Finding Edges and Straight Lines

Computer Vision

CS 543 / ECE 549

University of Illinois

Derek Hoiem

Many slides from Lana Lazebnik, Steve Seitz, David Forsyth, David Lowe, Fei-Fei Li

Last class

- How to use filters for
  - Matching
  - Denoising
  - Compression

- Image representation with pyramids

- Texture and filter banks

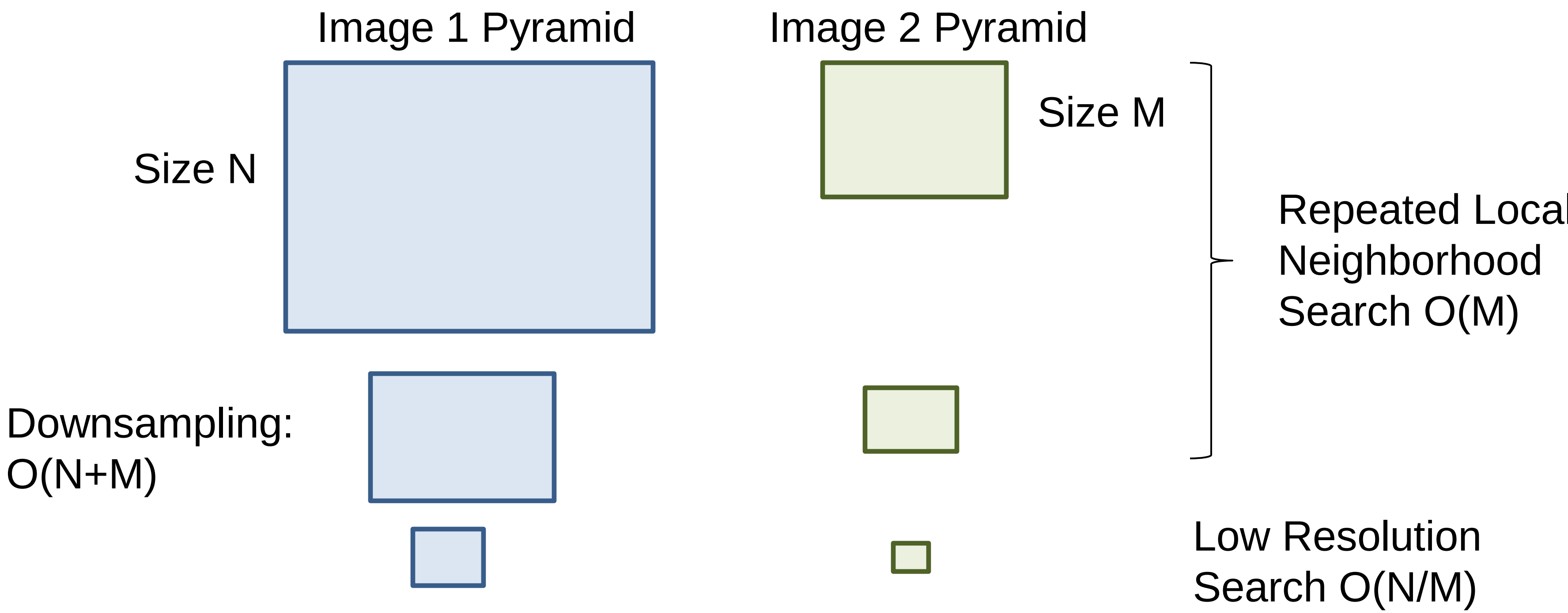
A couple remaining questions from earlier

- Does the curvature of the earth change the horizon location?

Illustrations from
Amin Sadeghi

A couple remaining questions from earlier

- Computational complexity of coarse-to-fine search?

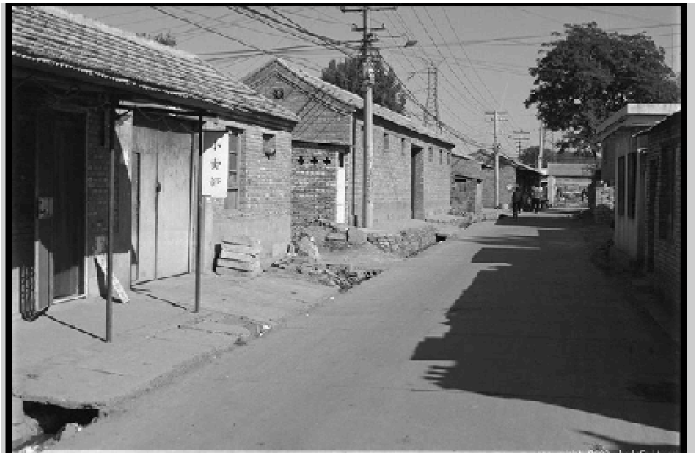


Overall complexity: O(N+M)
Original high-resolution full search: O(NM) or O(N logN)

A couple remaining questions from earlier

- Why not use an ideal filter?

Answer: has infinite spatial extent, clipping results in ringing

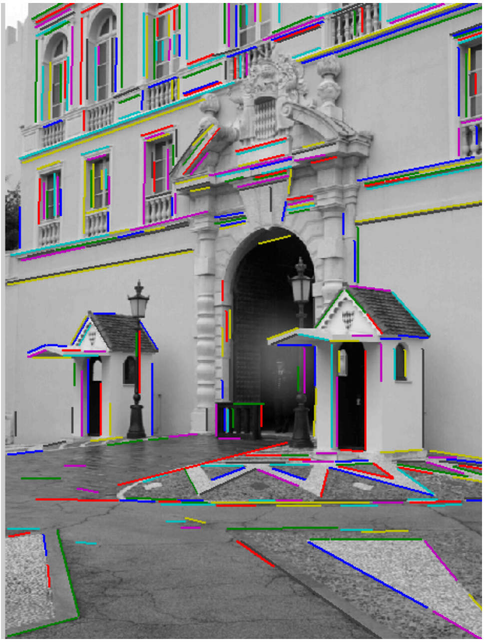


Attempt to apply ideal filter in frequency domain

Today's class

- Detecting edges

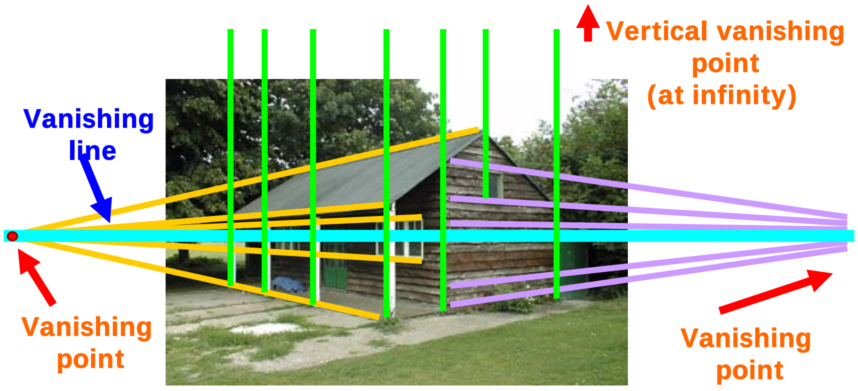
- Finding straight lines



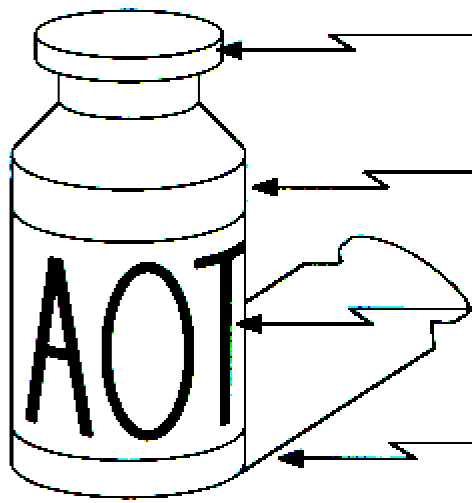
Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint



Origin of Edges



surface normal discontinuity

depth discontinuity

surface color discontinuity

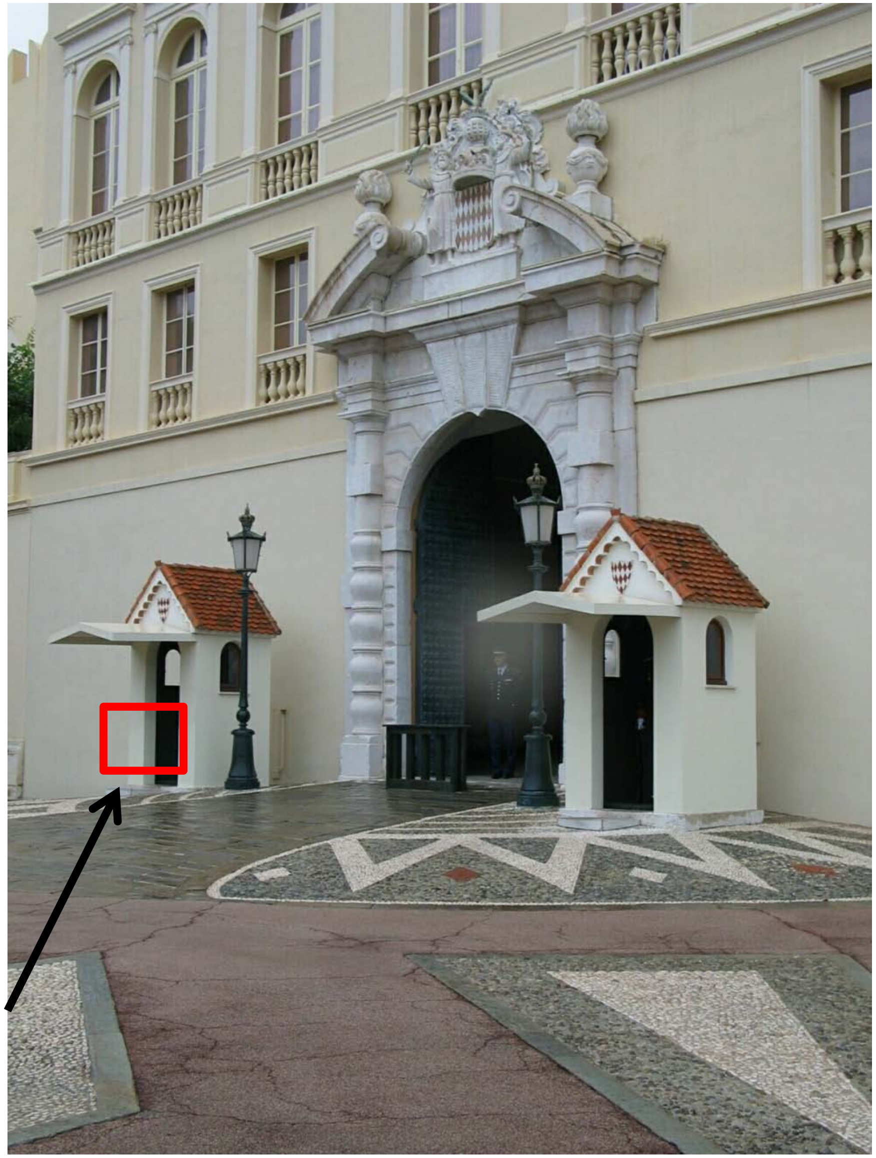
illumination discontinuity

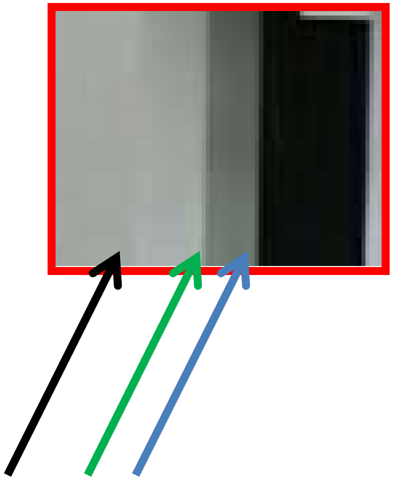
- Edges are caused by a variety of factors

