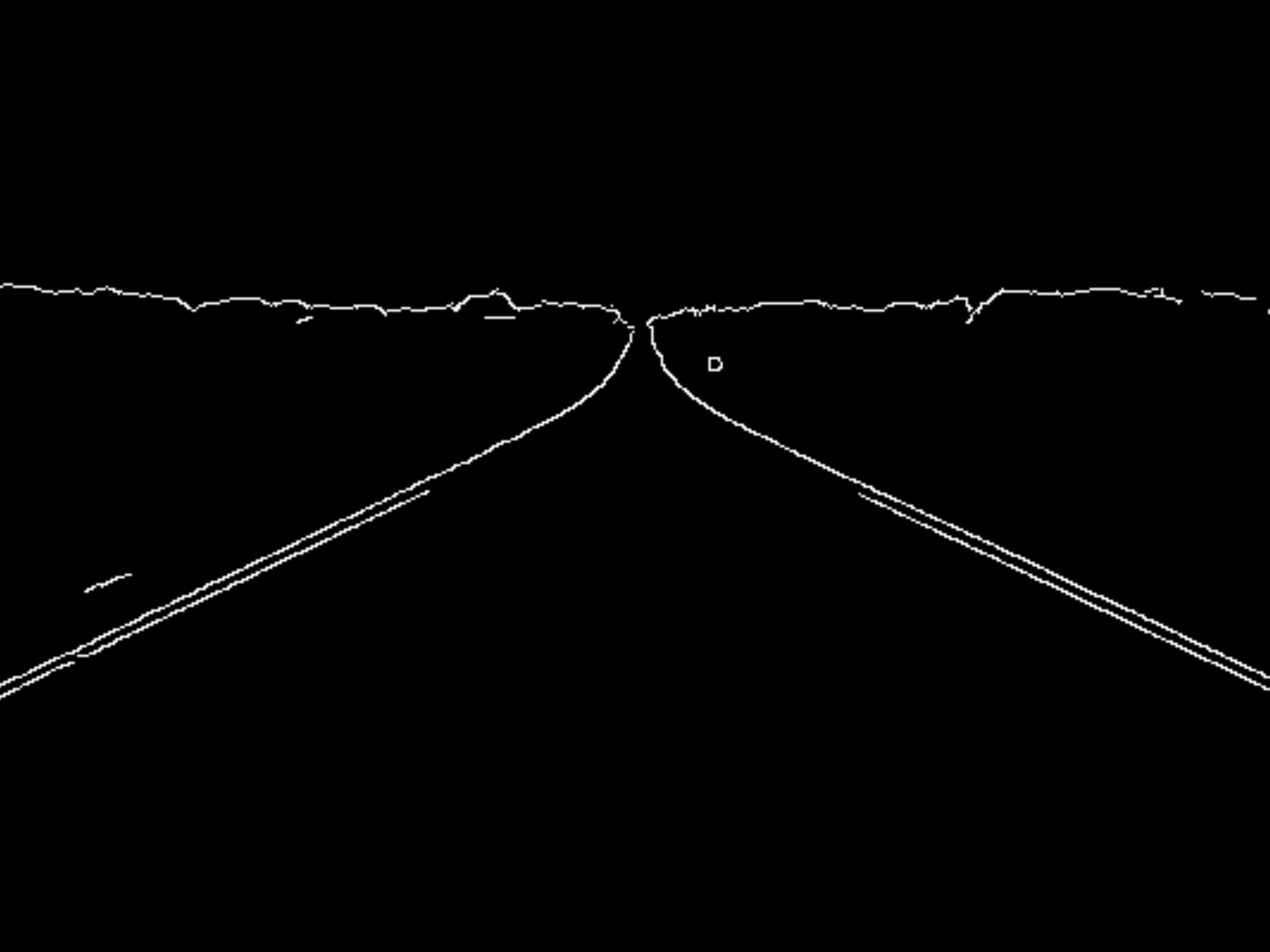
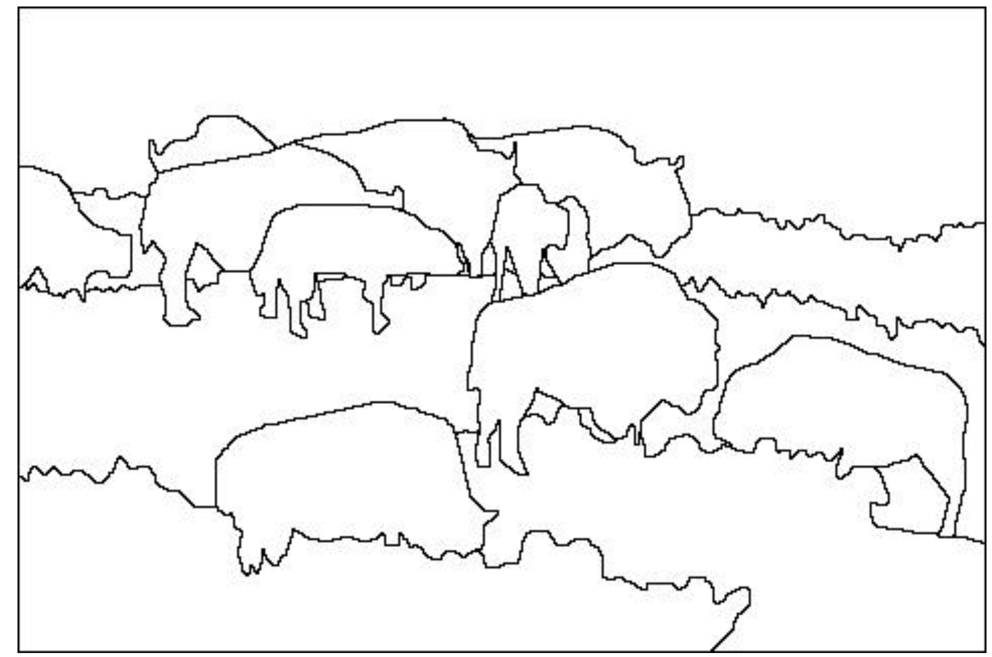
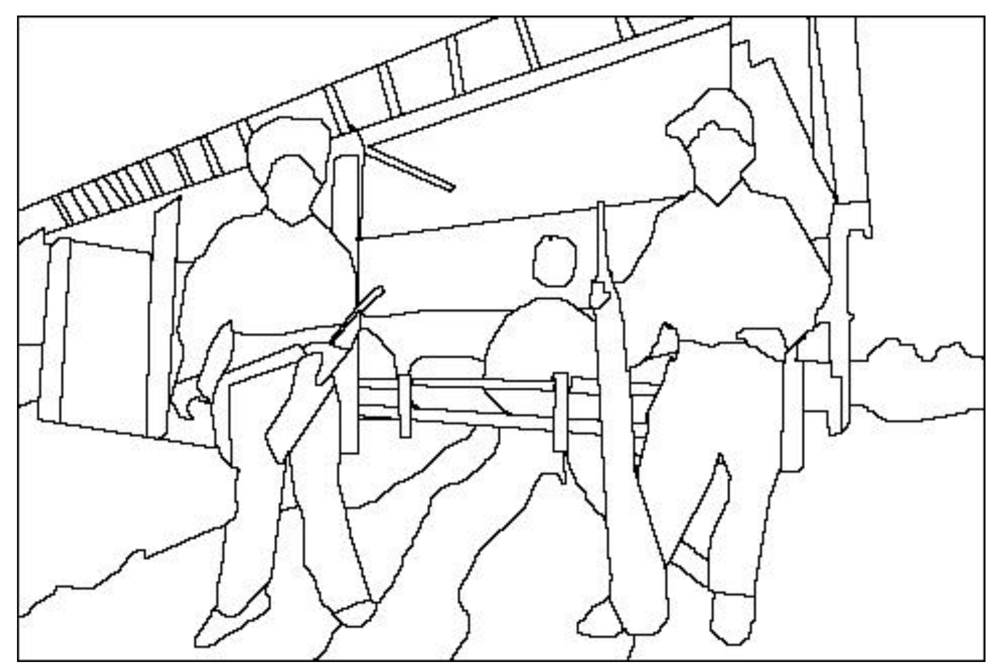
Edge Detection and Texture Analysis

