

Computer Vision & Image Processing CSE 473 / 573

Instructor - Kevin R. Keane, PhD

TAs - Radhakrishna Dasari, Yuhao Du, Niyazi Sorkunlu

Lecture 9

September 18, 2017

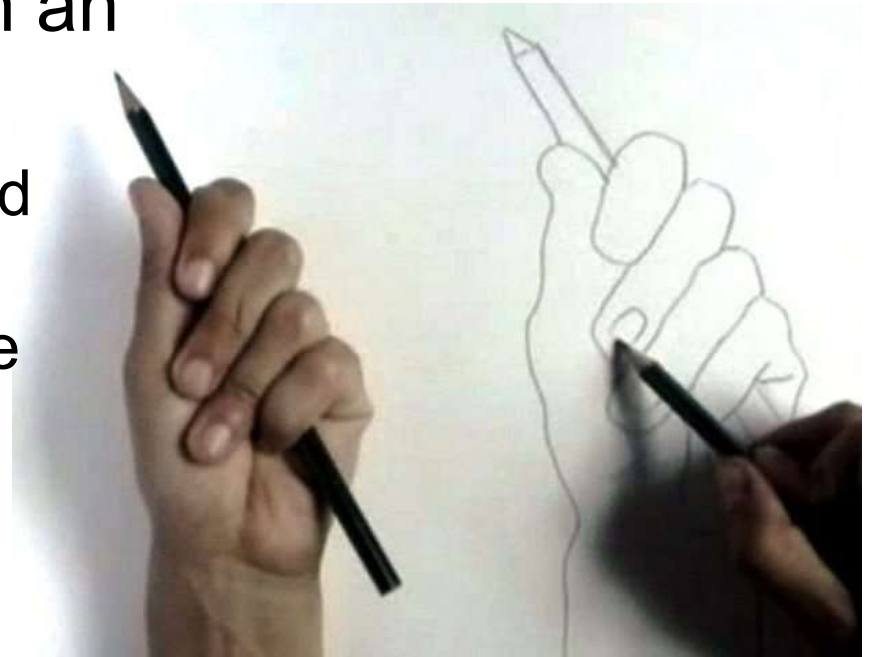
Edges and Pyramids

Schedule

- Last week
 - Linear filters and mathematical morphology
- Today
 - Edges and pyramids
- Recitation
 - HW1 questions
- Readings for today: Forsyth and Ponce 4.7

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
 - Intuitively, most semantic and shape information from the image can be encoded in the edges
 - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

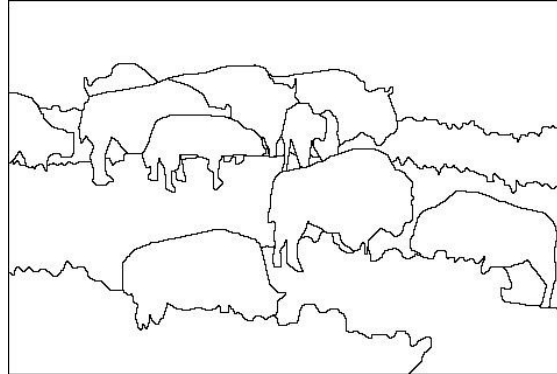


Human vs machine edges

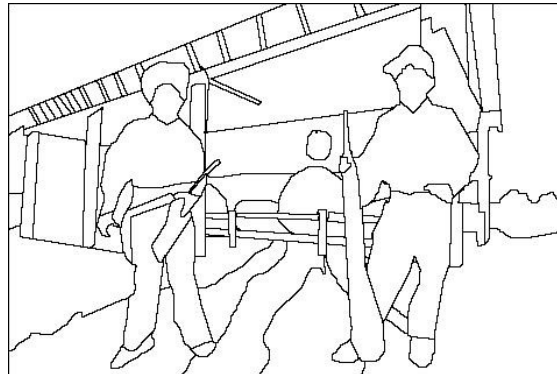
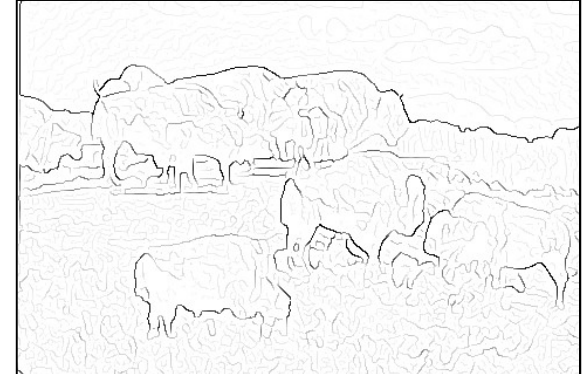
image



human segmentation



gradient magnitude



- Berkeley segmentation database:

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

Intuitions guiding edge detection

- **Pixels tend strongly to be like their neighbors**
 - This is the single most important experimental fact about images
 - Consequences
 - We can estimate a pixel value using its neighbors
 - ***Pixels that are different from their neighbors are important***
- Smoothing with a Gaussian
 - Works because pixels look like their neighbors
 - Suppresses noise because positive, negative errors tend to cancel

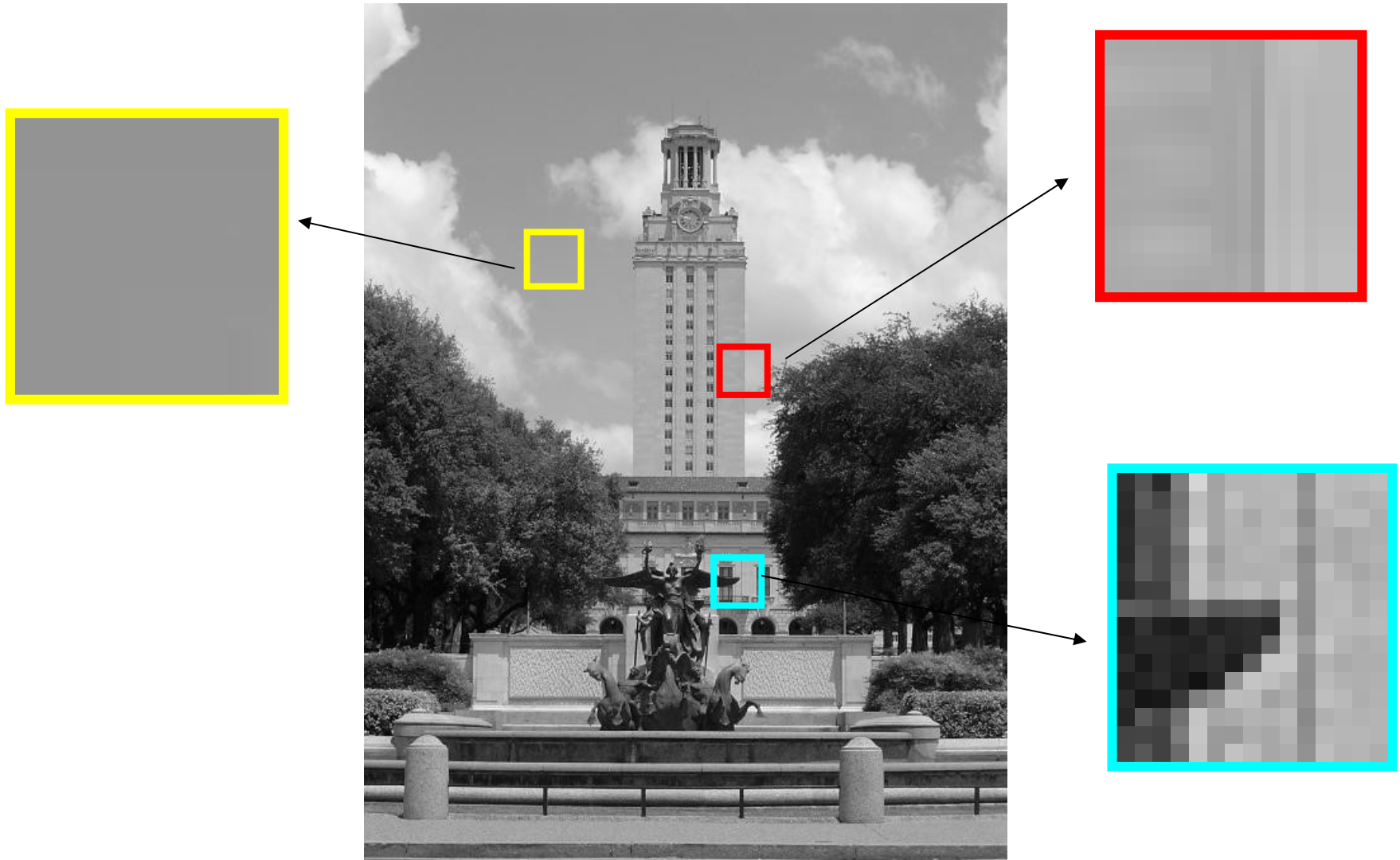
Intuitions guiding edge detection

- Pixel could differ from neighbor because
 - they have different albedos
 - they are on different objects
 - they have different surface normals
 - there is a big difference in shading (e.g. an outdoor shadow)
- Pixels that differ from their neighbors are interesting
 - they occur when the gradient is large
 - but image noise has large gradients, too