Source: Steve Seitz

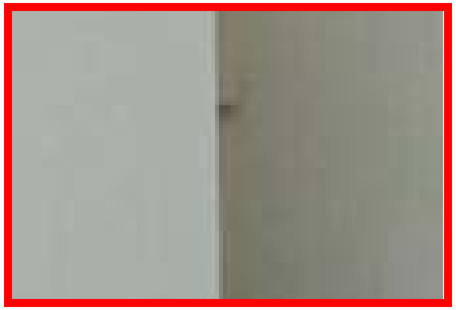
Closeup of edges

Closeup of edges





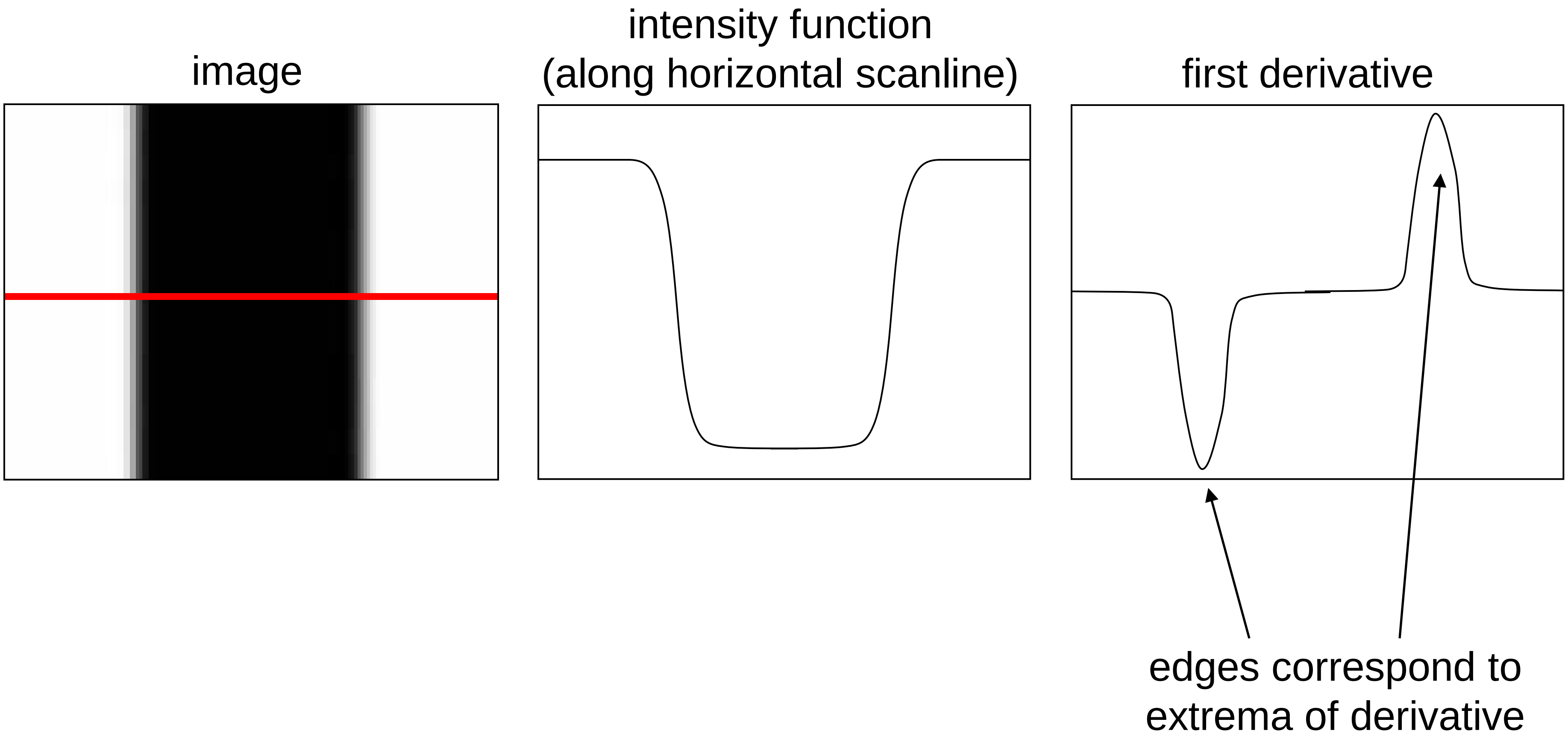
Closeup of edges



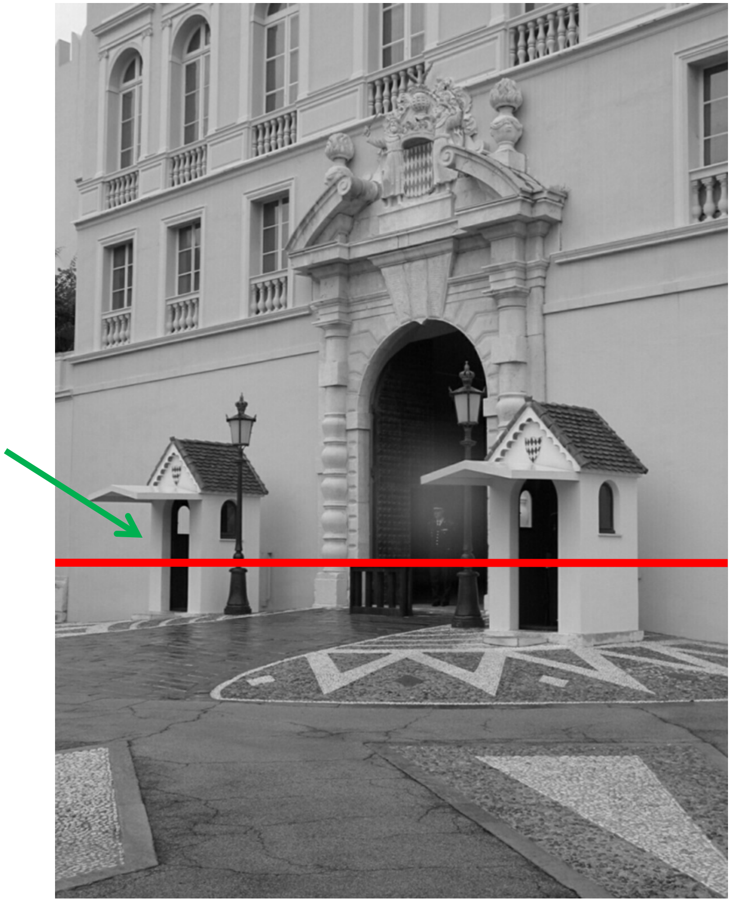
Closeup of edges

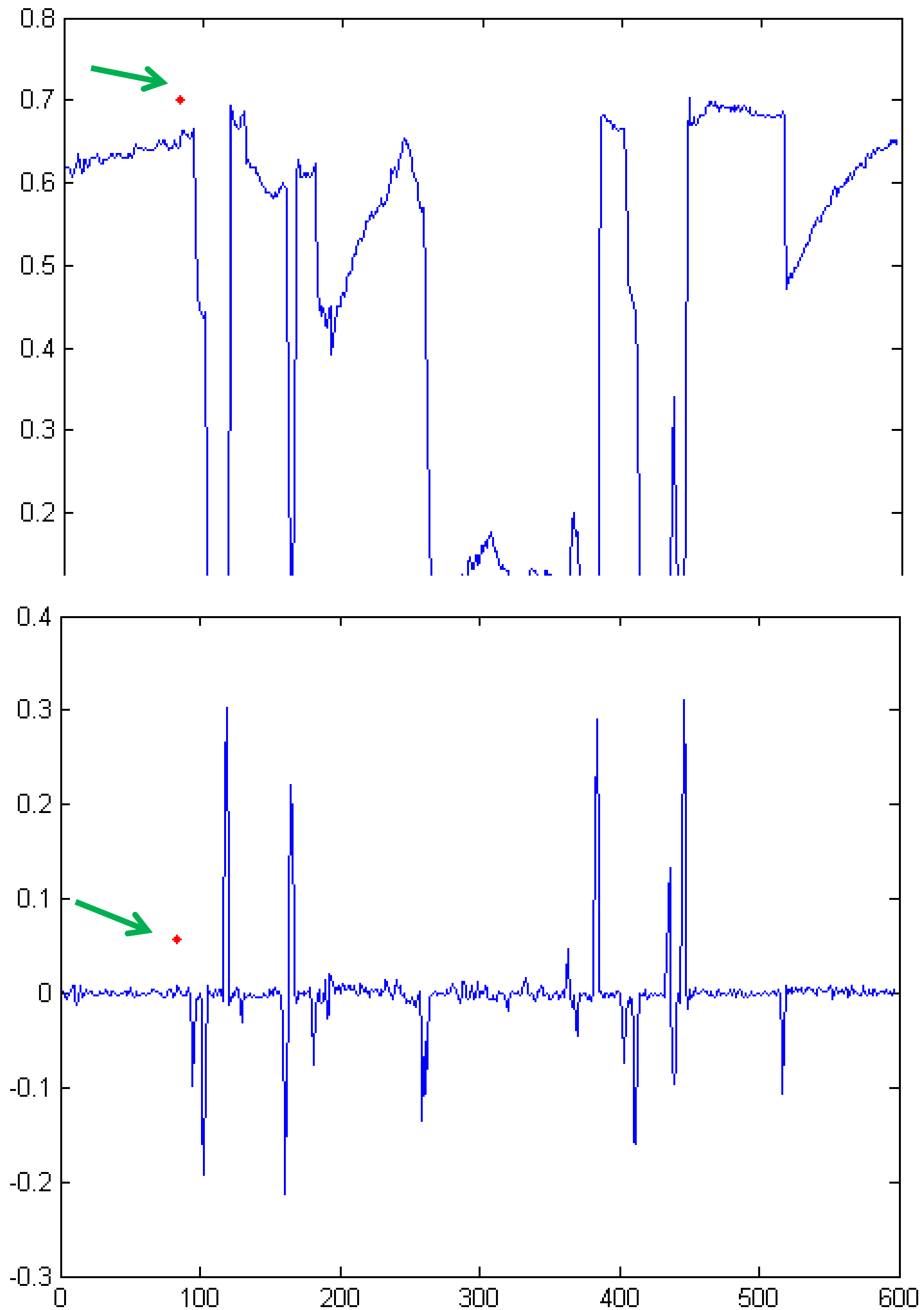
Characterizing edges

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

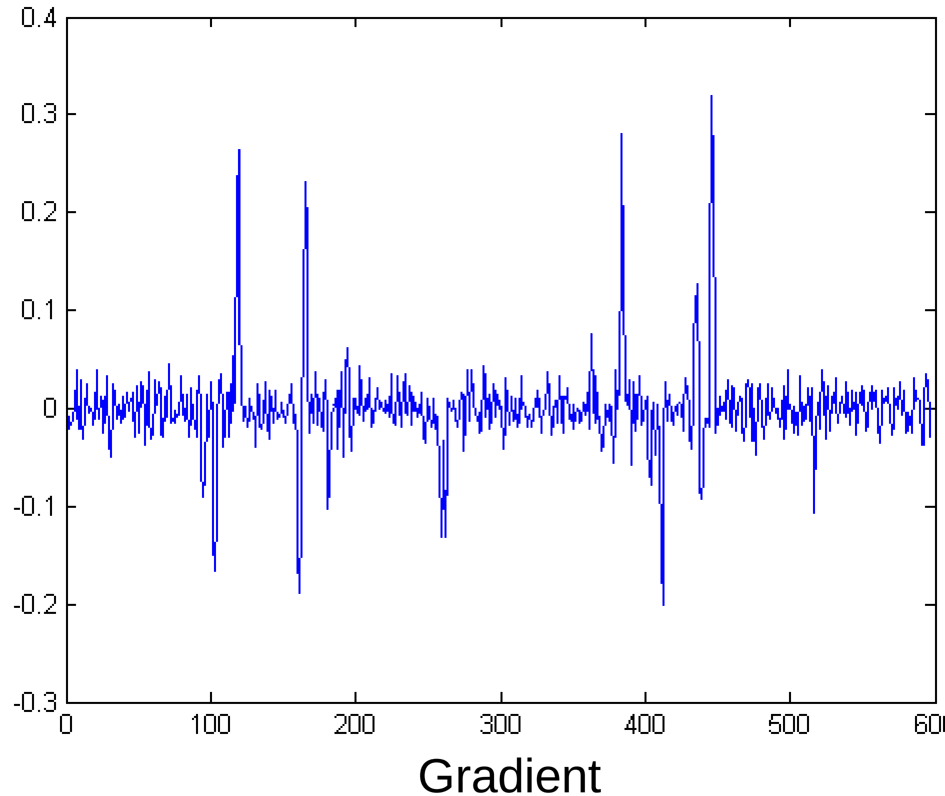


Intensity profile



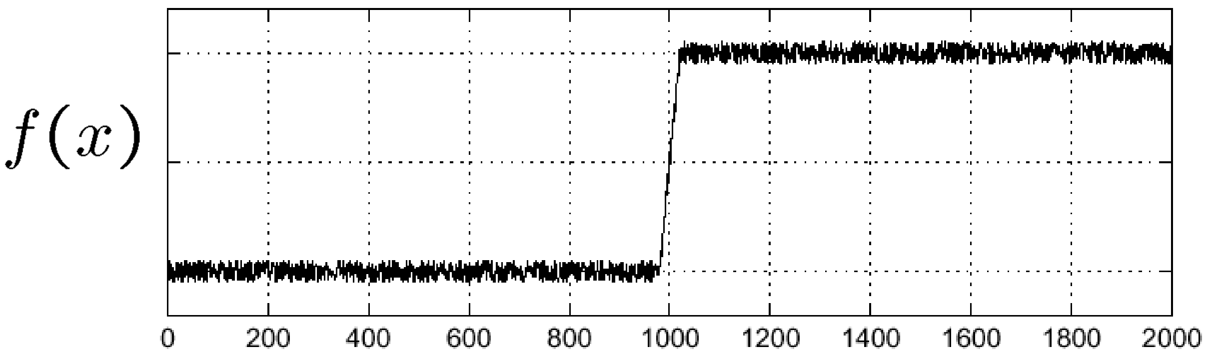


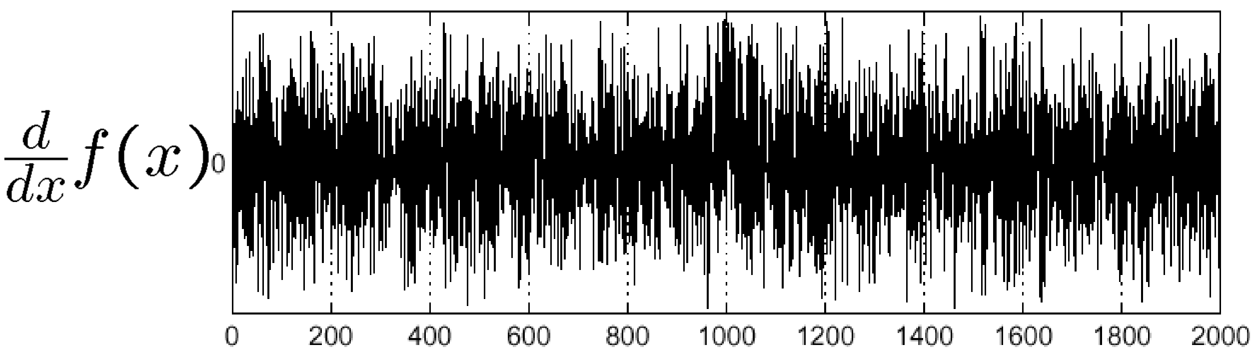
With a little Gaussian noise



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?