Faisal Qureshi
faisal.qureshi@uoit.ca



Edge Detection





Edge Detection

- Identify sudden changes (discontinuities) in an image

- Most semantic and shape information seen in an image can be encoded using the edges

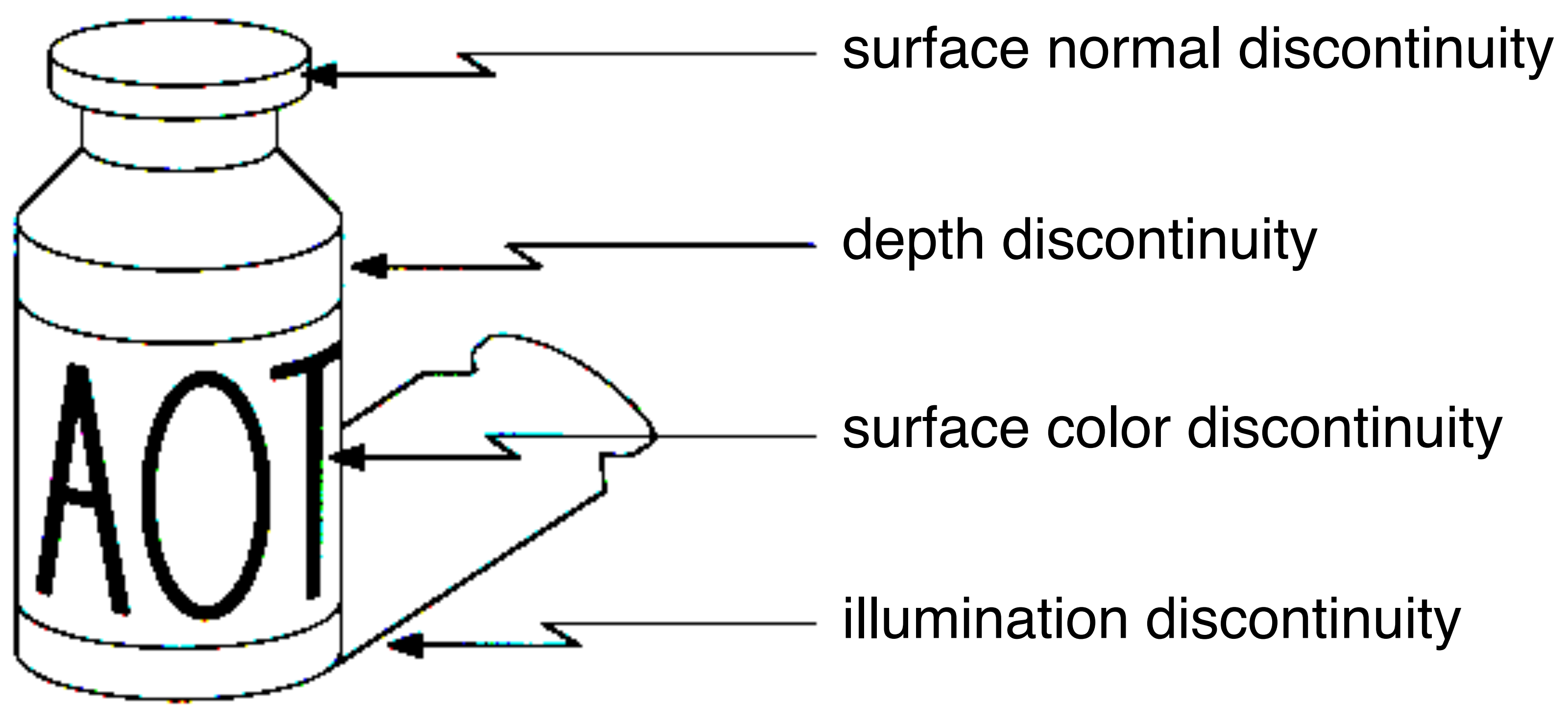
- Edges are more compact than pixels

An artist's line drawing

[Source: D. Lowe]

