Finding Boundaries

Computer Vision

CS 143, Brown

James Hays

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

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

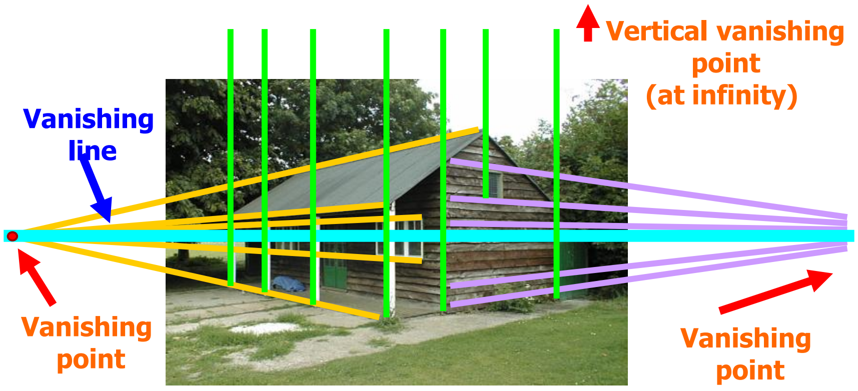
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

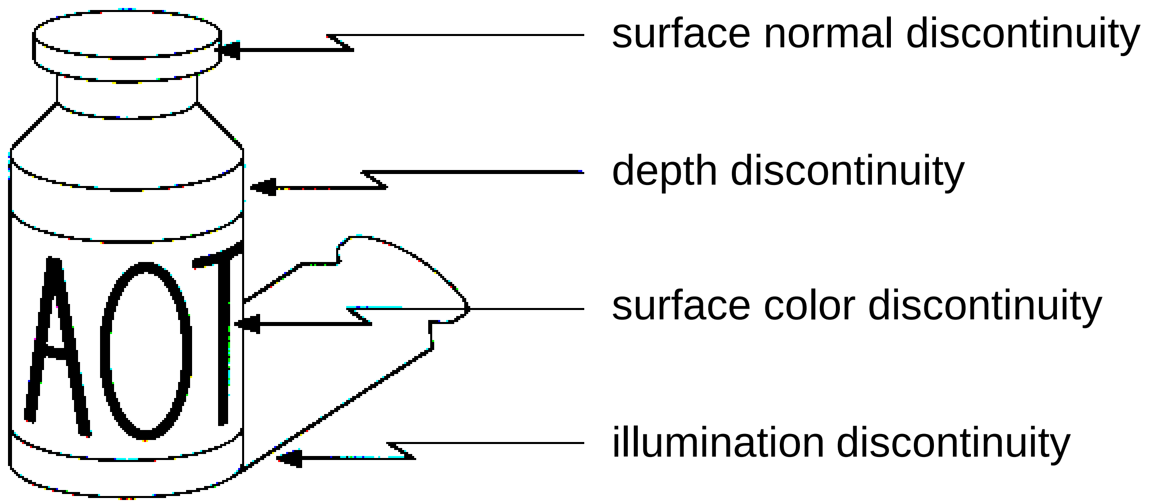
Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint



Origin of Edges



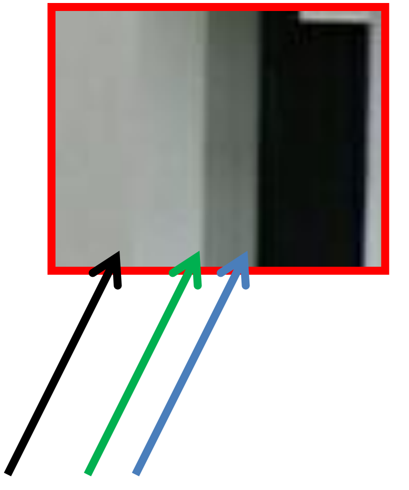
- Edges are caused by a variety of factors

