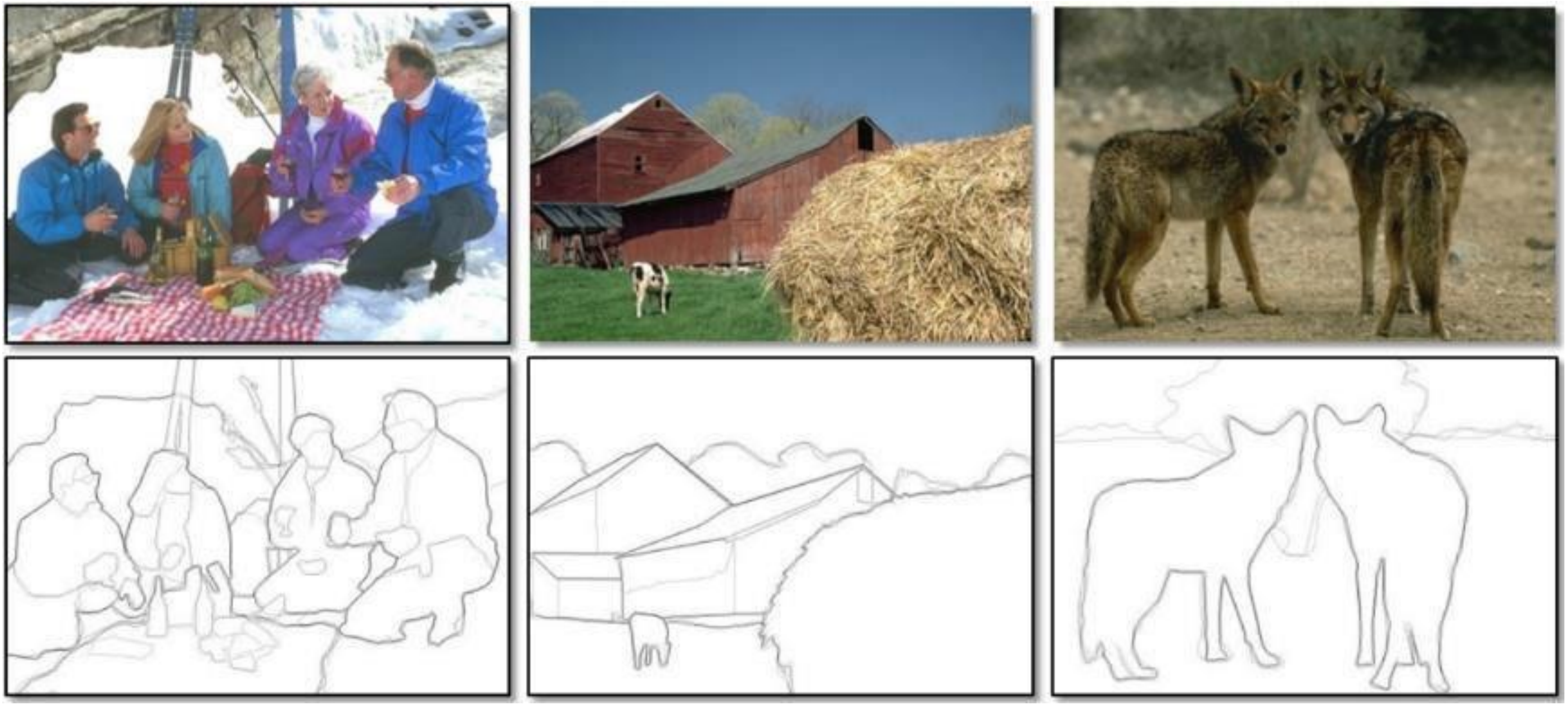
Computer Vision
Edge Detection

Dr. Mrinmoy Ghorai

Indian Institute of Information Technology
Sri City, Chittoor

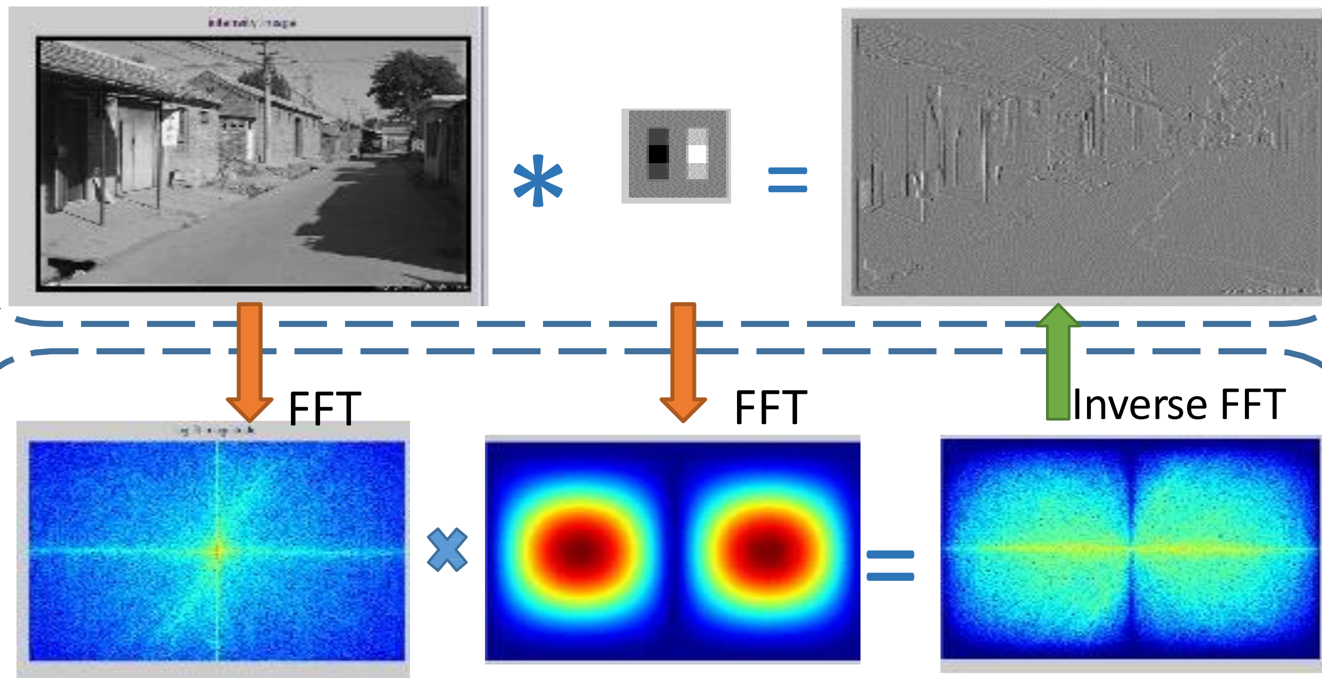
Today's Agenda: Edge Detection



Previous classes: Image Filtering

Spatial domain

- Smoothing, sharpening, measuring texture



Frequency domain

- Denoising, sampling

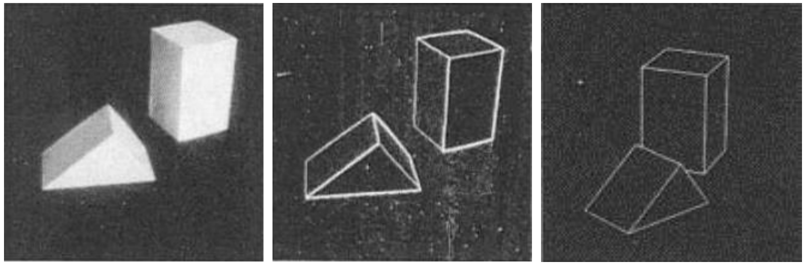
Today's class

- Detecting edges

- Finding straight lines

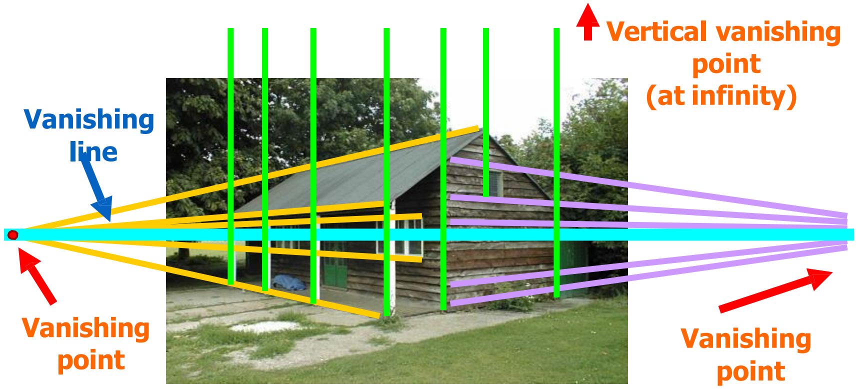
Why finding edges is important?

- Cues for 3D shape

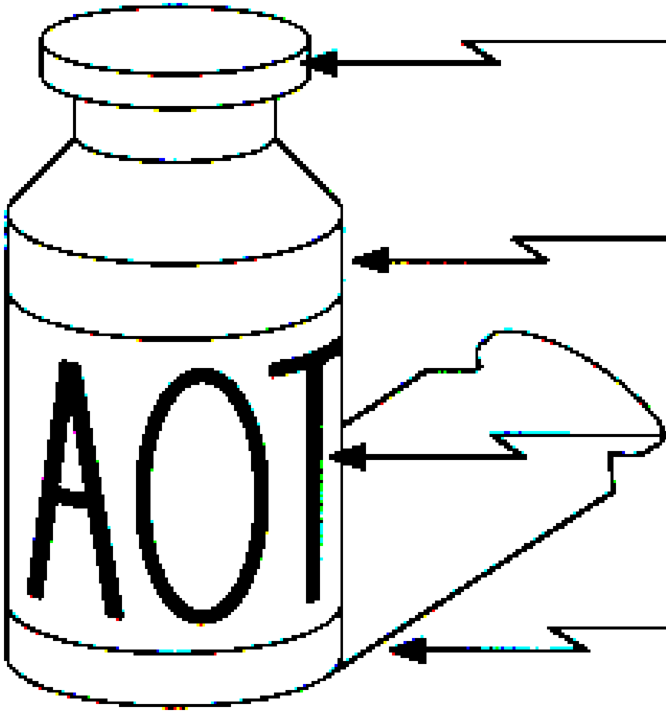


- Group pixels into objects or parts

- Shape analysis
- Recover geometry and viewpoint



Origin of Edges



surface normal discontinuity

depth discontinuity surface

color discontinuity

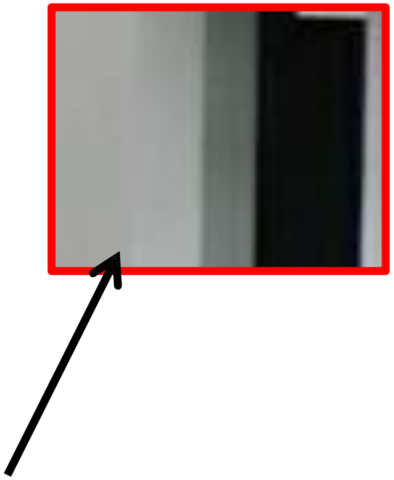
illumination discontinuity

Edges are caused by a variety of factors

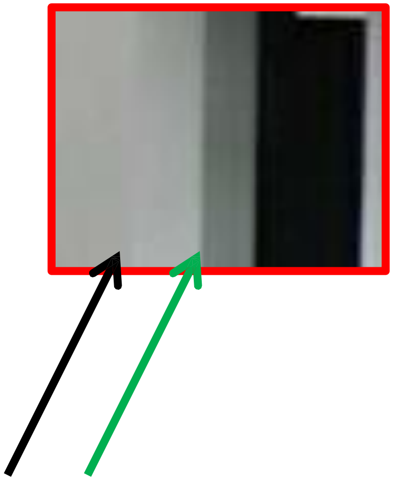
Source: Steve Seitz

Closeup of edges

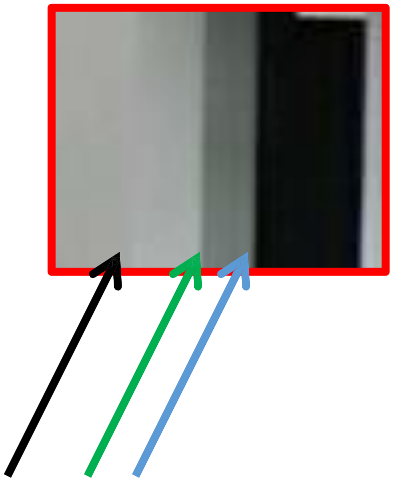
Closeup of edges



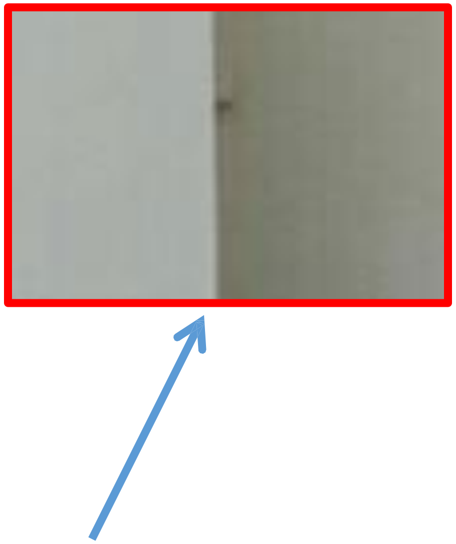
Closeup of edges



Closeup of edges



Closeup of edges

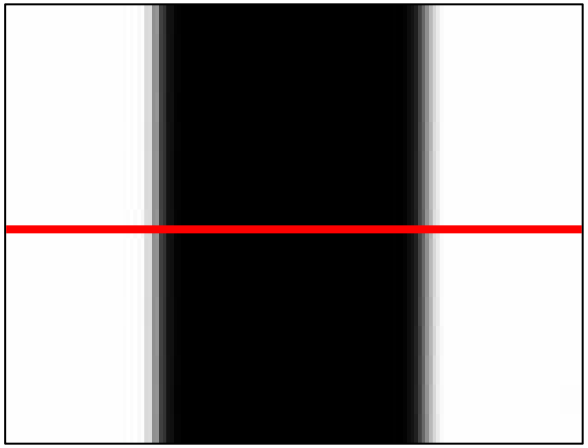


Closeup of edges

Characterizing edges

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

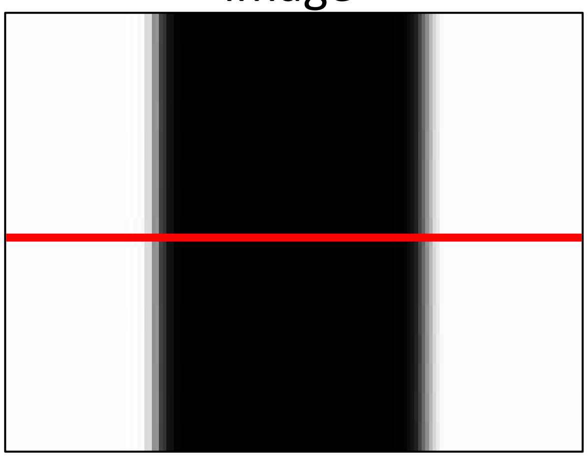
image



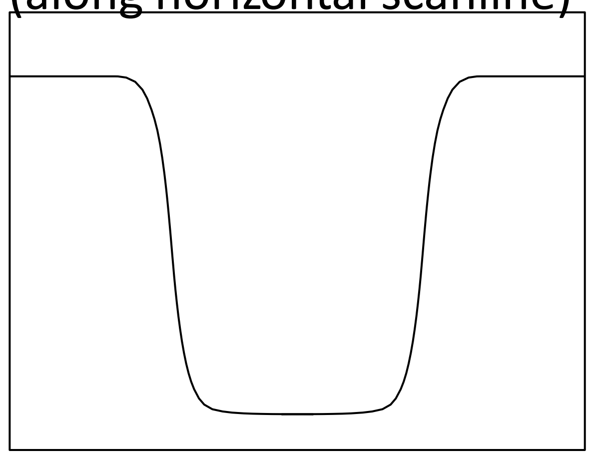
Characterizing edges

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

image



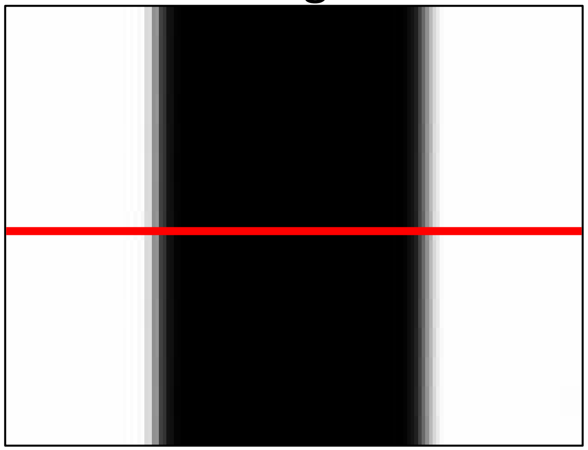
intensity function
(along horizontal scanline)