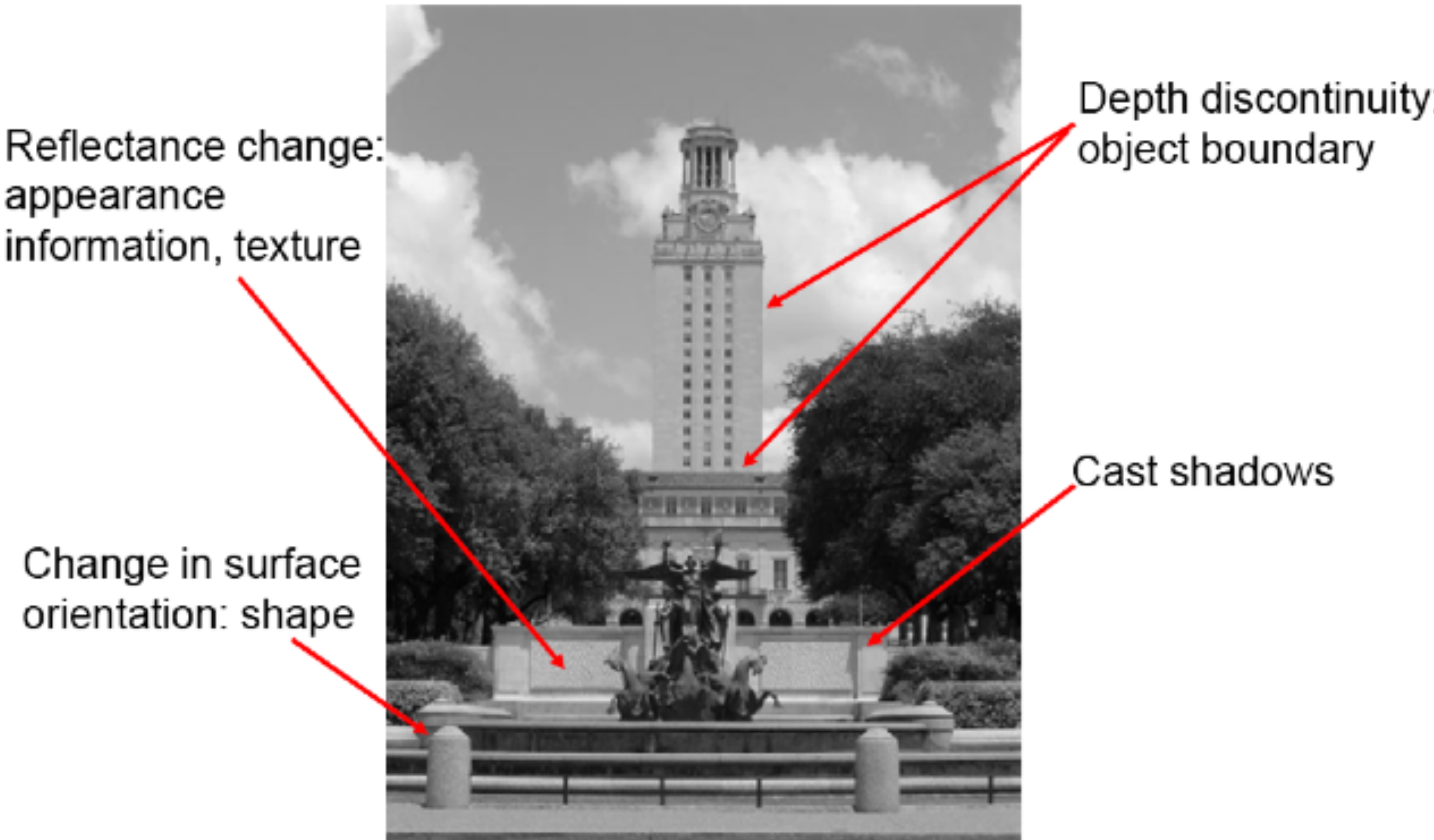
Origin of Edges

- Edges are caused by a variety of factors



[Source: Steve Seitz]

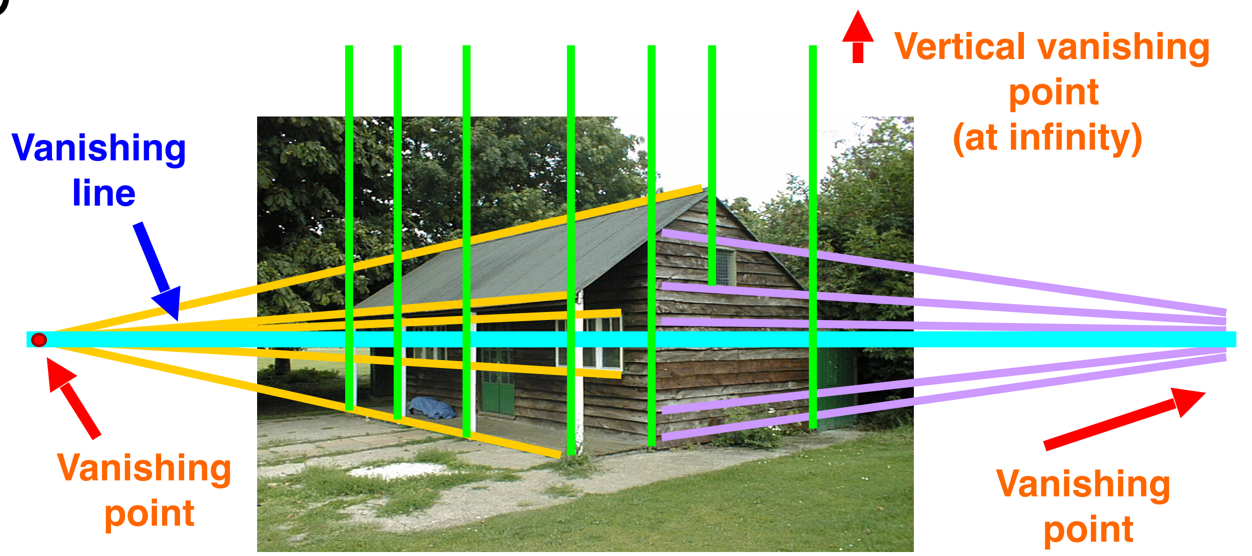
What causes edges?

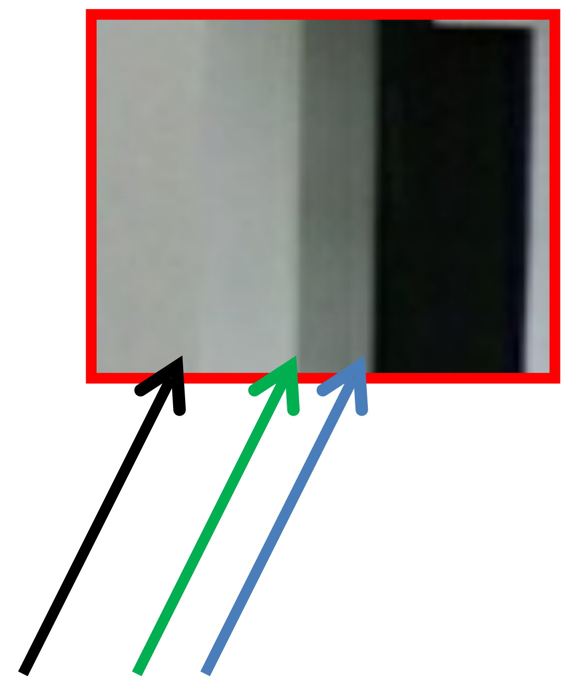


[Source: K. Grauman]

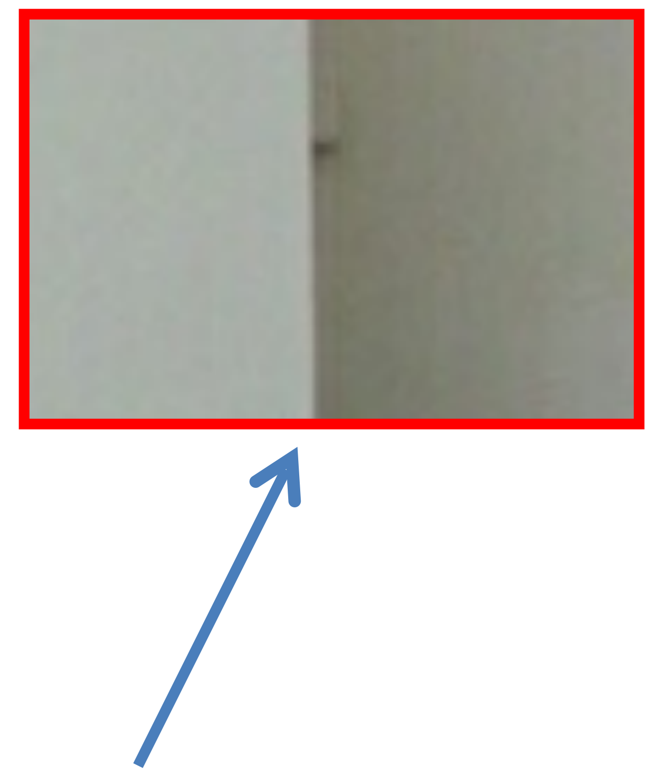
Why edge detection?

