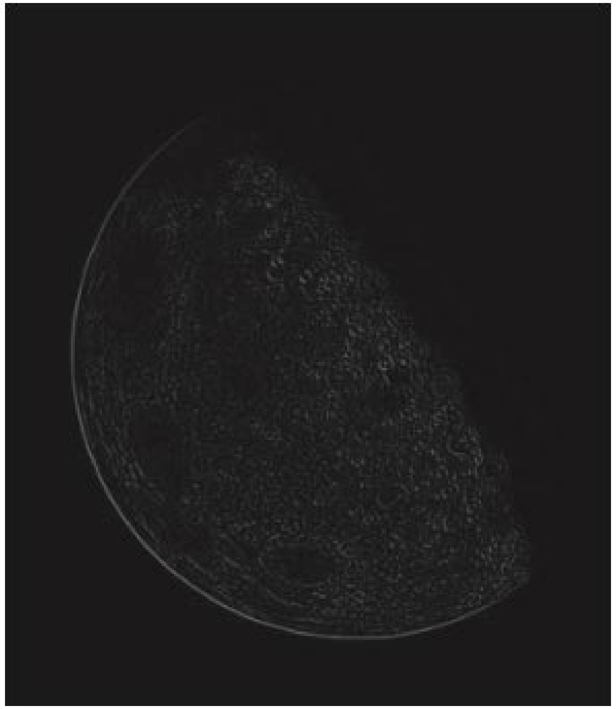
2D Laplace filter

0	1	0
1	-4	1
0	1	0

Adapted from I. Gkioulekas

Sharpening with Laplacians

A: original



B: filtered with

0	1	0
1	-4	1
0	1	0

Derivative filter: highlights transitions. Mostly black because negative values are clipped to 0 by display.

A−B: sharpening effect

Add original image to sharpened details. Use a subtraction sign because centre coefficient of kernel (-4) is negative!

A−filtering with

1	1	1
1	-8	1
1	1	1

Extension of Laplacian kernel with diagonal terms.

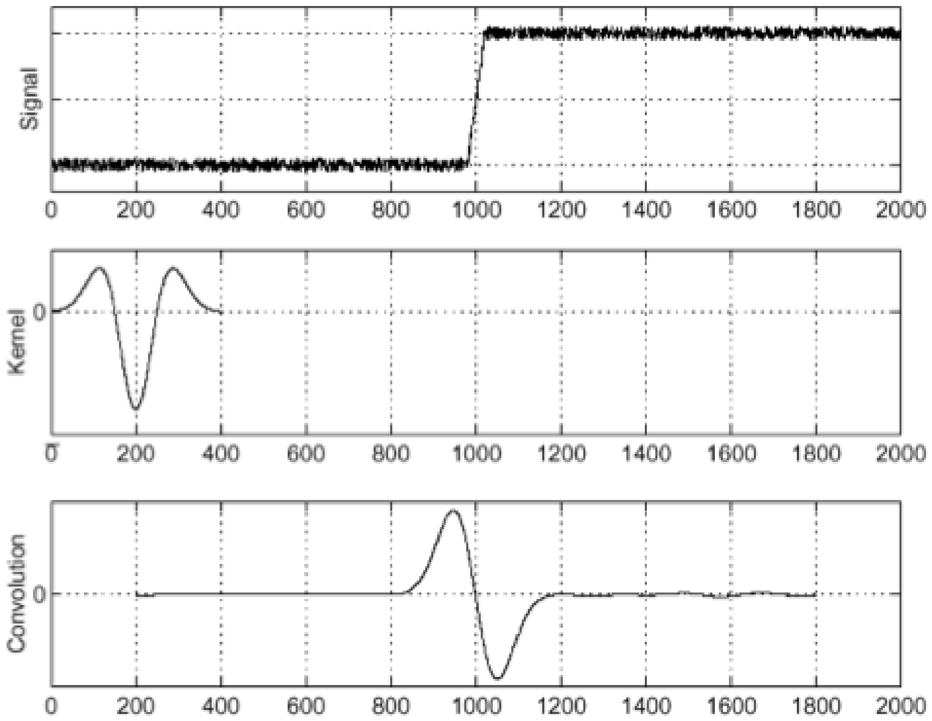
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