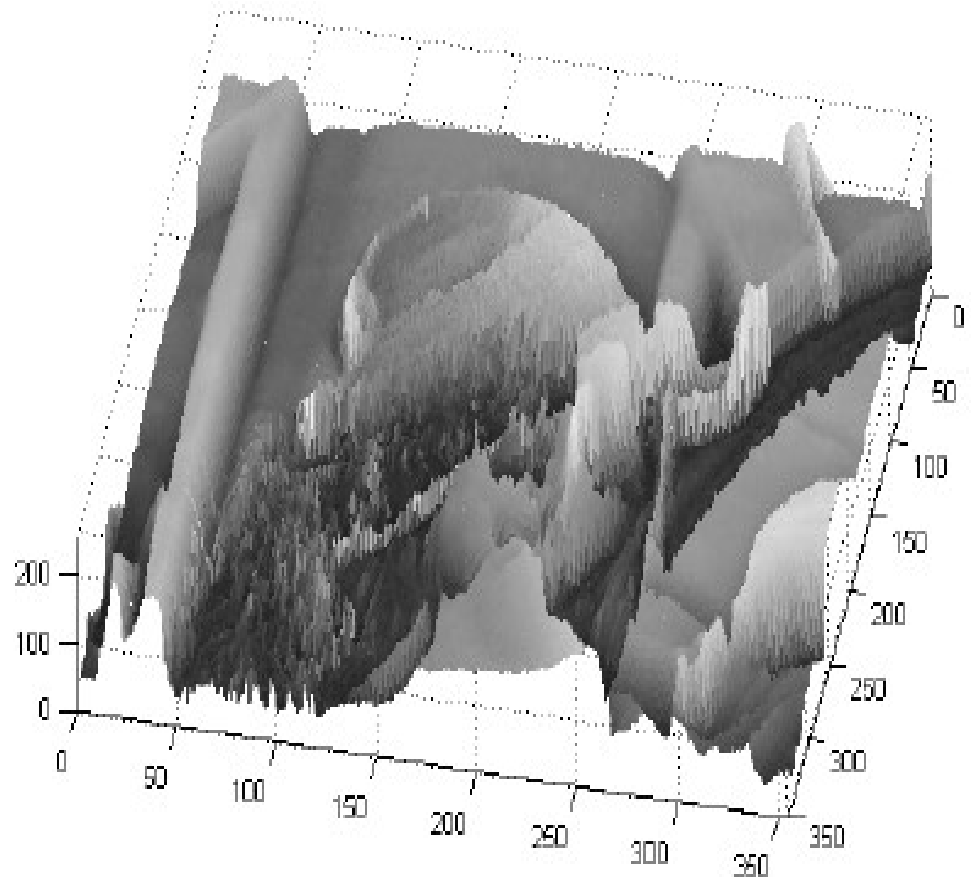
Key idea

- suppress image noise by **smoothing**, then take **gradients**

Contrast and invariance



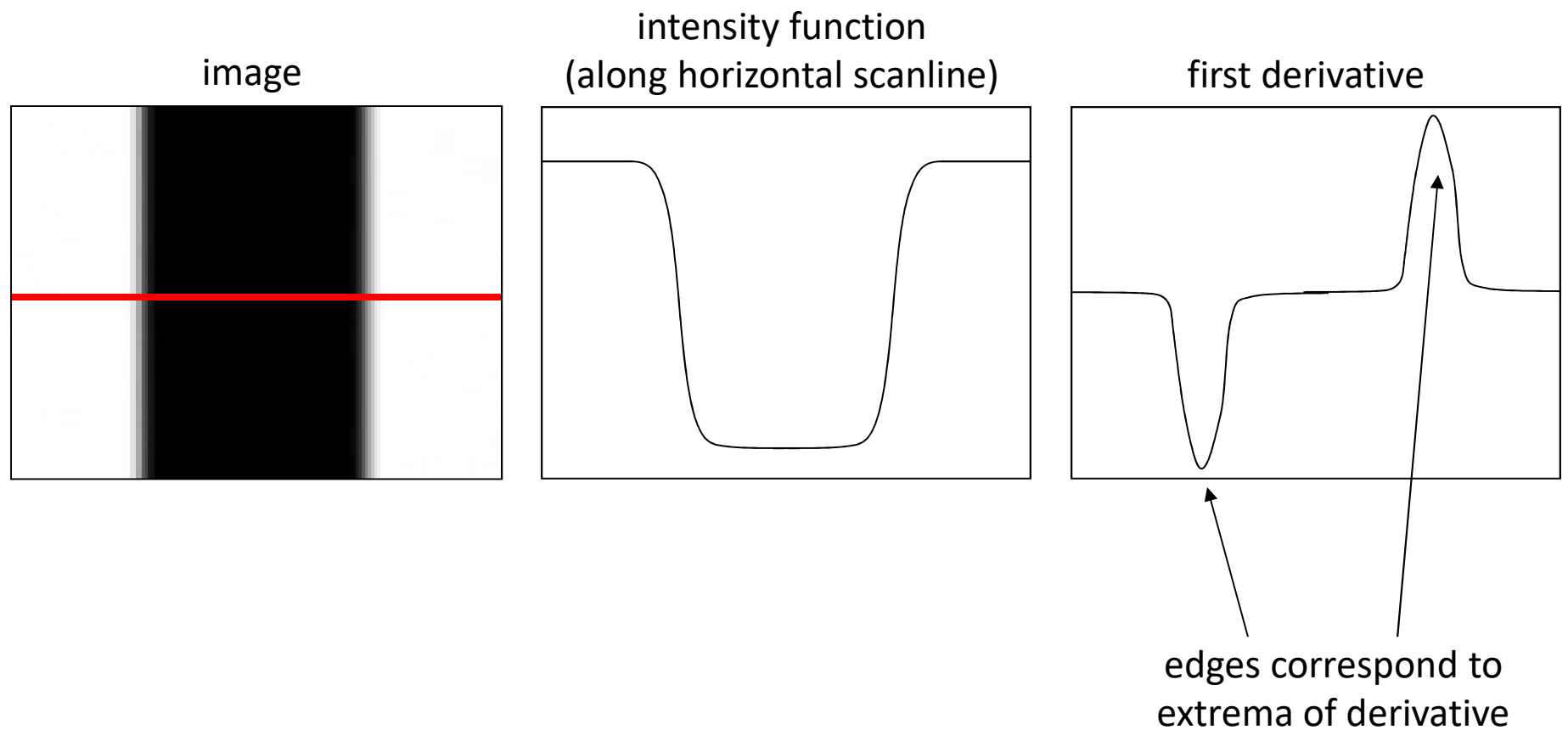
Recall : Images as functions



- Edges look like steep cliffs

Derivatives and edges

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



Derivatives and edges cont'd

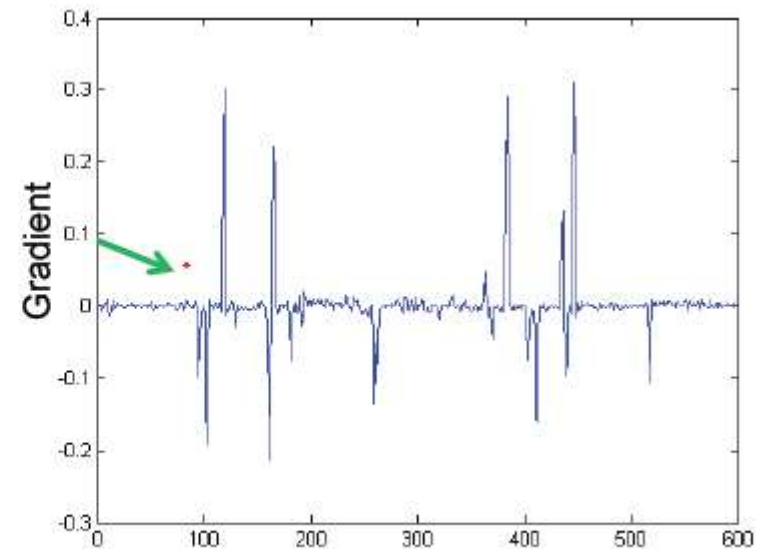
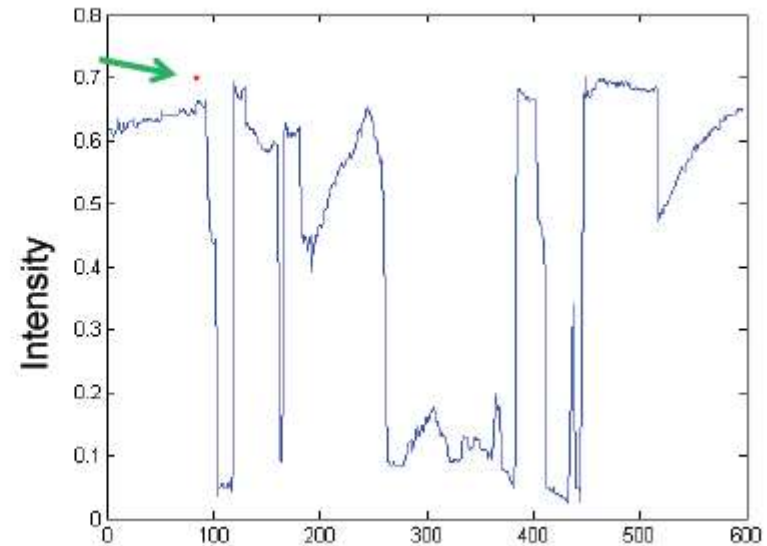
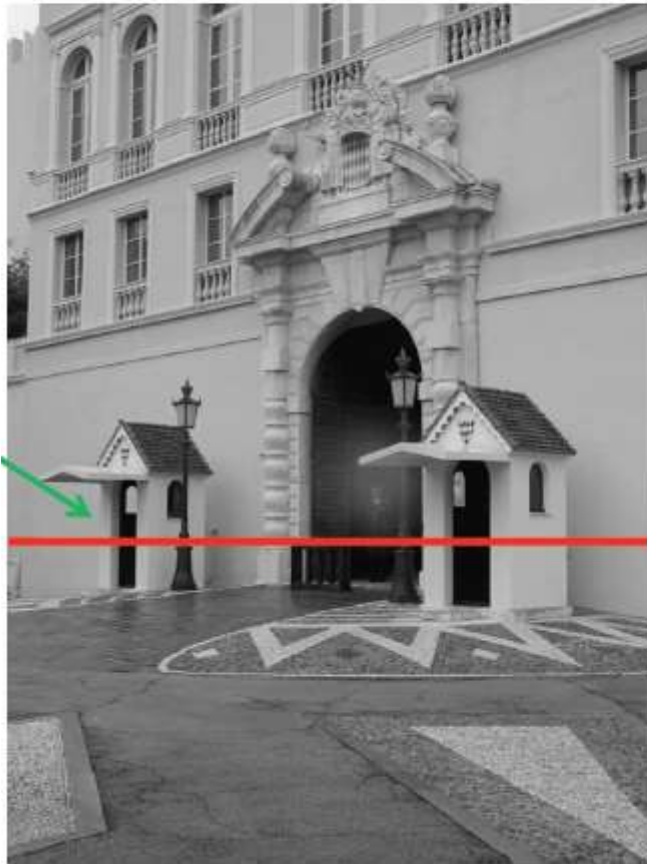
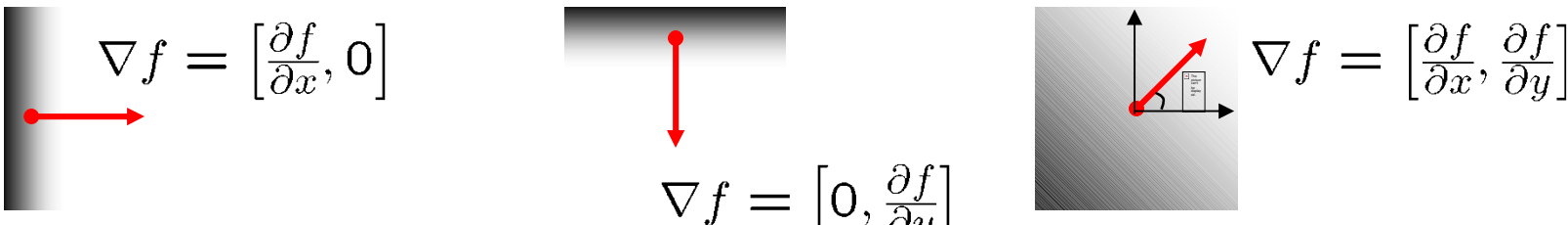


Image gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$

- 

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

- How does this direction relate to the direction of the edge?