Source: Steve Seitz

Closeup of edges

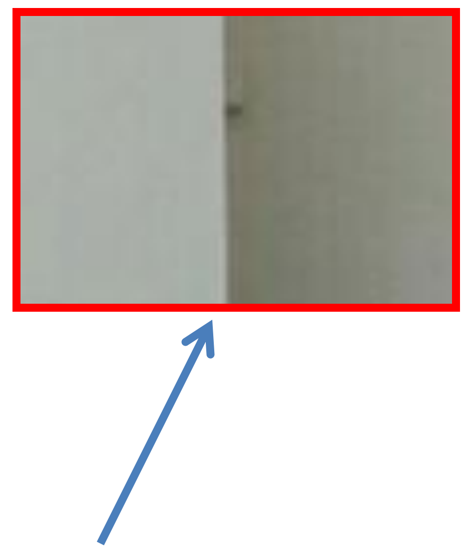
Source: D. Hoiem

Closeup of edges



Source: D. Hoiem

Closeup of edges



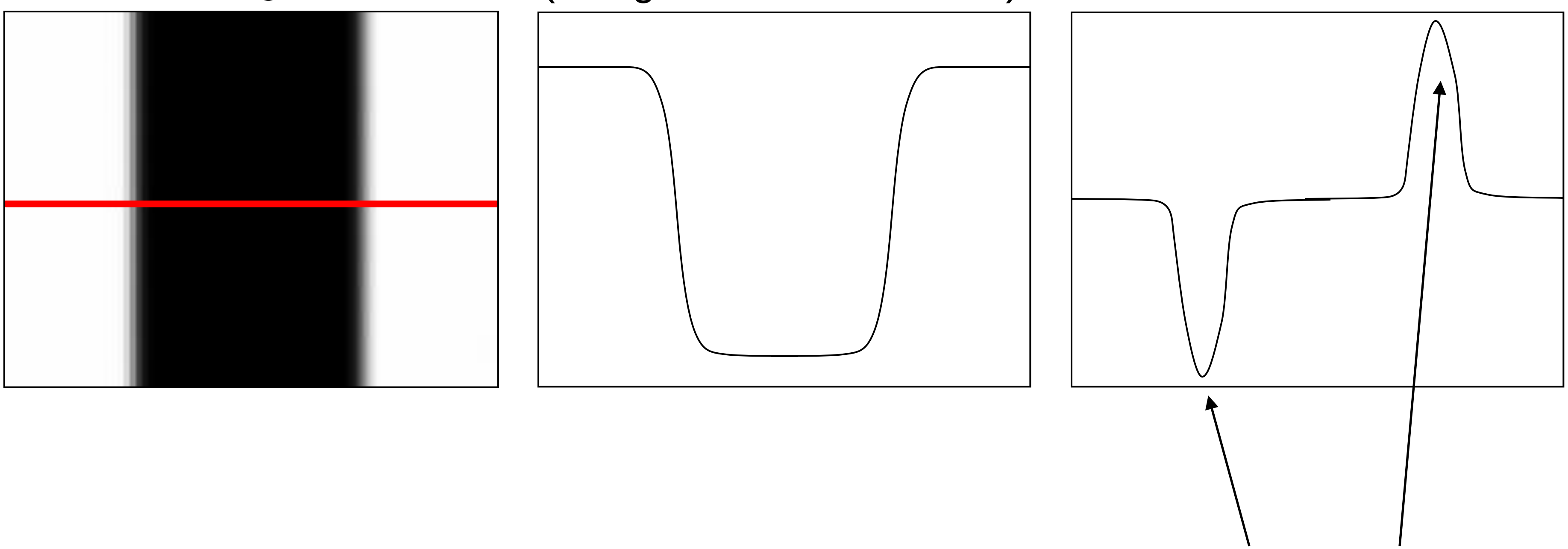
Source: D. Hoiem

Closeup of edges

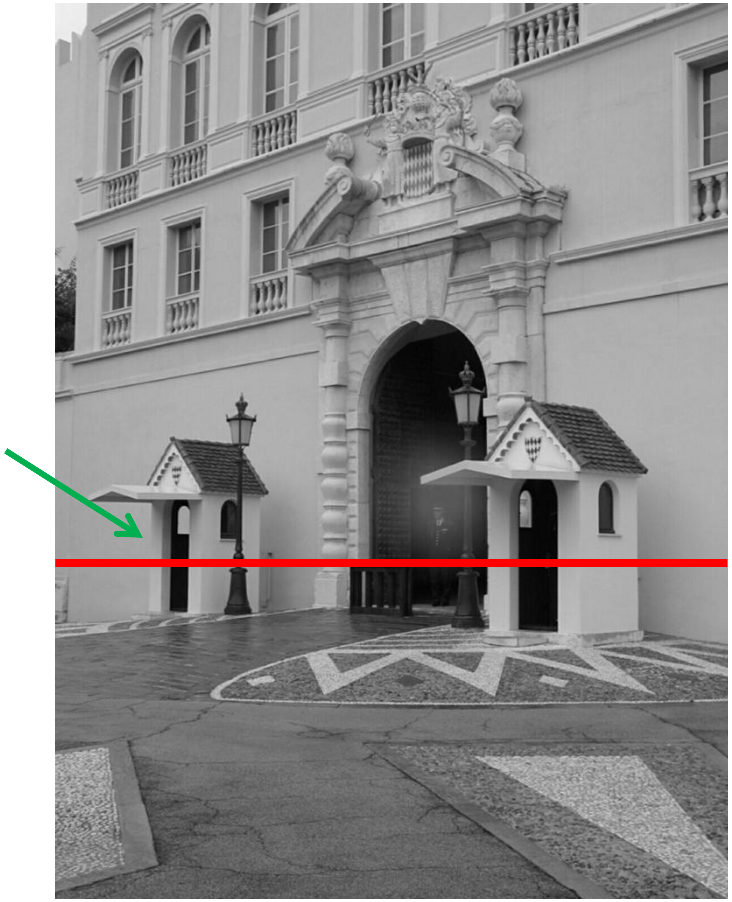
Source: D. Hoiem

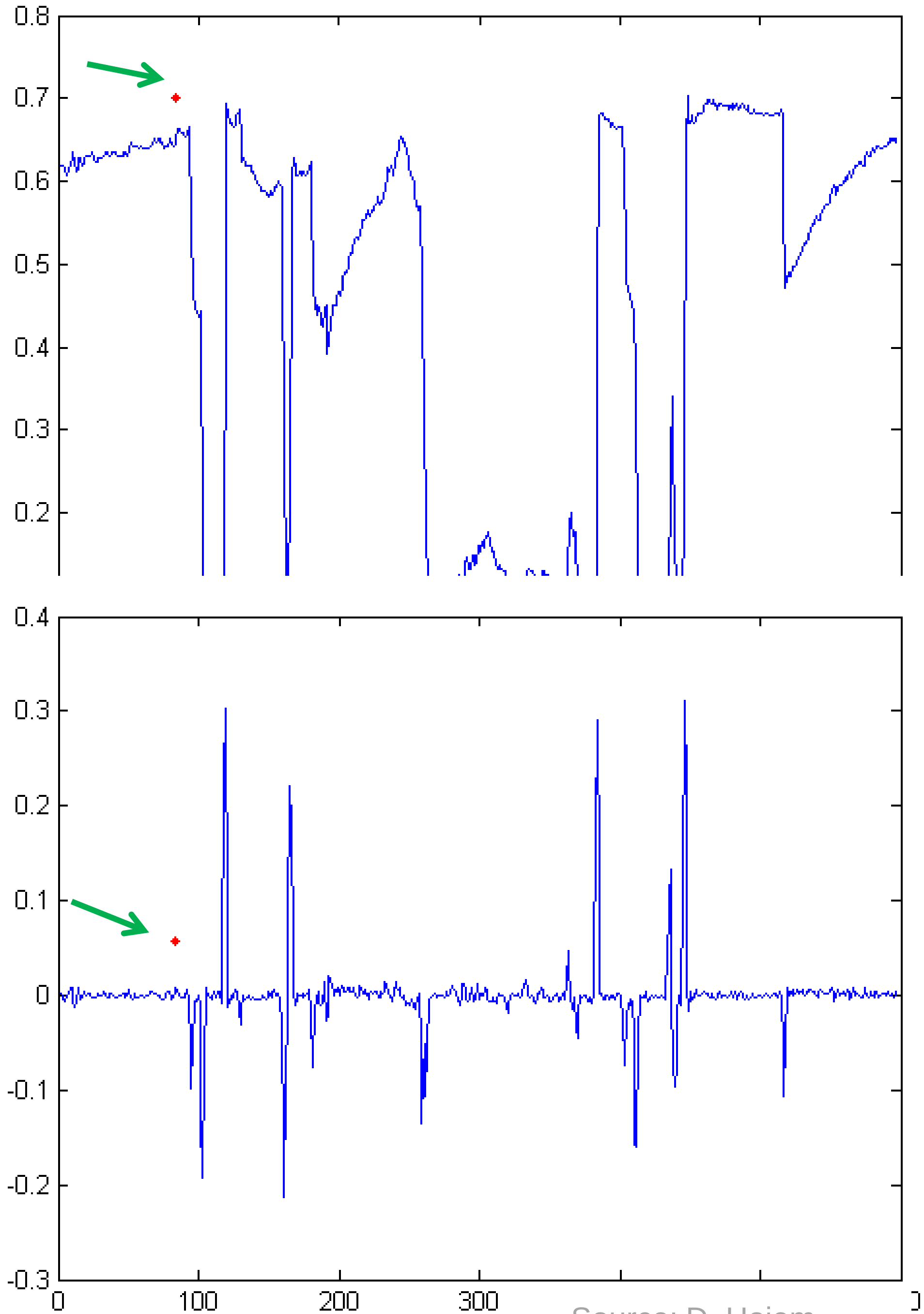
Characterizing edges

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



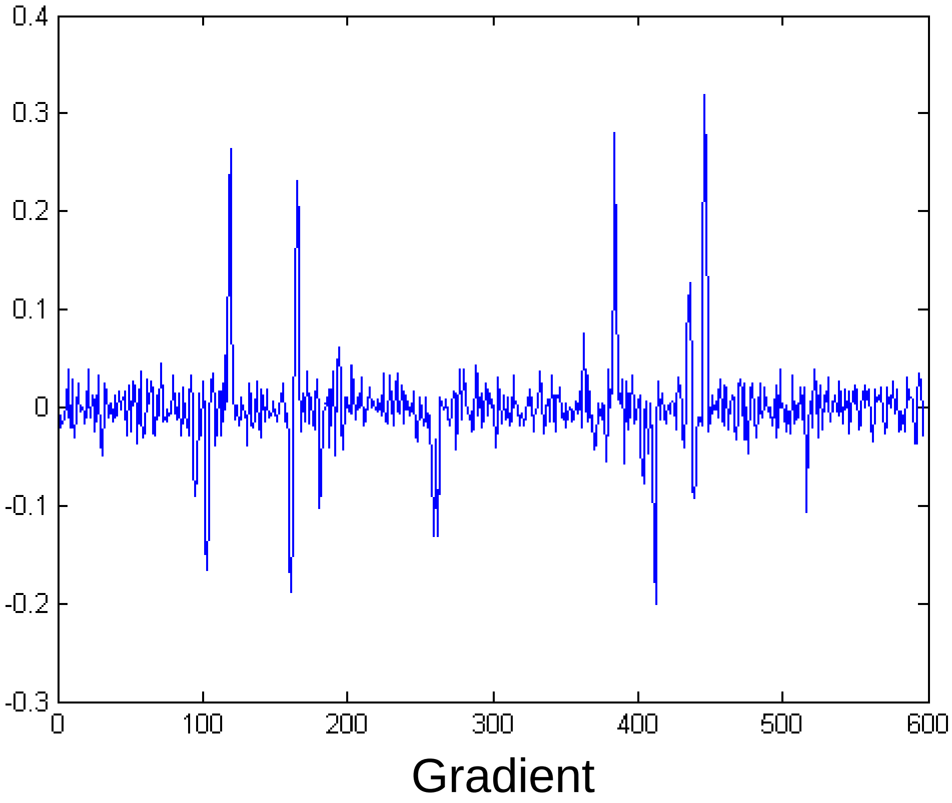
Intensity profile





Source: D. Hoiem

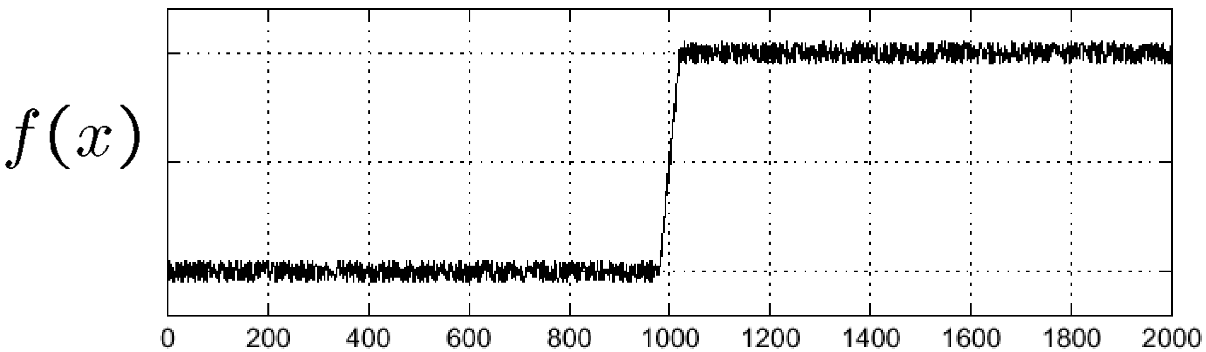
With a little Gaussian noise

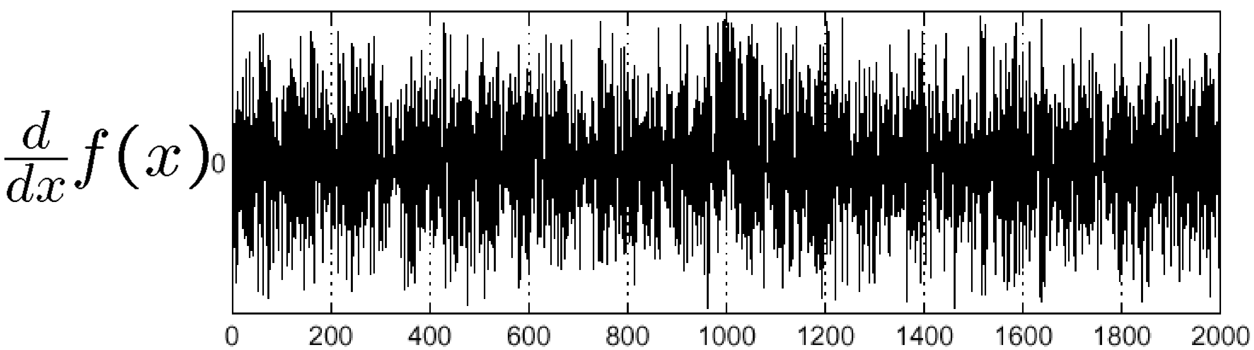


Source: D. Hoiem

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?