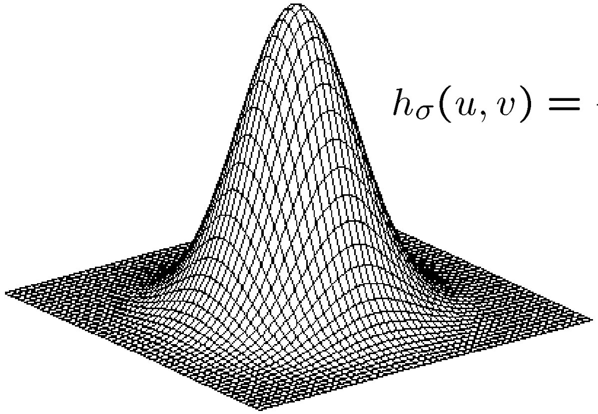
output



"zero crossings" at edges

Adapted from S. Seitz

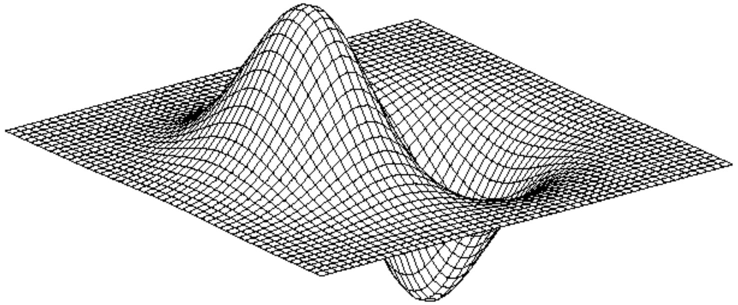
2D Gaussian filters



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

Gaussian

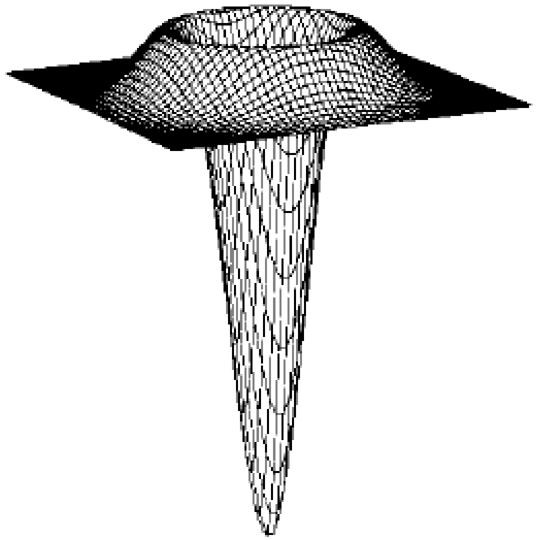
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