- Extract information
- Recognize objects
- Understand scene
- Reconstruct 3D from images
  - Recover viewpoint and geometry





[Source: Steve Seitz]

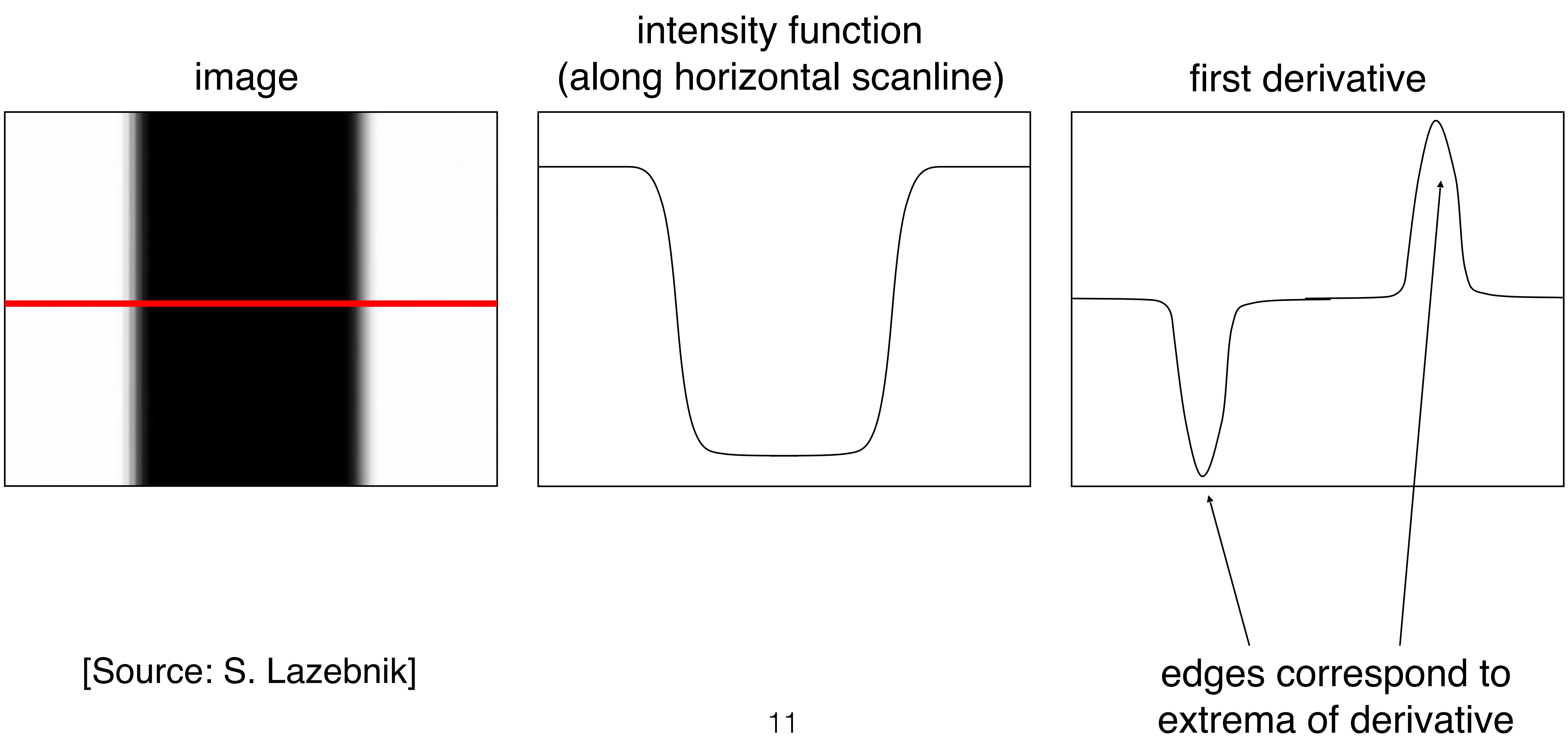


[Source: Steve Seitz]

[Source: Steve Seitz]

Characterizing Edges

- An edge is a place of rapid change in intensity



[Source: S. Lazebnik]

How to compute image derivatives?

- Option 1: reconstruct a continuous function f , then compute partial derivatives as

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

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

[Source: S. Fidler]

How to compute image derivatives?

- Option 2: use finite differences to take a discrete derivative as

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

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

[Source: S. Fidler]

How to compute image derivatives?

- Option 2: use finite differences to take a discrete derivative as

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

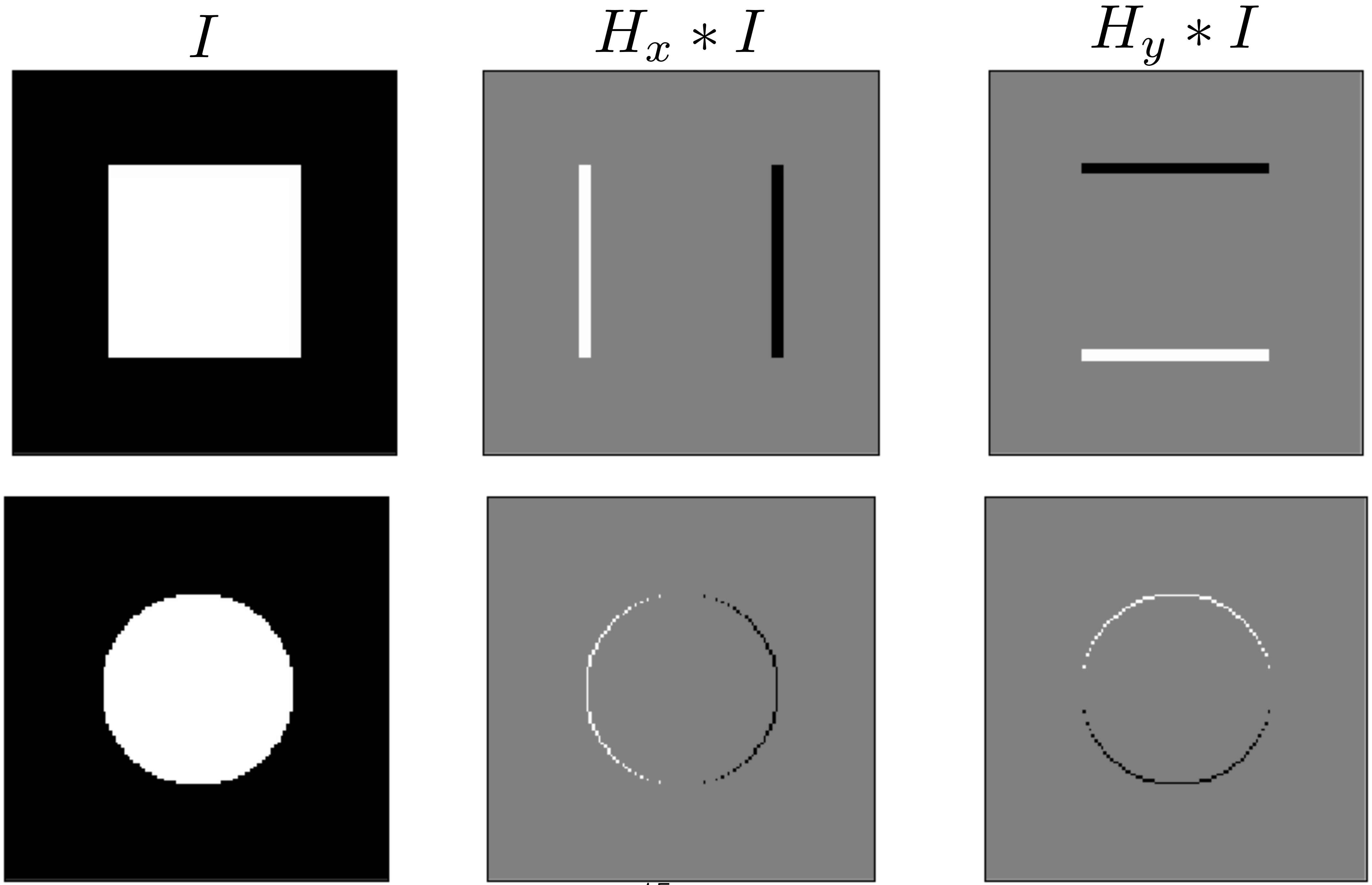
$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f[x,y+1]-f[x,y]}{1}$$