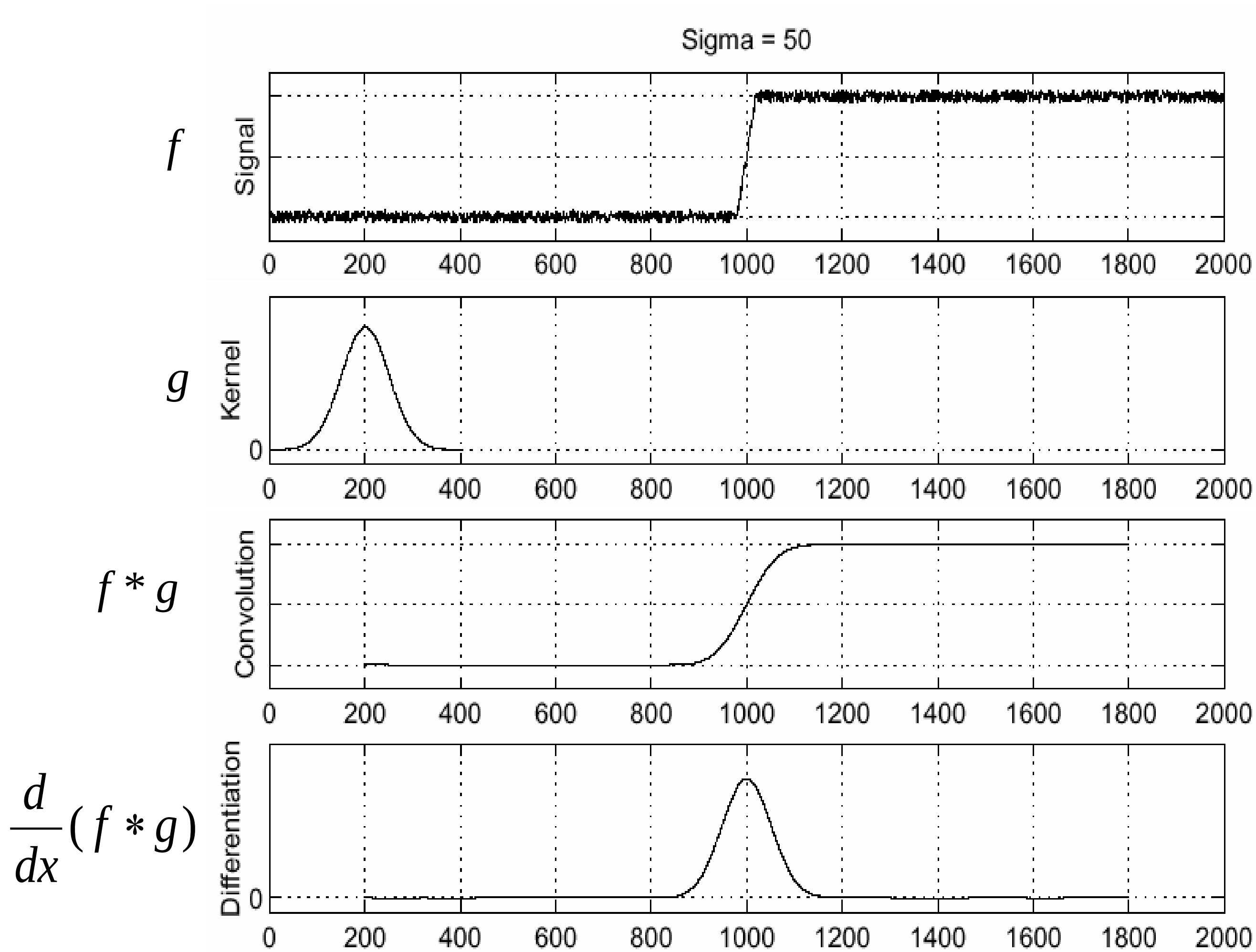
Source: S. Seitz

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Source: D. Forsyth

Solution: smooth first



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

Source: S. Seitz

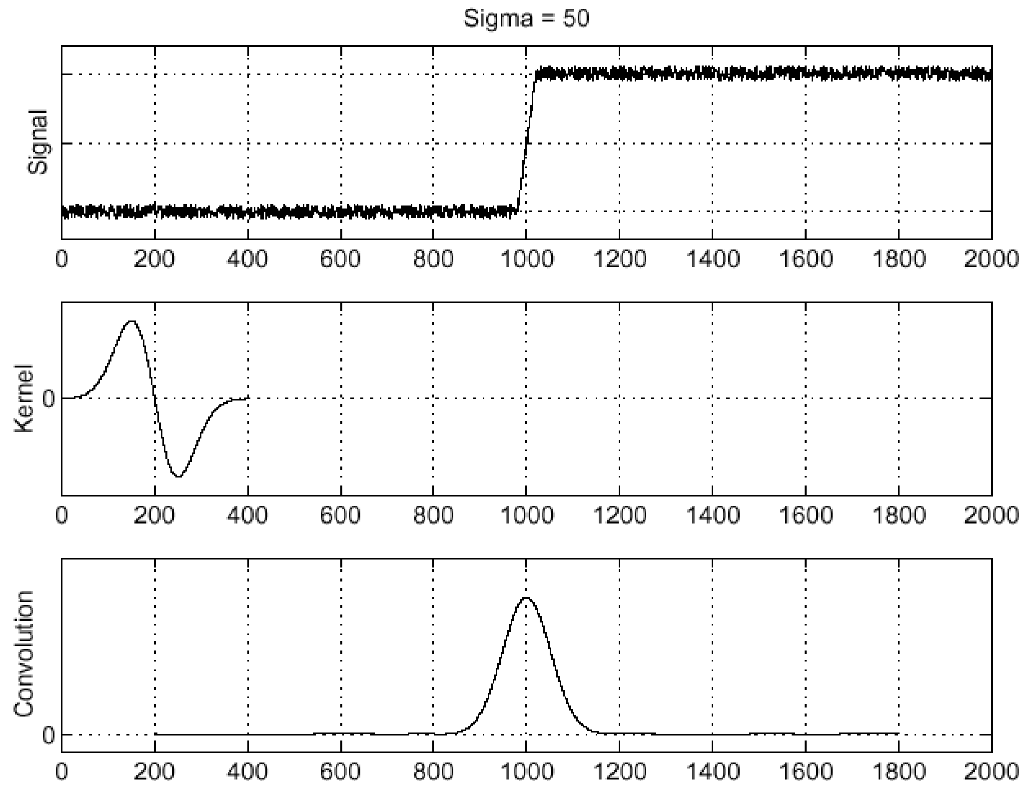
Derivative theorem of convolution

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

f

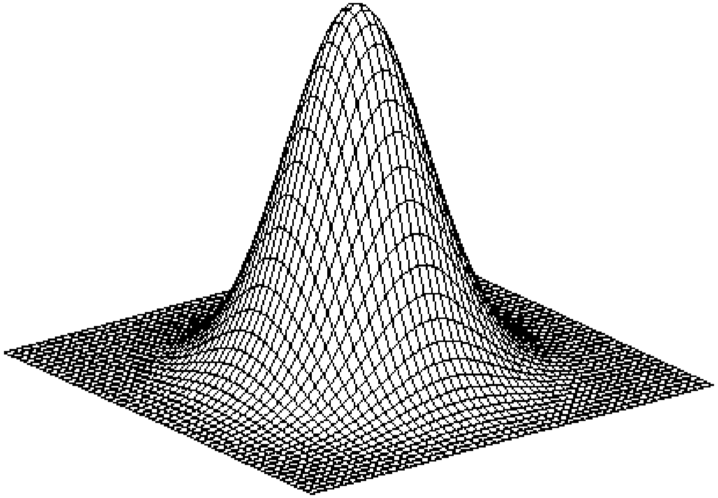
$\frac{d}{dx} g$

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

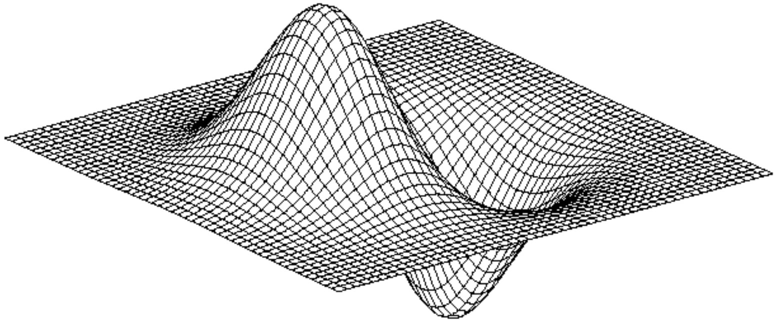


Source: S. Seitz

Derivative of Gaussian filter

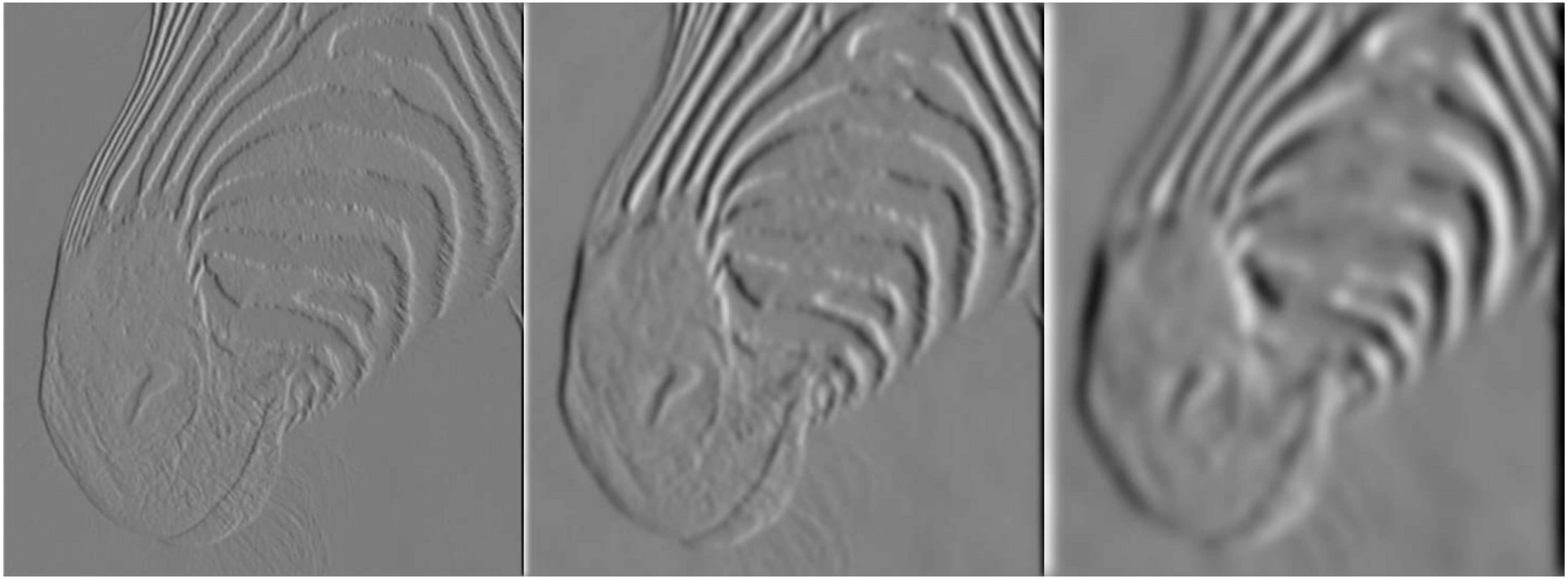


* [1 -1] =



- Is this filter separable?

Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth

Designing an edge detector

- Criteria for a good edge detector:
  - **Good detection:** the optimal detector should find all real edges, ignoring noise or other artifacts
  - **Good localization**
    - the edges detected must be as close as possible to the true edges
    - the detector must return one point only for each true edge point
- Cues of edge detection
  - Differences in color, intensity, or texture across the boundary
  - Continuity and closure
  - High-level knowledge

Source: L. Fei-Fei

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

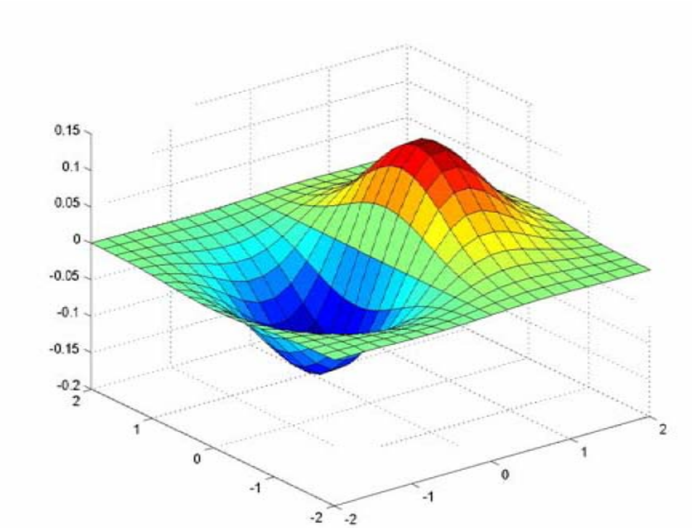
J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

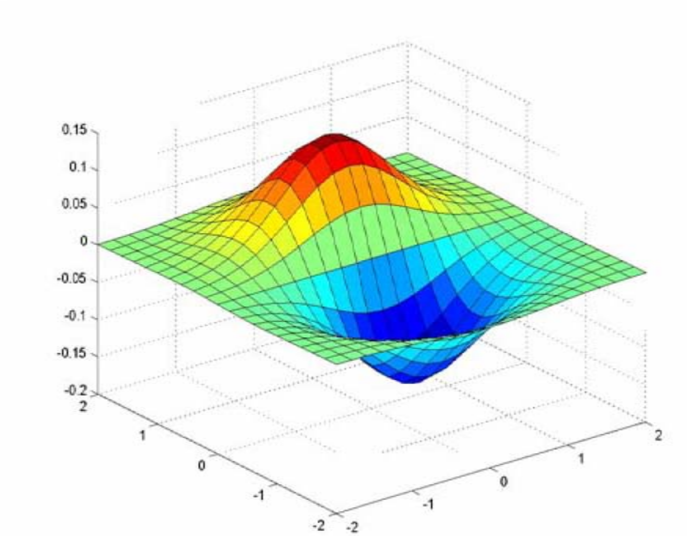
Example

original image (Lena)