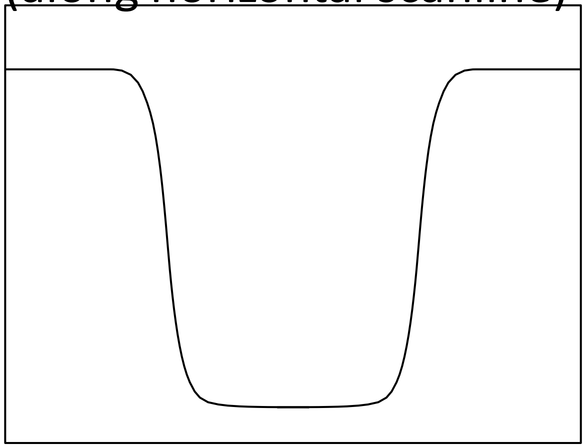
Characterizing edges

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

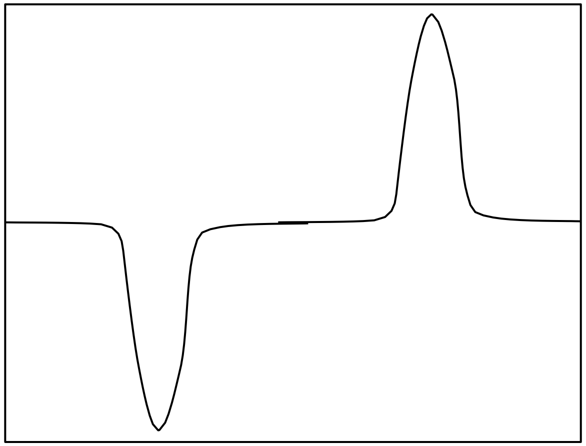
image



intensity function
(along horizontal scanline)

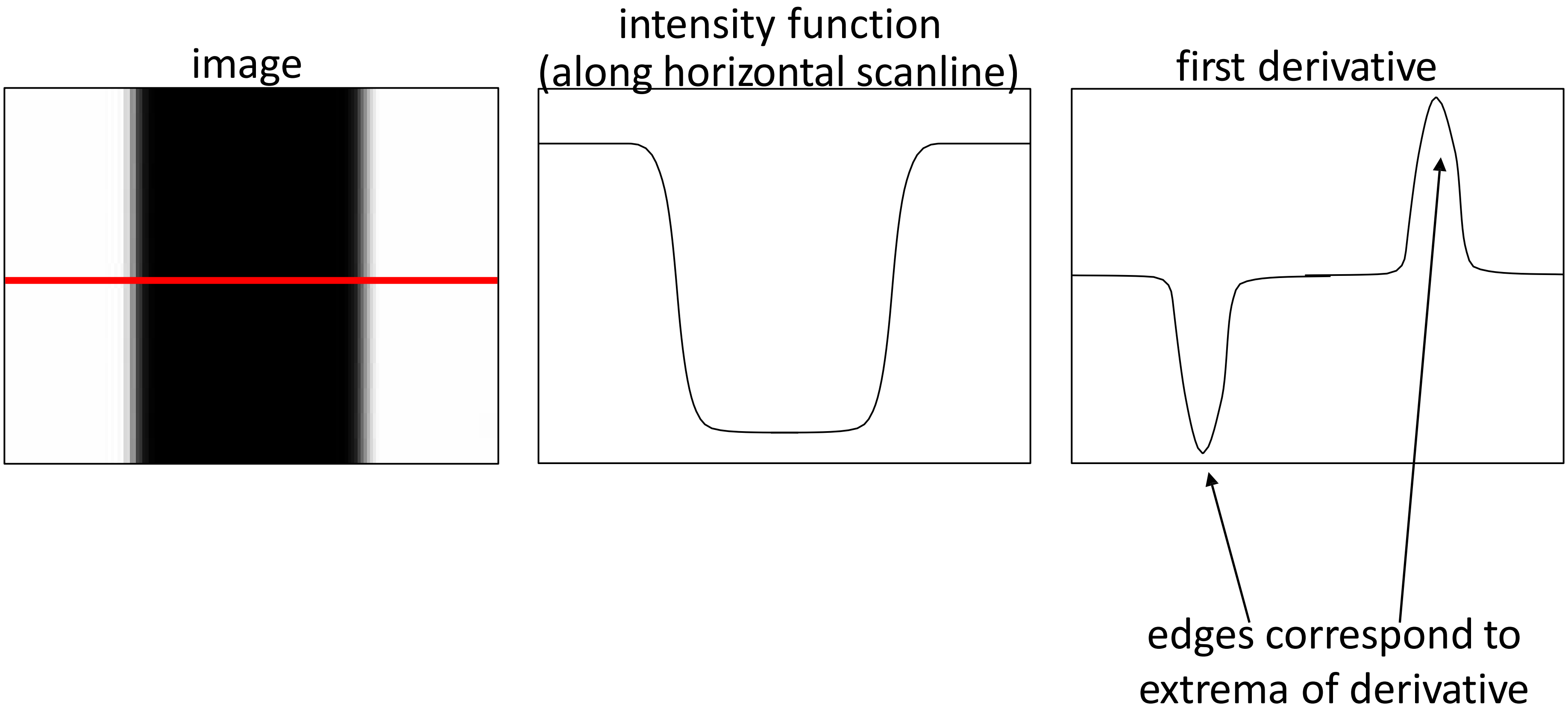


first derivative

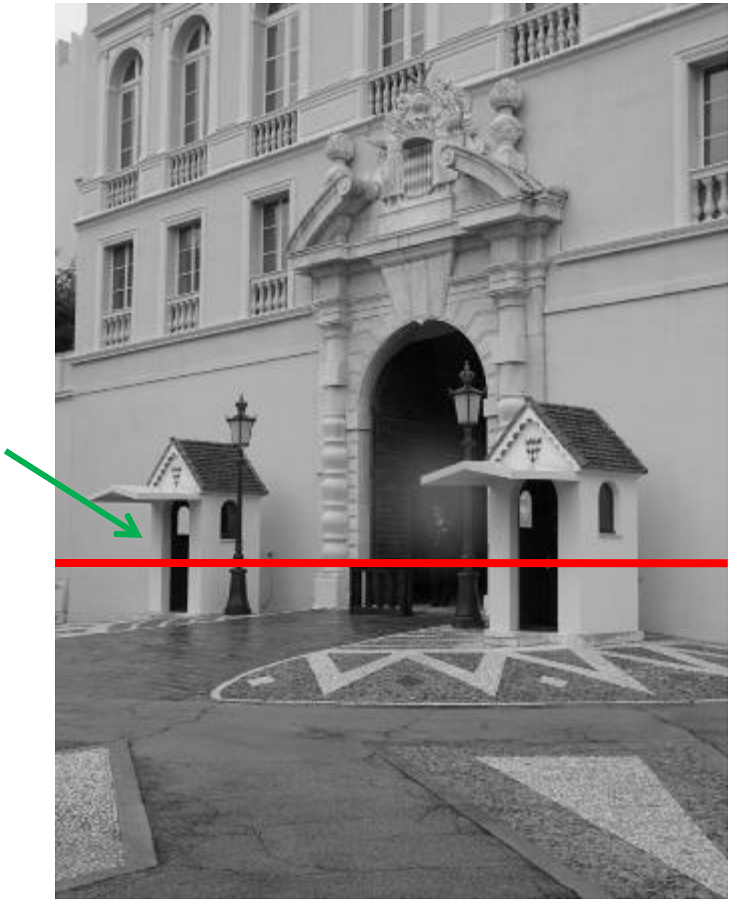


Characterizing edges

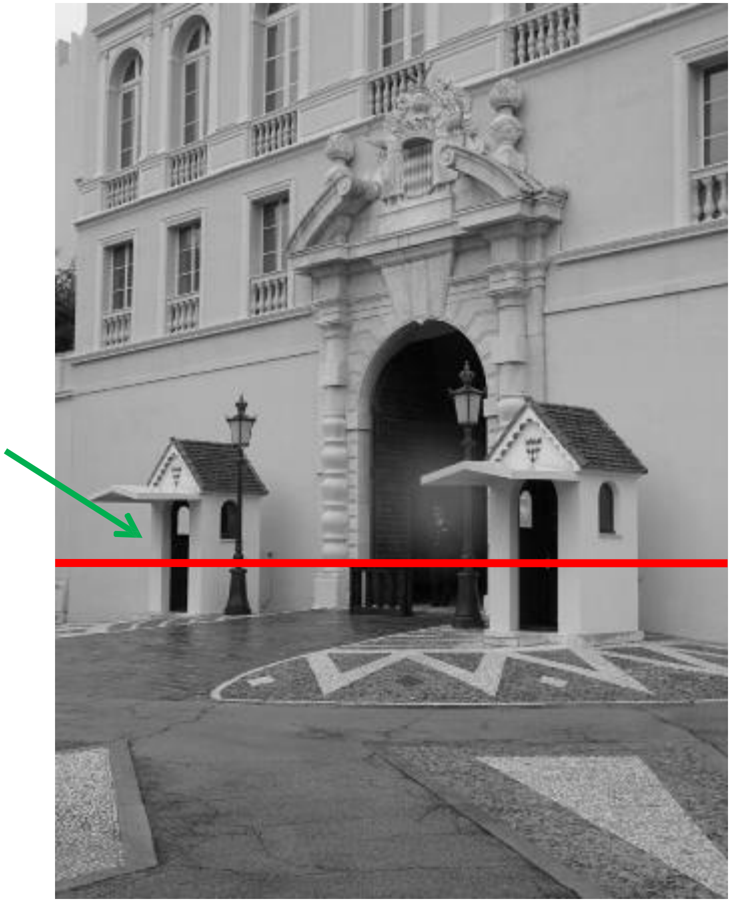
- An edge is a place of rapid change in the image intensity function