- We can achieve this using *convolution*?

$$H_x = \begin{bmatrix} -1 & 1 \end{bmatrix} \qquad H_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

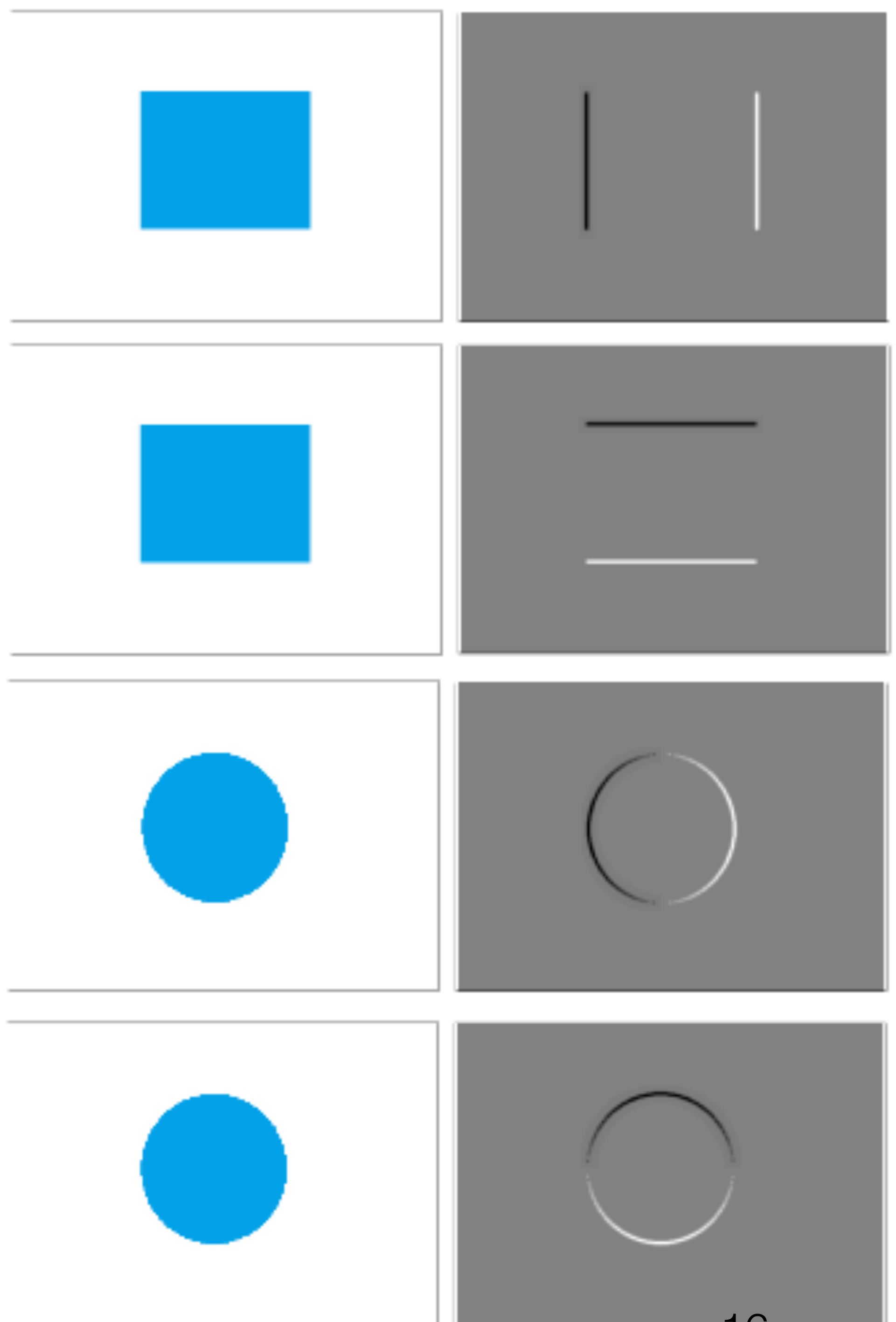
[Source: S. Fidler]