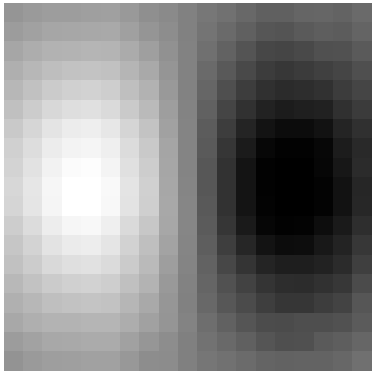
Derivative of Gaussian filter

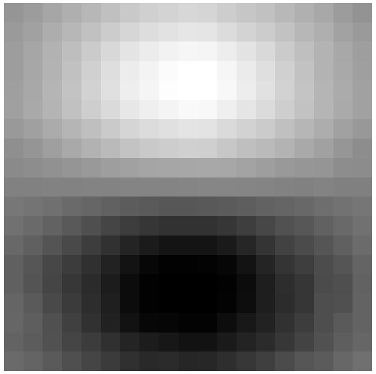


x-direction

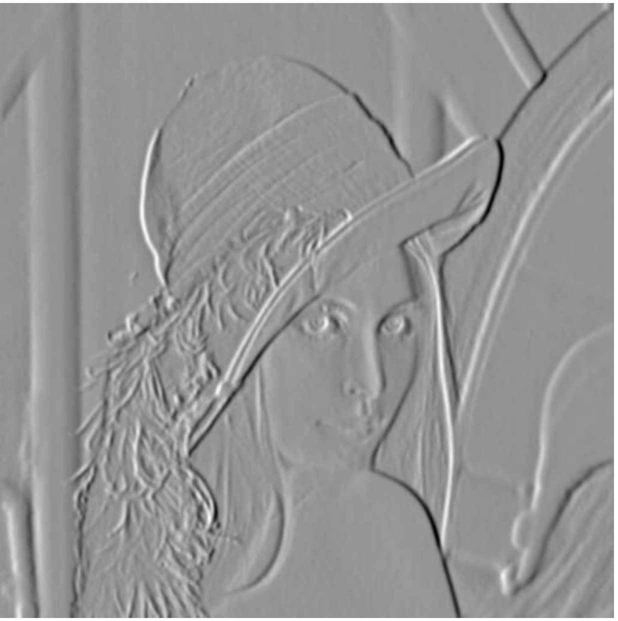


y-direction





Compute Gradients (DoG)



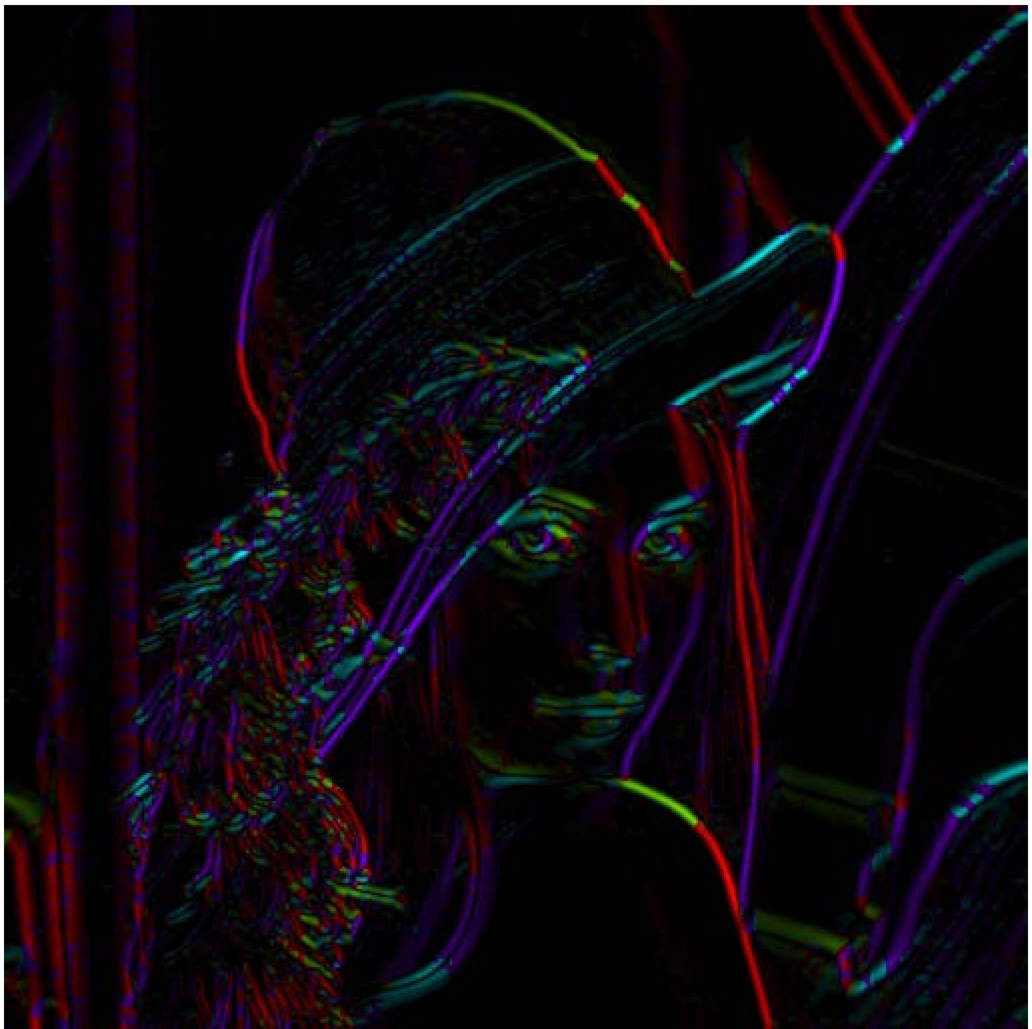
X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude

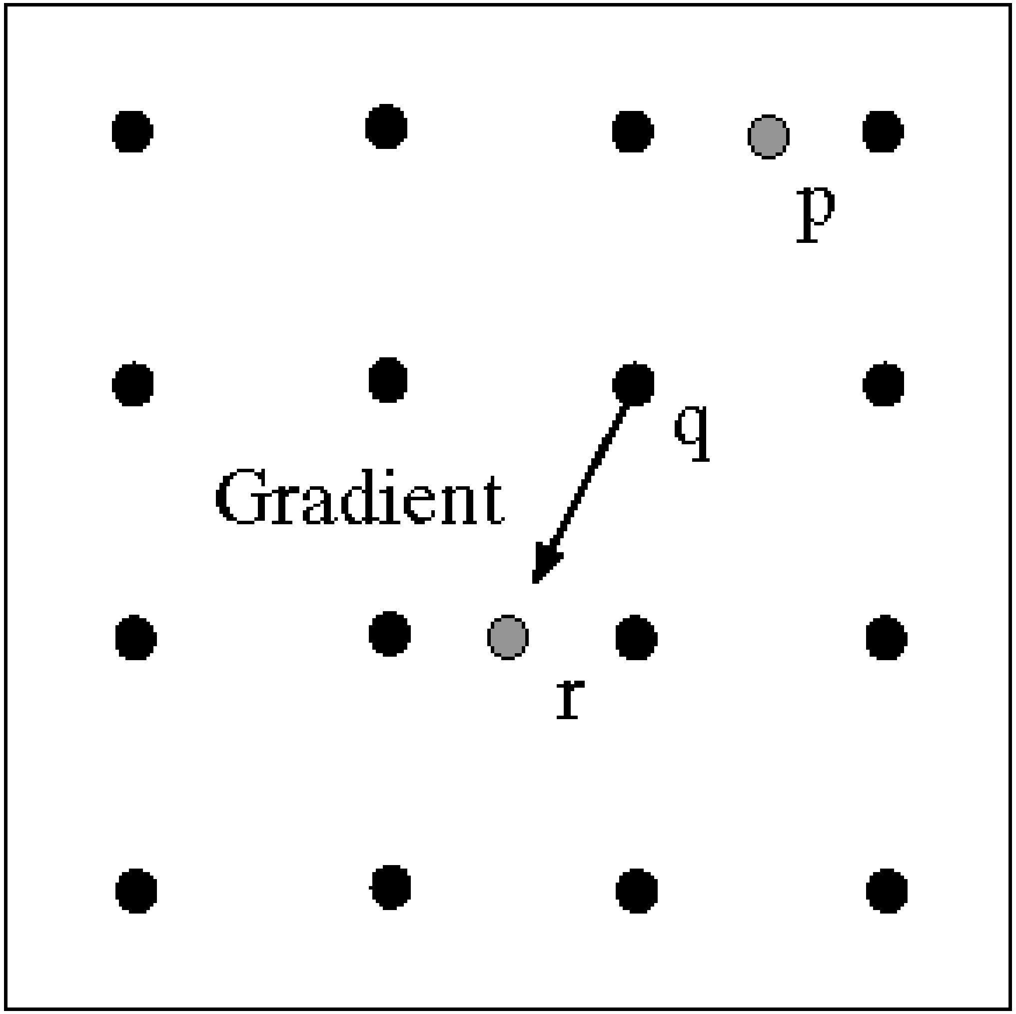
Get Orientation at Each Pixel

- Threshold at minimum level
- Get orientation

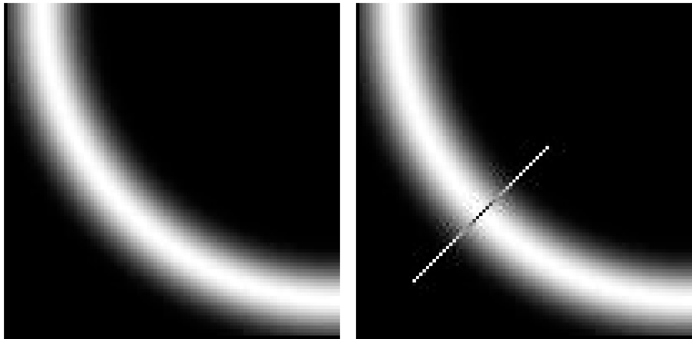


theta = atan2(gy, gx)