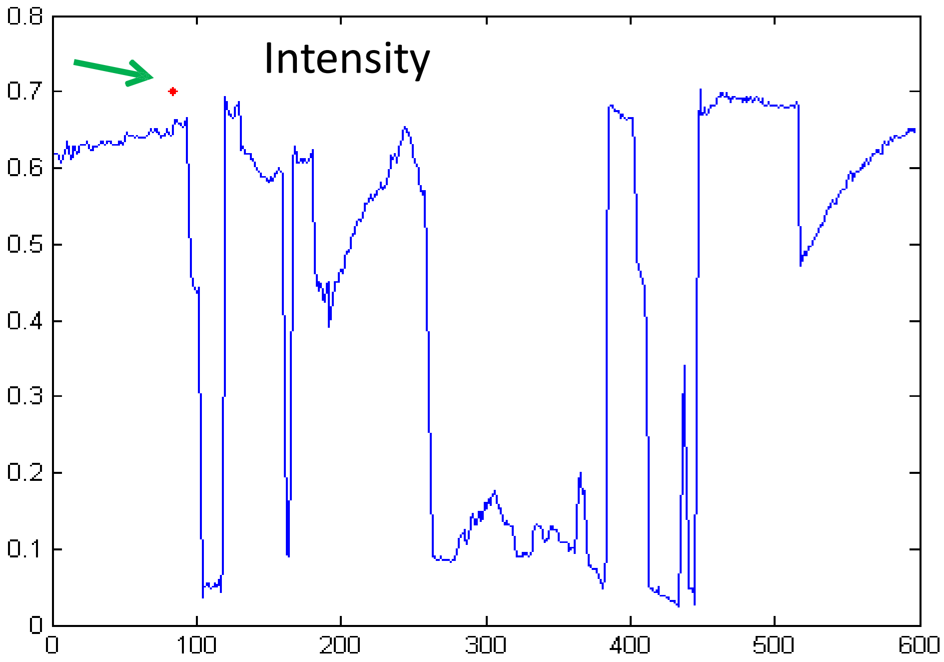


Intensity profile

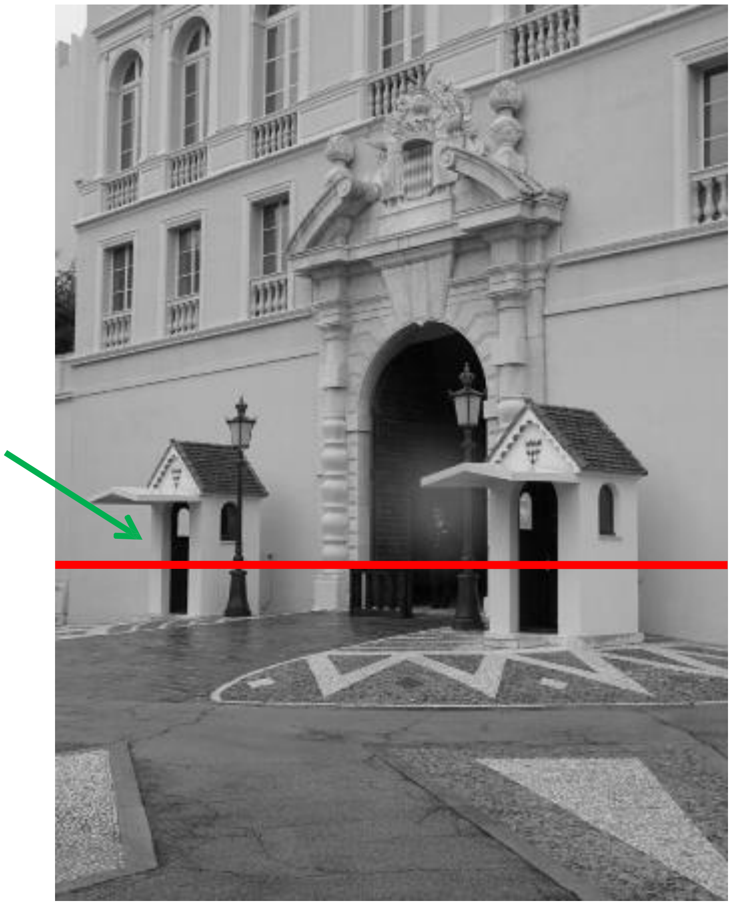


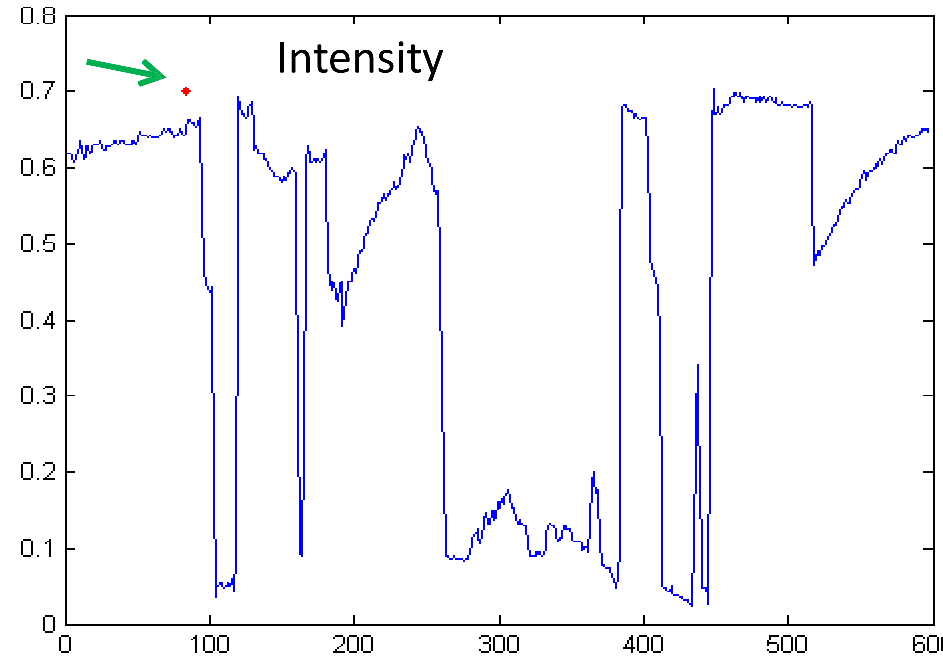
Intensity profile

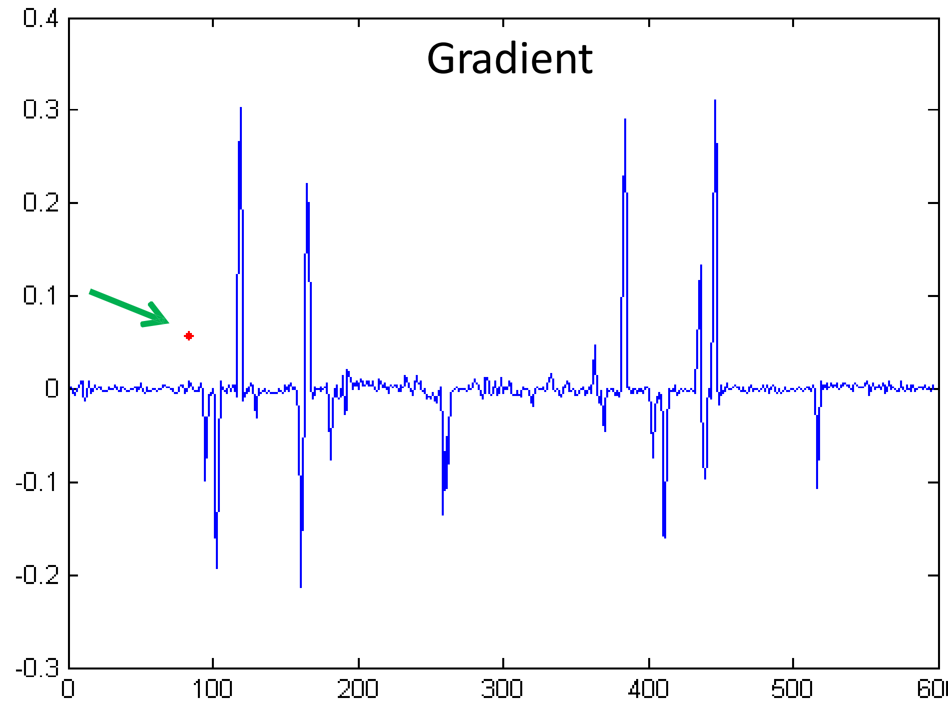




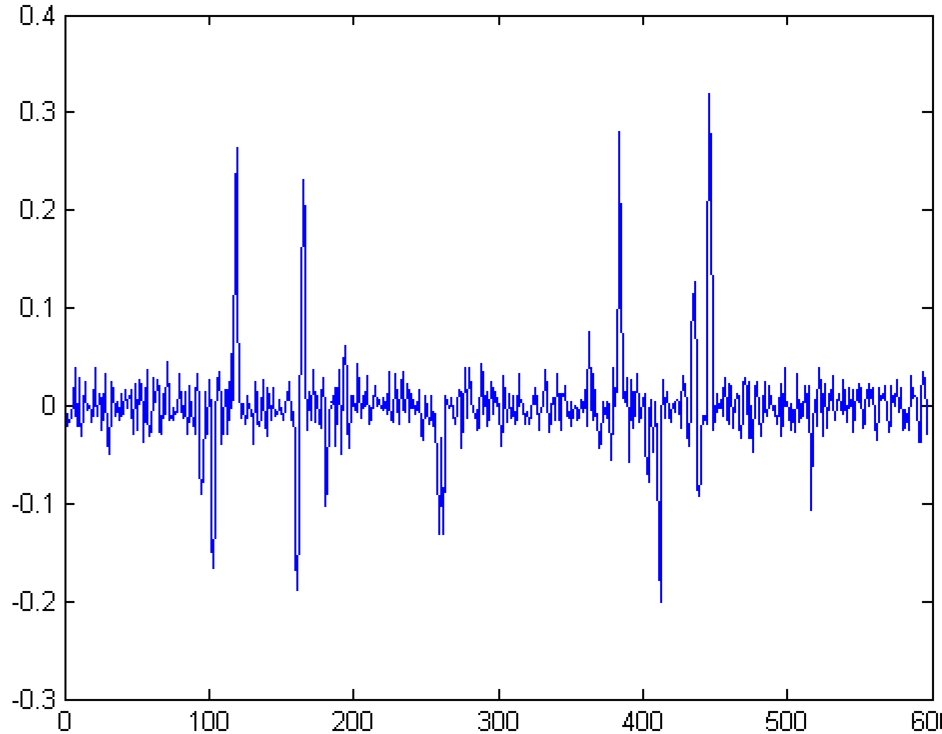
Intensity profile







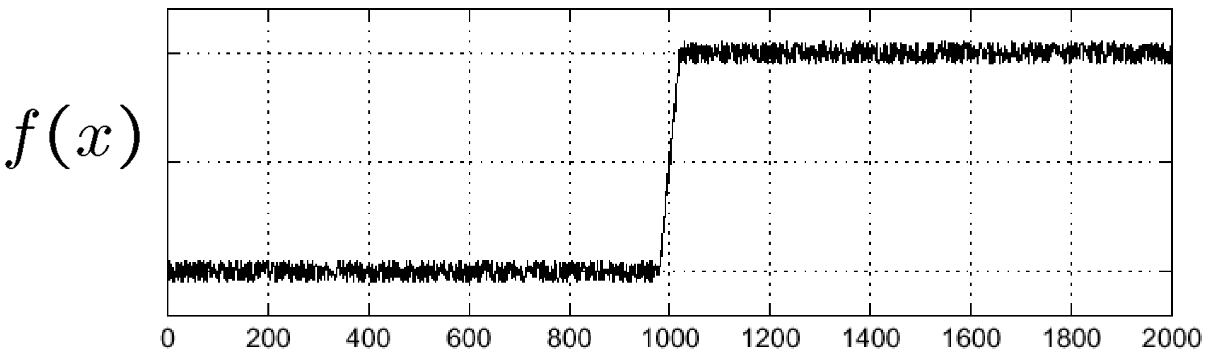
With a little Gaussian noise



Gradient

Effects of noise

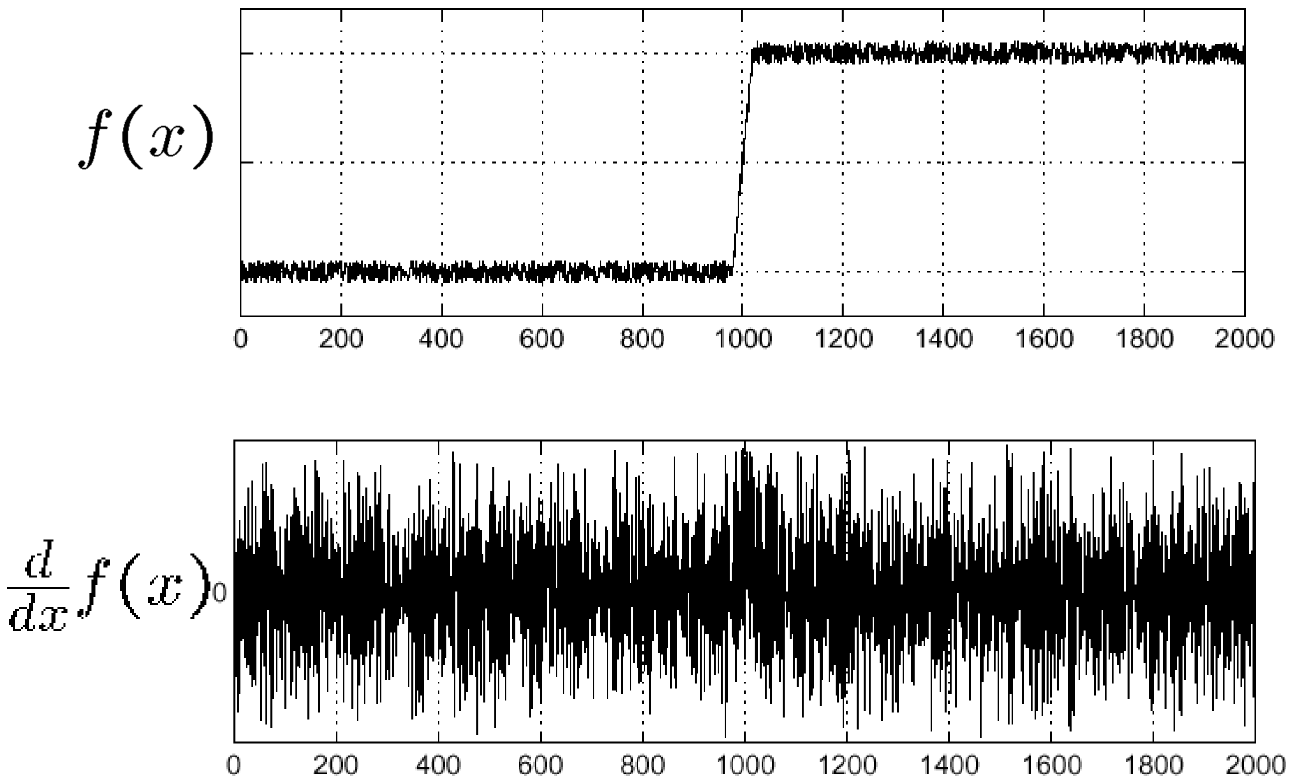
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



Source: S. Seitz

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



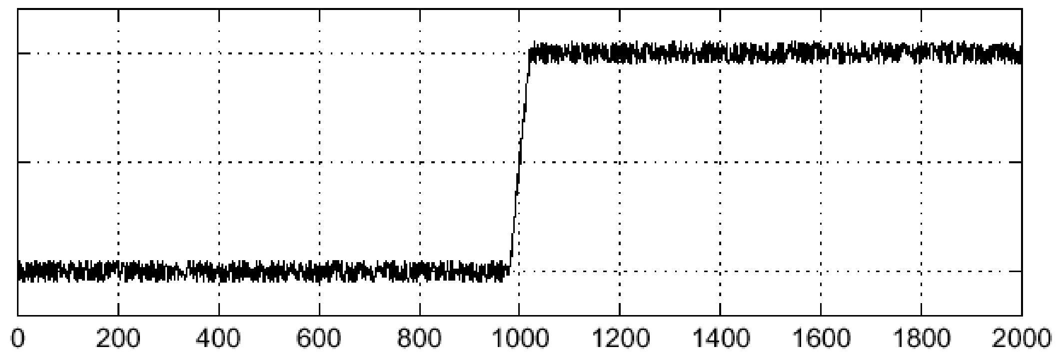
Where is the edge?

Source: S. Seitz

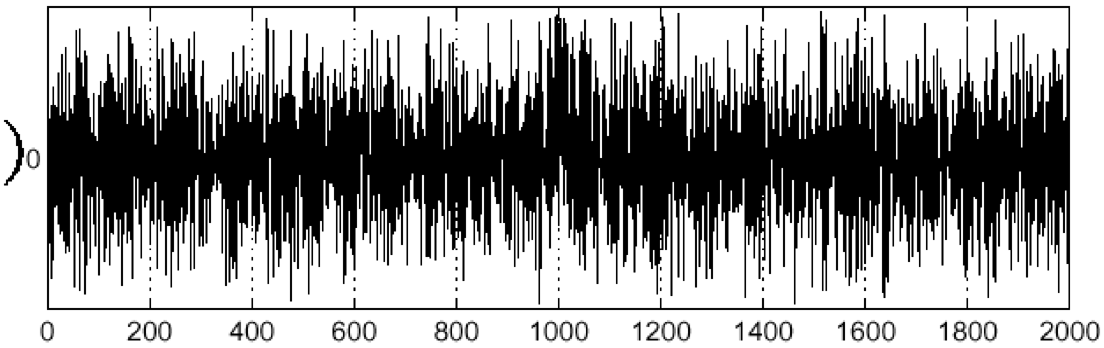
Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

$f(x)$



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

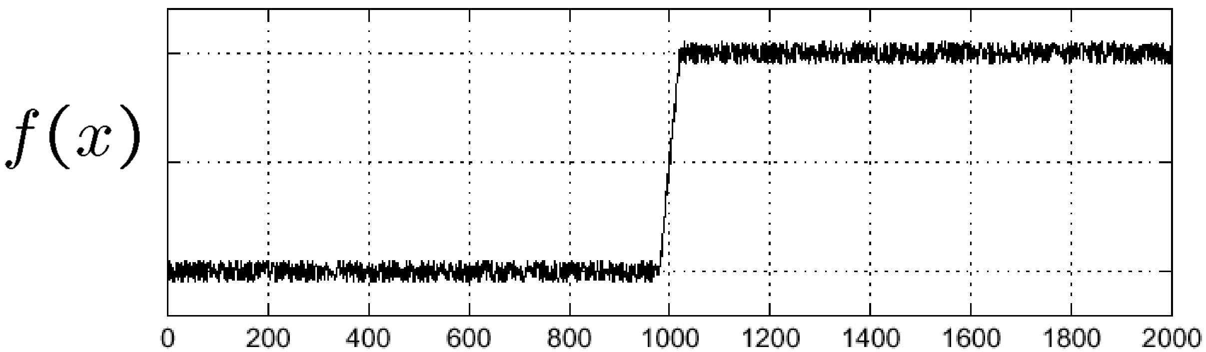


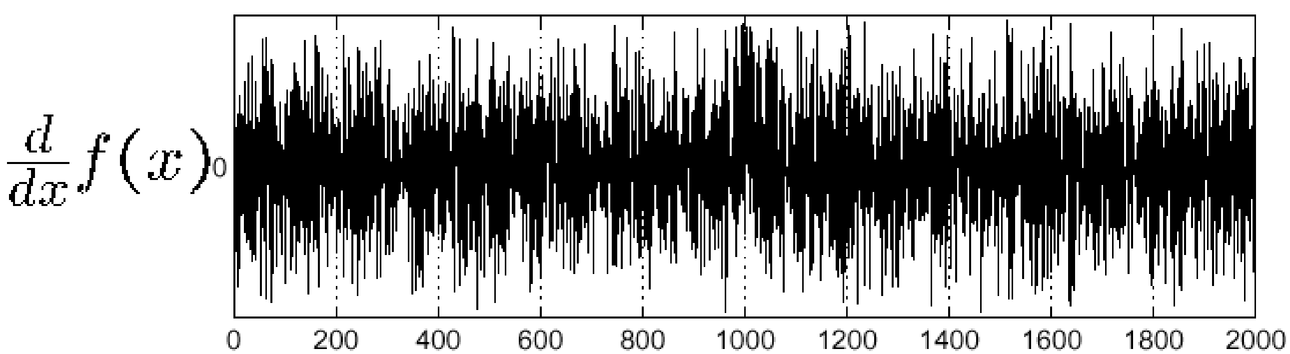
Where is the edge?