Partial Derivative of an Image



Partial Derivative of an Image

I



$H_x * I$

$H_y * I$

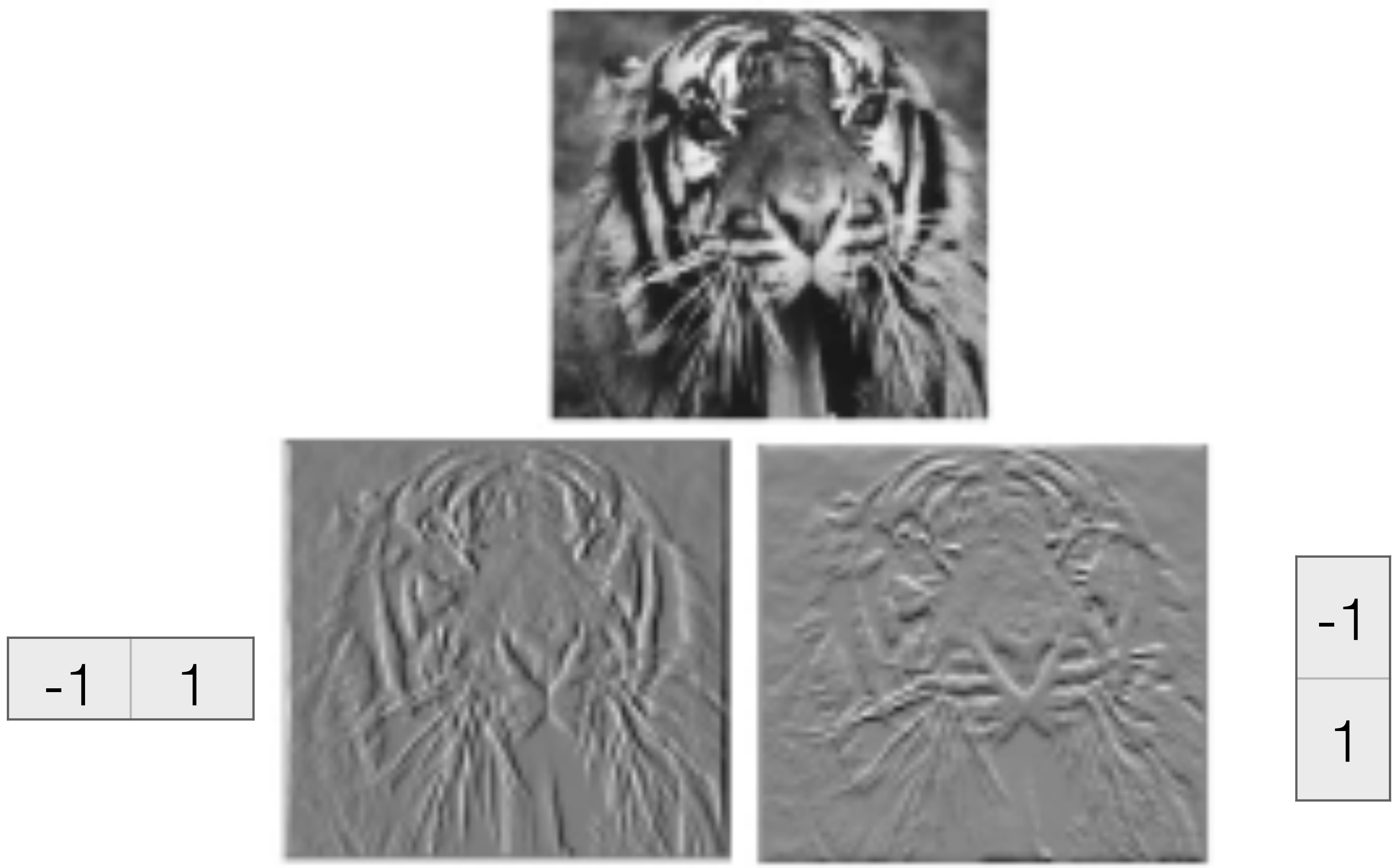
$H_x * I$

$H_y * I$

$$H_x = \boxed{-1 \quad 1}$$

$$H_y = \begin{array}{|c|}\hline -1 \\ \hline 1 \\ \hline \end{array}$$

Partial Derivatives of an Image



[Source: K. Grauman]

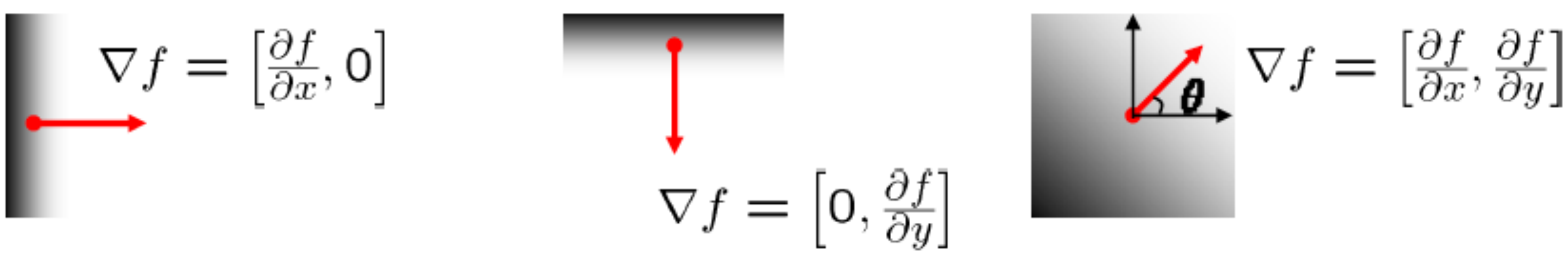
Finite Difference Filters

	H_x	H_y
• Prewire	$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$	$\begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$
• Sobel	$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$	$\begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$
• Roberts	$\begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}$	$\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$

Image Gradient

- The gradient of an image ∇f points to the direction of most rapid change in intensity

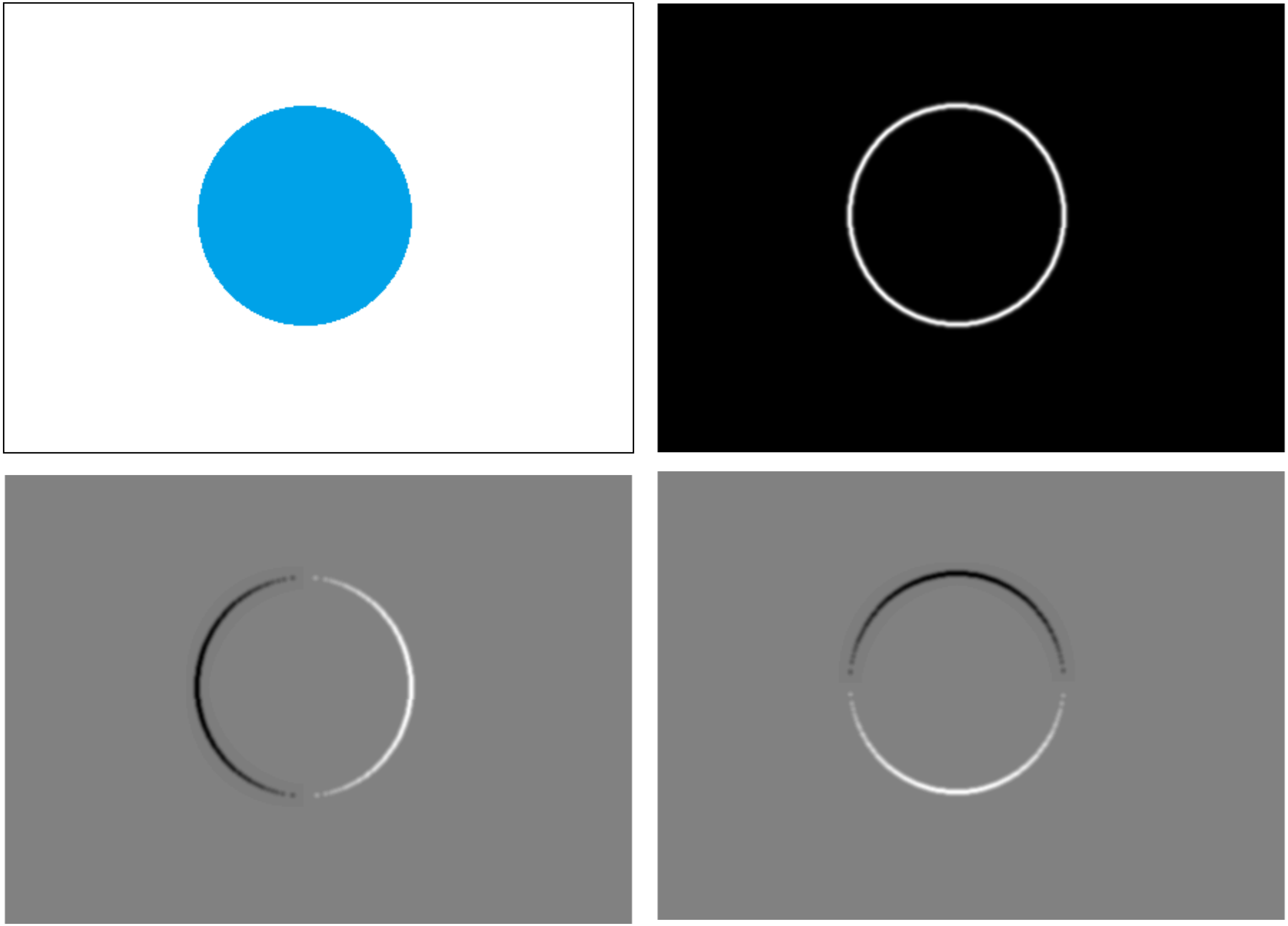
- Image gradient $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