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

Before Non-max Suppression

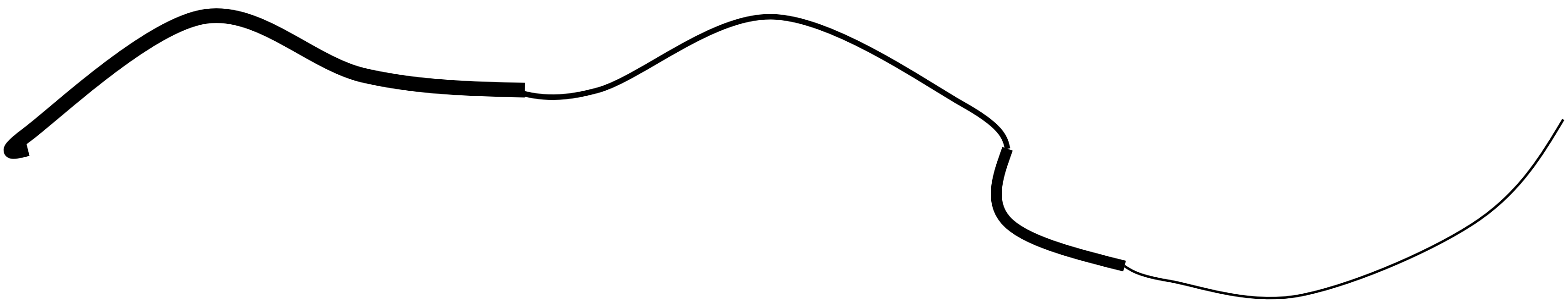
After non-max suppression

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
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