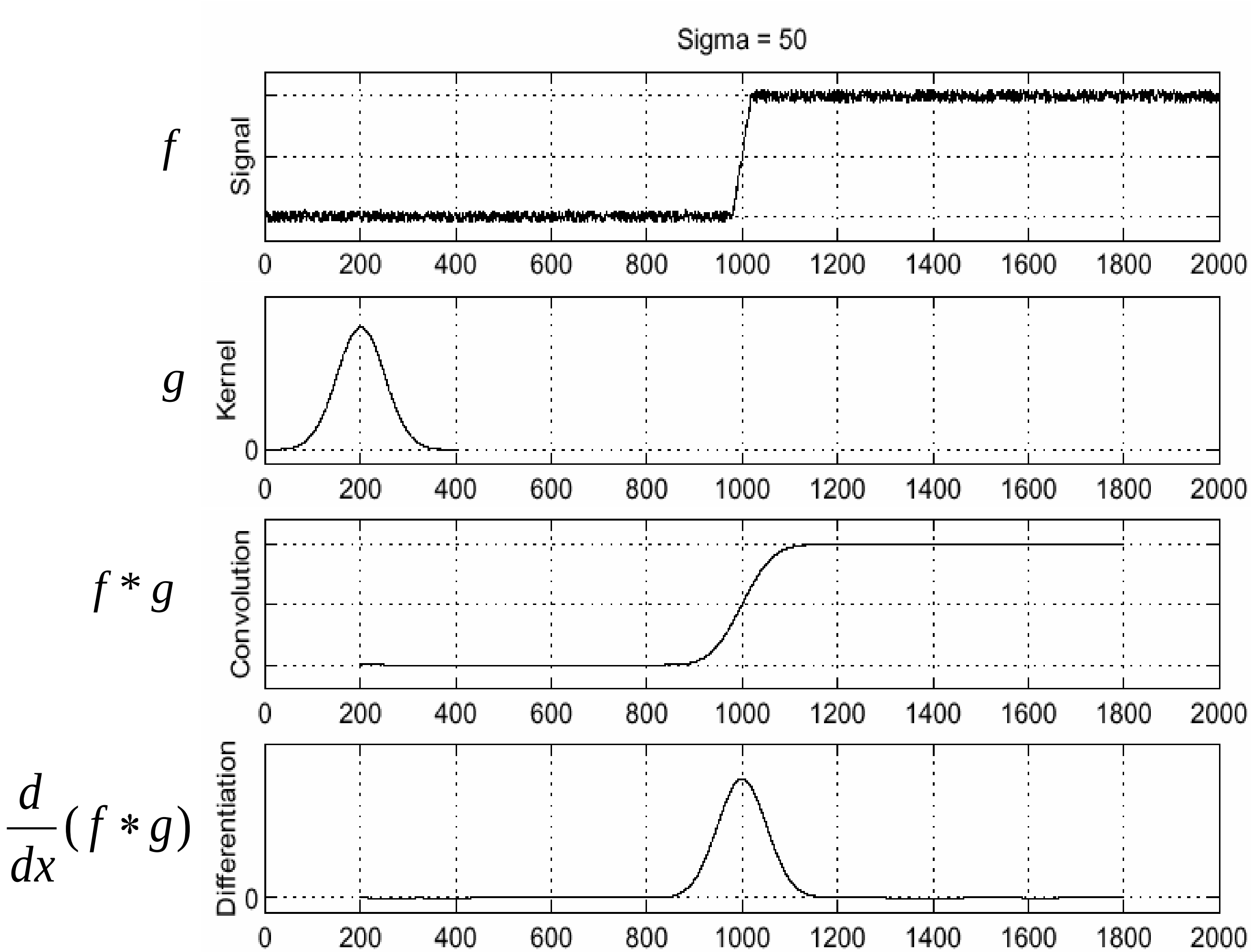
Source: S. Seitz

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

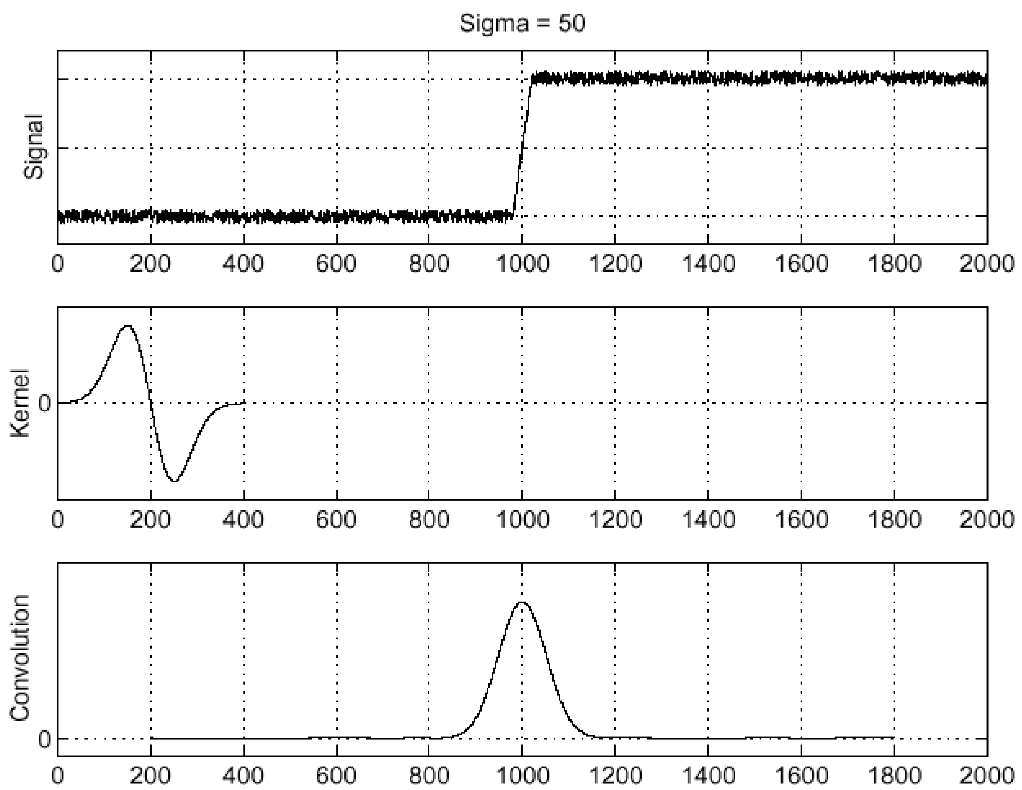
Derivative theorem of convolution

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

f

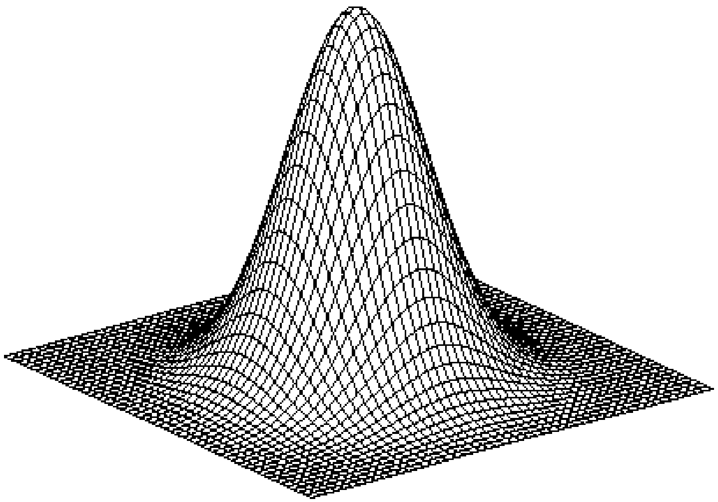
$\frac{d}{dx}g$

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

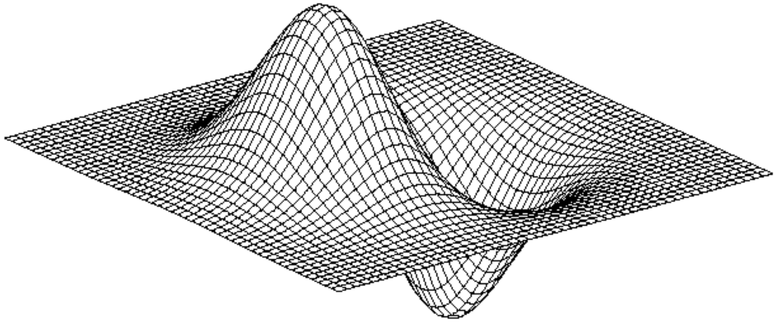


Source: S. Seitz

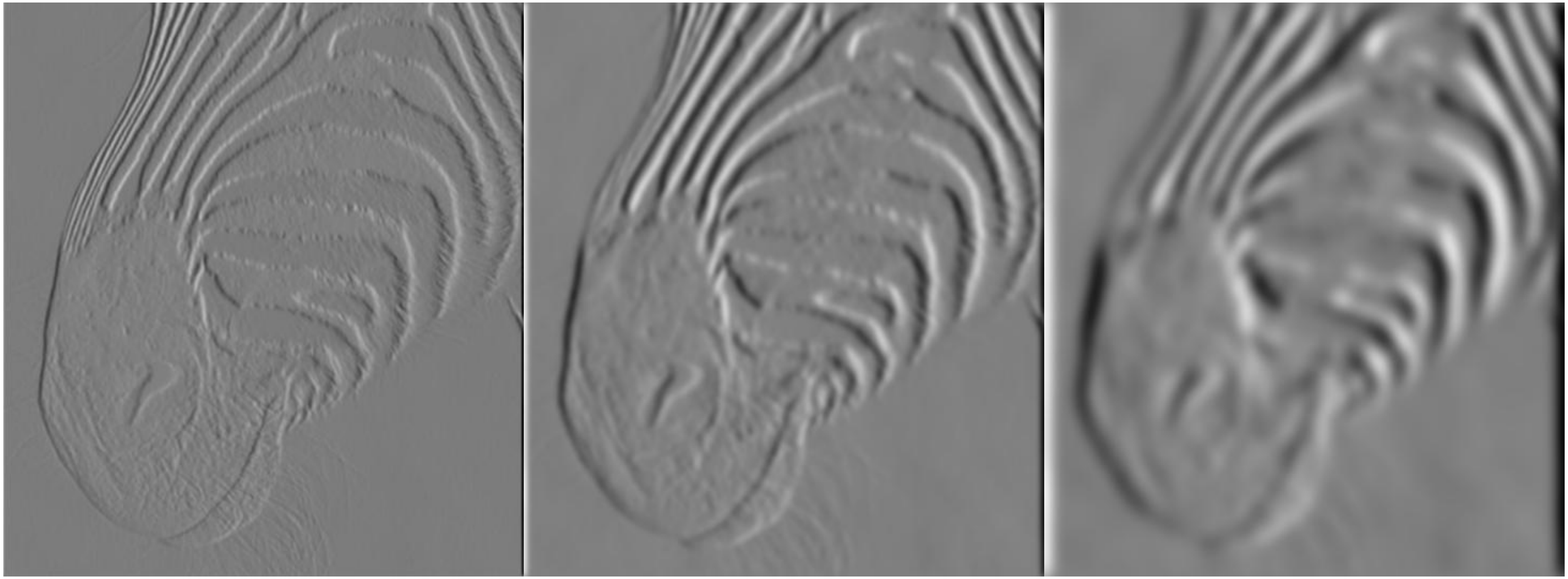
Derivative of Gaussian filter



* [1 -1] =



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

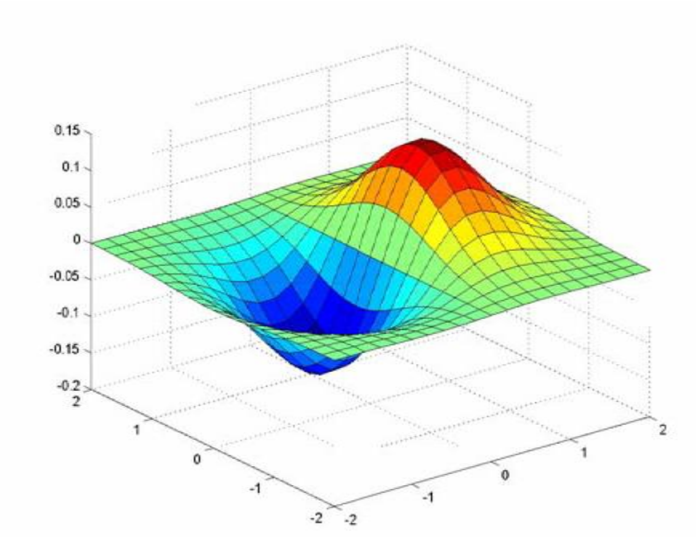