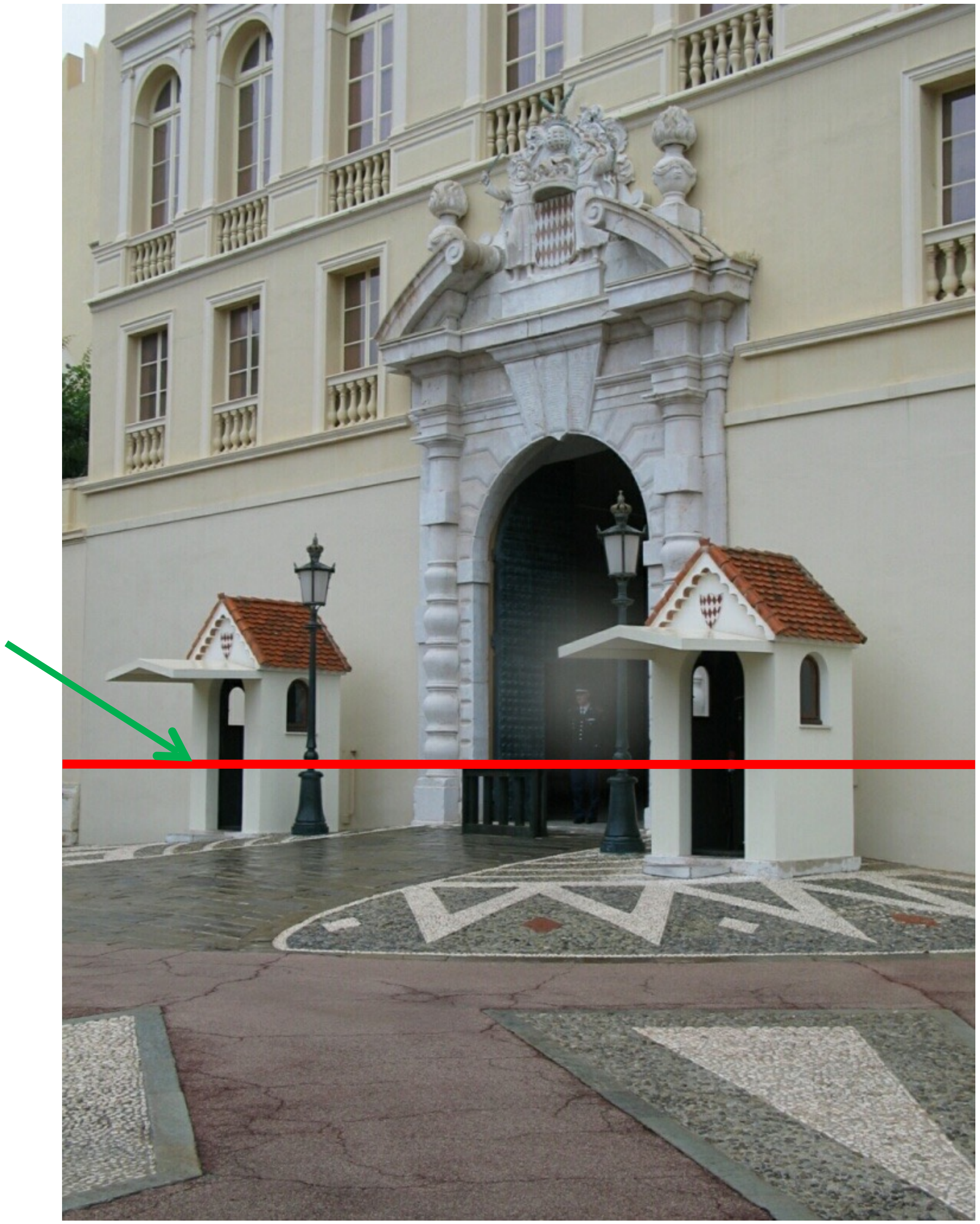


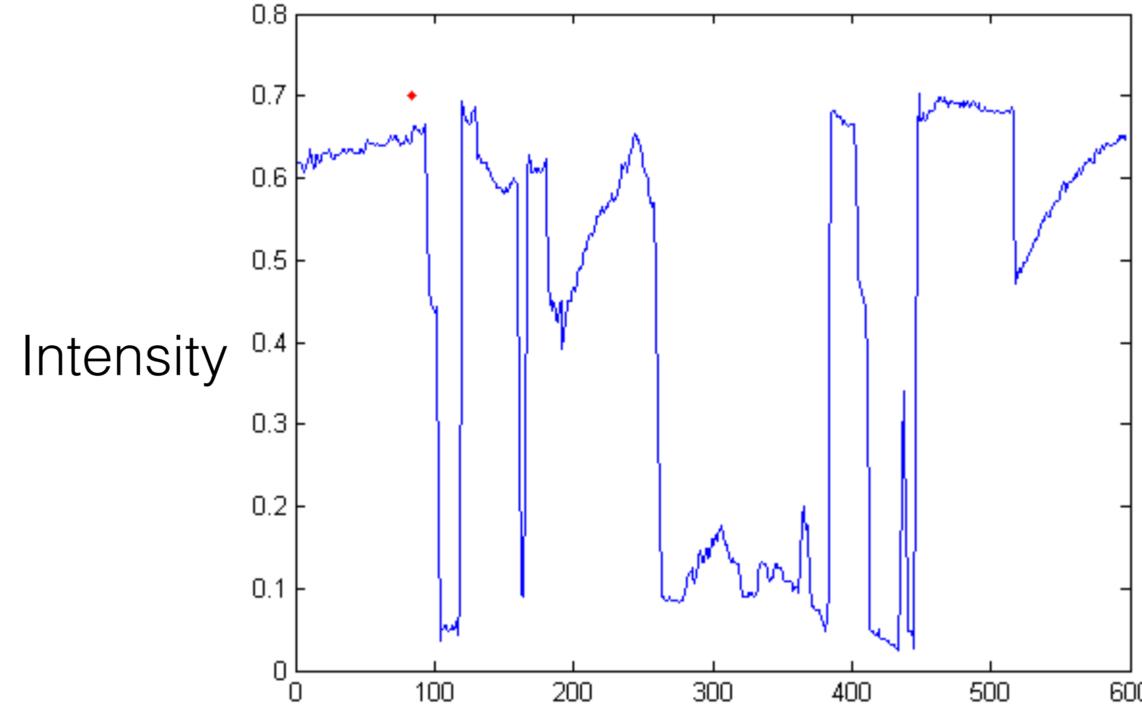
- Gradient magnitude $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