Derivative of Gaussian

Laplacian Operator

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



Laplacian of Gaussian

Adapted from S. Seitz

Laplace and LoG filtering examples

Also sensitive to noise

Smoothing w/ Gaussian helps

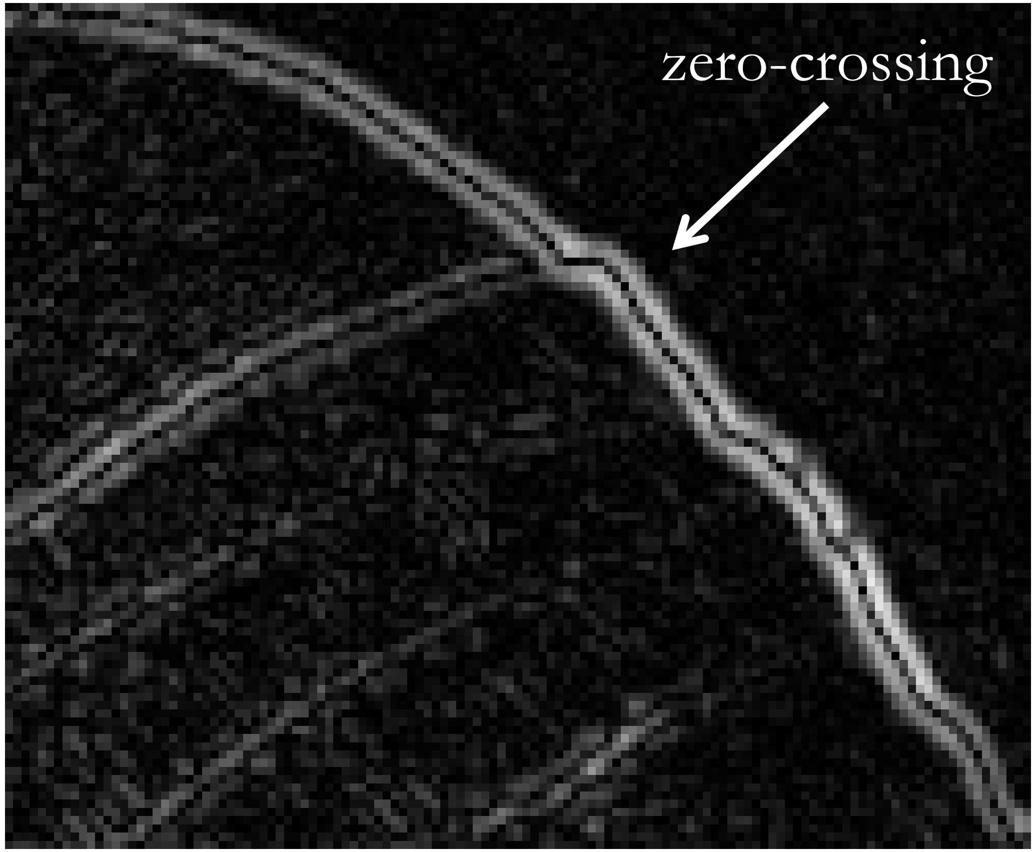
Laplace filtering

Laplacian of Gaussian filtering

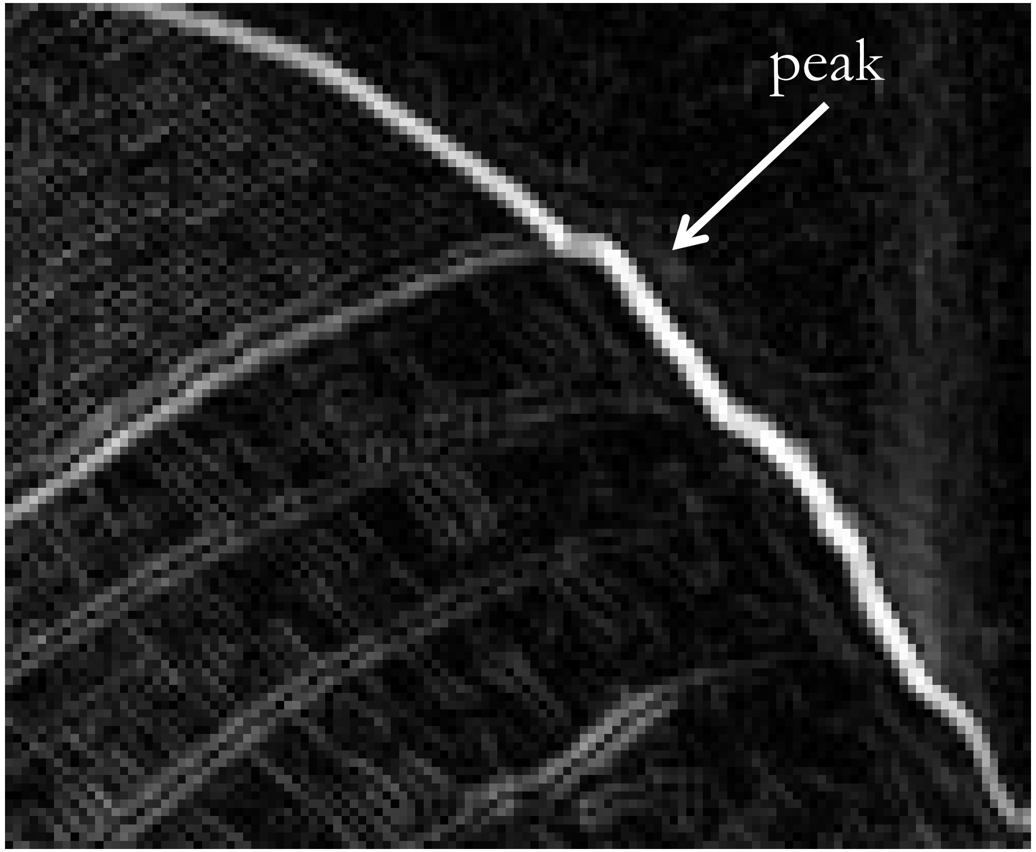
Derivative of Gaussian filtering

Adapted from I. Gkioulekas

Laplacian of Gaussian vs Derivative of Gaussian



Laplacian of Gaussian filtering



Derivative of Gaussian filtering

Zero crossings localize edges more accurately but are less convenient to use.

Slide Credit: Ioannis Gkioulekas

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986 679

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE

Abstract—This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a *comprehensive* set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges,

detector as input to a program which could isolate simple geometric solids. More recently the model-based vision system ACRONYM [3] used an edge detector as the front end to a sophisticated recognition program. Shape from motion [29], [13] can be used to infer the structure of

Canny Edge Detector

Adds non-maximum suppression and hysteresis thresholding to find edges from gradient outputs.