J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

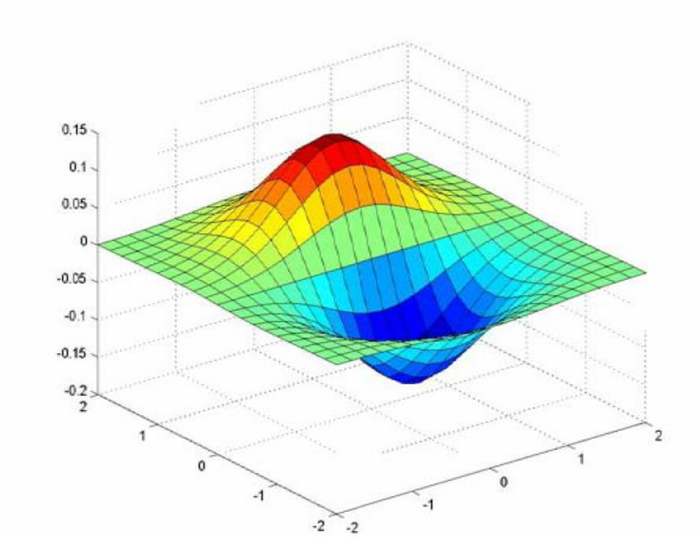
Example

original image (Lena)

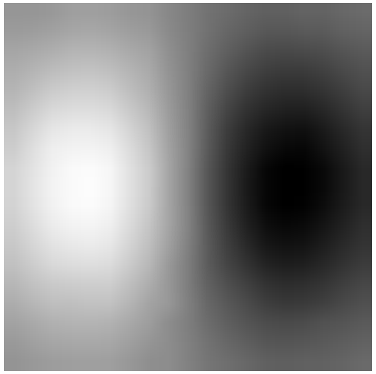
Derivative of Gaussian filter

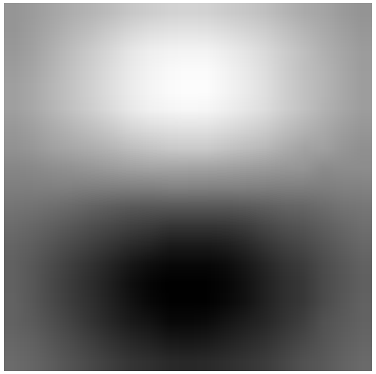


x-direction

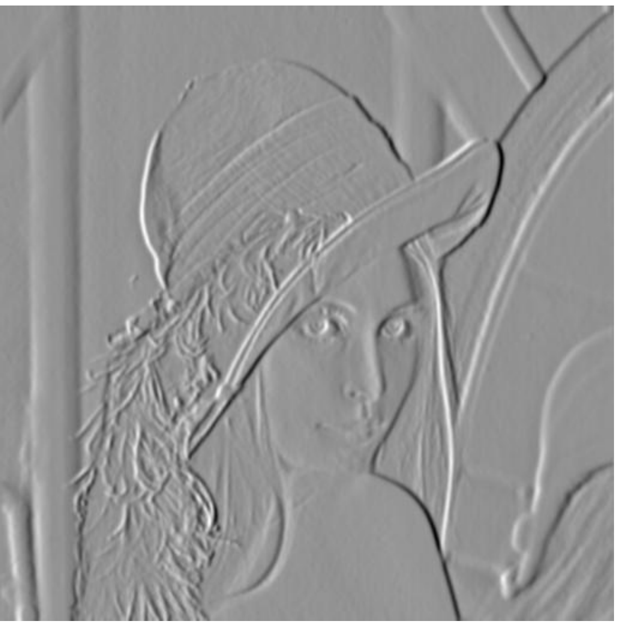


y-direction





Compute Gradients (DoG)



X-Derivative of Gaussian

Y-Derivative of Gaussian

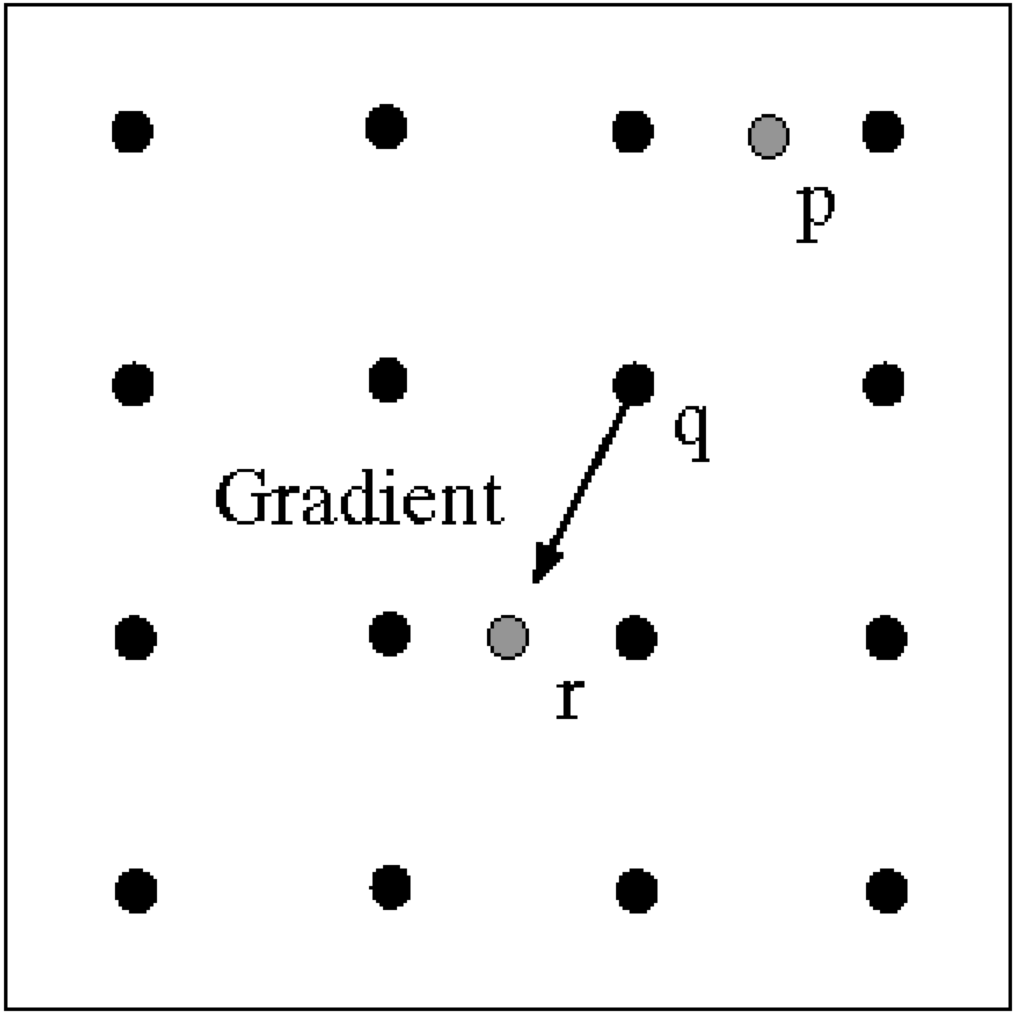
Gradient Magnitude

Get Orientation at Each Pixel

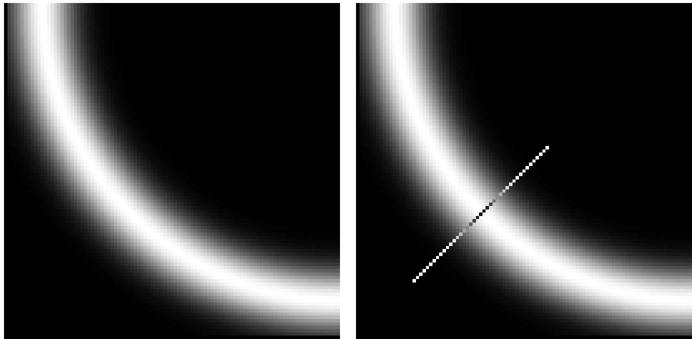
- Threshold at minimum level
- Get orientation