The gradient direction 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:

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

For discrete data, we can approximate using finite differences:

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

This is linear and shift invariant (must be the result of a convolution)

To implement the above as convolution, what would be the associated filter?

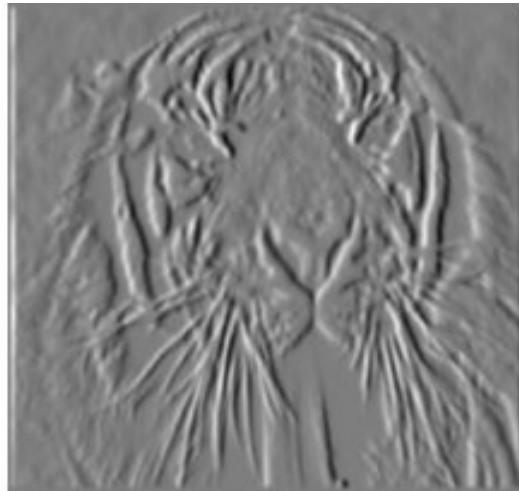
-1	1
----	---

Partial derivatives of an image



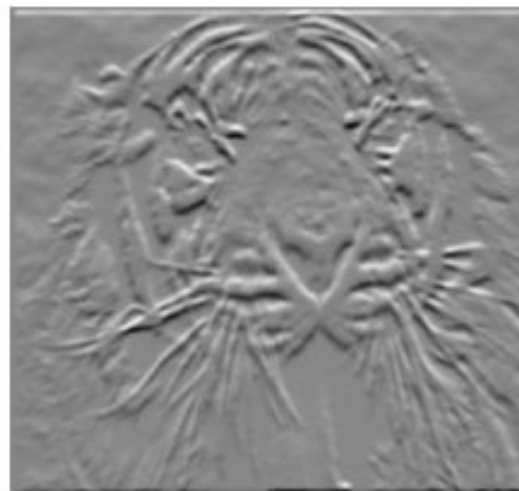
$$\frac{\partial f(x, y)}{\partial x}$$

-1	1
----	---



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

-1	?	1
1	or	-1



Which shows changes with respect to x?

Finite difference filters

- Other approximations of derivative filters exist:

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

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

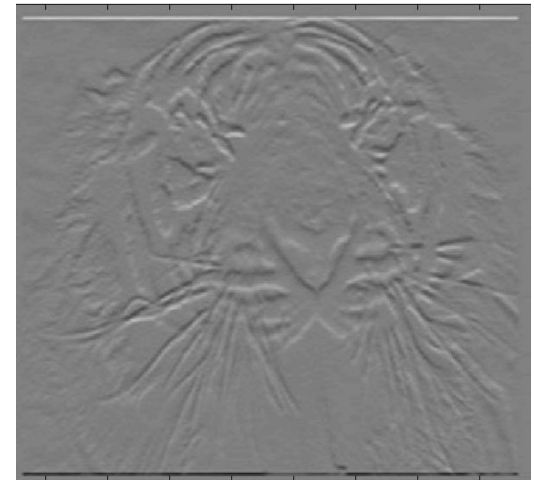
Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} ; M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Finite difference filters cont'd

Sobel_y =

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

```
>> My = fspecial('sobel');  
>> outim = imfilter(double(im), Sobel_y);  
>> imagesc(outim);  
>> colormap gray;
```

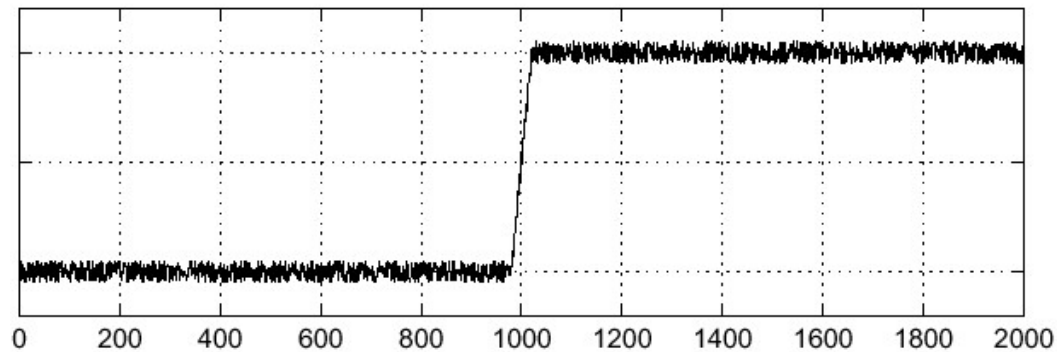


Noise

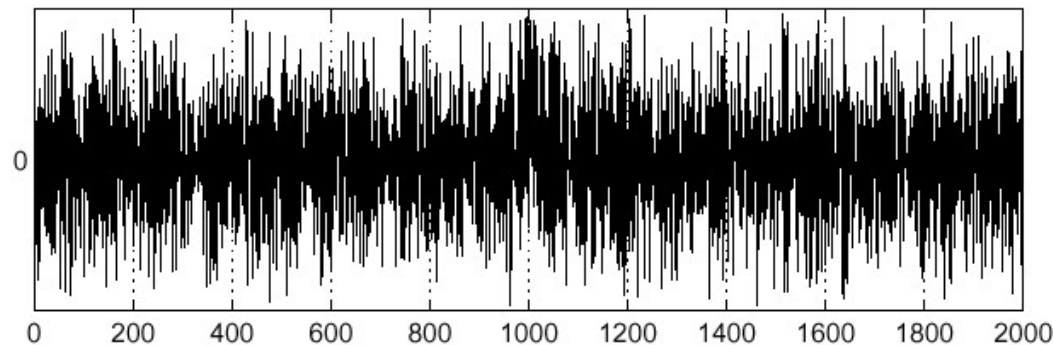
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$$f(x)$$

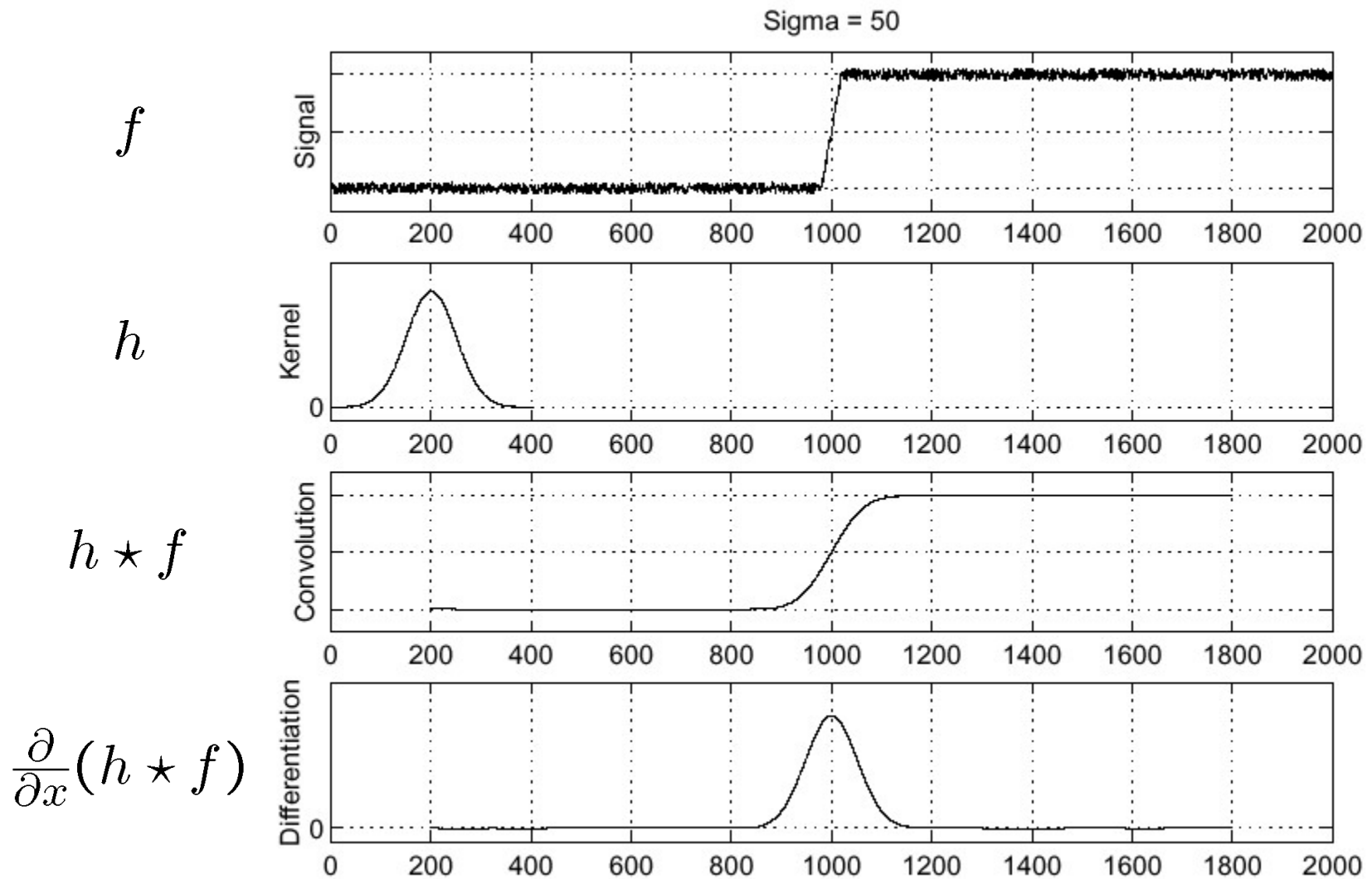


$$\frac{d}{dx}f(x)$$



Where is the edge?

Solution: smooth first



Where is the edge?