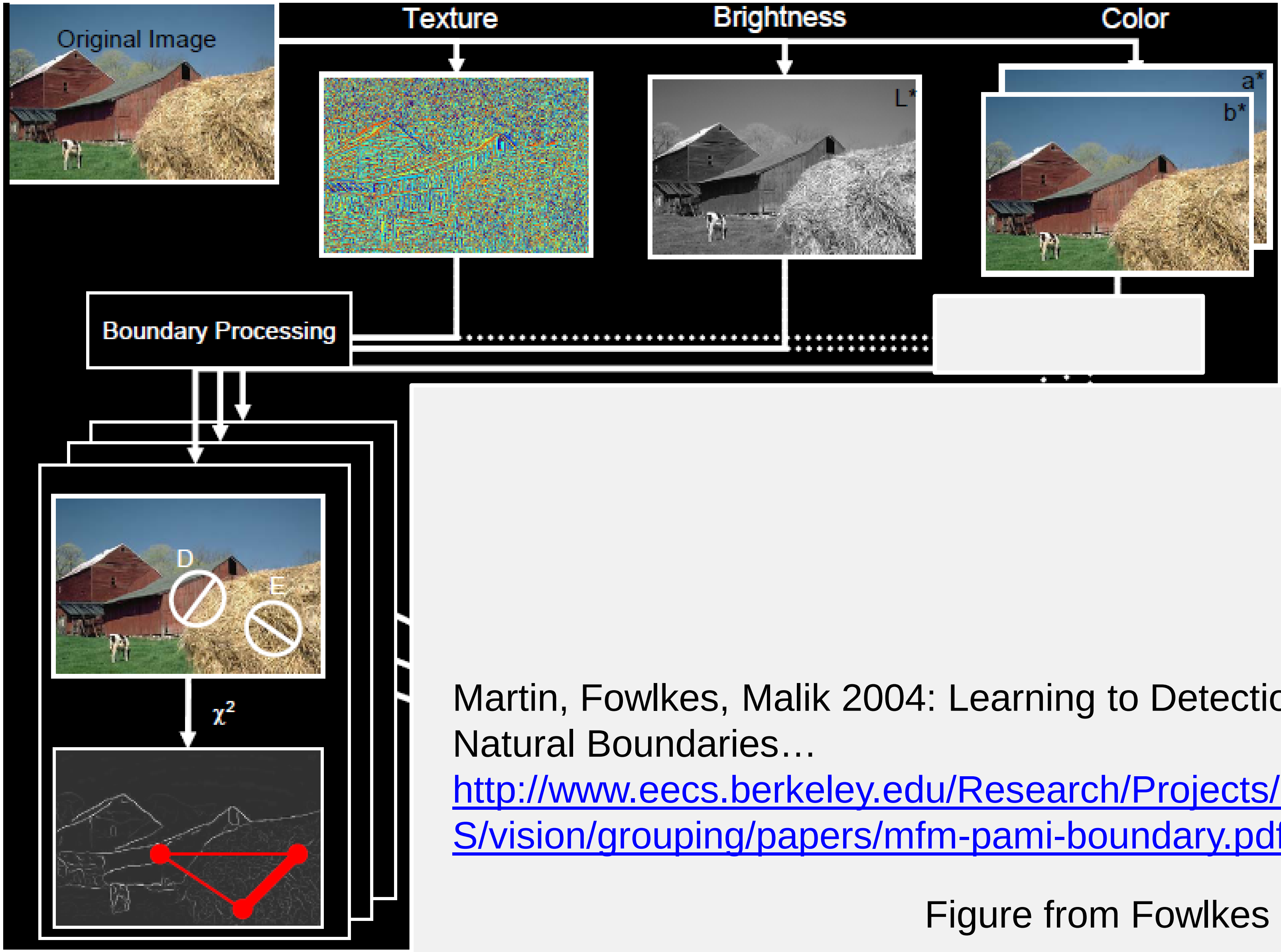
Source: S. Seitz

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

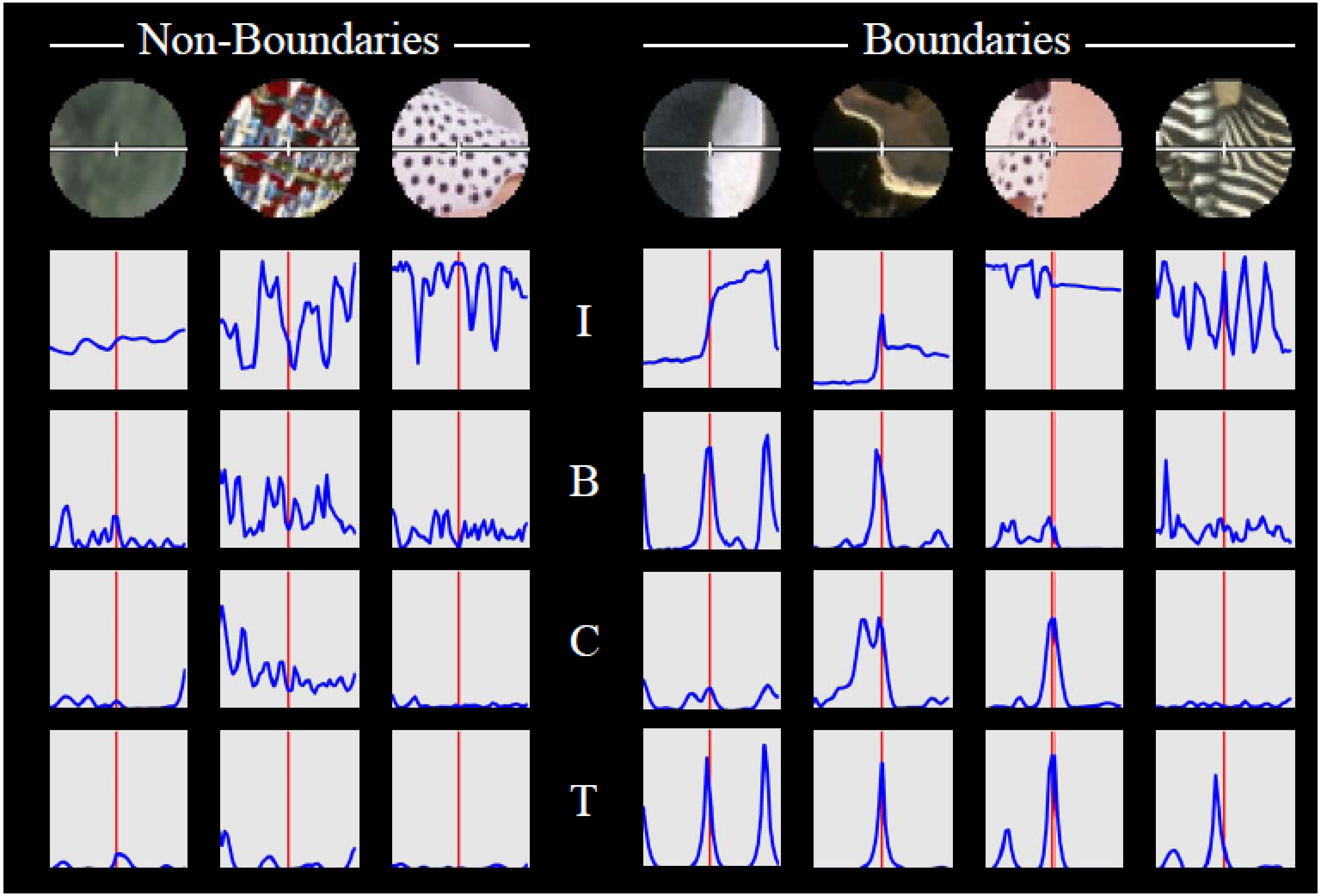


Figure from Fowlkes

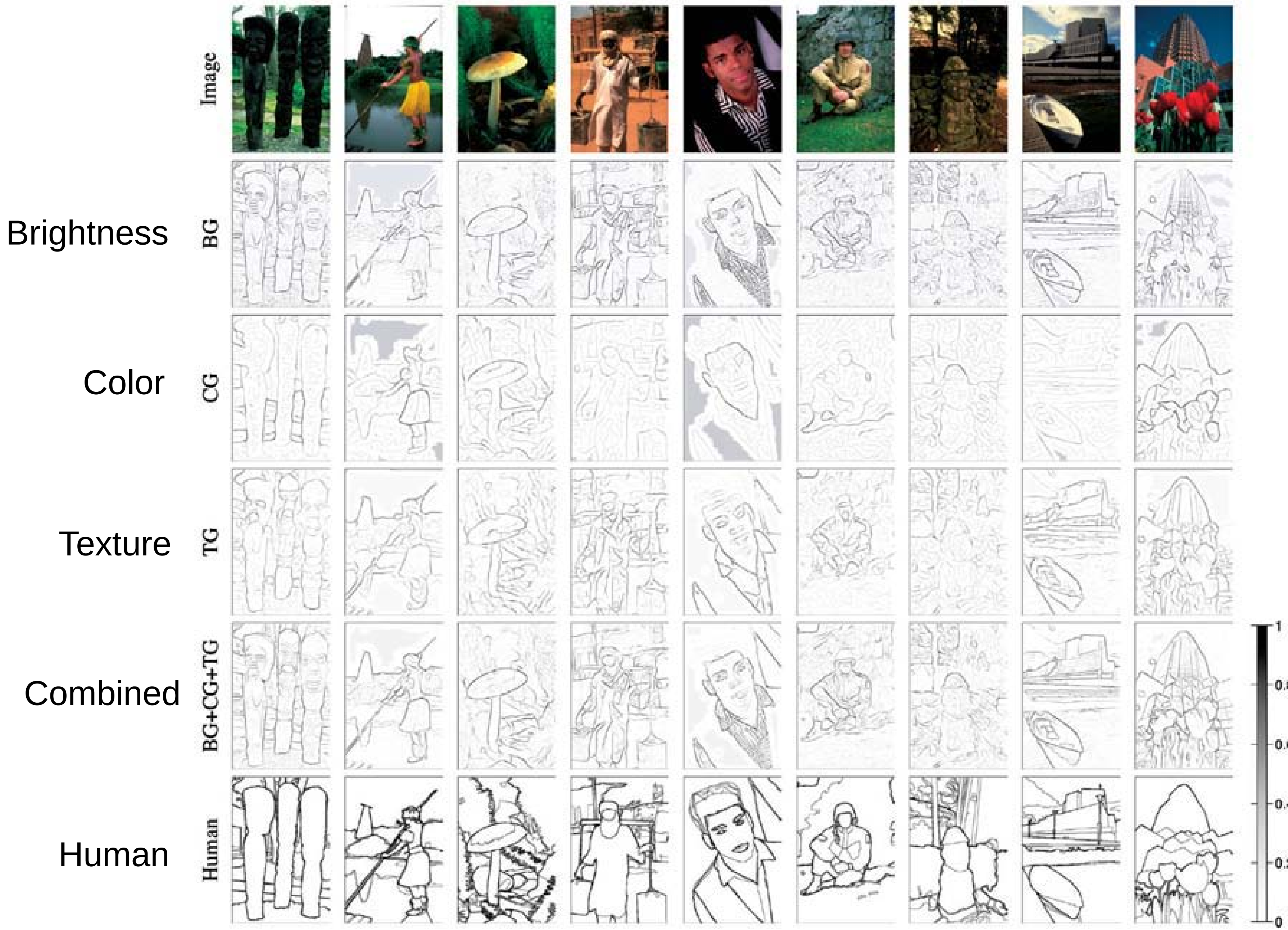


Image
Brightness
BG
Color
CG
Texture
TG
Combined
BG+CG+TG
Human
1
0.8
0.6
0.4
0.2
0

Finding straight lines

- One solution: try many possible lines and see how many points each line passes through

- Hough transform provides a fast way to do this