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

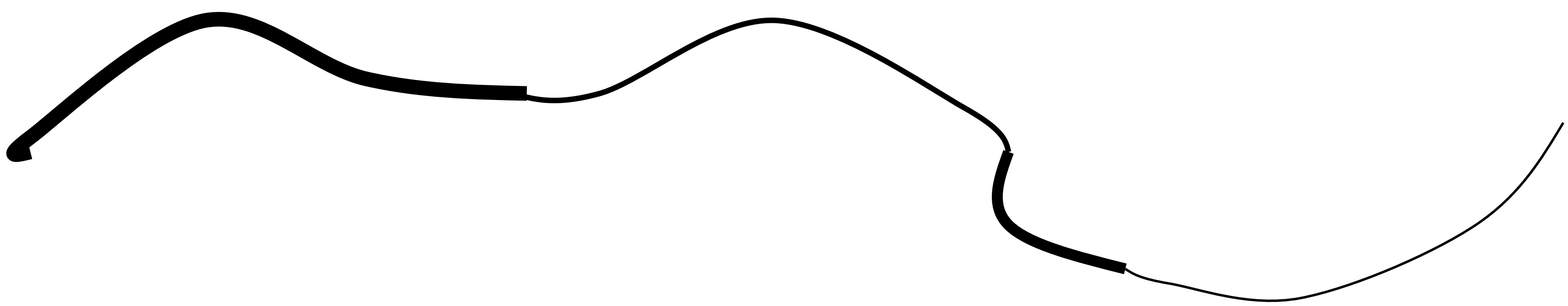
After non-max suppression

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

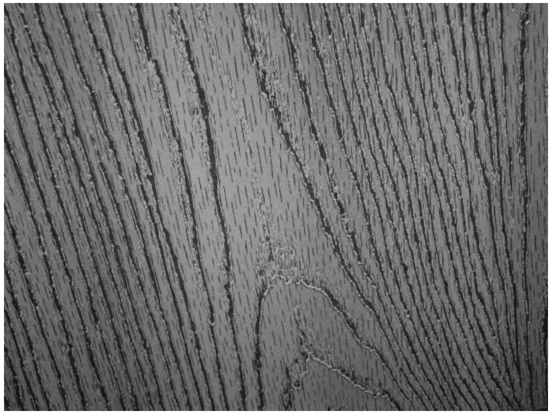
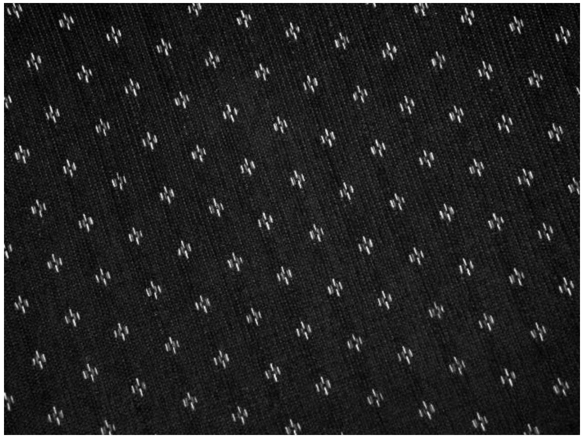
image | human segmentation | gradient magnitude

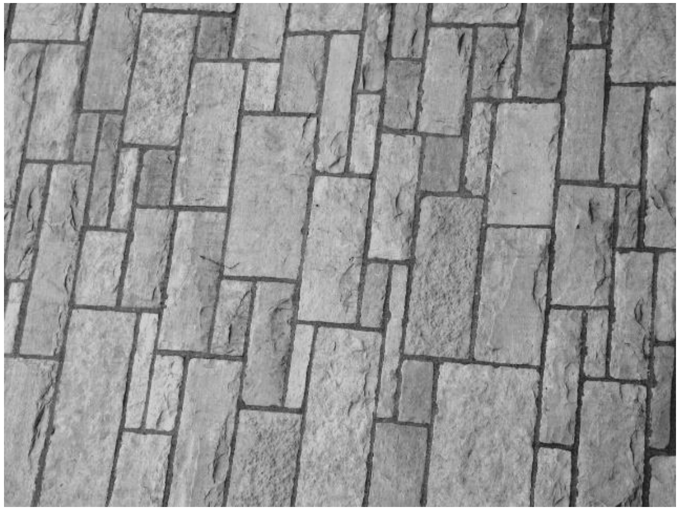
- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Representing Texture

Source: Forsyth

Texture and Material

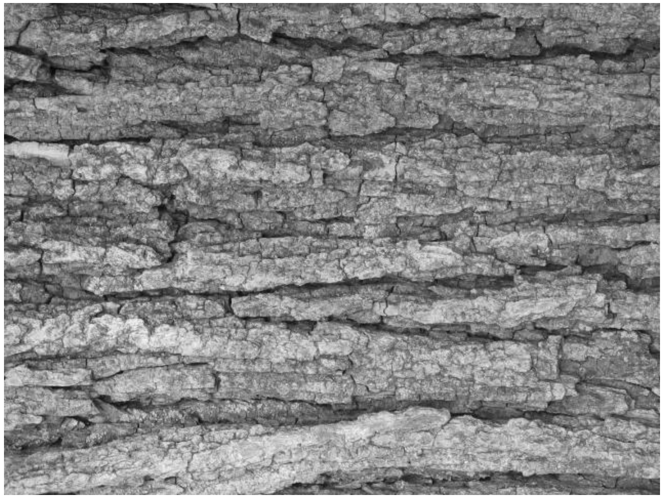





http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Orientation





http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Scale

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

What is texture?

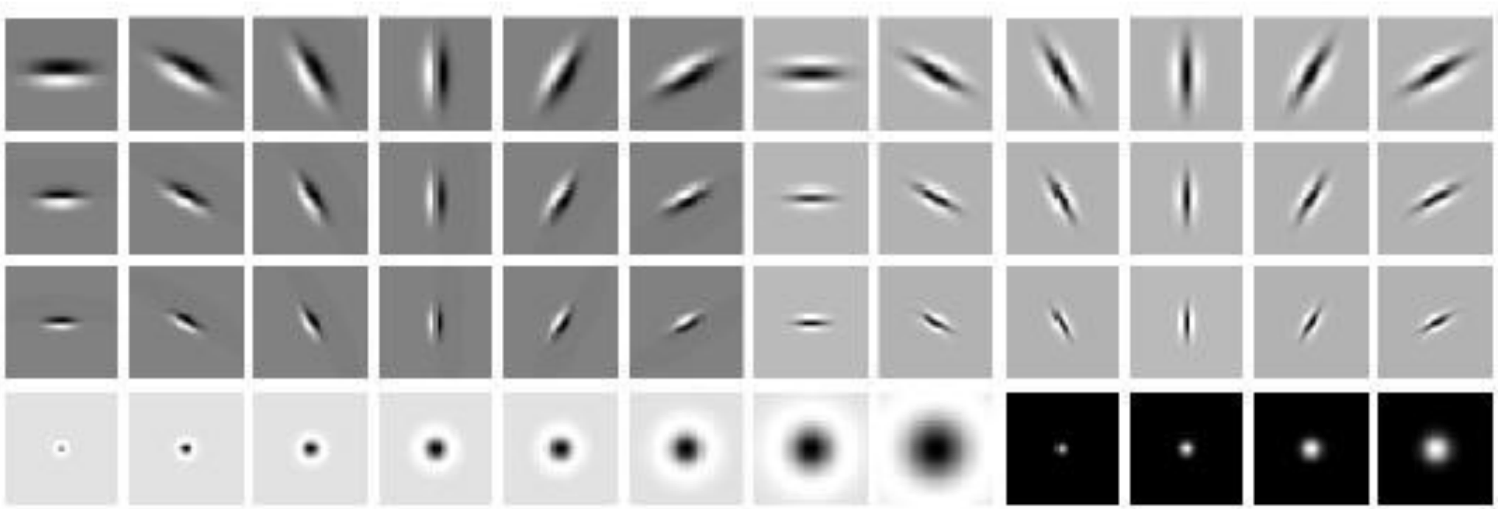
Regular or stochastic patterns caused by bumps, grooves, and/or markings

How can we represent texture?

- Compute responses of blobs and edges at various orientations and scales

Overcomplete representation: filter banks

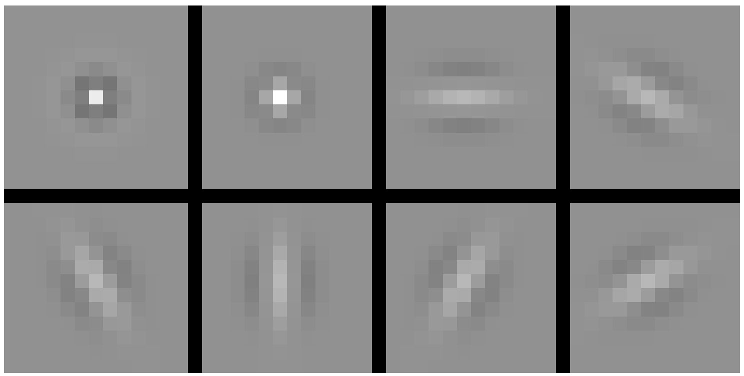
LM Filter Bank

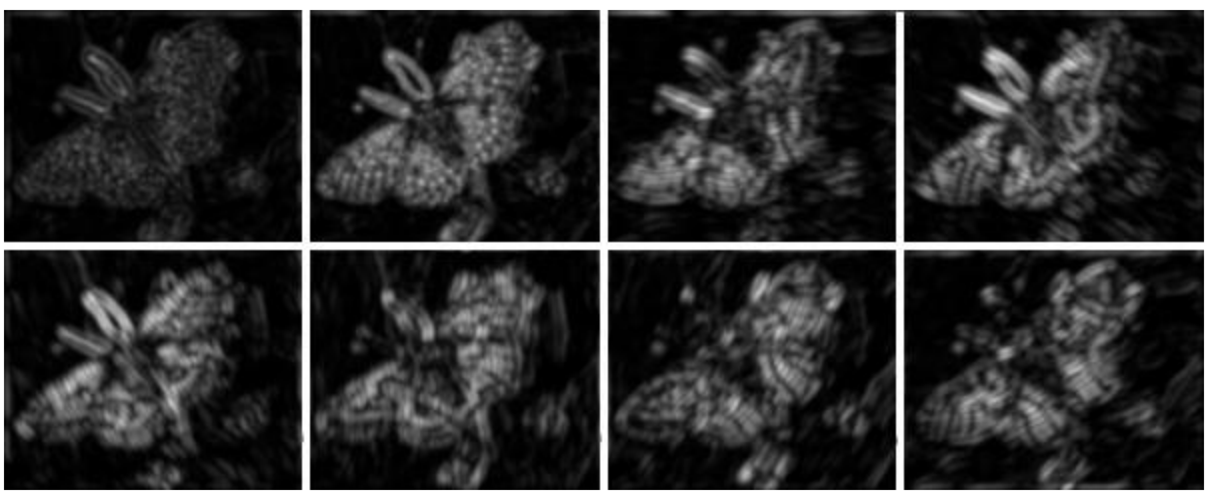


Code for filter banks: *www.robots.ox.ac.uk/~vgg/research/texclass/filters.html*

Filter banks

- Process image with each filter and keep responses (or squared/abs responses)