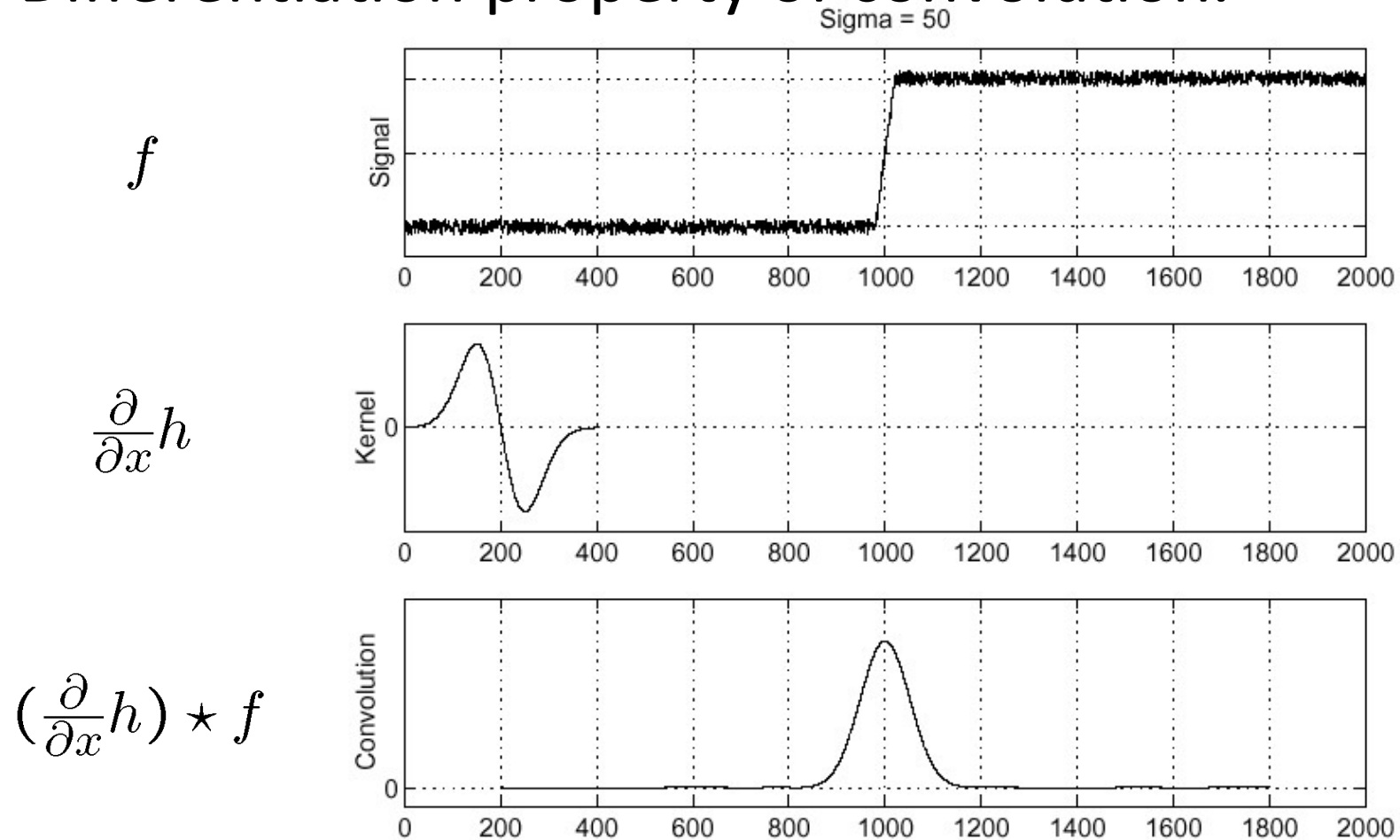
Look for peaks in

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

Derivative theorem of convolution

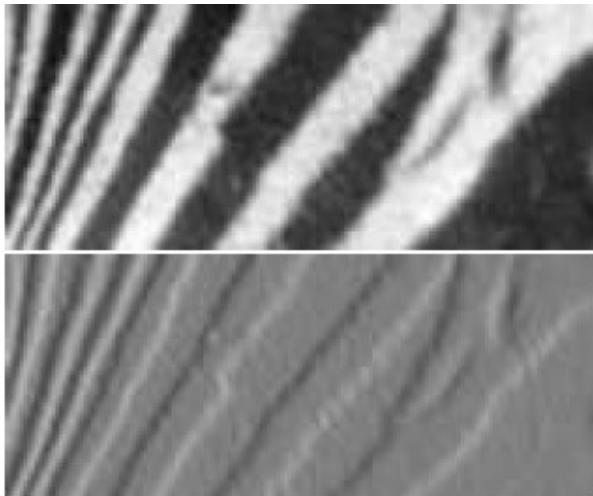
$$\frac{\partial}{\partial x}(h \star f) = \left(\frac{\partial}{\partial x}h\right) \star f$$

Differentiation property of convolution.

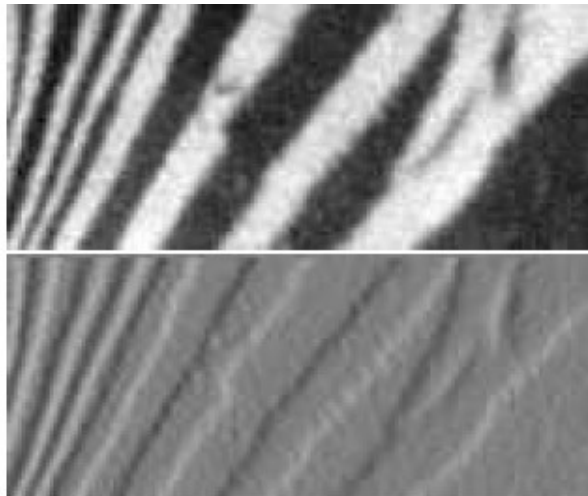


But derivatives amplify noise

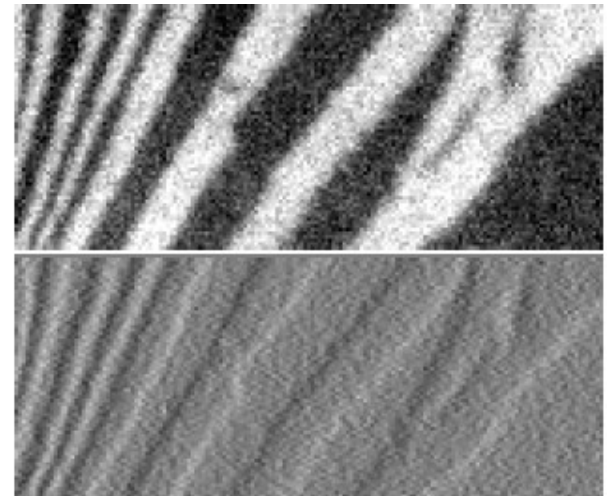
Input image



Noisier Input image



Noisiest Input image



- this is noise amplified by differentiation
- derivative filters respond strongly to pixels that differ from their neighbors

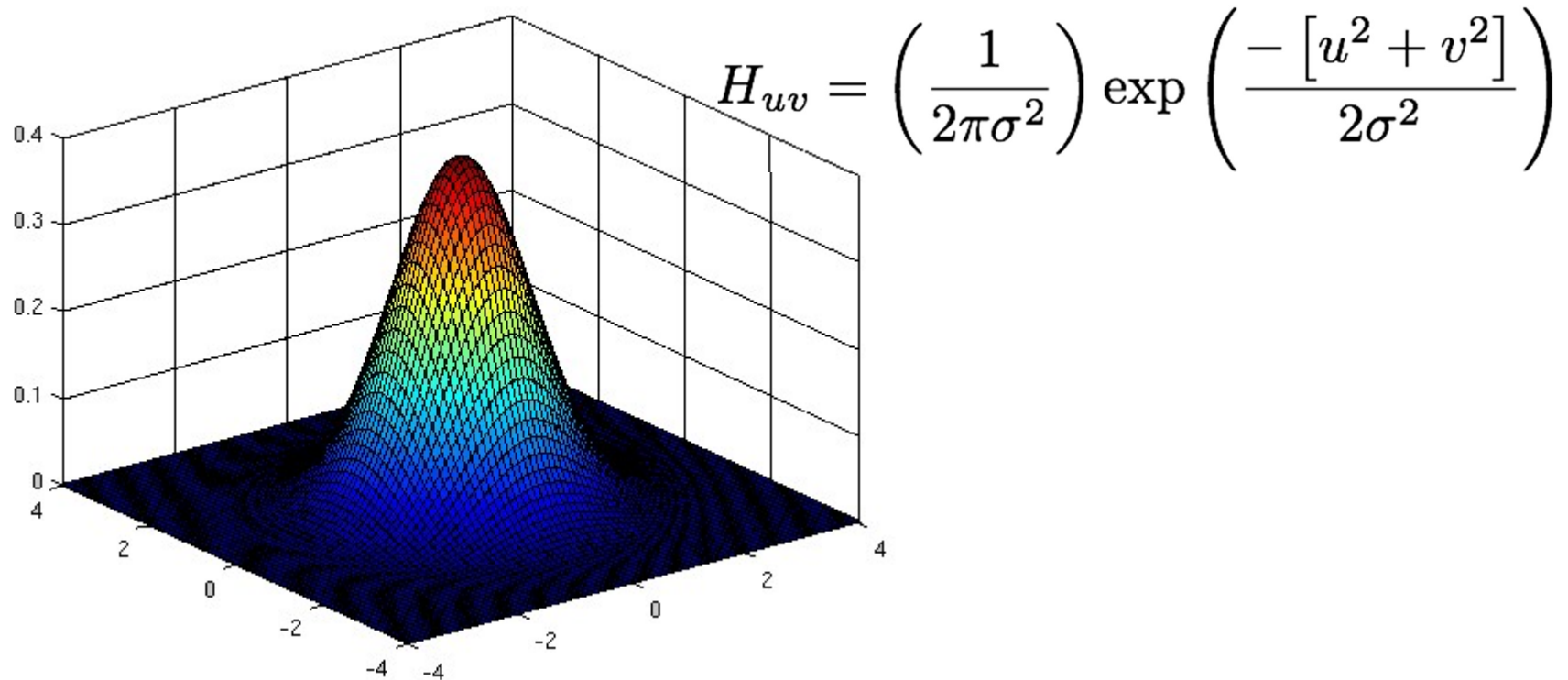
And smoothing reduces noise

- Pixels tend to “be like” their neighbors
 - surfaces turn slowly
 - relatively few reflectance changes
- Expect noise to be independent from pixel to pixel
 - Implies that smoothing suppresses noise, for appropriate noise models
- Scale
 - the parameter in the symmetric Gaussian
 - as this parameter goes up, more pixels are involved in the average
 - and the image gets more blurred
 - and noise is more effectively suppressed

$$H_{uv} = \left(\frac{1}{2\pi\sigma^2} \right) \exp \left(\frac{-[u^2 + v^2]}{2\sigma^2} \right)$$

Scale

The Gaussian smoothing kernel



- The symmetric Gaussian kernel in 2D; scaled so that its sum equal 1; Also, $\sigma = 1$
- Convolution with this kernel forms a weighted average where strongest response is at the center
 - Image point at the middle gets little contribution from points at the boundary

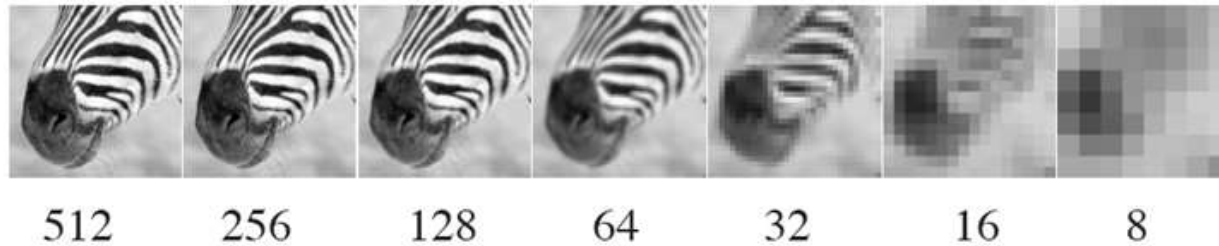
Gaussian pyramids and scale

- A smoothed image can be resampled
 - result:
 - lower resolution version
 - emphasizing large scale trends over detail
 - and again, ...

```
Set the finest scale layer to the image
For each layer, going from next to finest to coarsest
    Obtain this layer by smoothing the next finest
    layer with a Gaussian, and then subsampling it
end
```

Forming a Gaussian Pyramid.