Outline of Hough Transform

1. Create a grid of parameter values

2. Each point votes for a set of parameters, incrementing those values in grid

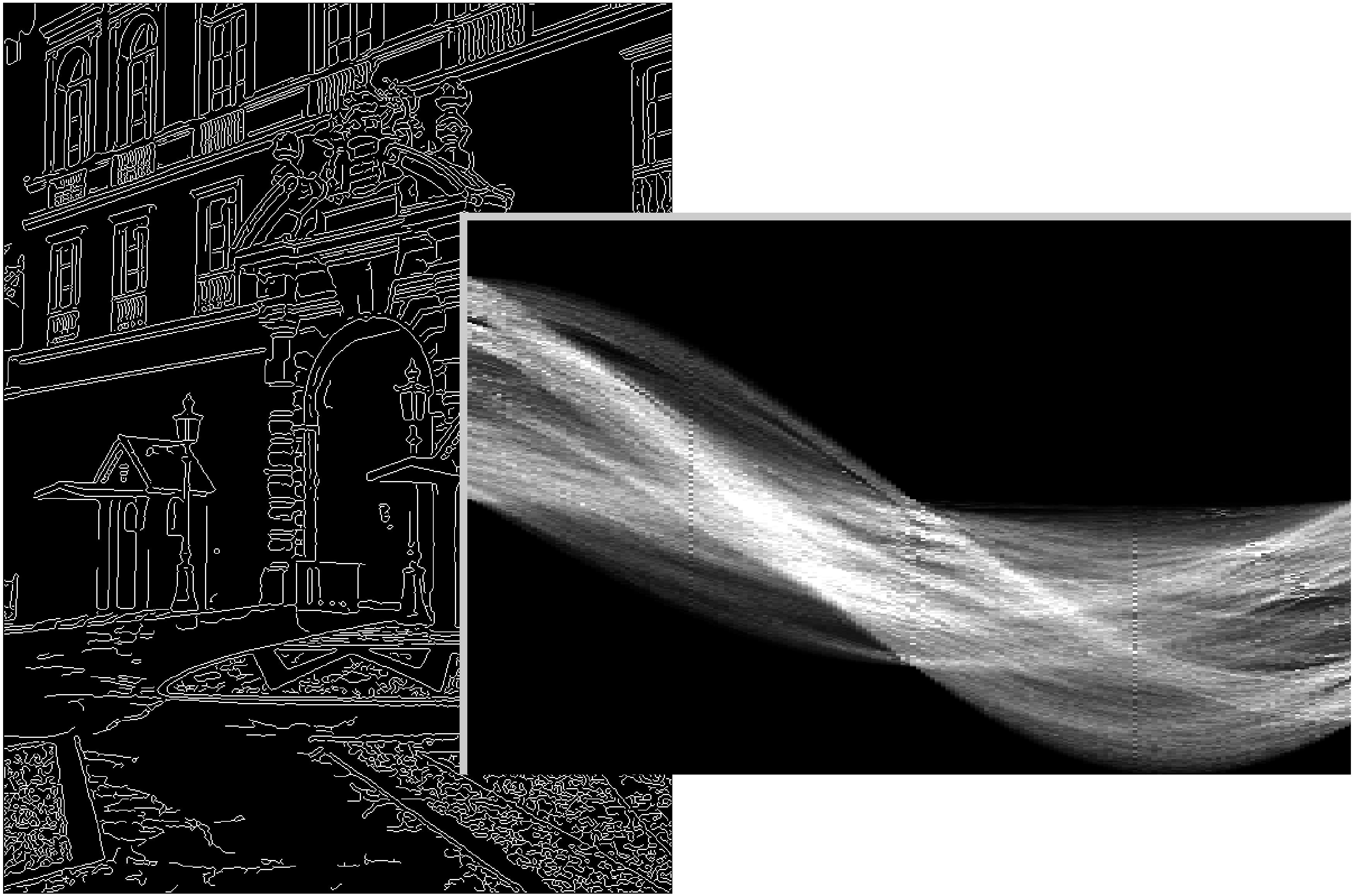
3. Find maximum or local maxima in grid

Finding lines using Hough transform

- Using m,b parameterization
- Using r, theta parameterization
  - Using oriented gradients
- Practical considerations
  - Bin size
  - Smoothing
  - Finding multiple lines
  - Finding line segments