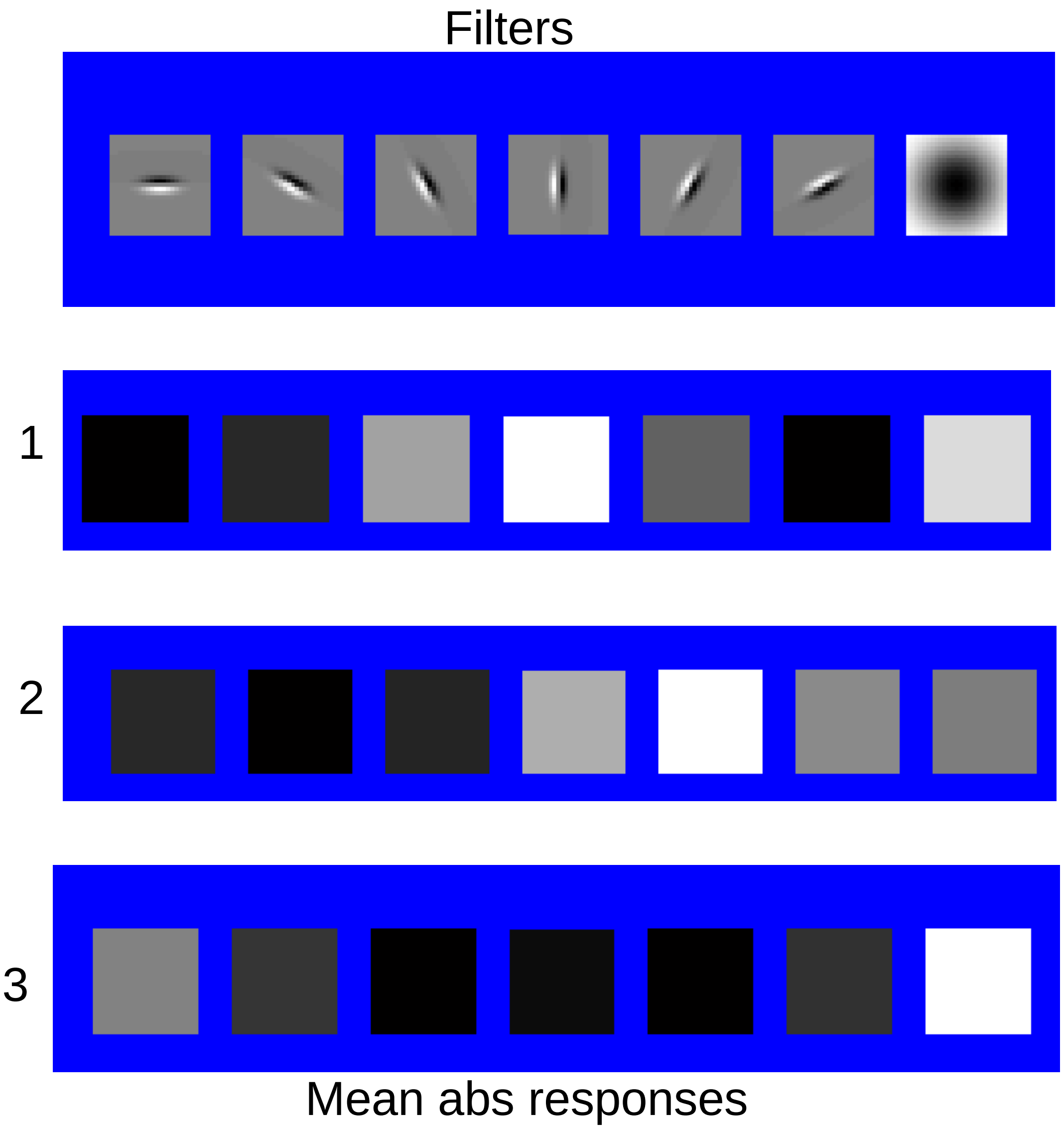




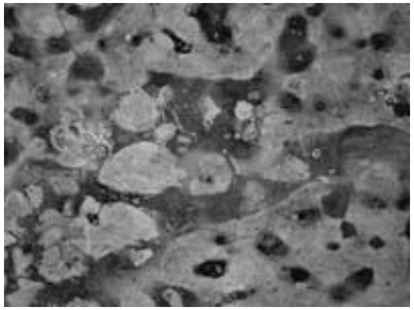
How can we represent texture?

- Measure responses of blobs and edges at various orientations and scales
- Idea 1: Record simple statistics (e.g., mean, std.) of absolute filter responses

Can you match the texture to the response?

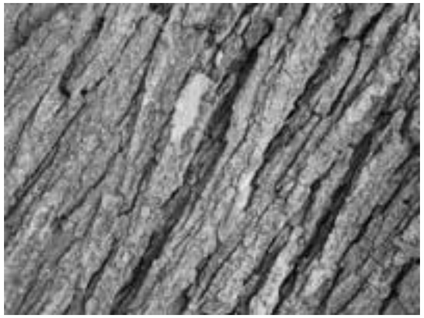


A



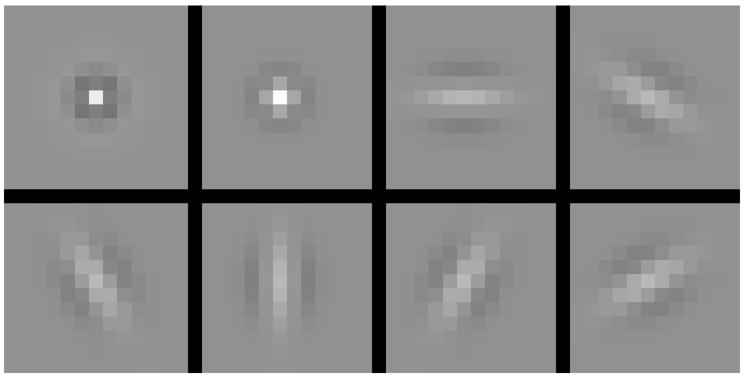
B

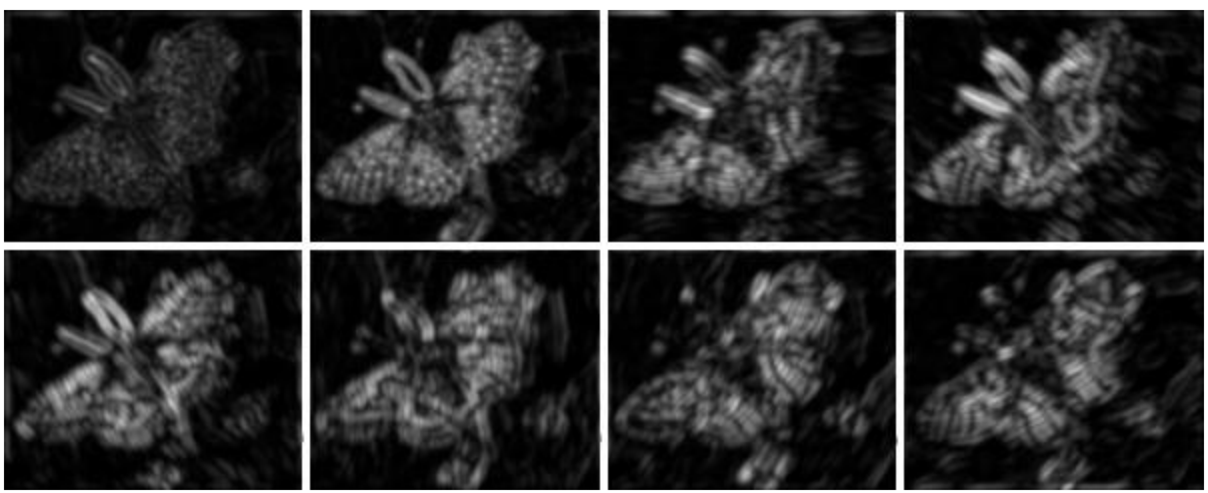
C



Representing texture

- Idea 2: take vectors of filter responses at each pixel and cluster them, then take histograms.