A Gaussian pyramid

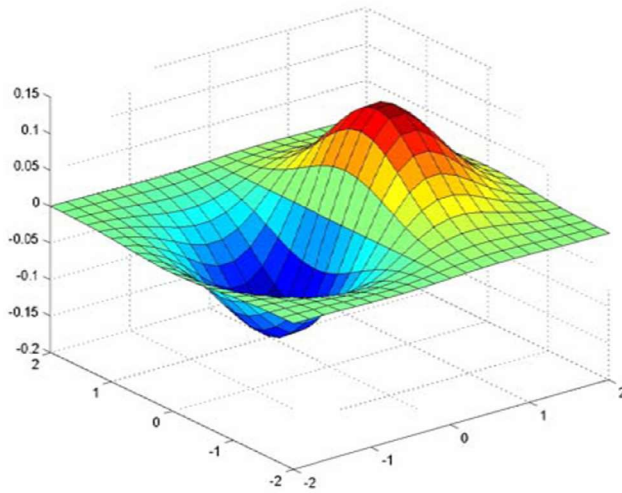


-A Gaussian pyramid running from 512 x 512 to 8 x 8

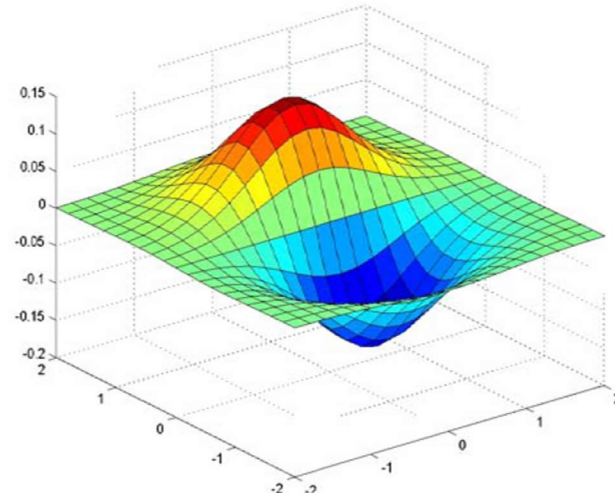
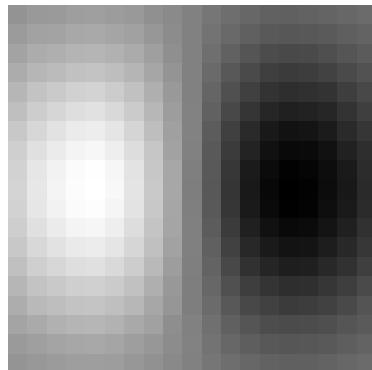
Back to smoothed gradients

- Fact: These two are the same
 - Smooth, then differentiate - $\frac{\partial (G_\sigma * I)}{\partial x} = \left(\frac{\partial G_\sigma}{\partial x}\right) * I$
 - Filter with derivative of Gaussian
- Exploit:
 - Filter image with derivative of Gaussian filters to get smoothed gradient

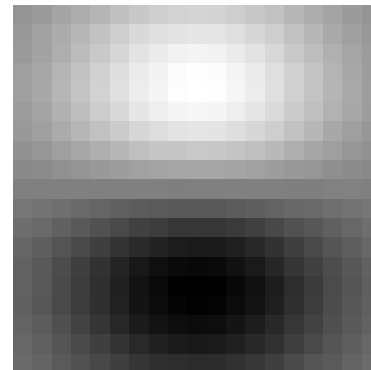
Derivative of Gaussian filters



x-direction



y-direction



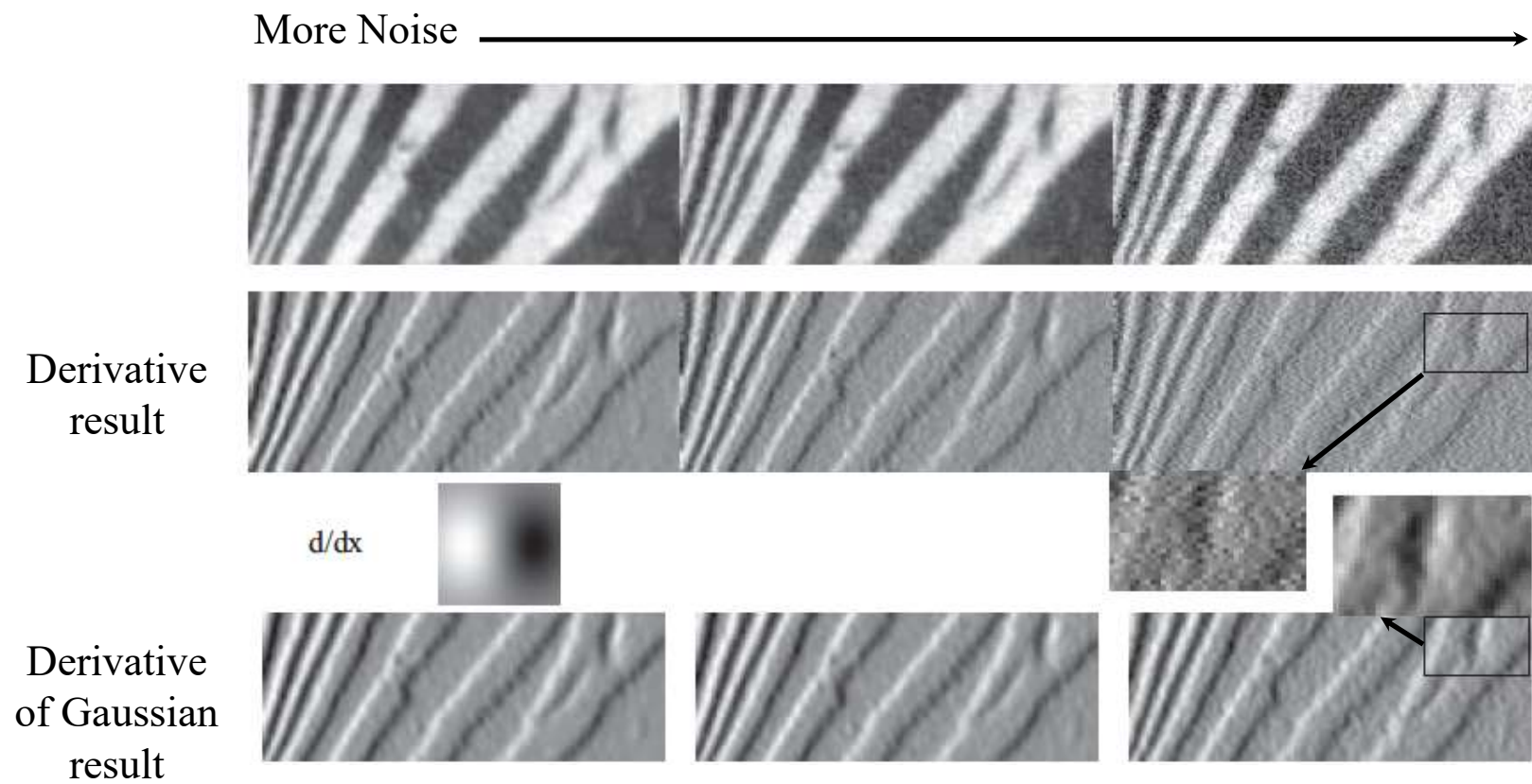
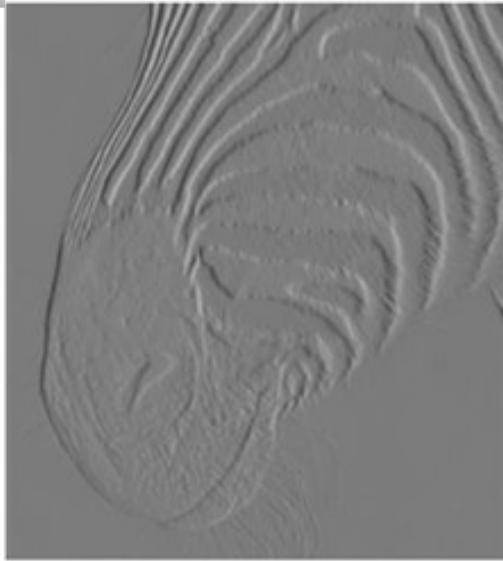