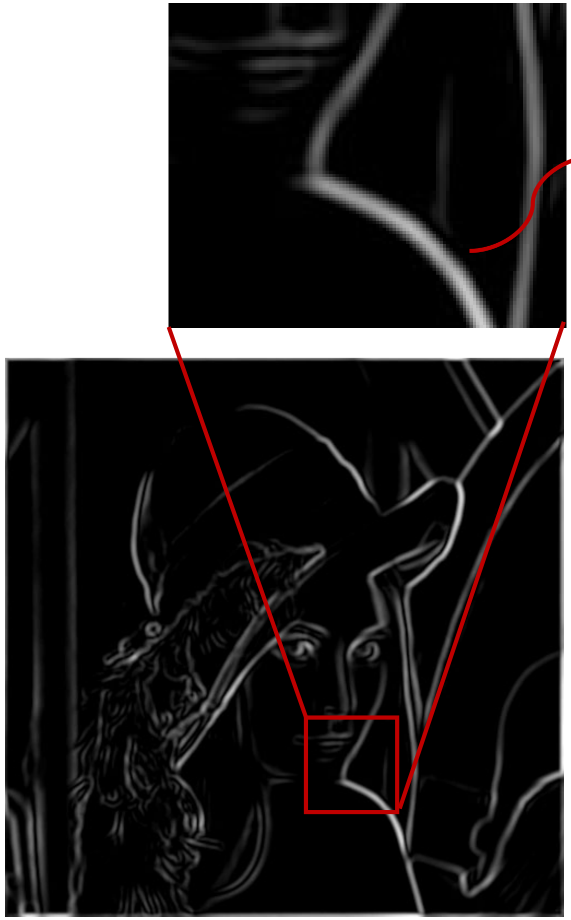
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf

Gradients to Edges

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

original* image

gradient magnitude



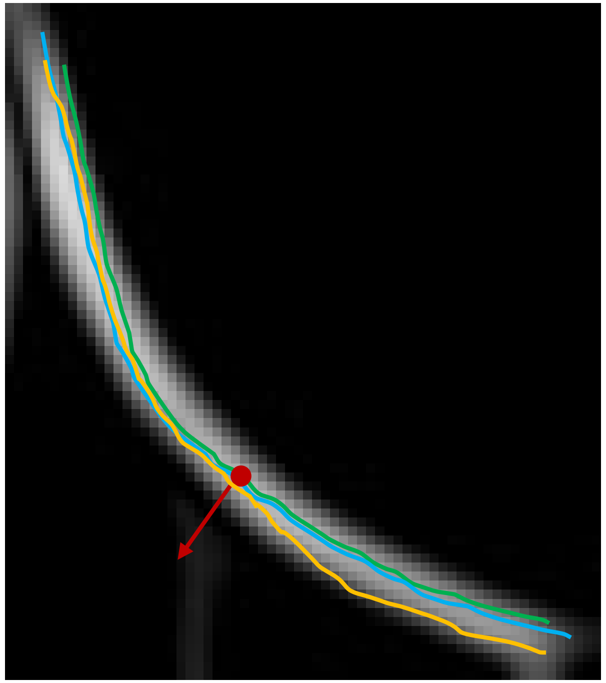
gradient magnitude after thresholding

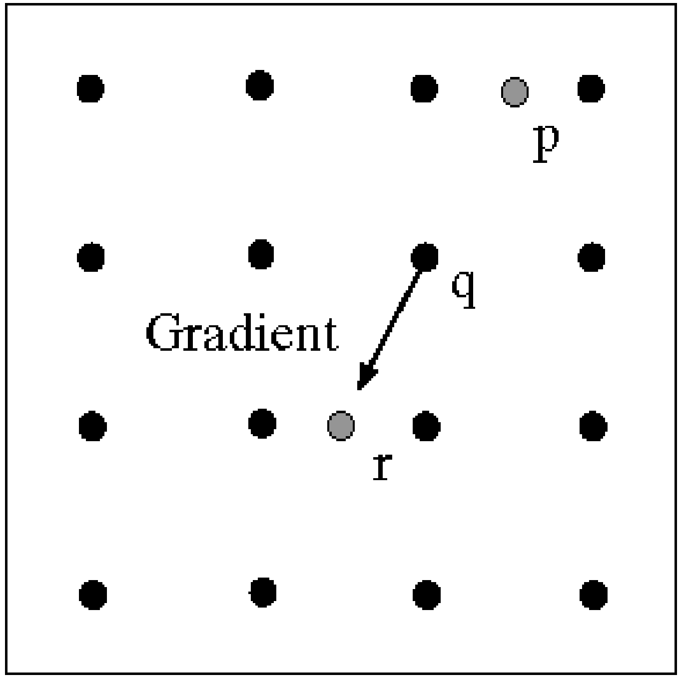
edge image

* Smoothing to suppress noise. Increase contrast to enhance edges.