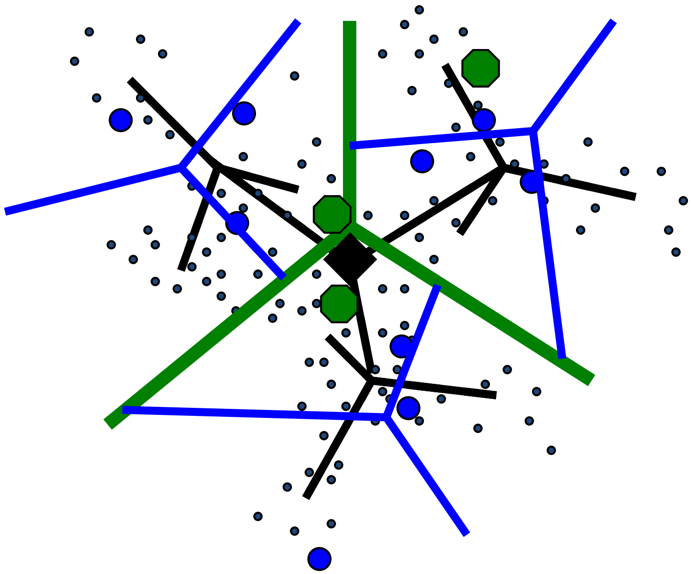




Building Visual Dictionaries

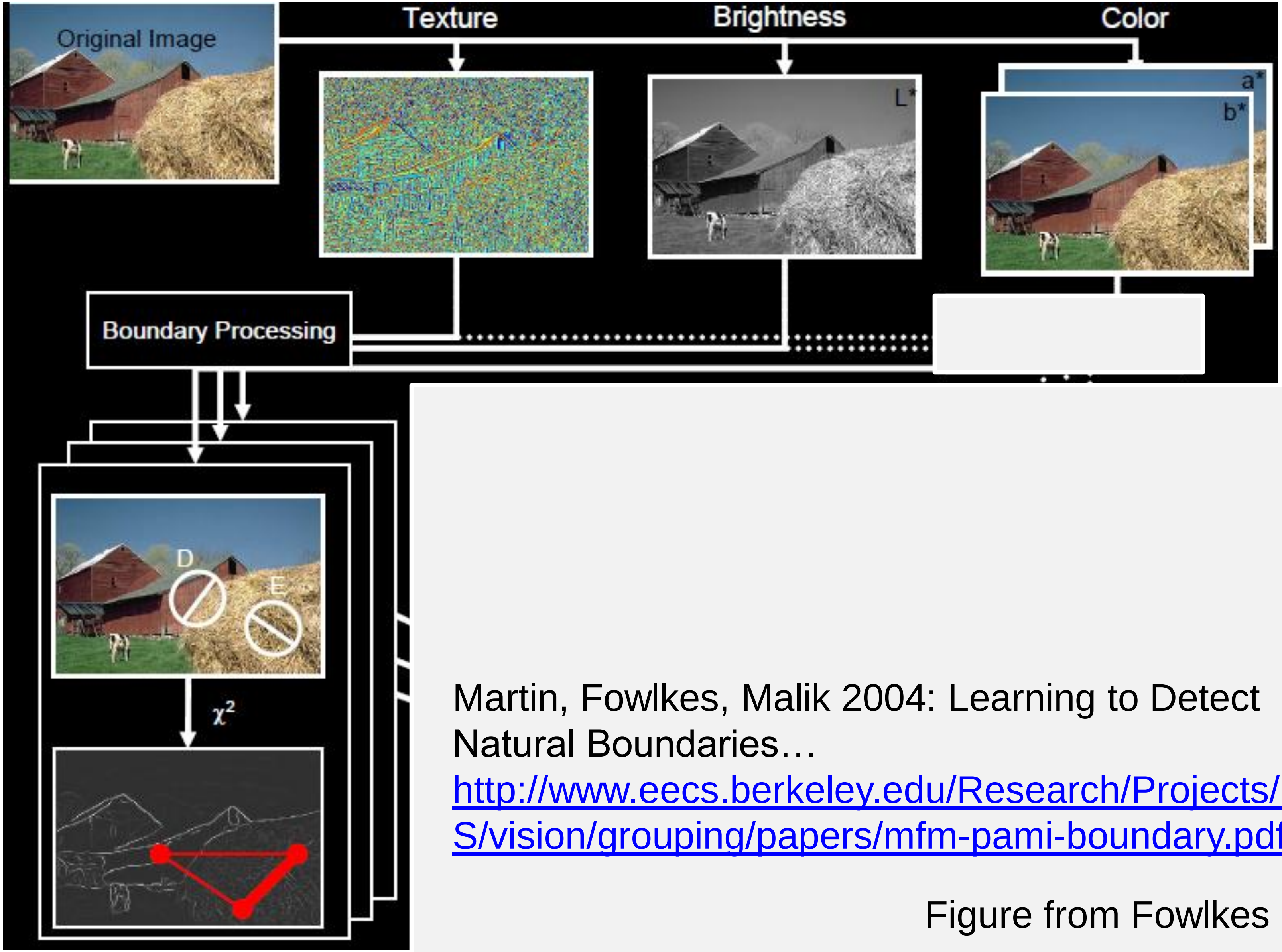
1. Sample patches from a database
   – E.g., 128 dimensional SIFT vectors

2. Cluster the patches
   – Cluster centers are the dictionary



3. Assign a codeword (number) to each new patch, according to the nearest cluster

pB boundary detector



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

Figure from Fowlkes

pB Boundary Detector

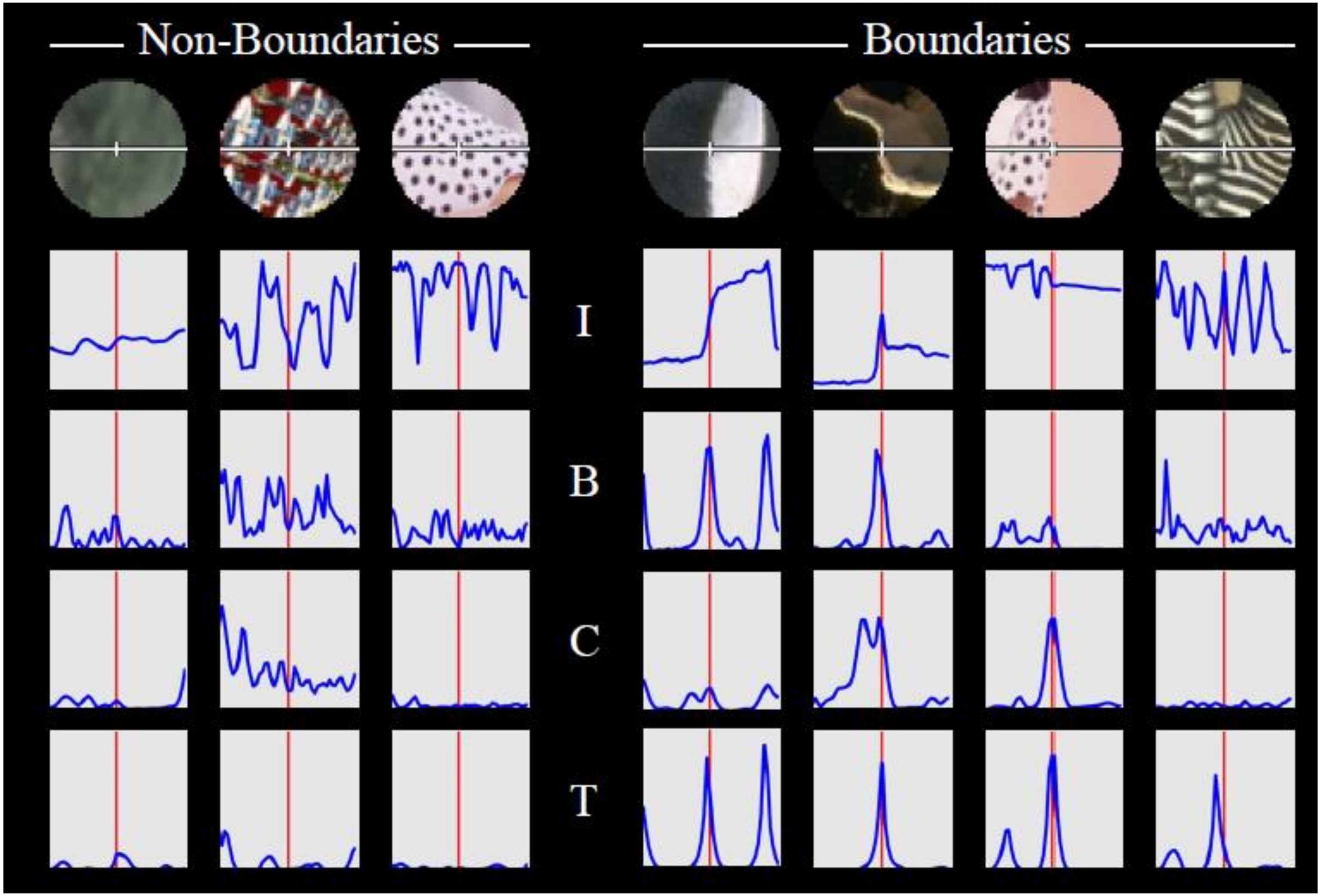
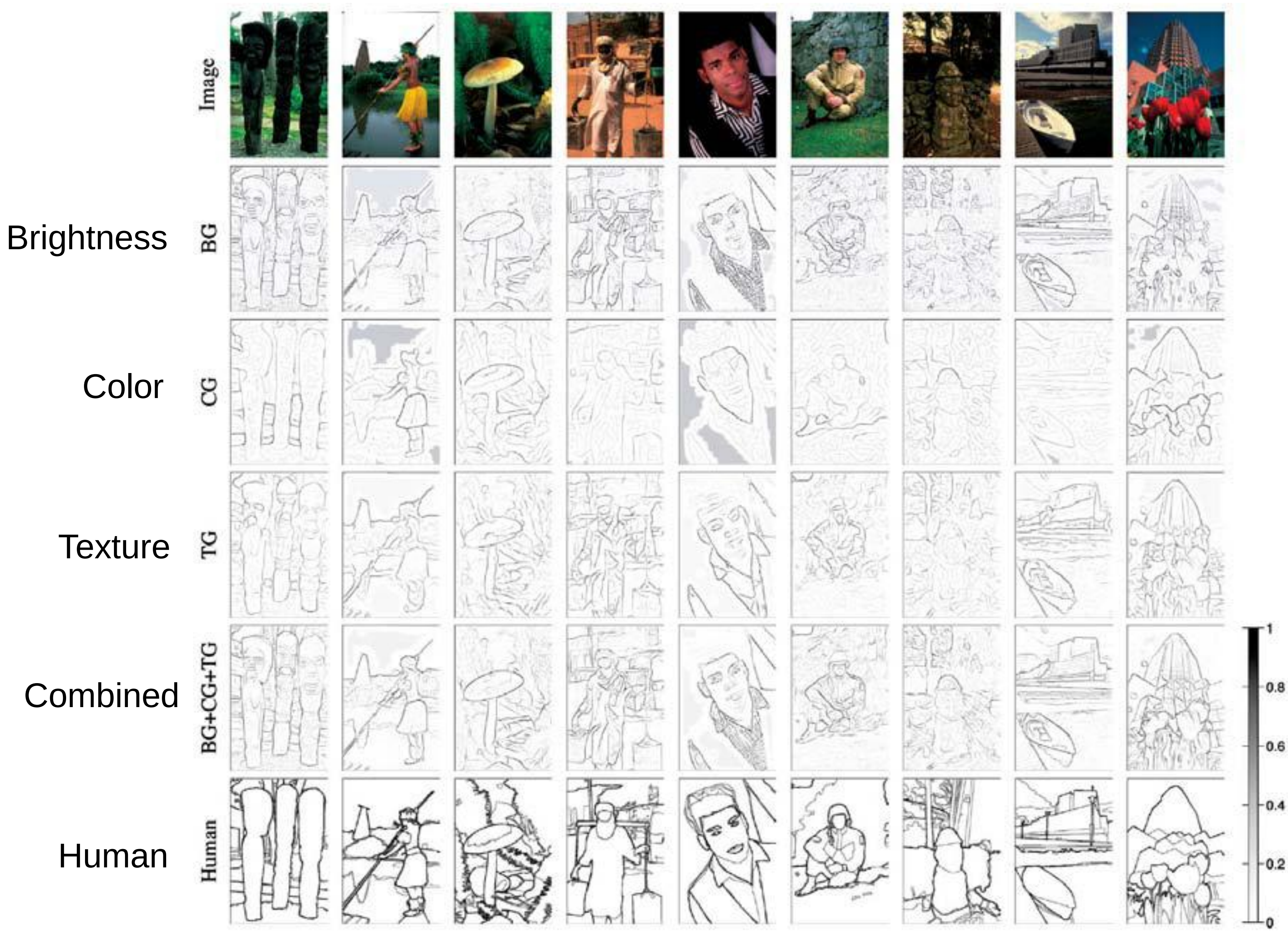