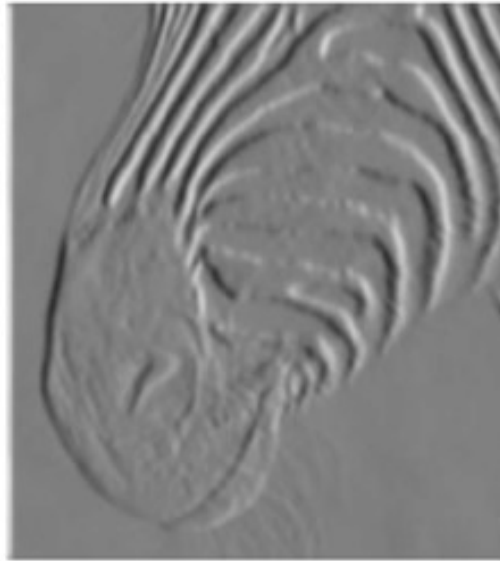
Figure 5.2



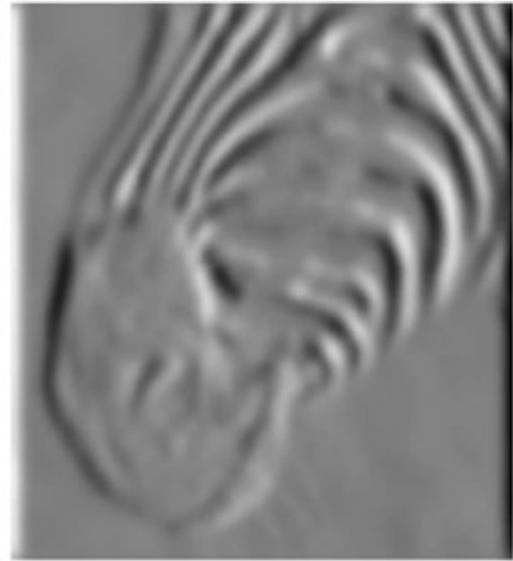
Scale of the Gaussian derivative filter



1 pixel



3 pixels



7 pixels

- The scale of the Gaussian used in the DoG filter has significant effects on the results
- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”
 - Small scale shows more details like hair while large scale loses some of the stripes at the muzzle

Implementation issues

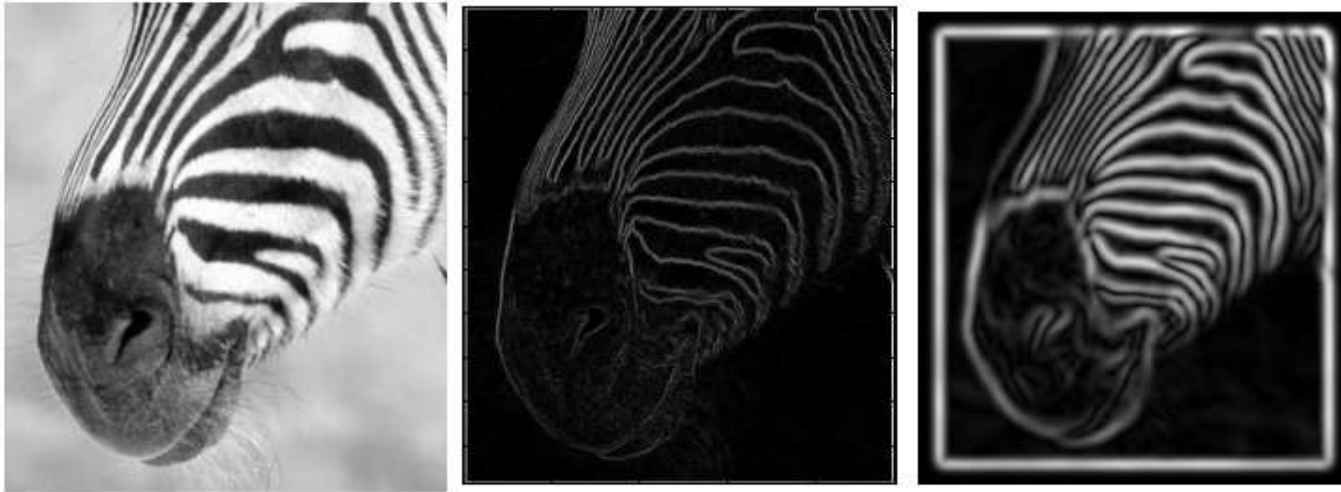


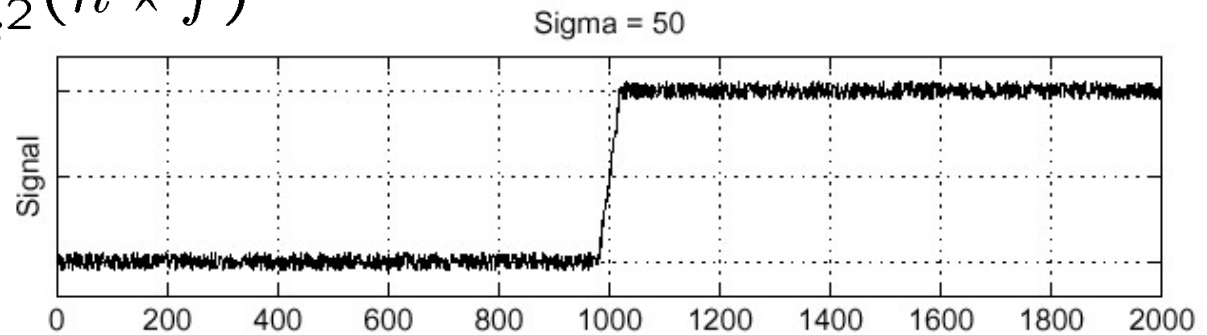
FIGURE 5.4: The gradient magnitude can be estimated by smoothing an image and then differentiating it. This is equivalent to convolving with the derivative of a smoothing kernel. The extent of the smoothing affects the gradient magnitude; in this figure, we show the gradient magnitude for the figure of a zebra at different scales. At the center, gradient magnitude estimated using the derivatives of a Gaussian with $\sigma = 1$ pixel; and on the right, gradient magnitude estimated using the derivatives of a Gaussian with $\sigma = 2$ pixel. Notice that large values of the gradient magnitude form thick trails.

- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

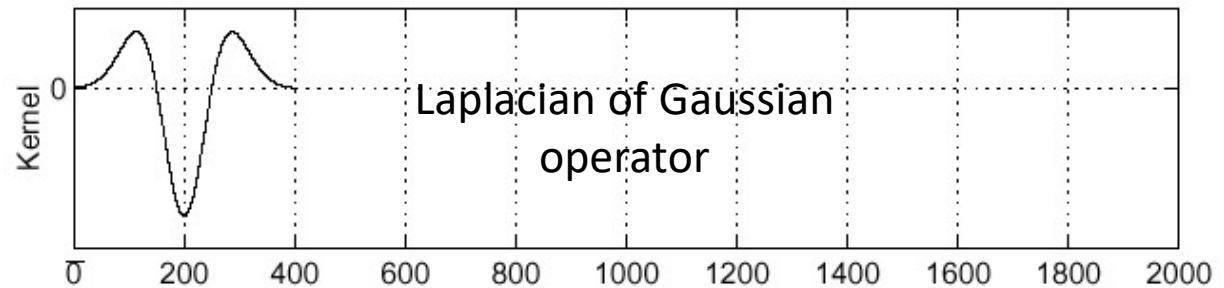
Laplacian of Gaussian

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

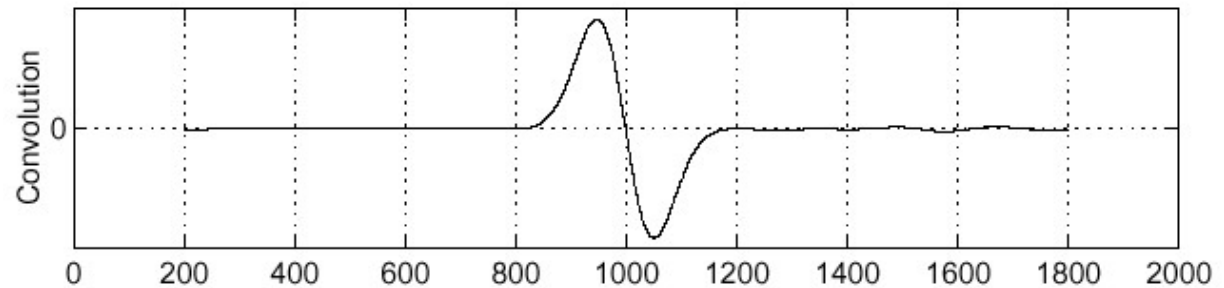
f



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



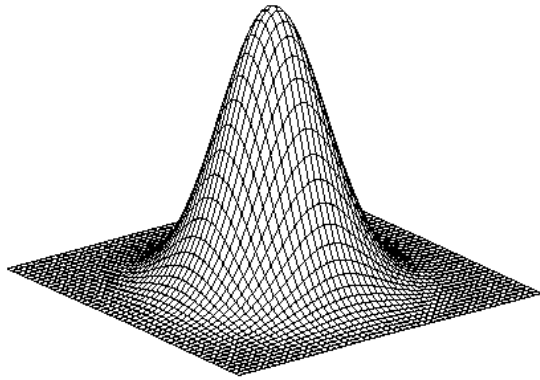
$(\frac{\partial^2}{\partial x^2}h) \star f$



Where is the edge?

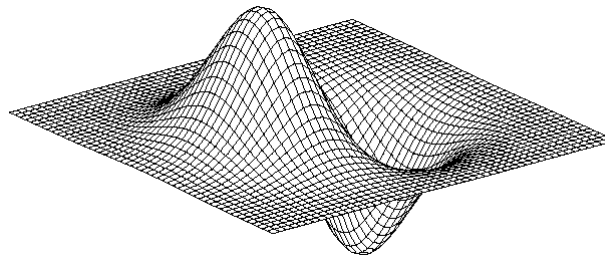
Zero-crossings of bottom graph

2D edge detection filters



Gaussian

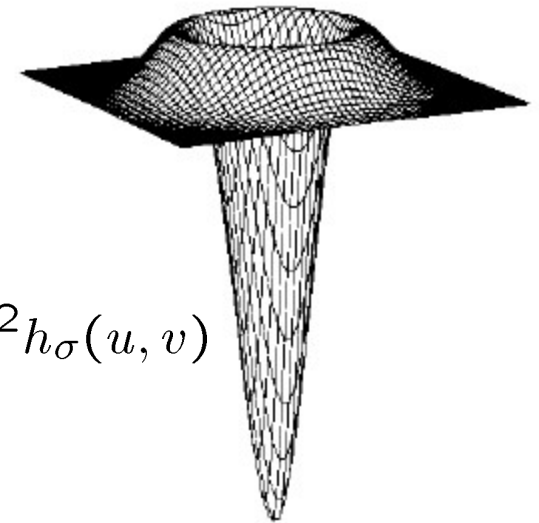
$$h_{\sigma}(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



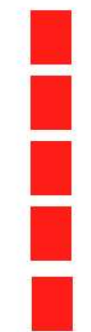
$$\nabla^2 h_{\sigma}(u, v)$$

- ∇^2 is the Laplacian operator:

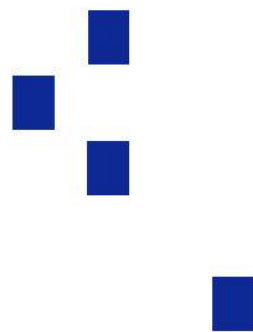
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Designing an edge detector

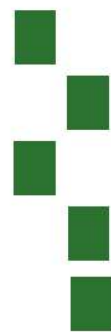
- Criteria for an “optimal” edge detector:
 - **Good detection:** the optimal detector must have a small number of false positives (detecting spurious edges caused by noise), and a small number of false negatives (missing real edges)
 - **Good localization:** the edges detected must be as close as possible to the true edges
 - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