Adapted from S. Seitz

Edge Thinning via Non-Maximum Suppression





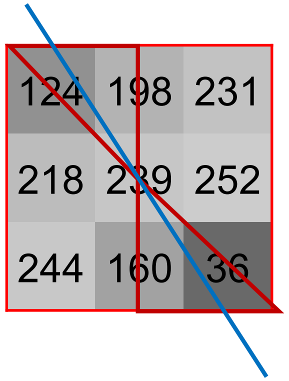
For each pixel, check if it is a local max along the gradient direction. If so, keep as edge, if not, discard (set to 0).
Note that this requires check interpolated values at pixel locations p and r!
Or simply approximate by finding the closest pixel locations to p and r.

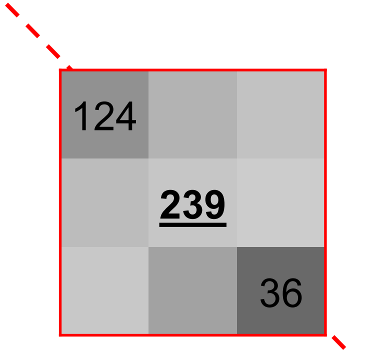
Adapted from S. Seitz

Non-Maximum Suppression Example

Magnitude

2	5	5	6	6	7	5	7
6	5	4	18	25	26	17	10
1	3	31	124	198	231	219	198
2	9	78	218	239	252	264	270
7	14	104	244	160	36	53	72
6	14	119	244	142	13	10	13
6	16	135	242	124	6	6	10
7	18	149	242	110	2	9	4

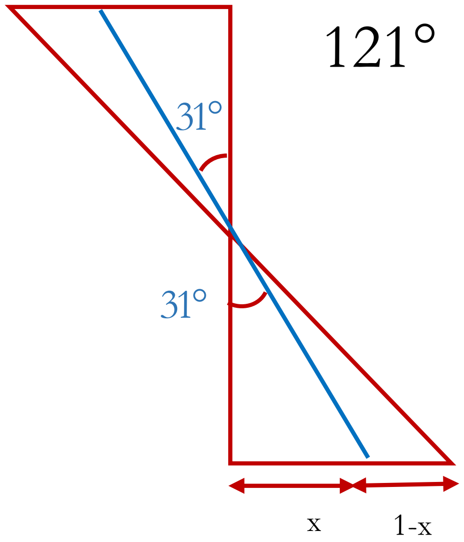




No interpolation approximation

Orientation

90	79	79	121	90	45	79	124
108	63	90	131	97	67	80	90
-135	-18	138	128	103	86	86	86
-180	117	151	148	121	88	85	85
135	146	169	172	168	96	79	81
162	172	176	177	179	-135	-90	-72
-180	173	175	175	175	-180	-59	-53
146	167	176	175	173	90	-27	-45