The larger the noise the stronger the response

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

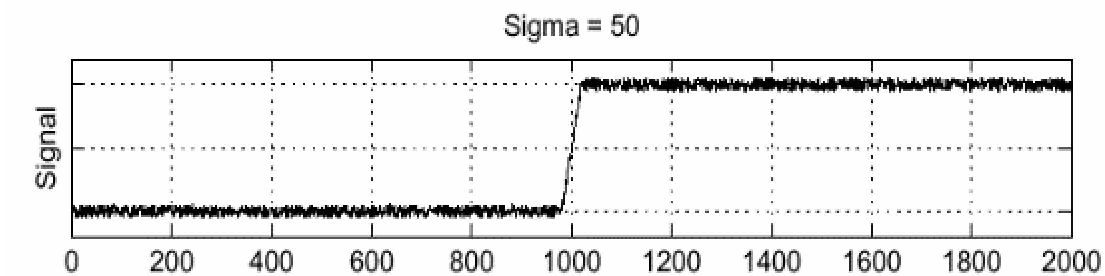




- **What can we do about it?**

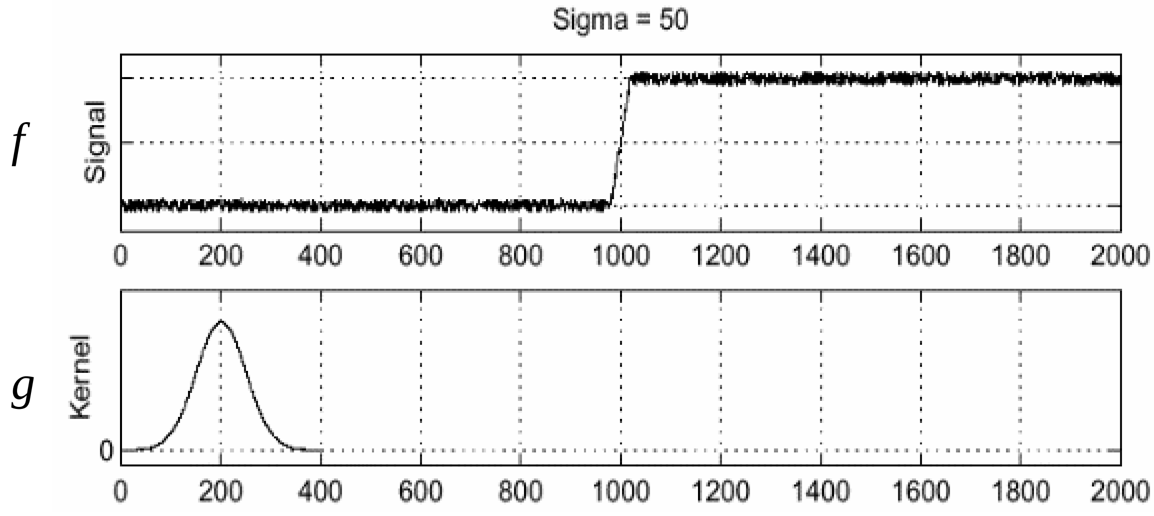
Solution: smooth first

f



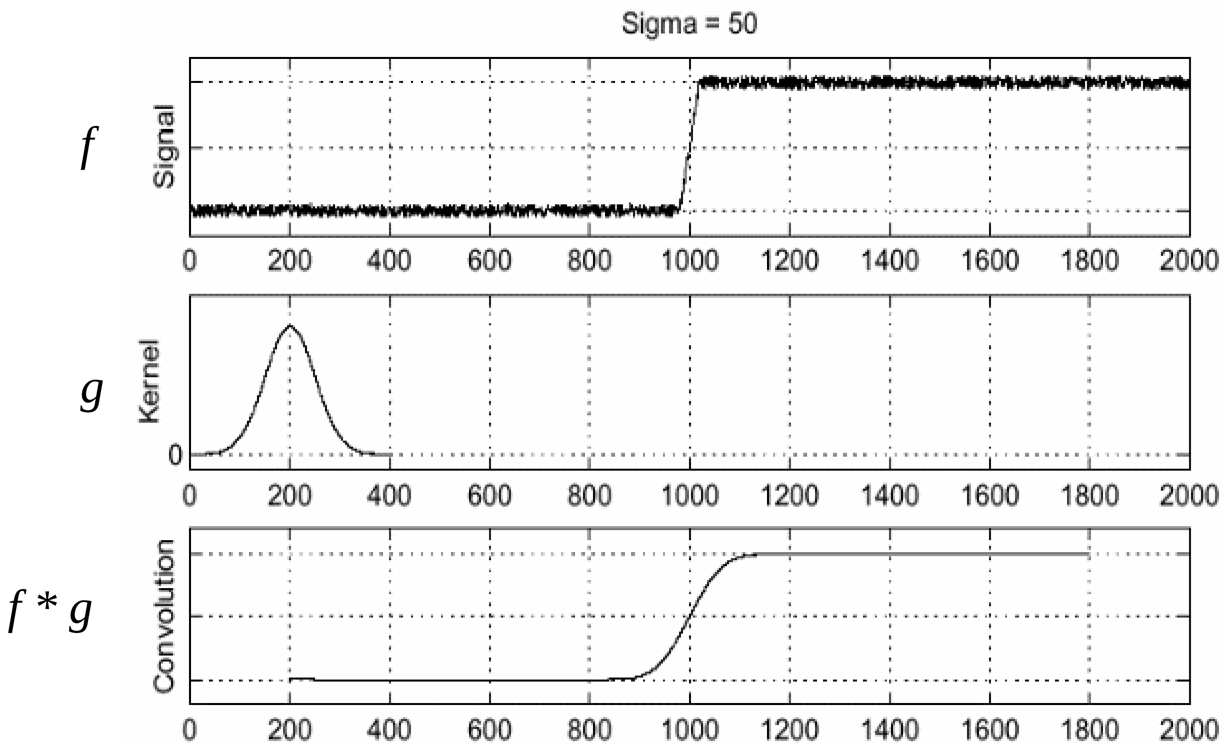
Source: S. Seitz

Solution: smooth first



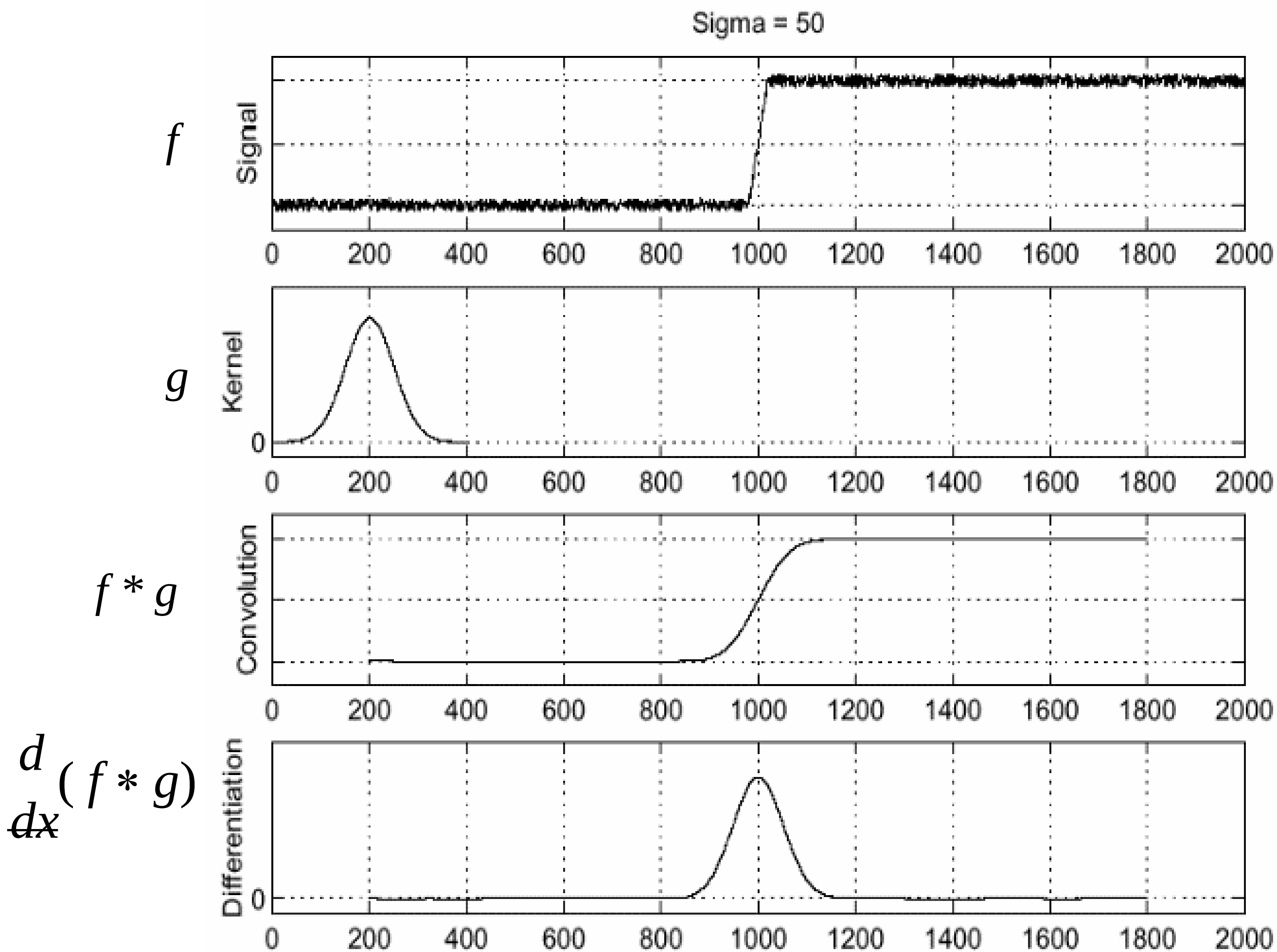
Source: S. Seitz

Solution: smooth first



Source: S. Seitz

Solution: smooth first



- To find edges, look for peaks in $$\frac{d}{dx}(f * g)$$

Source: S. Seitz

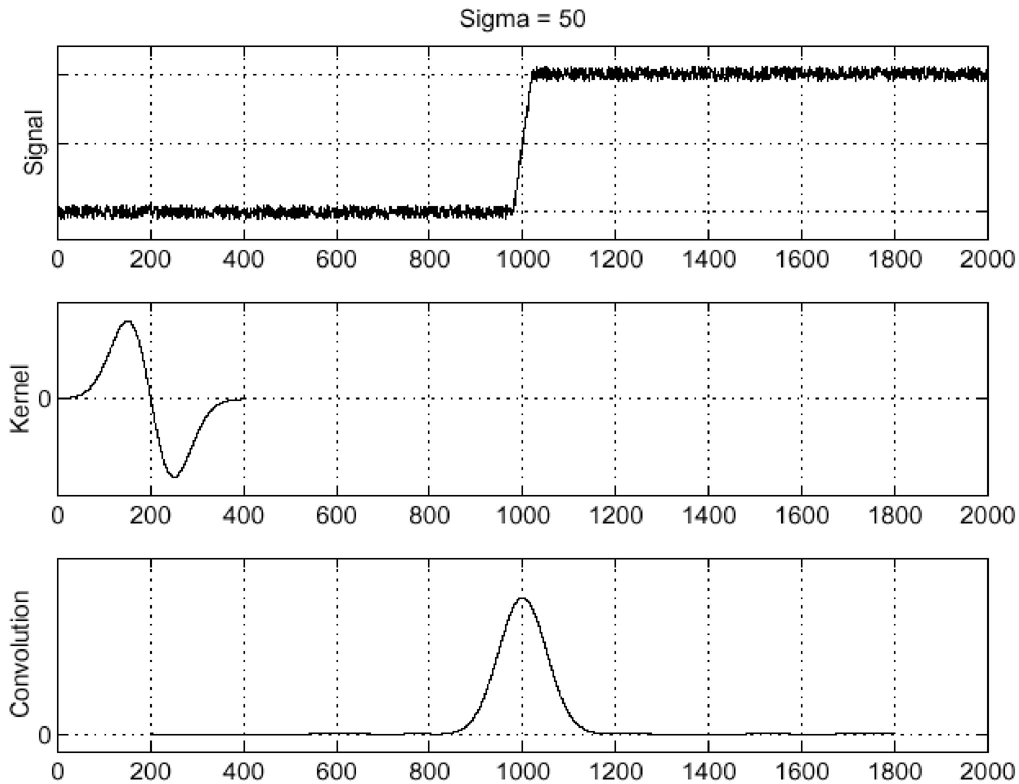
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative: $$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$
- This saves us one operation:

f

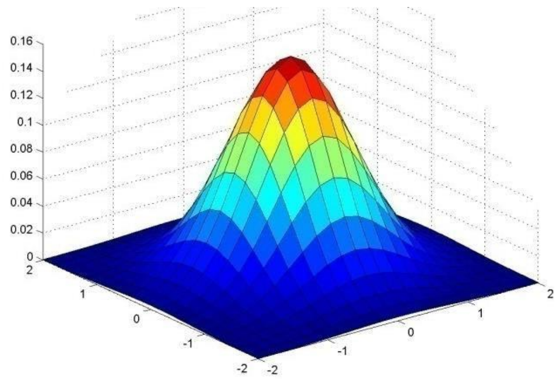
$\frac{d}{dx} g$

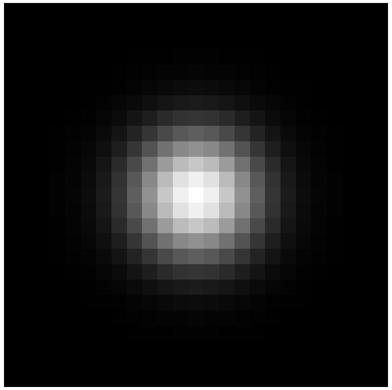
$f * \frac{d}{dx} g$



Source: S. Seitz

Gaussian Kernel

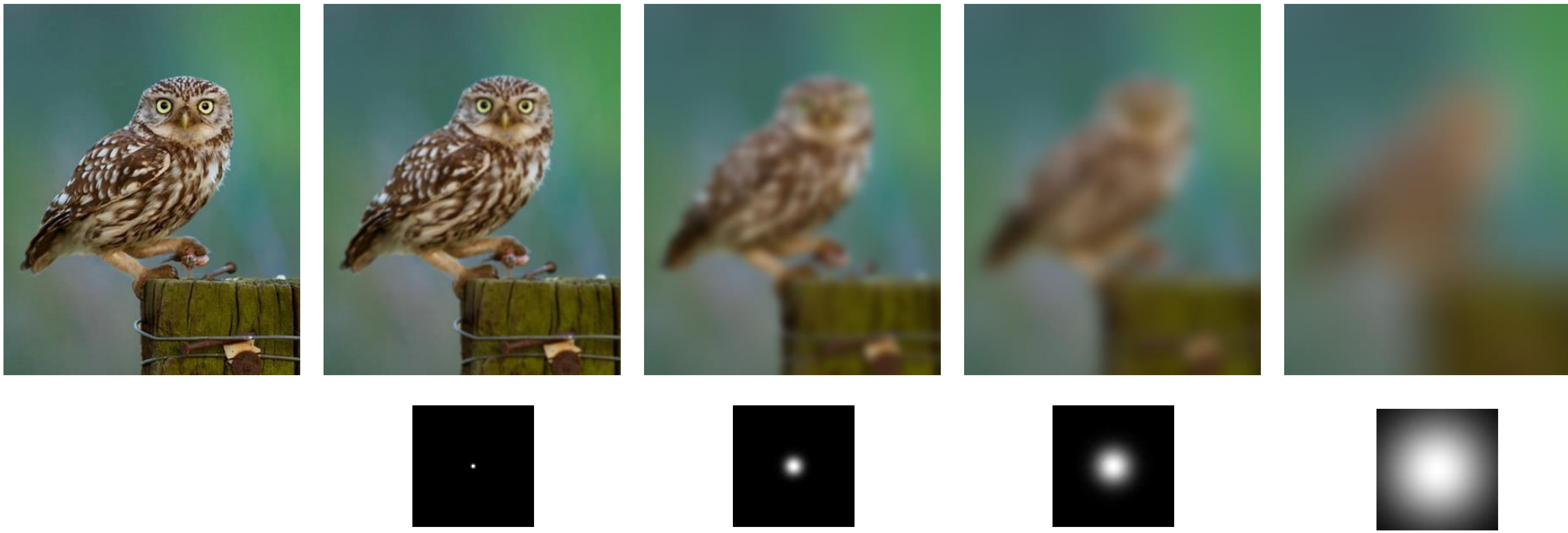




$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

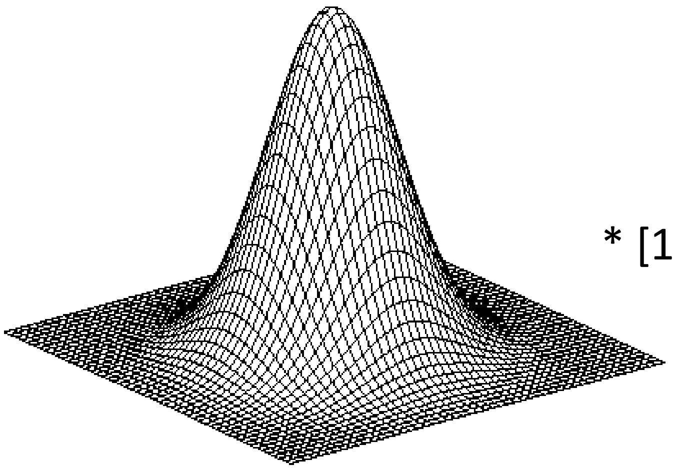
Source: C. Rasmussen

Gaussian filters



σ = 1 pixel　σ = 5 pixels　σ = 10 pixels　σ = 30 pixels

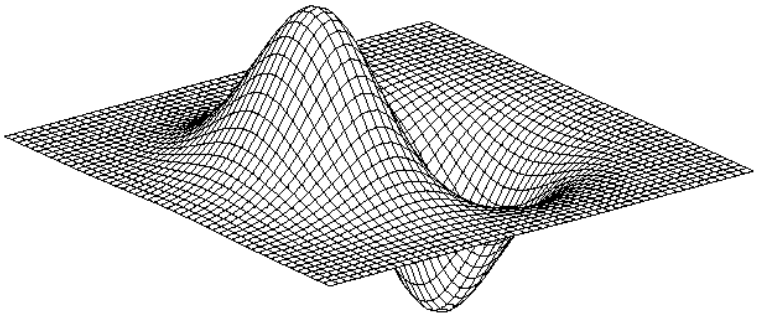
Derivative of Gaussian filter



Gaussian

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

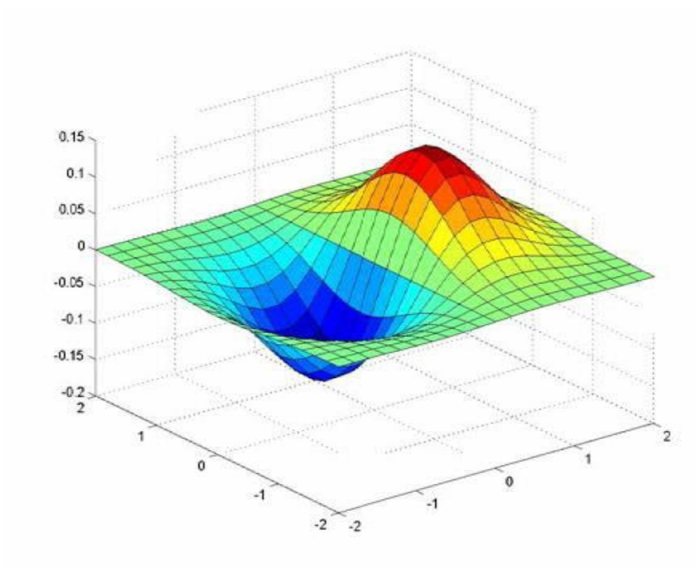
* [1 0 -1] =



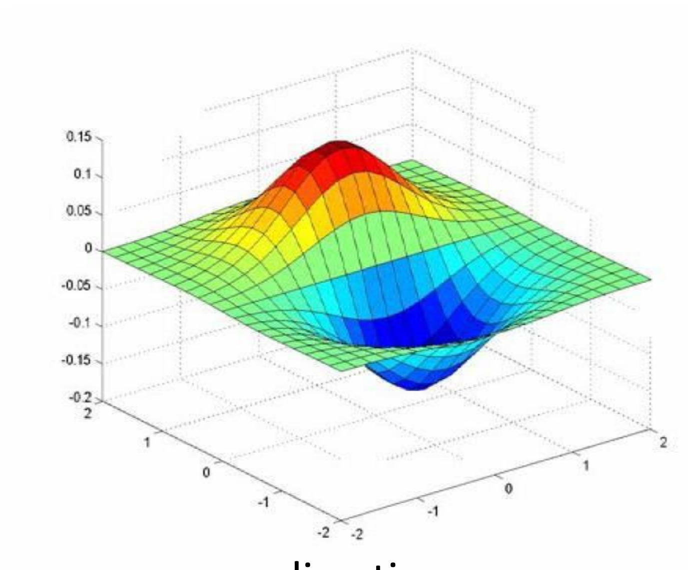
derivative of Gaussian (*x*)