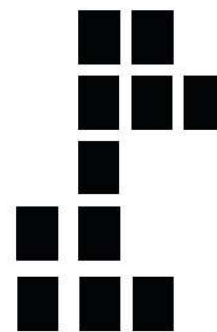
True
edge



Poor robustness
to noise



Poor
localization



Too many
responses

Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization
- MATLAB: `edge(image, 'canny')`

J. Canny, [**A Computational Approach To Edge Detection**](#), IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Canny edge detector

- Filter image with x- and y- derivatives of Gaussian
- Find magnitude and orientation of gradient
- Non-maximum suppression
 - Thin multi-pixel wide “ridges” down to single pixel width
- Thresholding and linking (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them

Canny edge detector

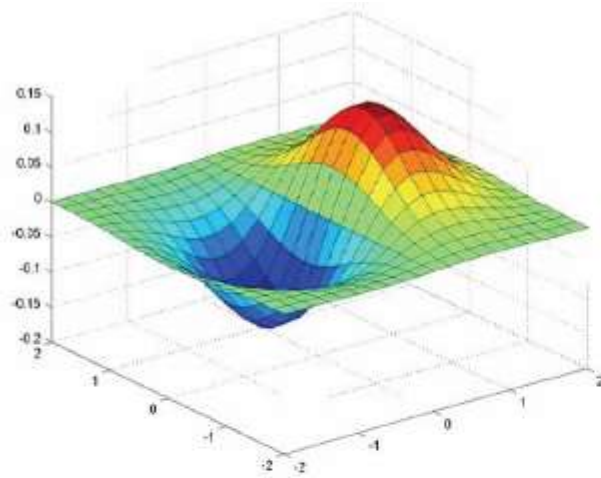
- Filter image with x- and y- derivatives of Gaussian
- Find magnitude and orientation of gradient
- Non-maximum suppression
 - Thin multi-pixel wide “ridges” down to single pixel width
- Thresholding and linking (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them

Example

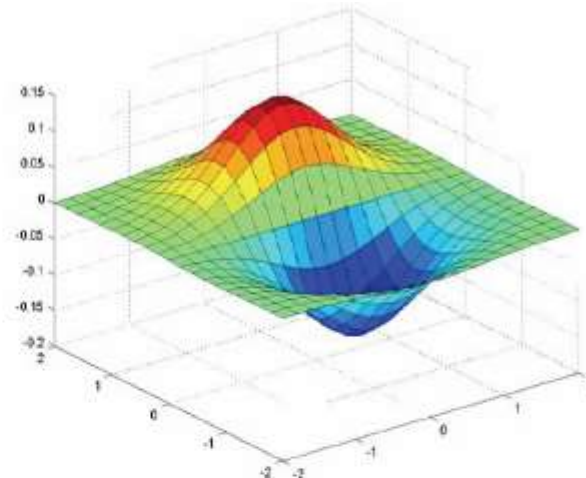
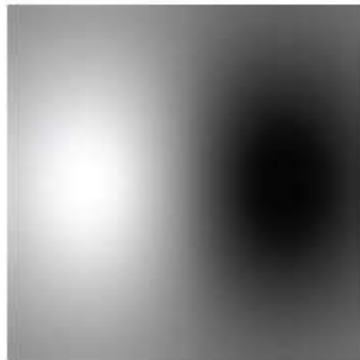


Original image (Lena)

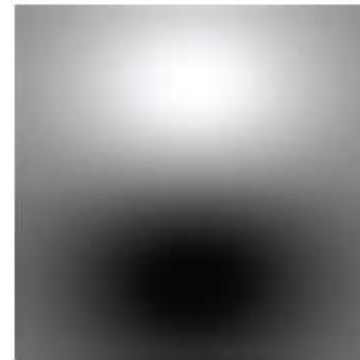
Derivative of Gaussian (DoG) filter



x-direction



y-direction



DoG filter responses



x -derivative
of Gaussian
filter response



y -derivative
of Gaussian
filter response

Canny edge detector

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

Gradient-based values

x -derivative
of Gaussian
filter response



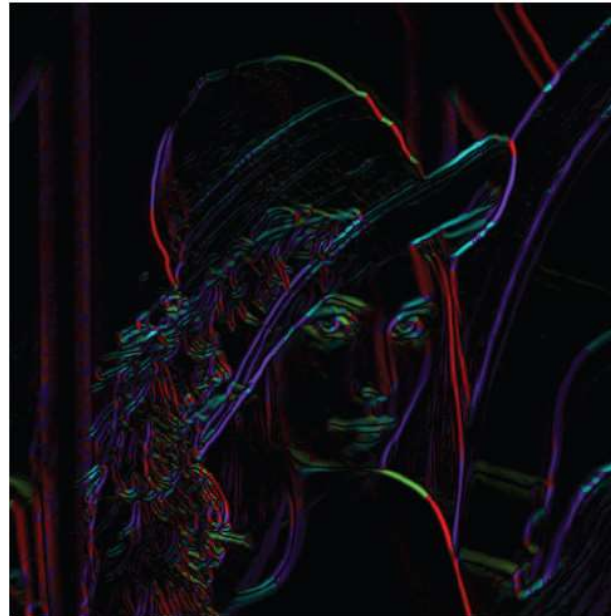
y -derivative
of Gaussian
filter response