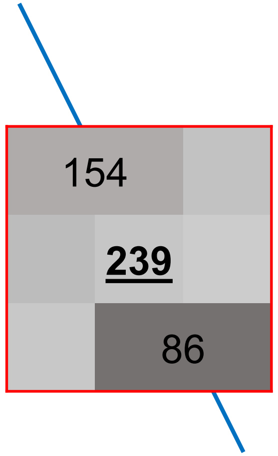


121°

$$x = \tan\left(\frac{31}{180}\cdot\pi\right);\quad x \approx 0.6$$

$$0.6\cdot(124 - 198) + 198 \approx 154$$

$$0.6\cdot(36 - 160) + 160 \approx 86$$



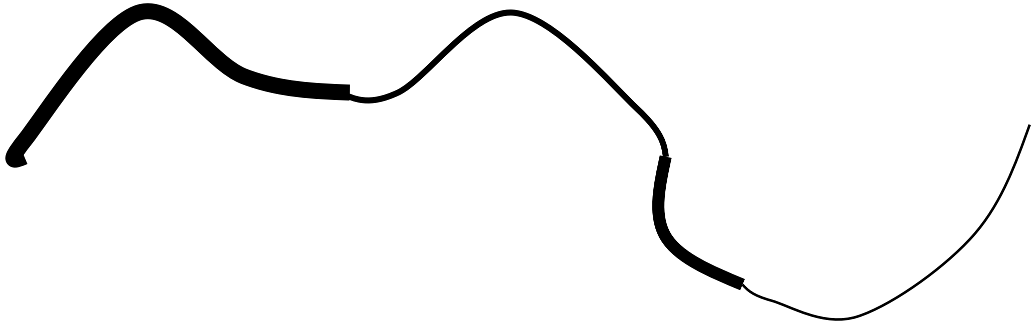
Keep as part of edge

Discontinuous Edges

gradient magnitude
after thresholding

edge image

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



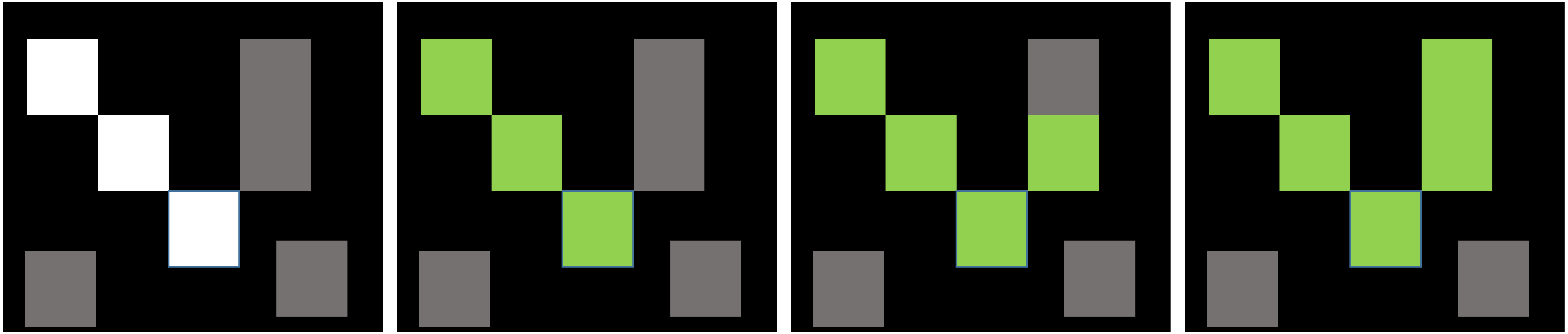
Hysteresis thresholding: use a high threshold to start edge curves, and a low threshold to continue growing them.

Hysteresis Thresholding

<u>White</u>: Clears high threshold
<u>Grey</u>: clears low threshold but not high threshold

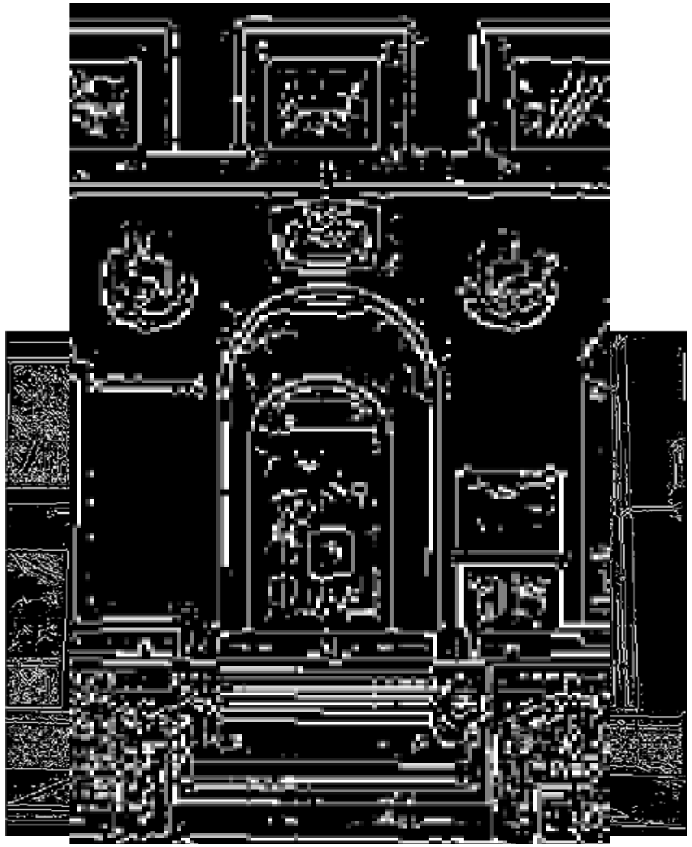
Mark pixels that clear high threshold as boundary

If there are pixels that clear low threshold and are connected to a boundary, mark them as boundary too