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

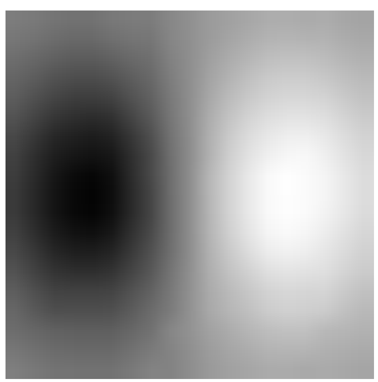
Derivative of Gaussian filter

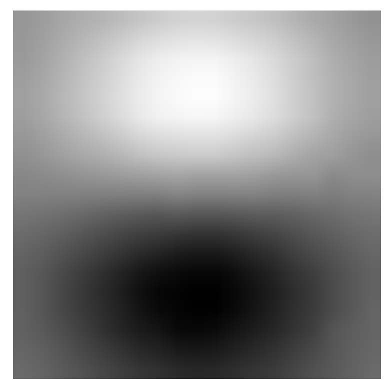


x-direction



y-direction





Designing an edge detector

- Criteria for a good edge detector:
  - **Good detection:**
    - find all real edges, ignoring noise or other artifacts

Source: L. Fei-Fei

Designing an edge detector

- Criteria for a good edge detector:
  - **Good detection:**
    - find all real edges, ignoring noise or other artifacts
  - **Good localization**
    - detect edges as close as possible to the true edges
    - return one point only for each true edge point

Source: L. Fei-Fei

Canny edge detector

- The most widely used edge detector

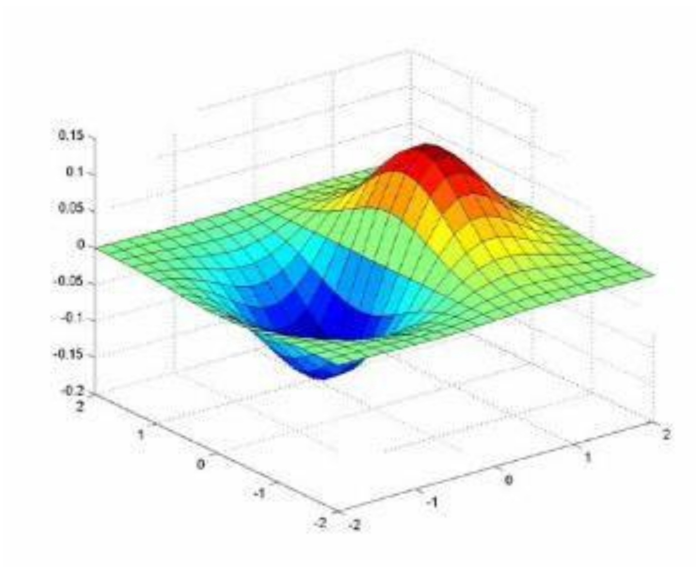
A computational approach to edge detection
J Canny - IEEE Transactions on pattern analysis and machine …, 1986 - ieeexplore.ieee.org
Abstract: This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a comprehensive set of goals for the 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, and present mathematical forms for ...
Cited by 27743 Related articles All 27 versions Import into BibTeX Save More

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

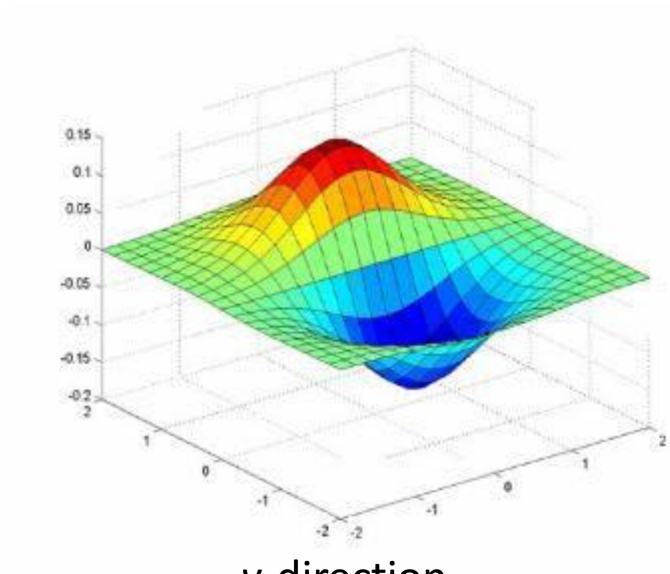
Example

input image (“Lena”)

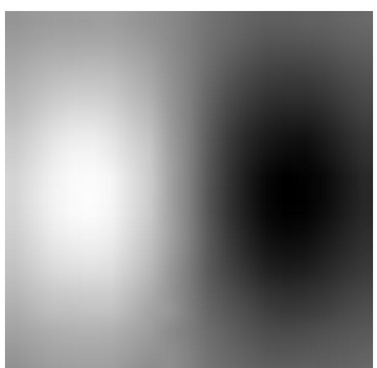
Derivative of Gaussian filter

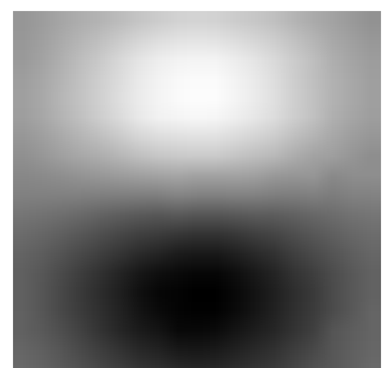


x-direction

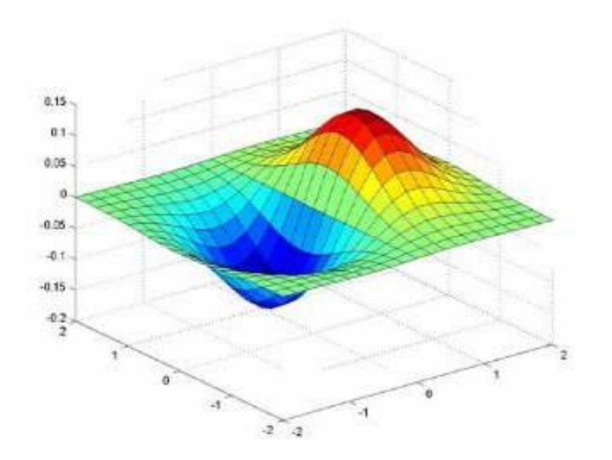


y-direction

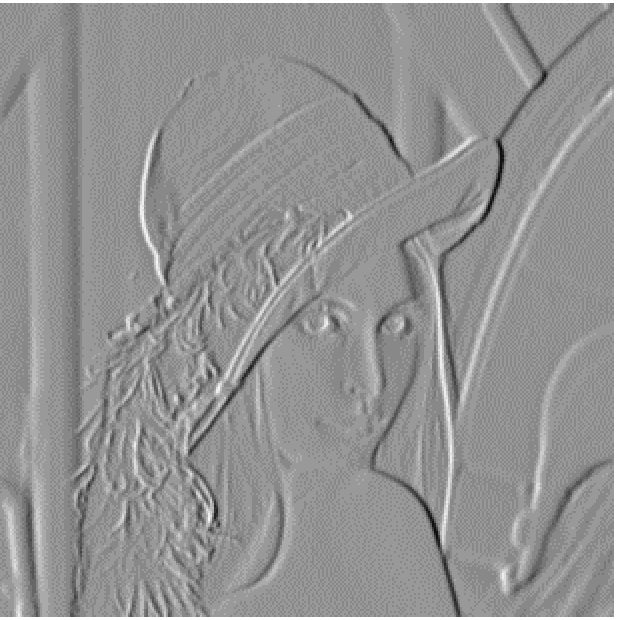




Compute Gradients (DoG)

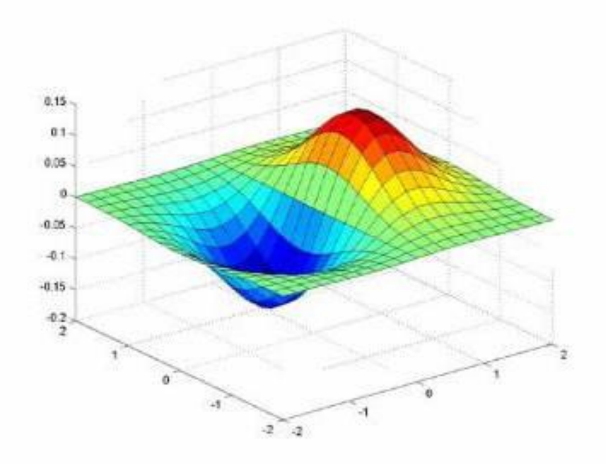


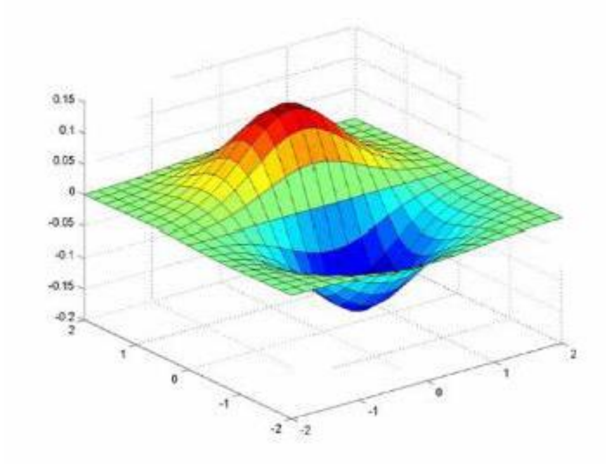
Input Image



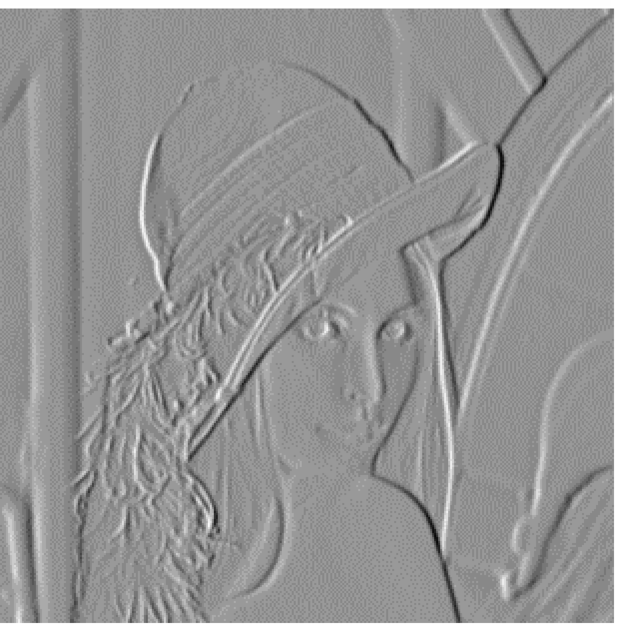
X-Derivative of Gaussian

Compute Gradients (DoG)





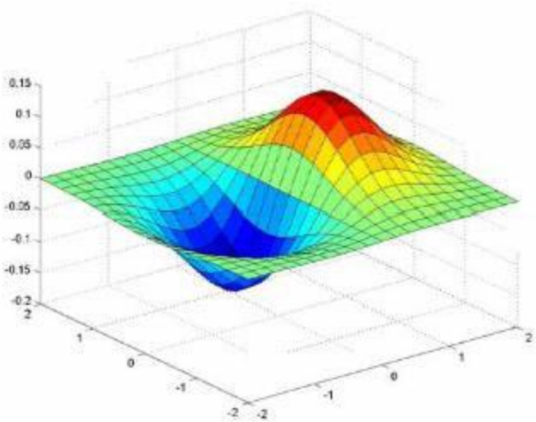
Input Image

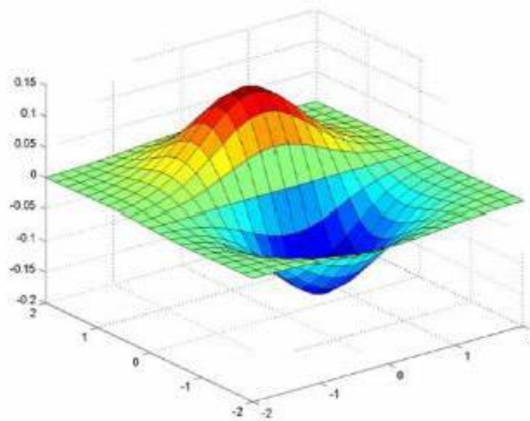


X-Derivative of Gaussian

Y-Derivative of Gaussian

Compute Gradients (DoG)