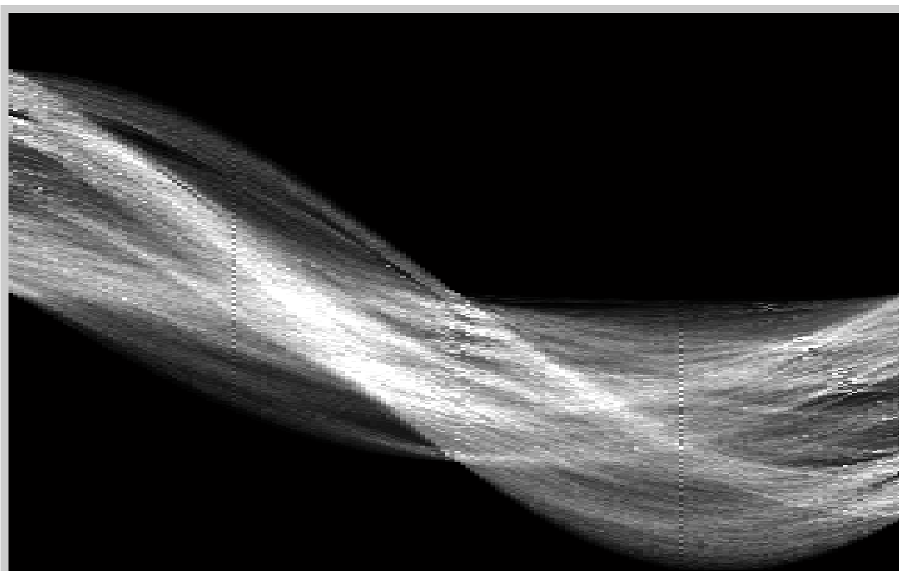
1. Image → Canny

2. Canny → Hough votes

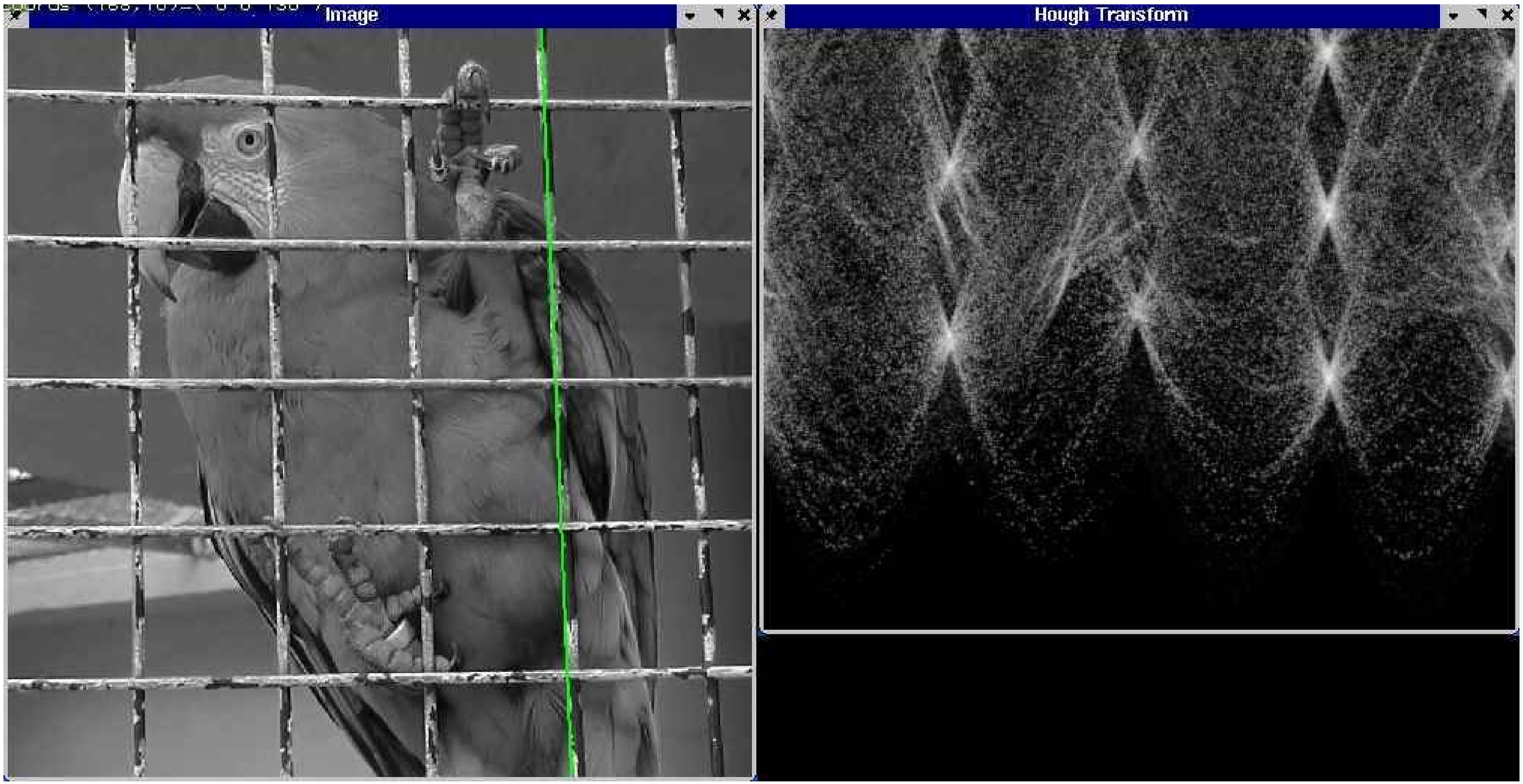


3. Hough votes → Edges

Find peaks and post-process



Hough transform example



http://ostatic.com/files/images/ss_hough.jpg

Finding circles using Hough transform

- Fixed r
- Variable r

Finding straight lines

- Another solution: get connected components of pixels and check for straightness

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

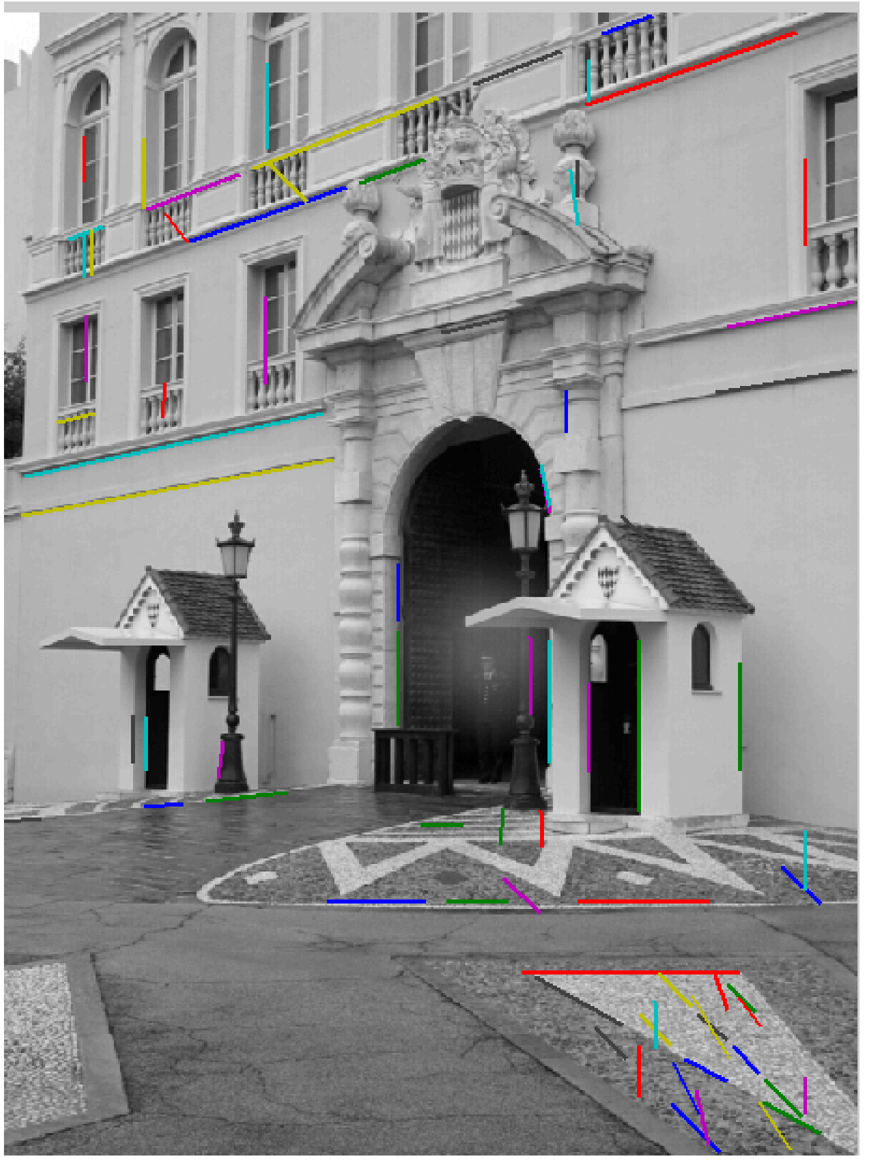
$$\theta = \mathrm{atan}\,2(v(2,2), v(1,2))$$
$$conf = \lambda_2 / \lambda_1$$

5. Threshold on straightness, store segment

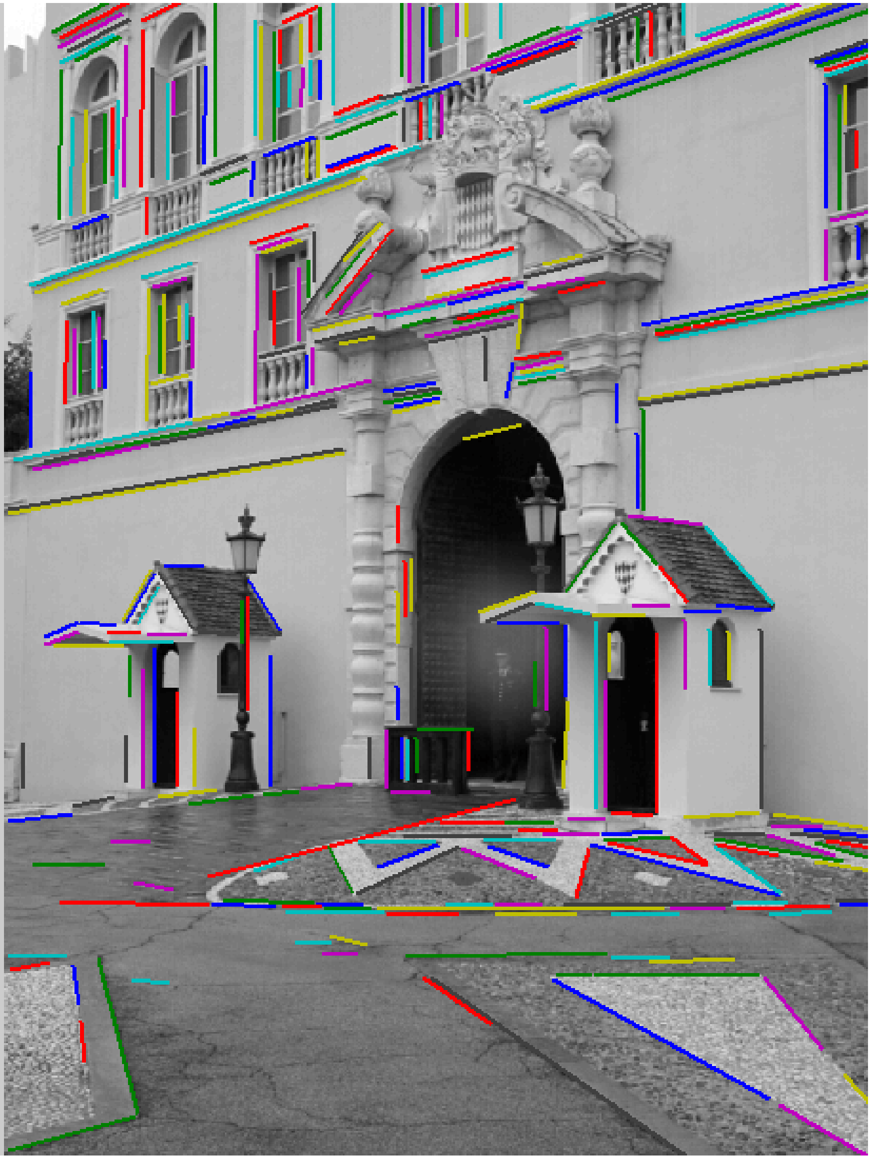
1. Image → Canny

2. Canny lines → … →straight edges

Comparison



Hough Transform Method



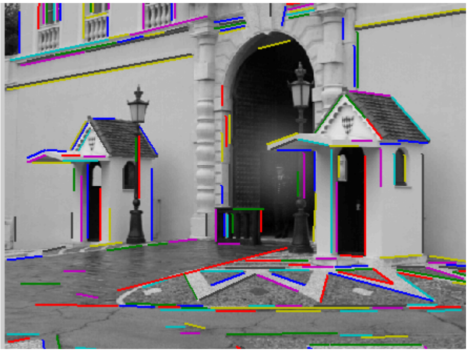
Connected Components Method

Things to remember

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

- Generalized Hough transform = points vote for shape parameters

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



Next classes

- Fitting and Registration
- Clustering
- EM (mixture models)

Questions