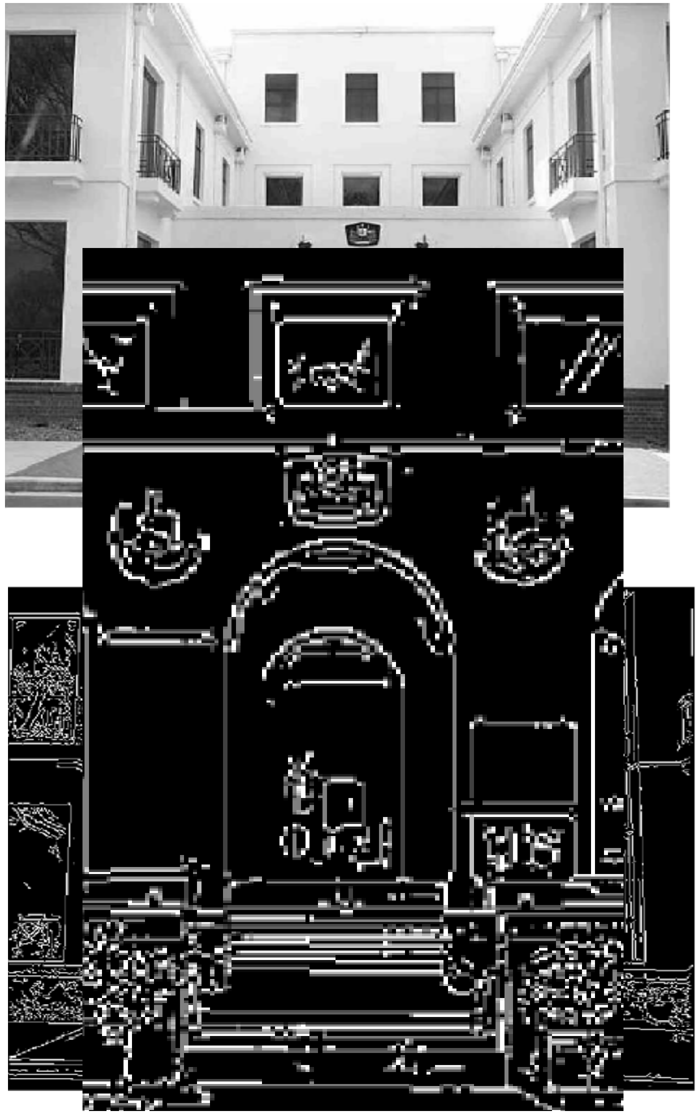


Hysteresis Thresholding

high threshold
(strong edges)



low threshold
(weak edges)



hysteresis threshold

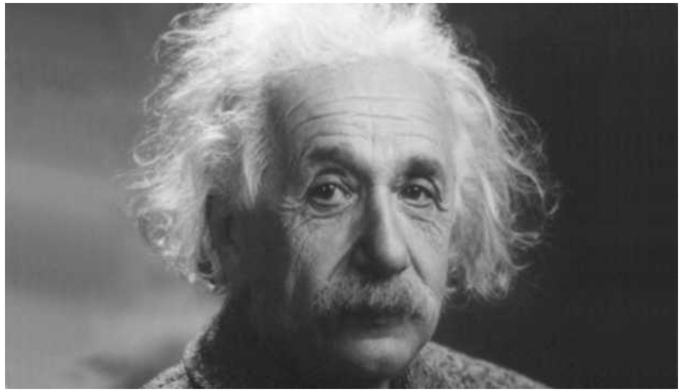
Source: L. Fei-Fei

Canny Edge Detector

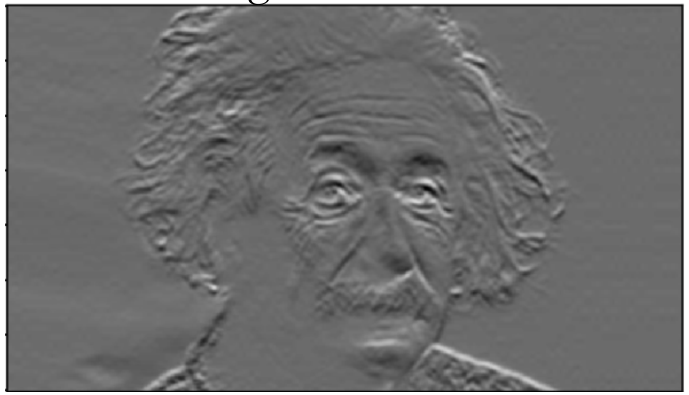
1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression (on low and high threshold):
   - Thin wide “ridges” down to single pixel width
4. Linking and thresholding (hysteresis):
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei