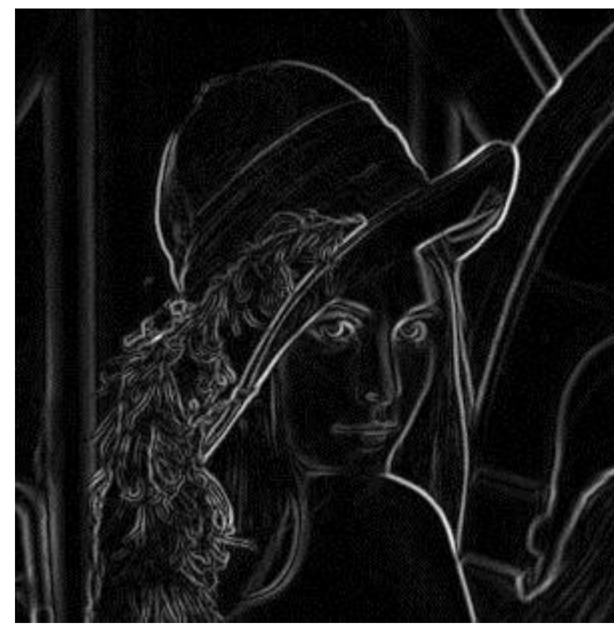




Input Image

X-Derivative of Gaussian

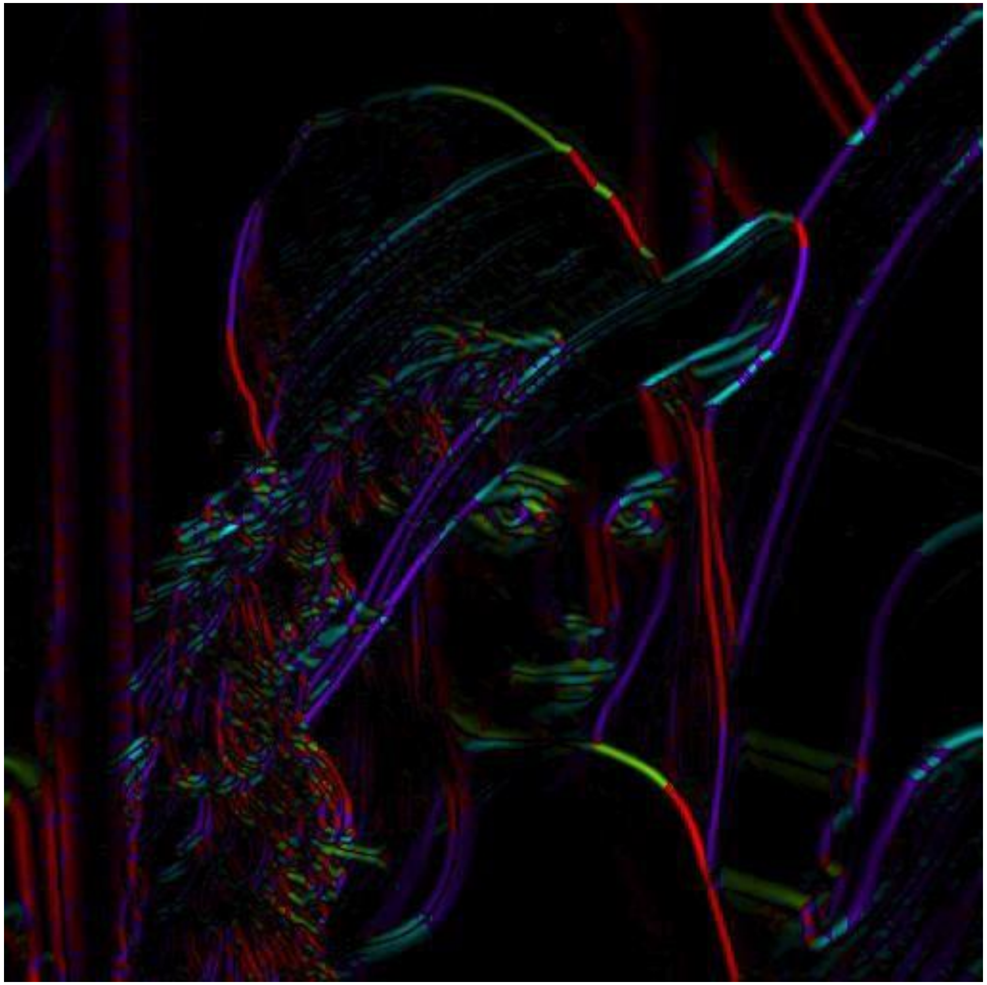
Y-Derivative of Gaussian



Gradient Magnitude

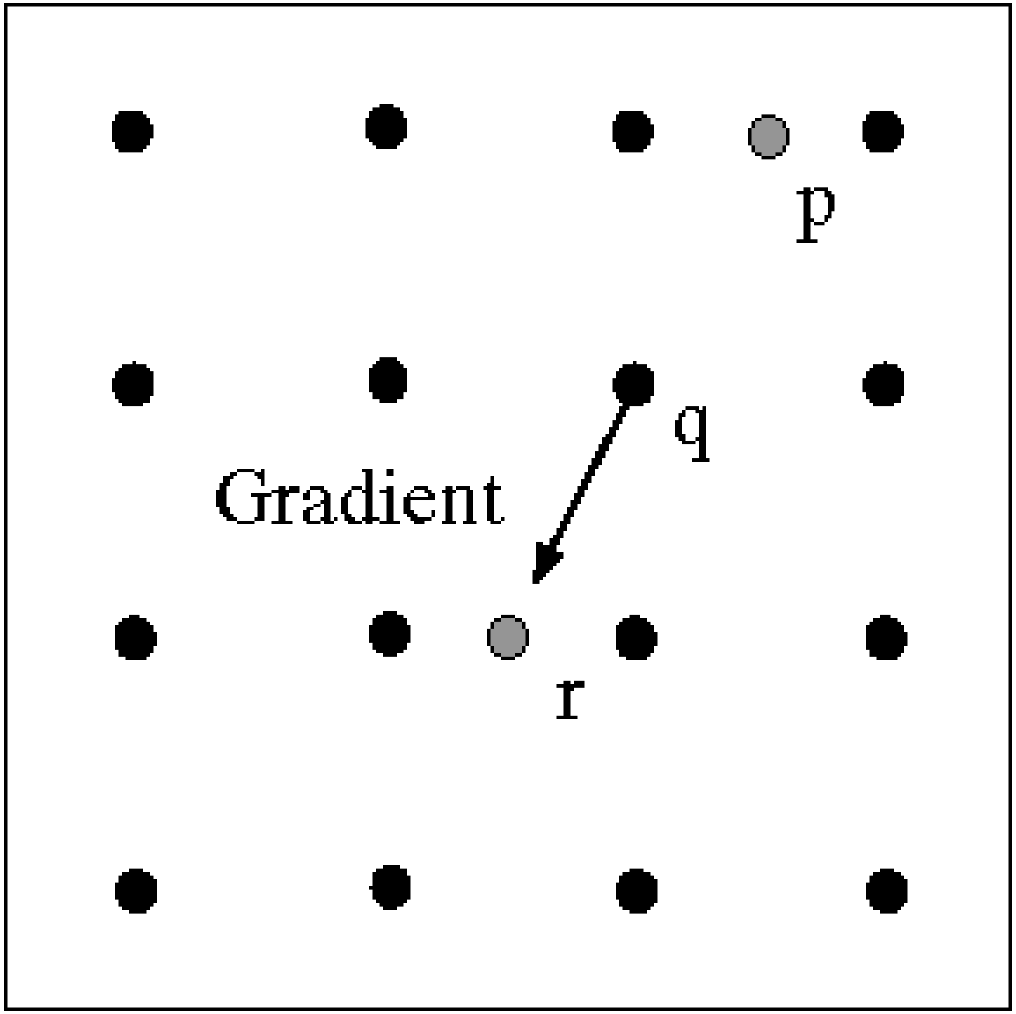
Get Orientation at Each Pixel

- Get orientation

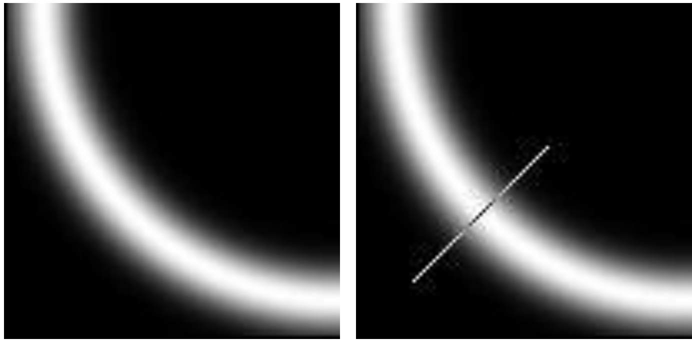


theta = atan2(gy, gx)

Non-maximum suppression for each orientation



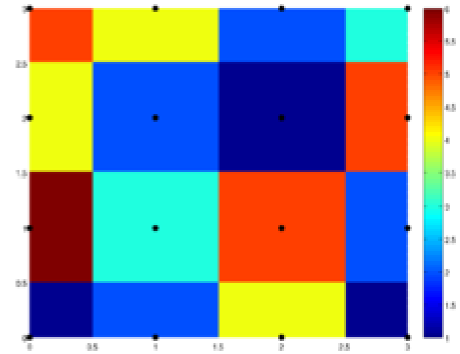
At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



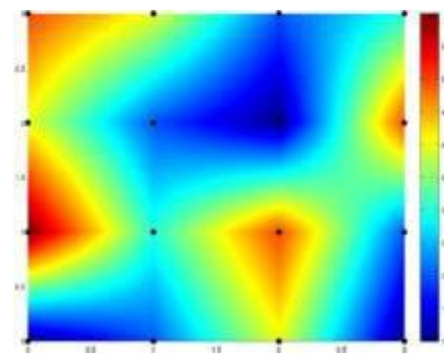
Source: D. Forsyth

Sidebar: Interpolation options

- ‘nearest’
  - Copy value from nearest known
  - Very fast but creates blocky edges

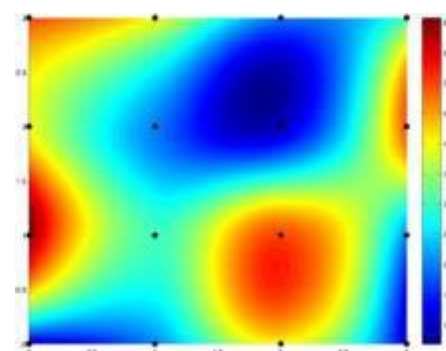


- ‘bilinear’
  - Weighted average from four nearest known pixels
  - Fast and reasonable results



- ‘bicubic’ (default)
  - Non-linear smoothing over larger area
  - Slower, visually appealing, may create negative

pixel values



Examples from http://en.wikipedia.org/wiki/Bicubic_interpolation

Before Non-max Suppression

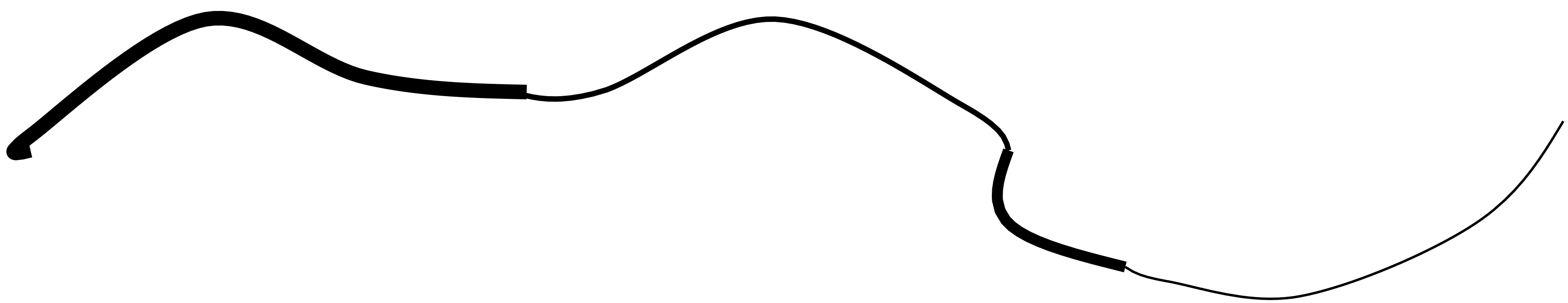
After non-max suppression

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
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Source: S. Seitz

Final Canny Edges

Canny edge detector

1. Filter image with x, y derivatives of Gaussian

Source: D. Lowe, L. Fei-Fei

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient

Source: D. Lowe, L. Fei-Fei

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
    - Thin multi-pixel wide “ridges” down to single pixel width

Source: D. Lowe, L. Fei-Fei

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. 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

Source: D. Lowe, L. Fei-Fei

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. 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

- MATLAB: `edge(image, ‘canny’)`

Source: D. Lowe, L. Fei-Fei

Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

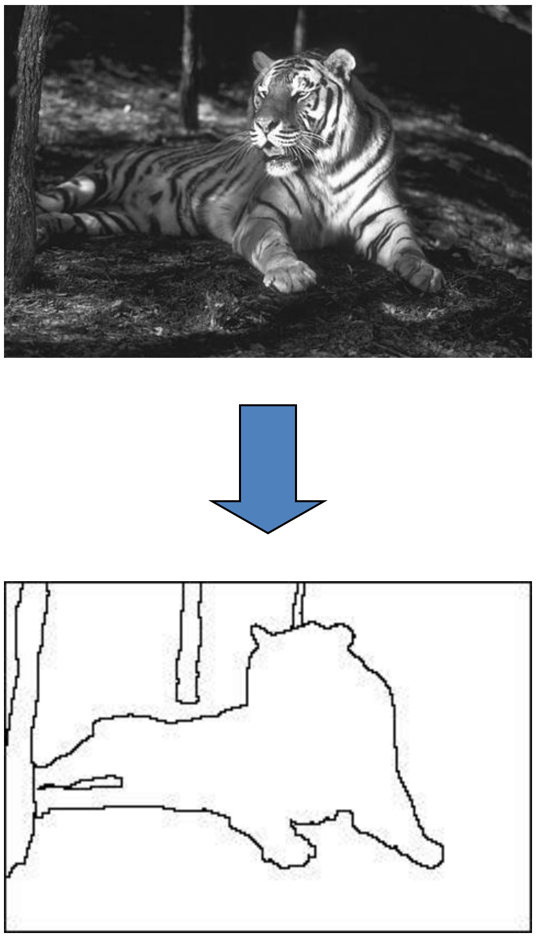
Canny with $\sigma = 2$

The choice of σ depends on desired behavior

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

Source: S. Seitz

Why edges?



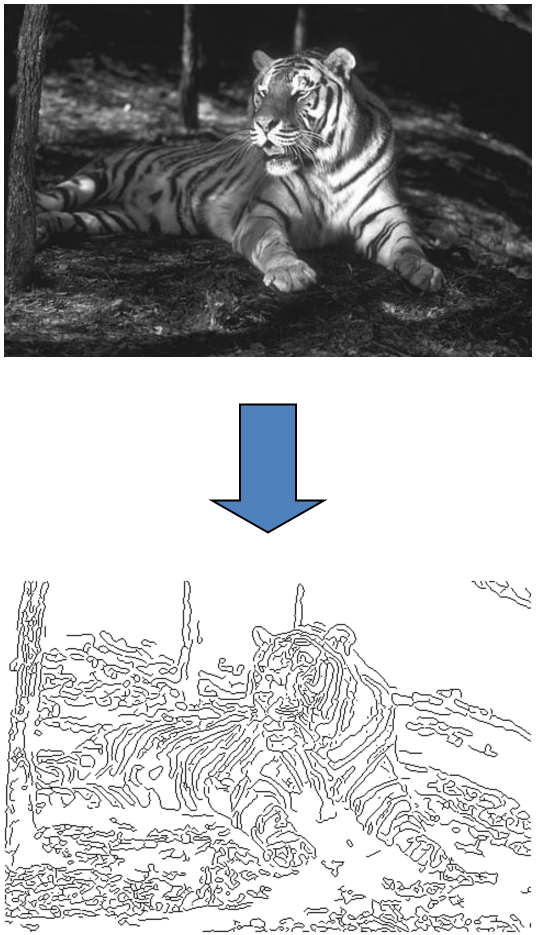
Reduce dimensionality of data

Preserve content information

Useful in applications such as:

- object detection
- structure from motion
- tracking

Why **not** edges?



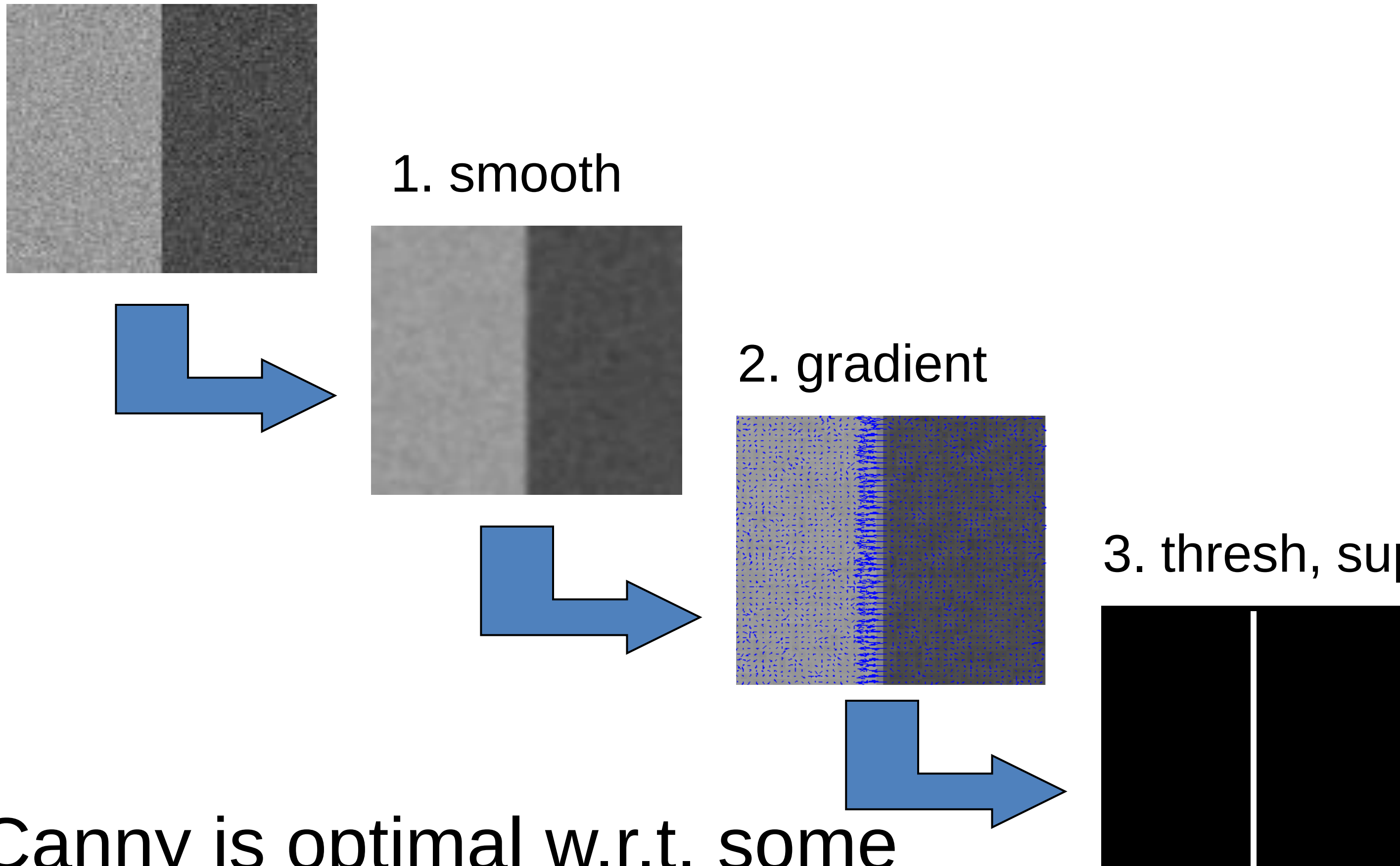
But, not that useful, **why**?