Figure from Fowlkes



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

Global pB boundary detector

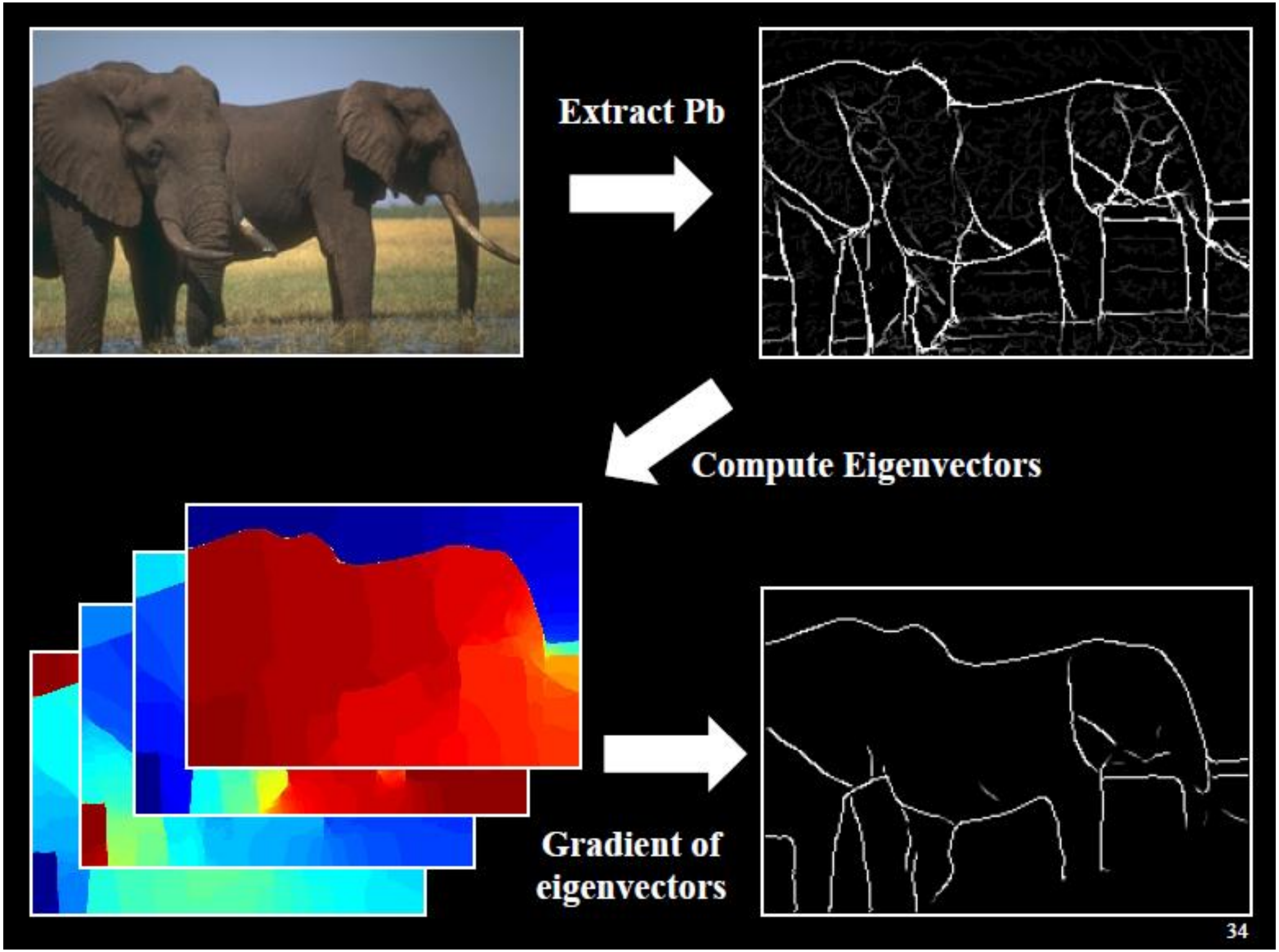
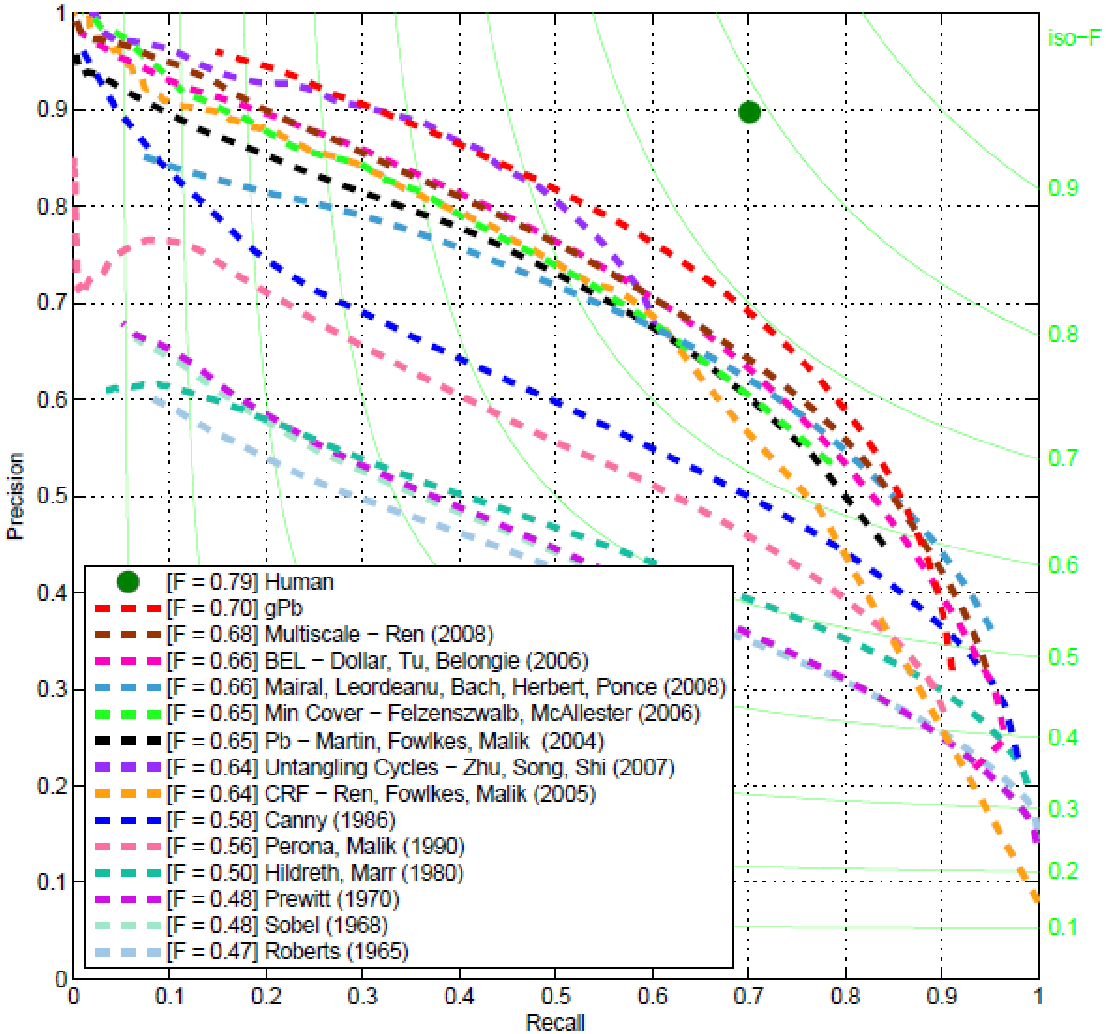


Figure from Fowlkes

45 years of boundary detection



Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)

Questions