Gradient
magnitude



Orientation
at each pixel

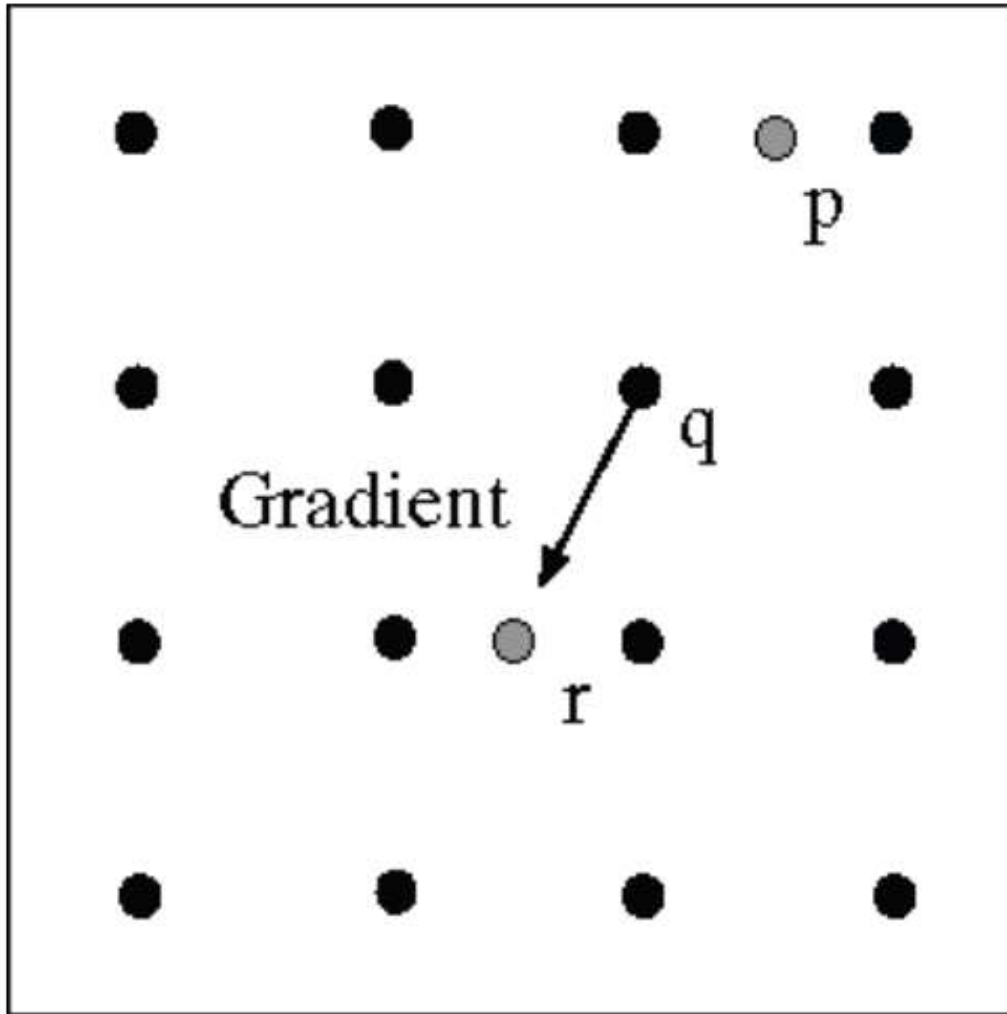


$$\theta = \tan^{-1} \frac{gy}{gx}$$

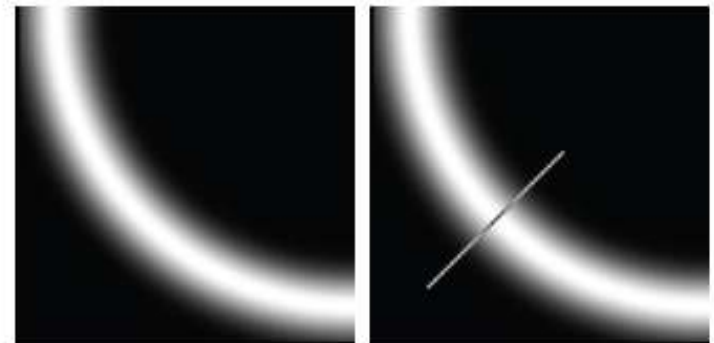
Canny edge detector

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

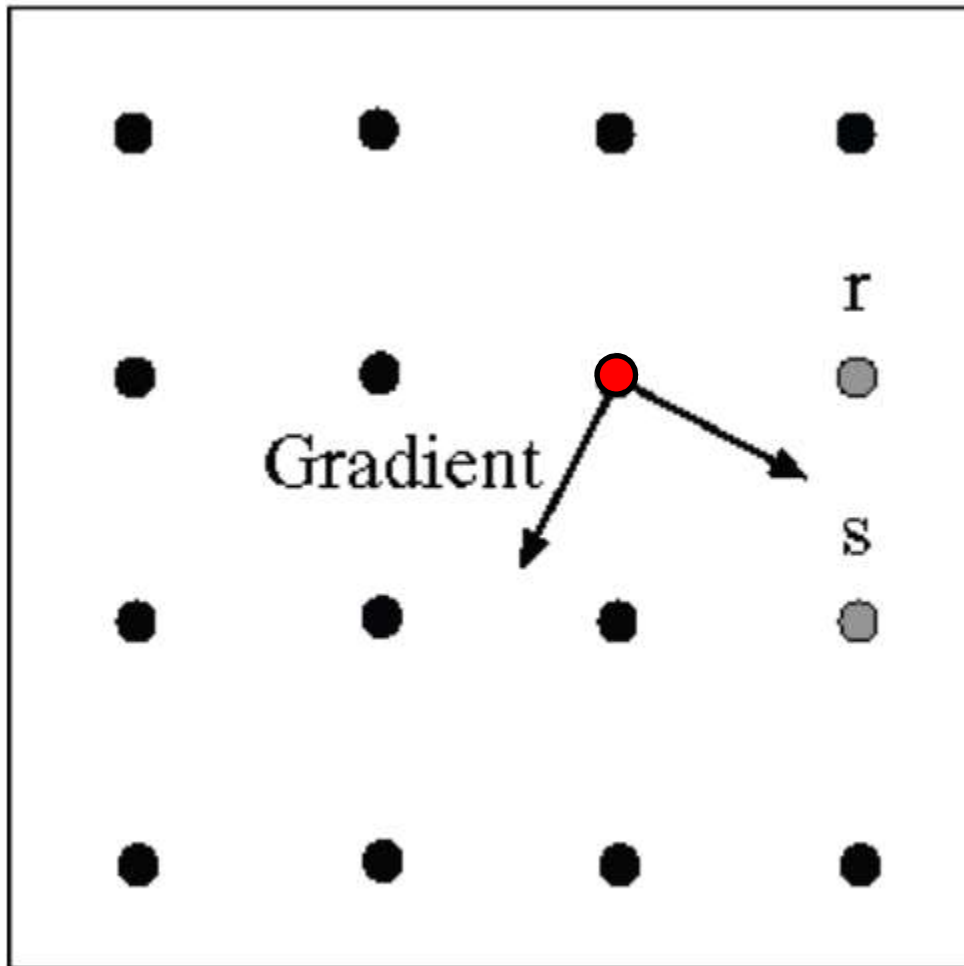
Non-maximum suppression



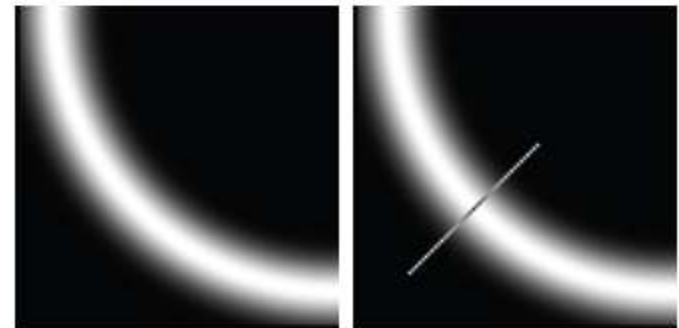
At q , we have a maximum if the value is larger than those at both p and at r . Interpolate to get these values.



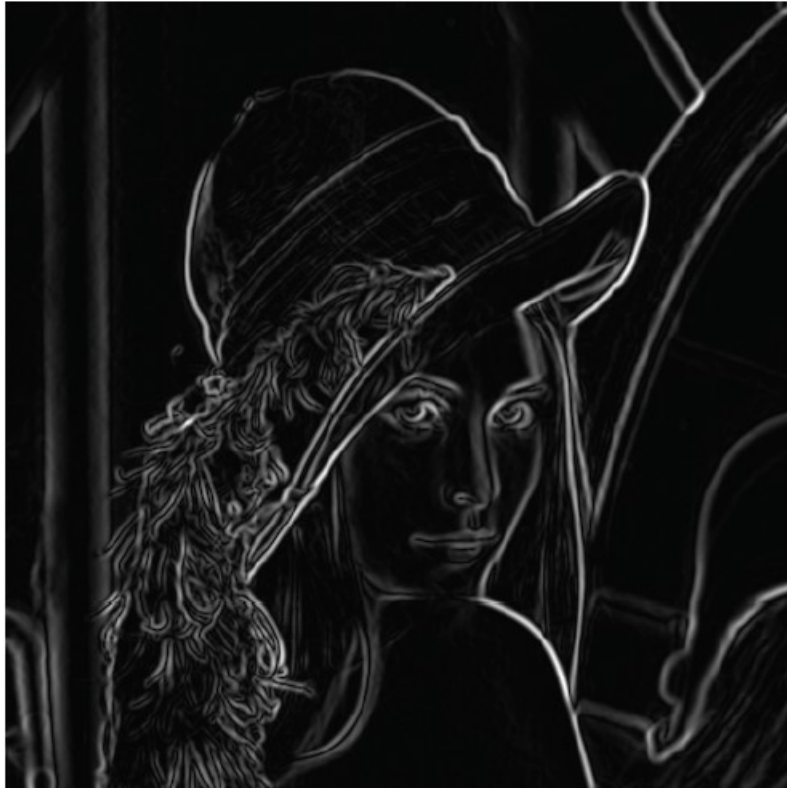
Edge Linking



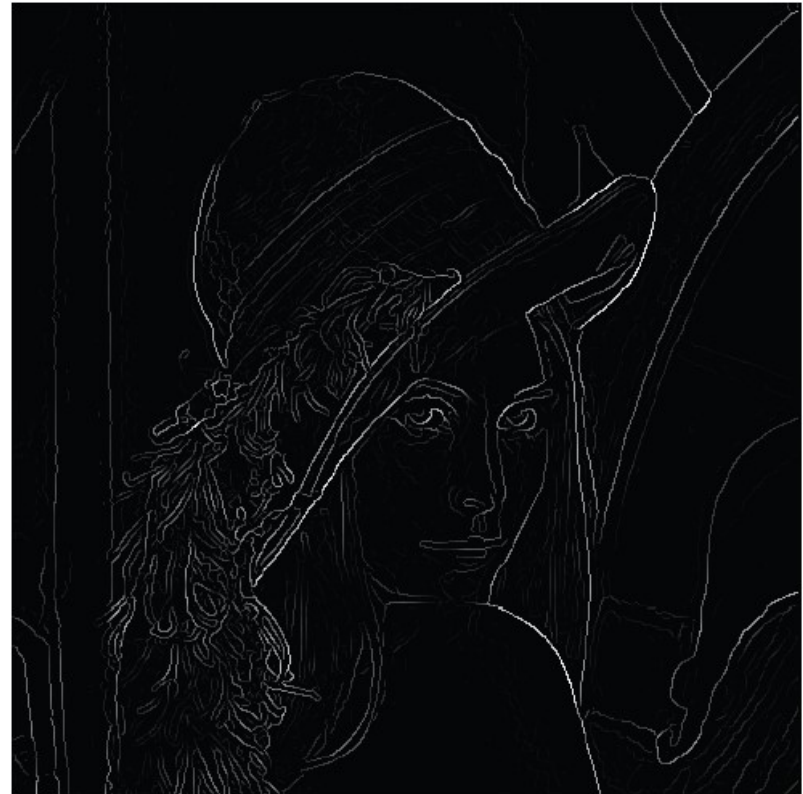
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Non-maximum suppression cont'd



Before non-max suppression



After non-max suppression

Canny edge detector

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

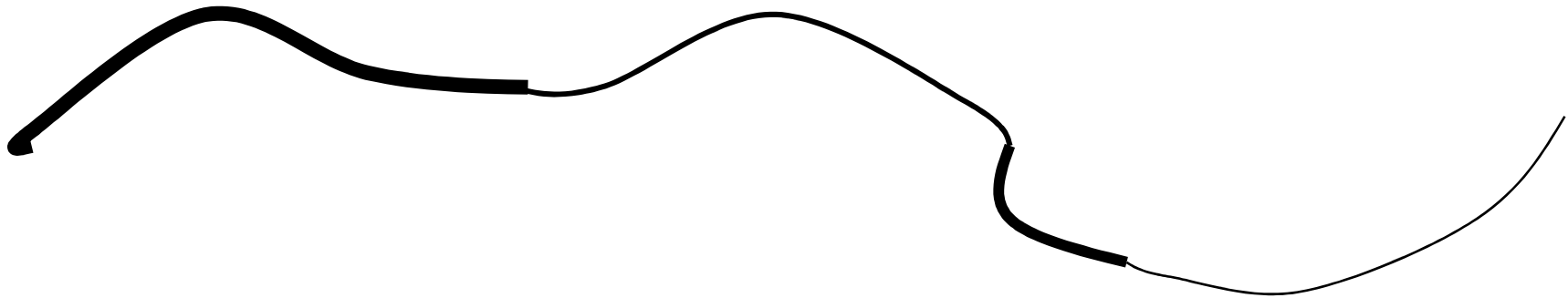
Hysteresis thresholding



- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Hysteresis thresholding

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



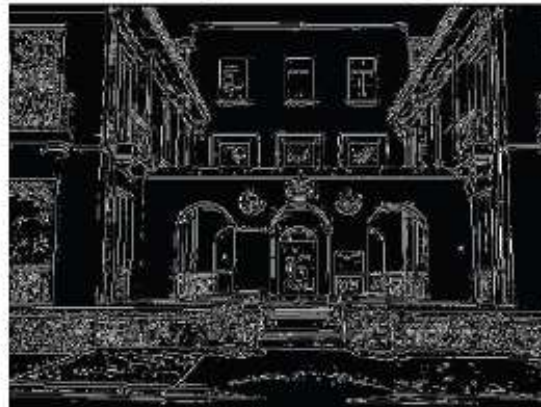
Hysteresis thresholding



original image



high threshold
(strong edges)



low threshold
(weak edges)



hysteresis threshold

Final Canny edges



Canny edge detector

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

Hysteresis thresholding



original image



high threshold
(strong edges)



low threshold
(weak edges)



hysteresis threshold

Effect of σ (Gaussian kernel spread/size)



original



Canny with $\sigma = 1$



Canny with $\sigma = 2$

The choice of σ depends on desired behavior

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

Summary

- We started last week with linear filters
 - including filter construction and separability, convolution methods and image blurring
- Today we discussed filter derivatives and scale space/pyramids
 - including 1st and 2nd derivatives of the Gaussian filters, the Gaussian pyramid and the Laplacian pyramid
- In this lecture we discussed how DoG filters detect edges and how post-processing works
 - specifically we focused on the Canny edge detector and its post-processing techniques

Slide Credits

- David A. Forsyth - UIUC
- Fei Fei Li - Stanford
- Svetlana Lazebnik – UIUC
- Rob Fergus – NYU

Next class

- Local features
- Readings for next lecture:
 - Forsyth and Ponce Chp 5, 4.2
 - (optional) Szeliski 3.1-3.3
- Readings for today:
 - Forsyth and Ponce FP 4.7

Questions