Difficulties:

1. Modeling assumptions
2. Parameters
3. Multiple sources of information (brightness, color, texture, …)
4. Real world conditions

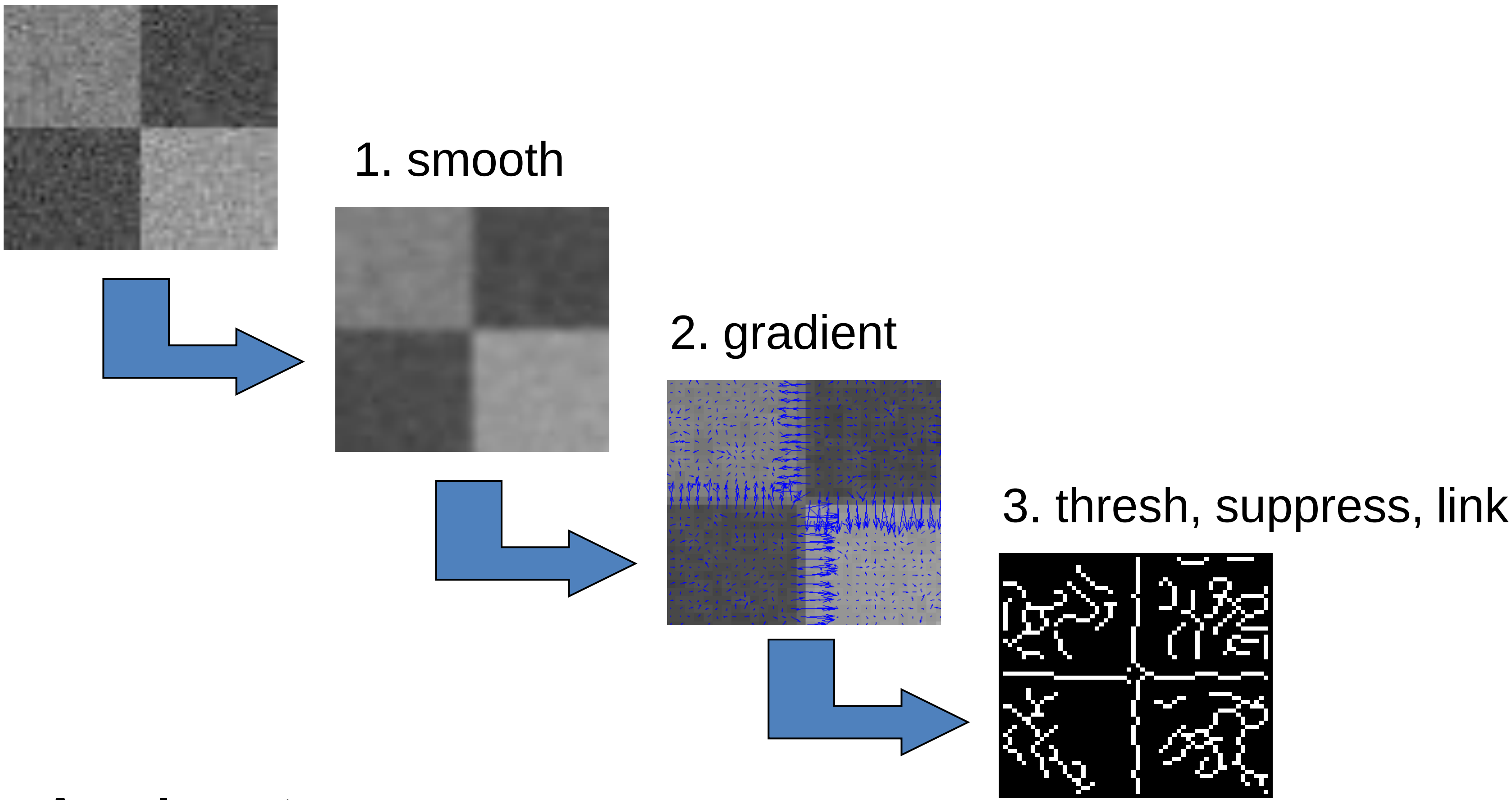
Is edge detection even well defined?

Canny edge detection



Canny is optimal w.r.t. some model.

Canny edge detection



And yet…

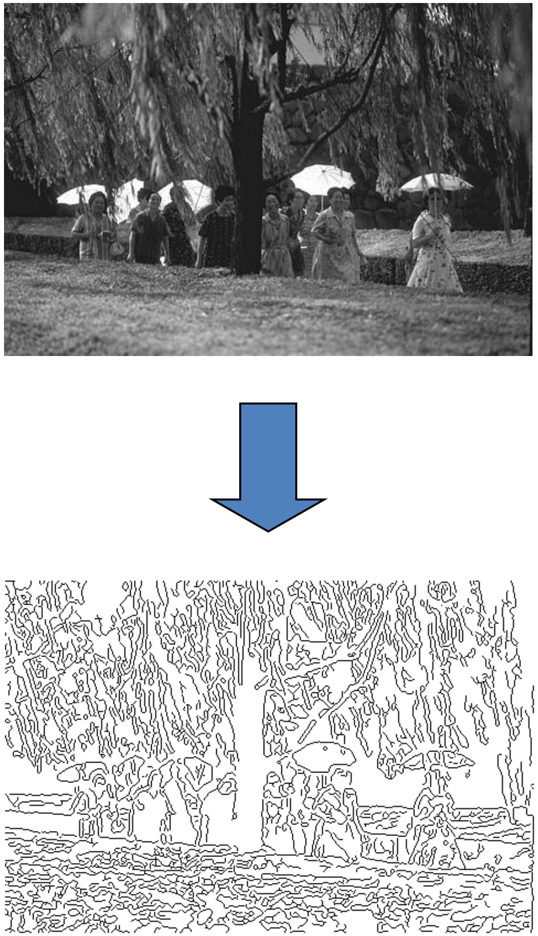
Canny difficulties

1. Modeling assumptions
   Step edges, junctions, etc.

2. Parameters
   Scales, threshold, etc.

3. Multiple sources of information
   Only handles brightness

4. Real world conditions
   Gaussian iid noise? Texture…

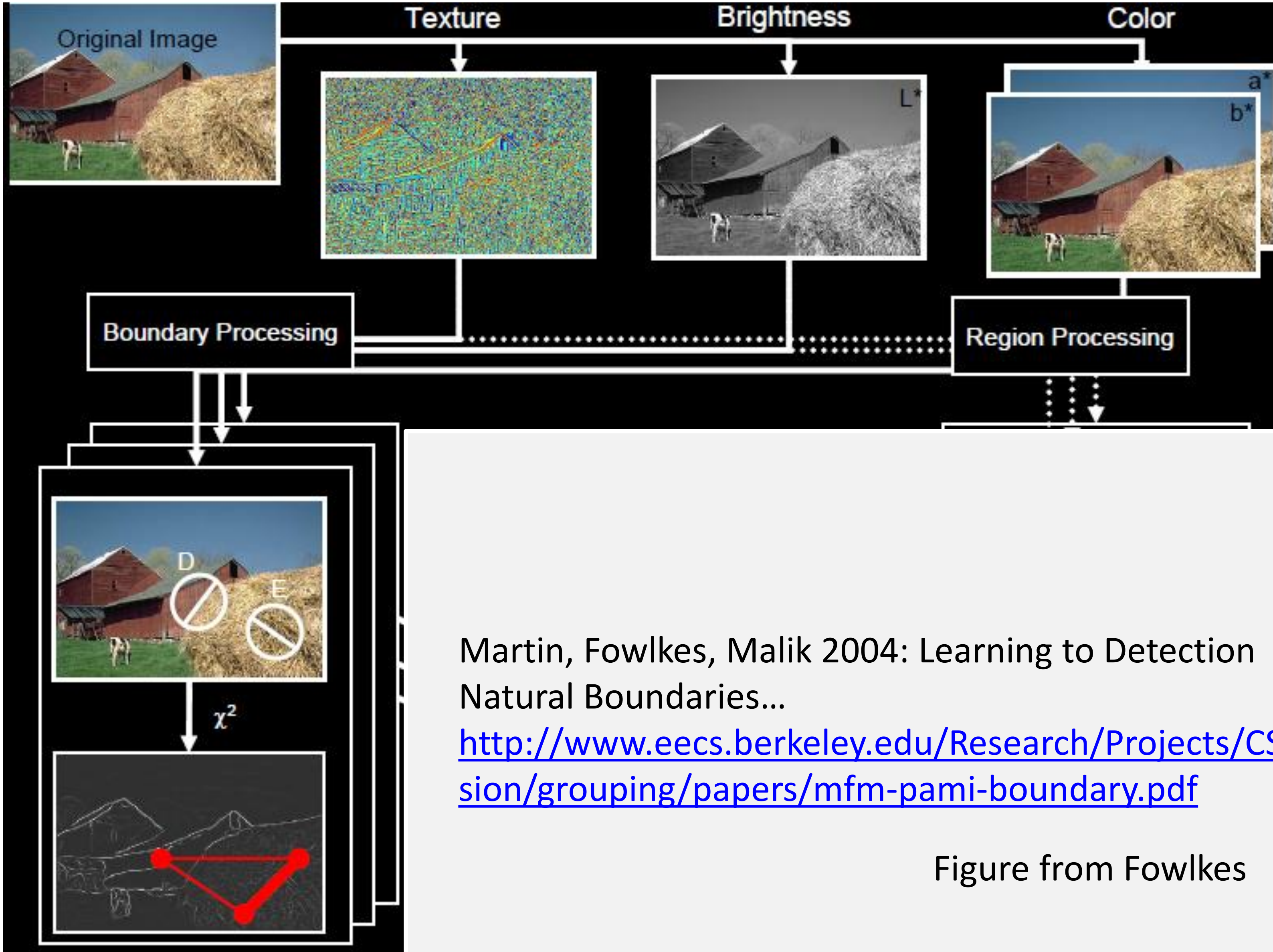


Learning to detect boundaries

image | human segmentation | gradient magnitude

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

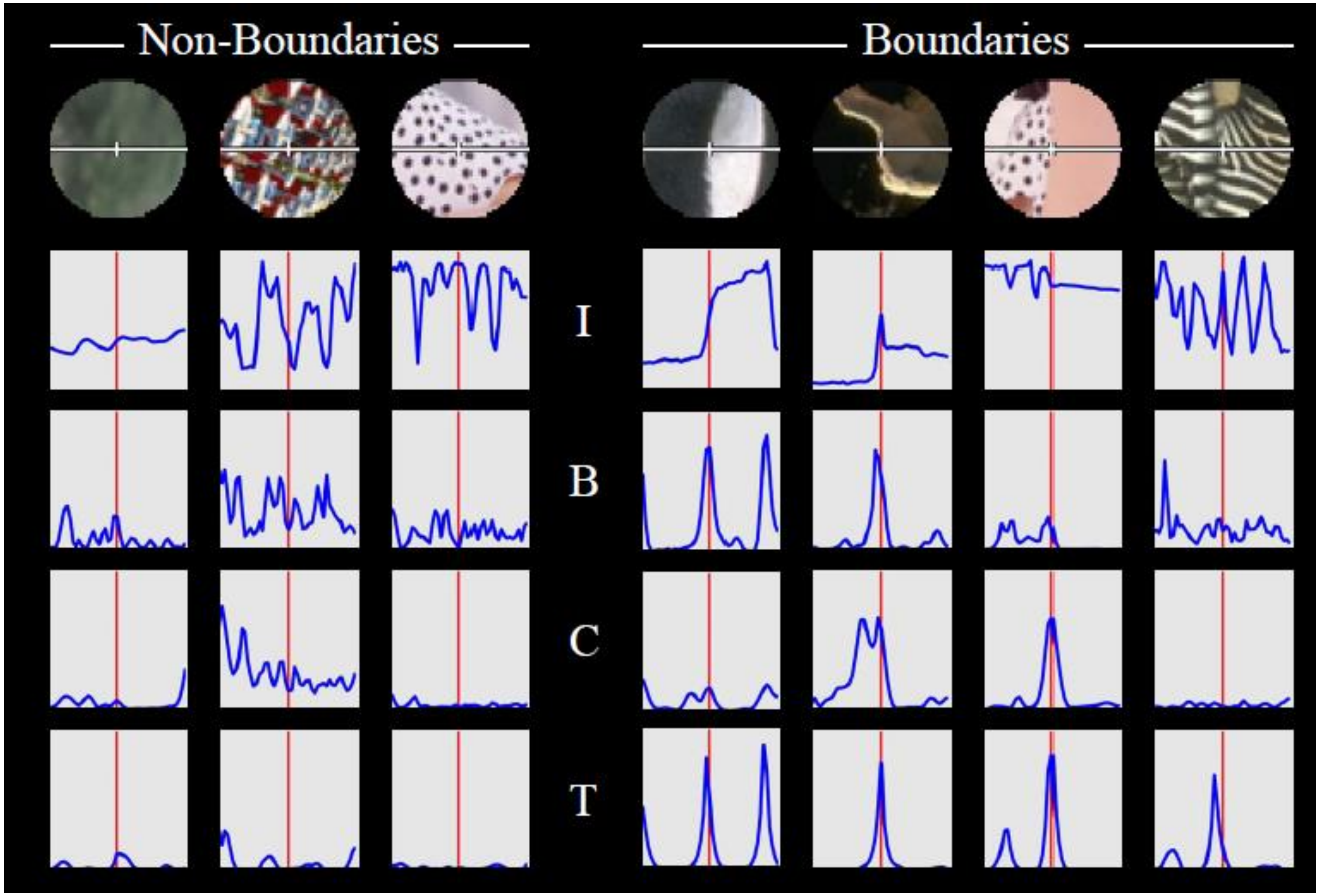
pB boundary detector



Martin, Fowlkes, Malik 2004: Learning to Detection Natural Boundaries…
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/mfm-pami-boundary.pdf

Figure from Fowlkes

pB Boundary Detector



- Estimate Posterior probability of boundary passing through centre point based on local patch based features
- Using a Supervised Learning based framework


Brightness
Color
Texture
Combined
Human
Image
BG
CG
TG
BG+CG+TG
Human
1
0.8
0.6
0.4
0.2
0

Features

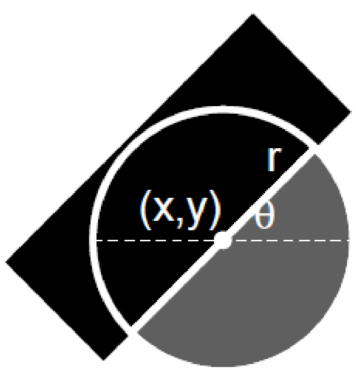
Brightness oriented energy,

$$\mathrm{OE}_{\theta,\sigma} = (I * f^e_{\theta,\sigma})^2 + (I * f^o_{\theta,\sigma})^2$$

Gaussian second derivative

Gradients computed from two disc halves:

Brightness gradient
Color gradient
Texture gradient



Texture features

(a) Filterbank (13 filters)

(b) Universal textons (64)

(c) Image

(d) Texton map (color-coded)