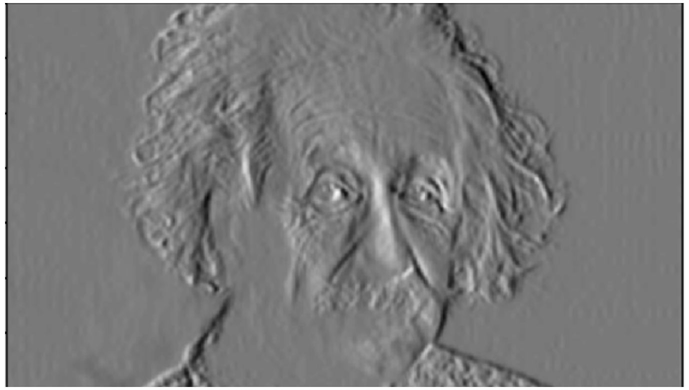
Canny Step-by-Step



1: Filter image with derivative of Gaussian



gradients in y-dir



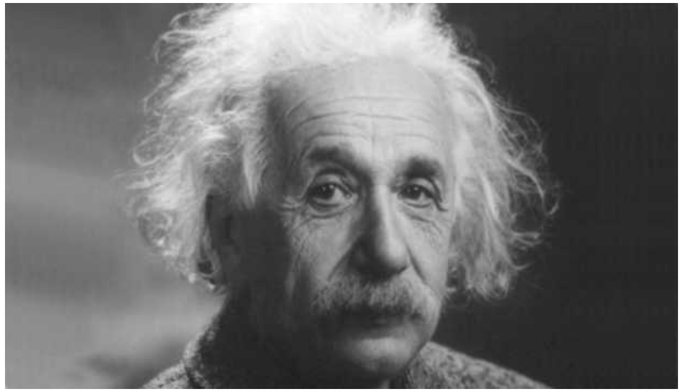
gradients in x-dir

2. Compute gradient magnitude and orientation

gradient magnitude

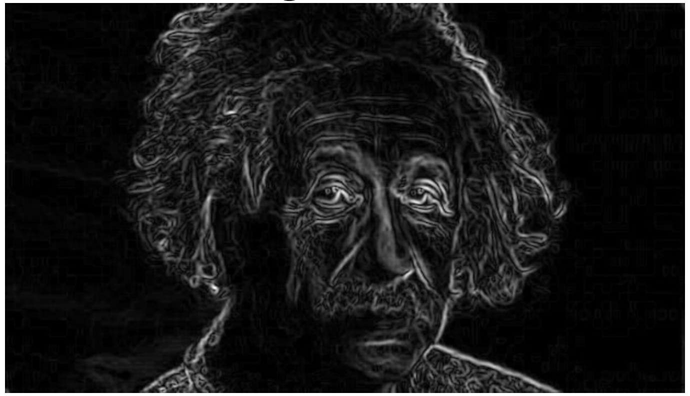
(visualization of) gradient orientation

Canny Step-by-Step

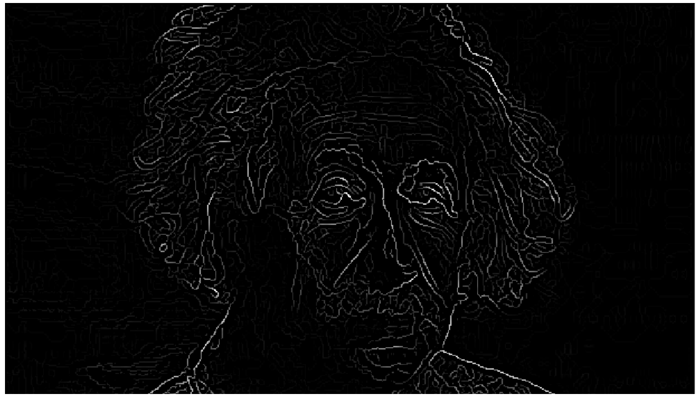


Final Canny Edge Image

3. Thresholding



gradient magnitude



(low) thresholded magnitude

4. Non-Maximum Suppression & Linking

high threshold
(keeps only most confident pixels)

low threshold
(produces spurious edges)

Summary

- Image edges are a compact way to convey information;
- Edges found by applying derivative filter (e.g. Sobel) to extract gradients
- Gradients have two components: magnitude & orientation
- Smoothing prior to estimating gradients reduces effects of noise, but trades off localization accuracy
- Canny edge detection converts gradients to edges w/ non-max suppression & hysteresis thresholding