Martin, Fowlkes, Malik, 2004: Berkeley (Pb) edge detector

Localization

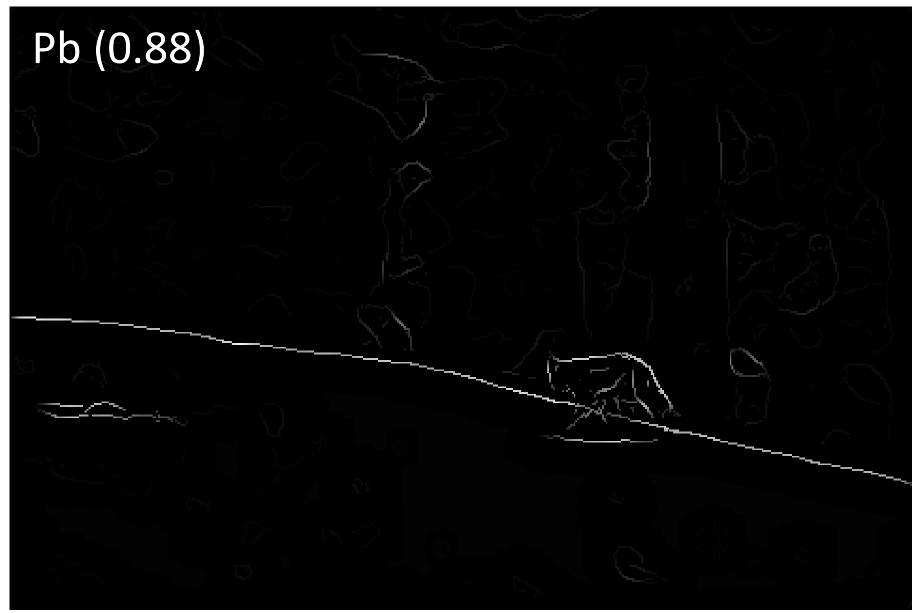
- edges (due to large filters) are poorly localized; double peaks
- Improve Localization by using derived feature
- Divide by distance to nearest maximum

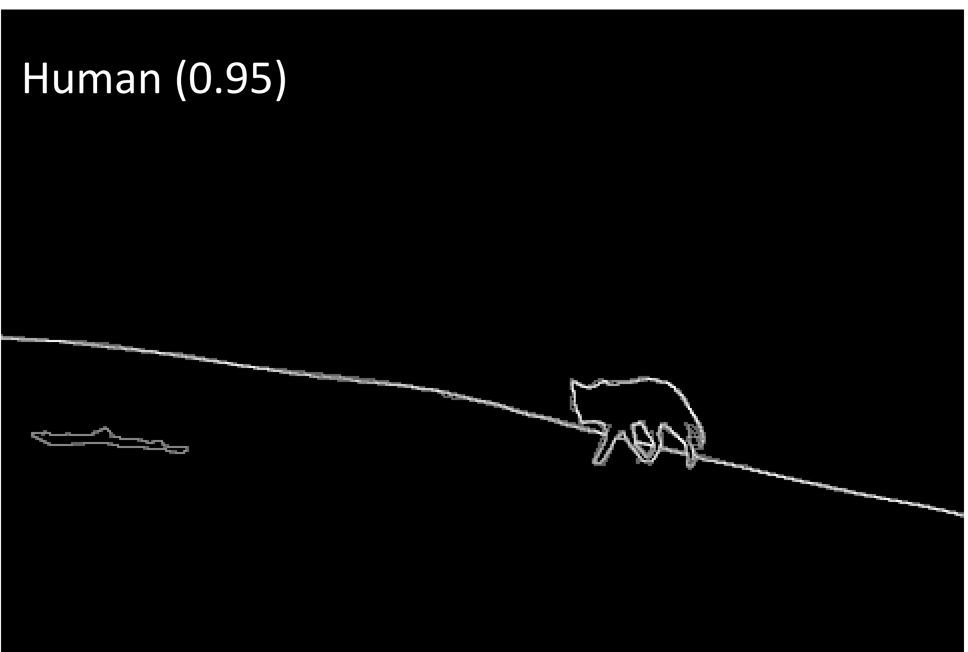
$$\hat{f}(x) = \tilde{f}(x) \cdot \left(\frac{-f''(x)}{|f'(x)| + \epsilon} \right)$$

where f(x) is feature and the estimated distance to the nearest maximum of f(x) is

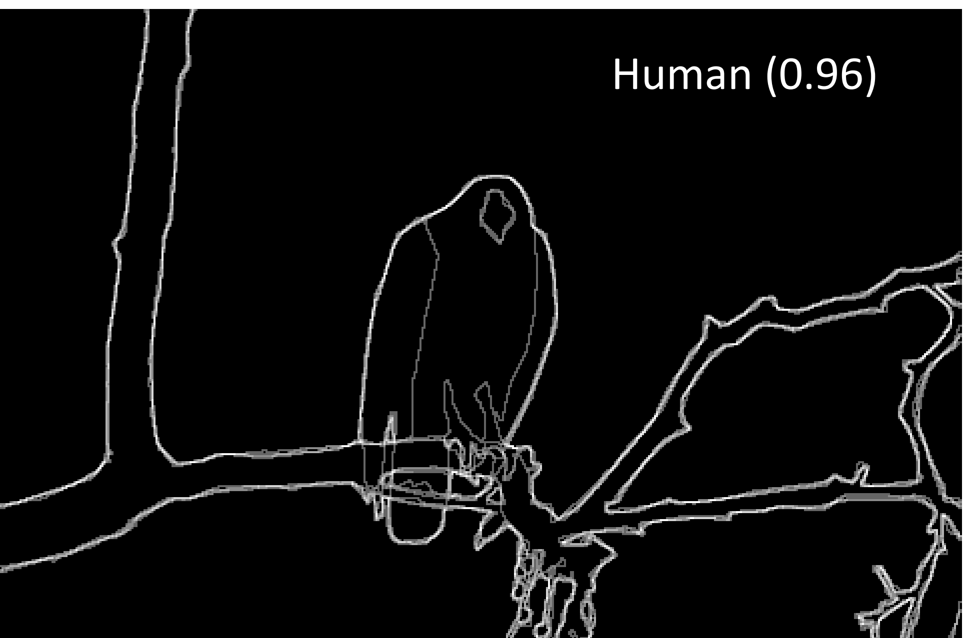
$d(x) = -|f'(x)|/f''(x)$

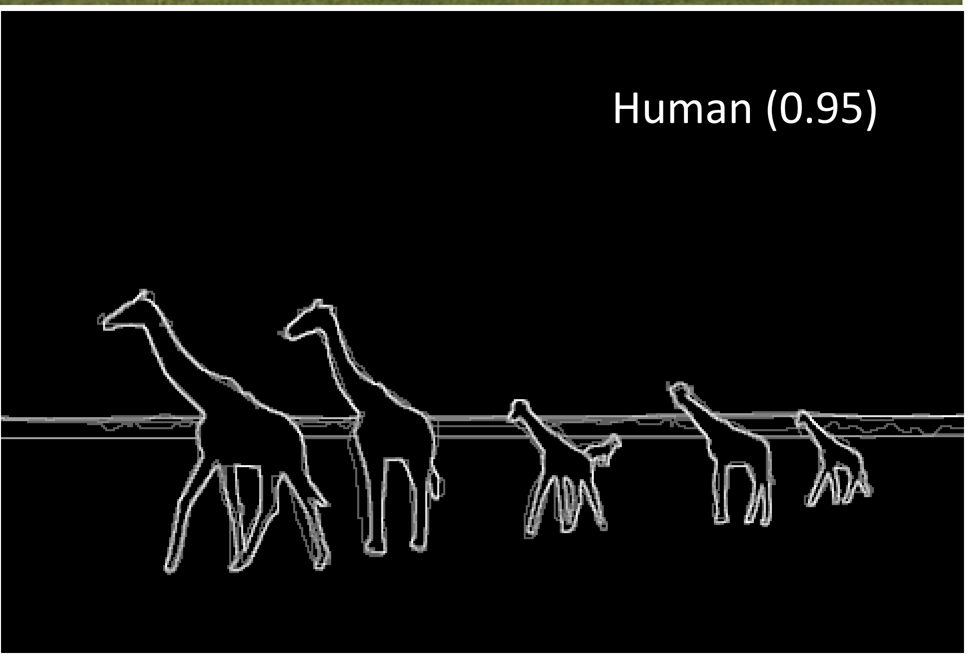
Results



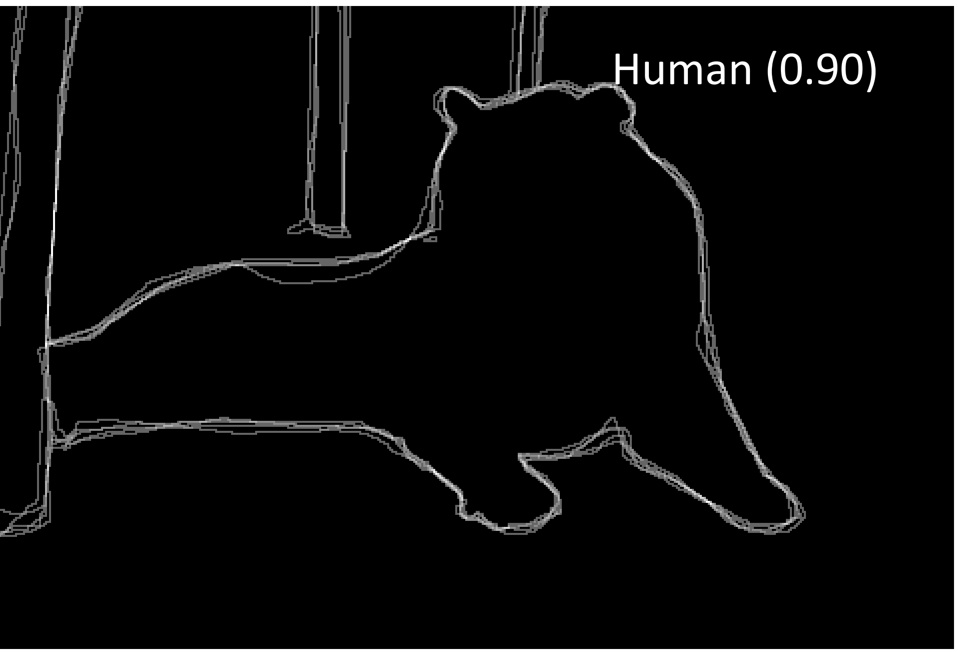


Results




Human (0.95)


Pb (0.63)



For more:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/108082-color.html

State of edge detection

- Local edge detection is mostly solved
  - Intensity gradient, color, texture
- Often used in combination with object detectors or region classifiers
- Deep learning approach is more common nowadays

Finding straight lines

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   - find connected components for edge pixels with directions in {d-1, d, d+1}

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   - find connected components for edge pixels with directions in {d-1, d, d+1}
4. Compute straightness and theta of edgelets using eig of x,y 2nd moment matrix of their points

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   - find connected components for edge pixels with directions in {d-1, d, d+1}
4. Compute straightness and theta of edgelets using eig of x,y 2nd moment matrix of their points

$$\mathbf{M} = \begin{bmatrix} \sum (x-\mu_x)^2 & \sum (x-\mu_x)(y-\mu_y) \\ \sum (x-\mu_x)(y-\mu_y) & \sum (y-\mu_y)^2 \end{bmatrix} \qquad [v, \lambda] = \mathrm{eig}(\mathbf{M})$$

Larger eigenvector

$$\theta = \mathrm{atan}\,2(v(2,2), v(1,2))$$

$$conf = \lambda_2 / \lambda_1$$

Finding line segments using connected components

1. Compute canny edges
   – Compute: gx, gy (DoG in x,y directions)
   – Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   – find connected components for edge pixels with directions in {d-1, d, d+1}
4. Compute straightness and theta of edgelets using eig of x,y 2nd moment matrix of their points

$$\mathbf{M} = \begin{bmatrix} \sum (x-\mu_x)^2 & \sum (x-\mu_x)(y-\mu_y) \\ \sum (x-\mu_x)(y-\mu_y) & \sum (y-\mu_y)^2 \end{bmatrix} \qquad [v, \lambda] = \mathrm{eig}(\mathbf{M})$$

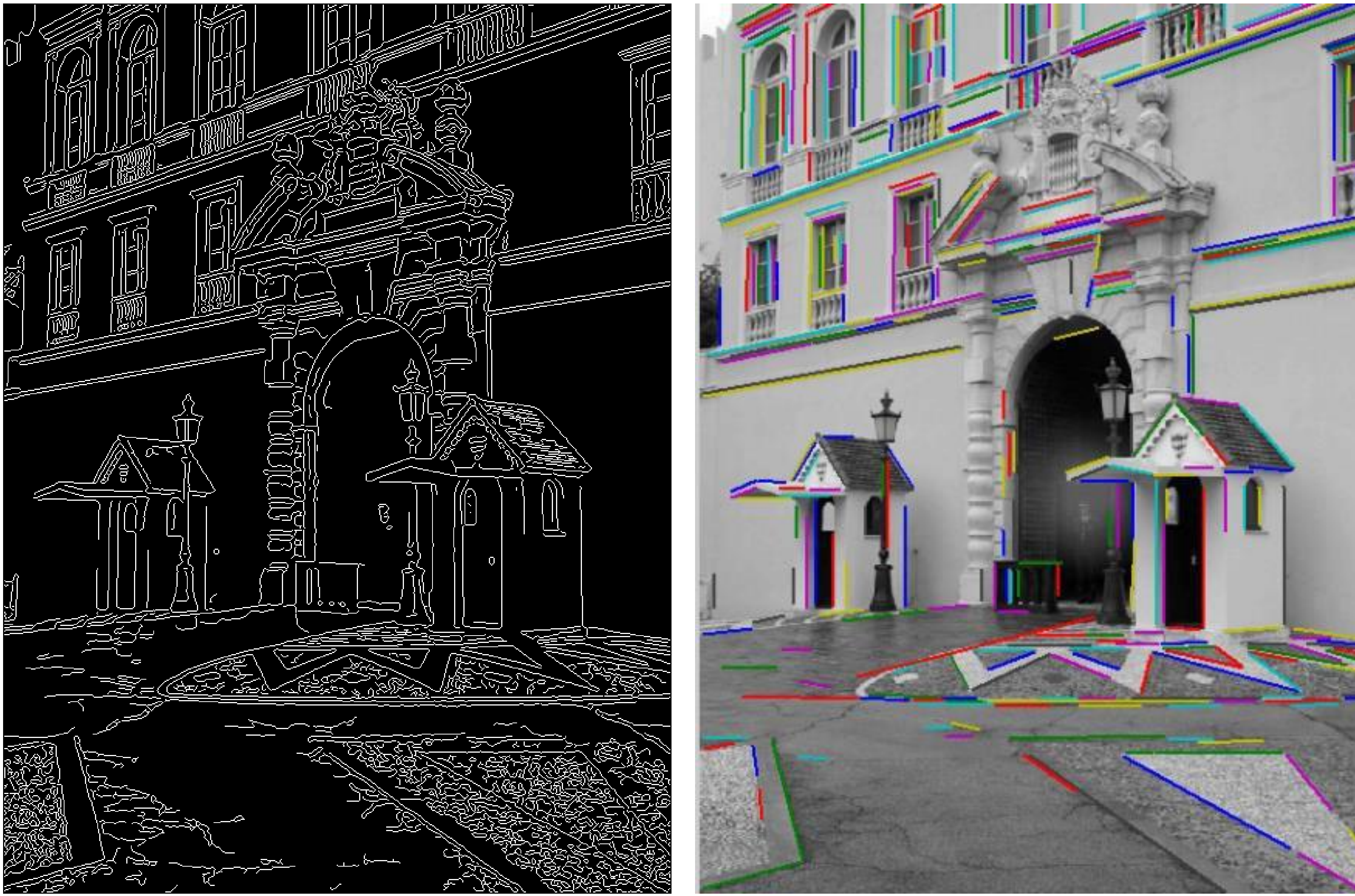
Larger eigenvector

$$\theta = \mathrm{atan}\,2(v(2,2), v(1,2))$$

$$conf = \lambda_2 / \lambda_1$$

5. Threshold on straightness, store segment

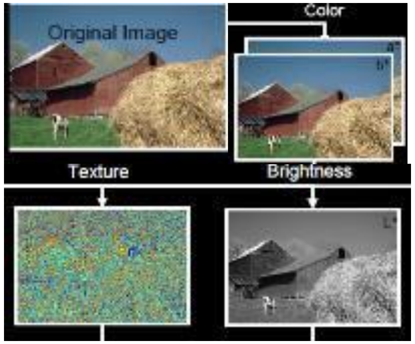
Canny lines → … →straight edges



Things to remember

- Canny edge detector =
  smooth → derivative → thin → threshold → link

- Pb: learns weighting of gradient, color, texture differences



- Straight line detector =
  canny + gradient orientations → orientation binning → linking → check for straightness

Acknowledgements

- Thanks to the following researchers for making their teaching/research material online
  - Forsyth
  - Steve Seitz
  - Noah Snavely
  - J.B. Huang
  - Derek Hoiem
  - D. Lowe
  - A. Bobick
  - S. Lazebnik
  - K. Grauman
  - R. Zaleski

Thank you: Question?