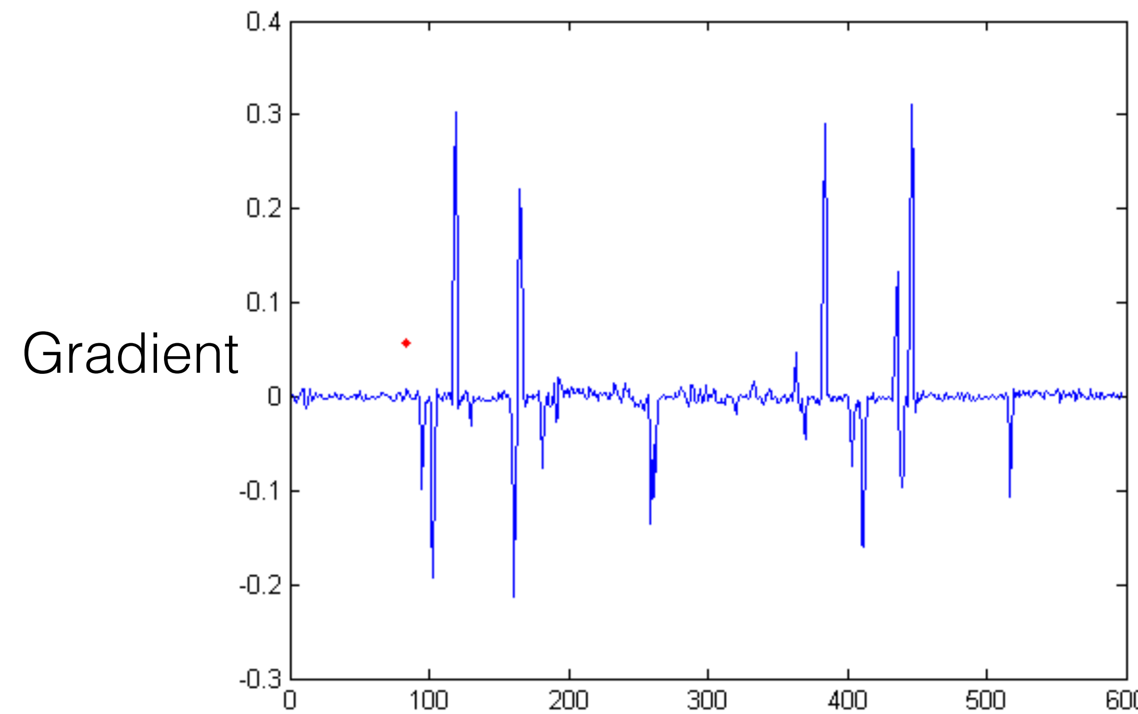
- Gradient direction $\theta = \tan^{-1}\left(\frac{\partial f}{\partial x} / \frac{\partial f}{\partial y}\right)$

[Source: S. Seitz]



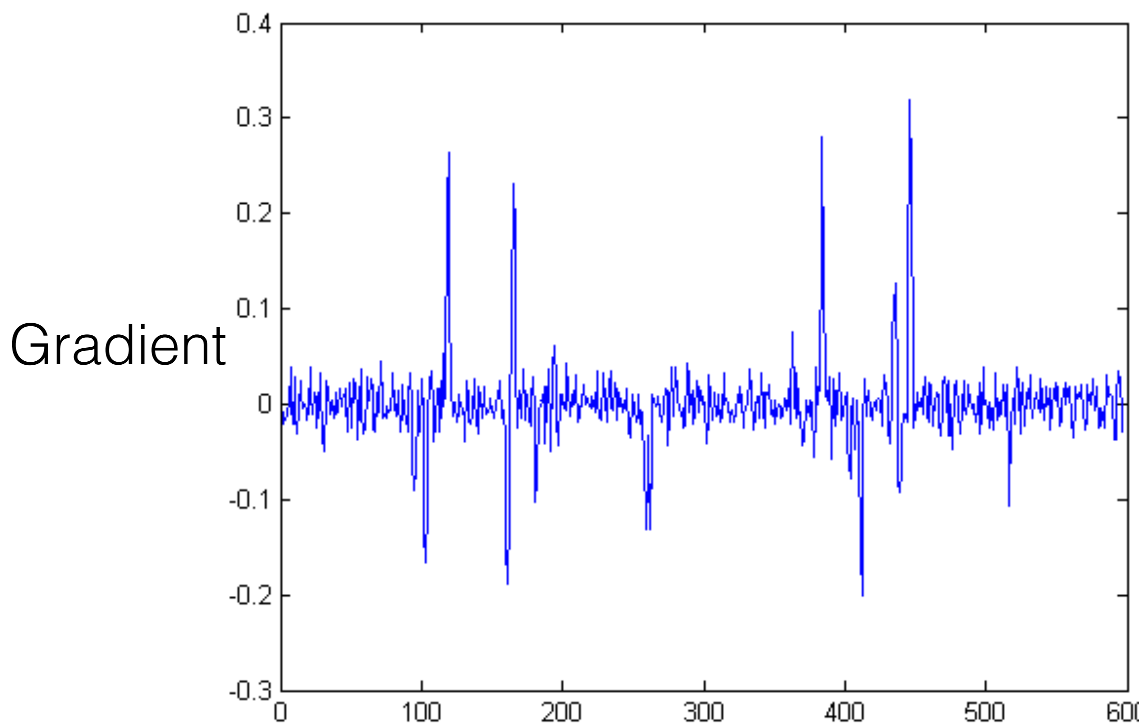






[Source: D. Hoiem]

Gradient when a little Gaussian noise is added to the image

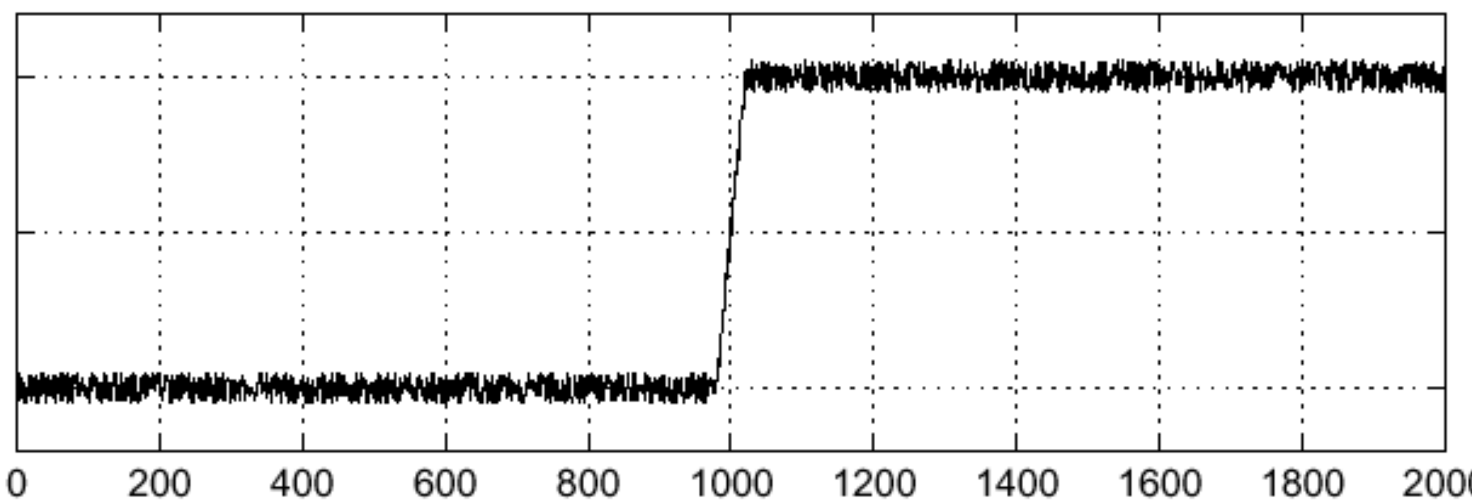


[Source: D. Hoiem]

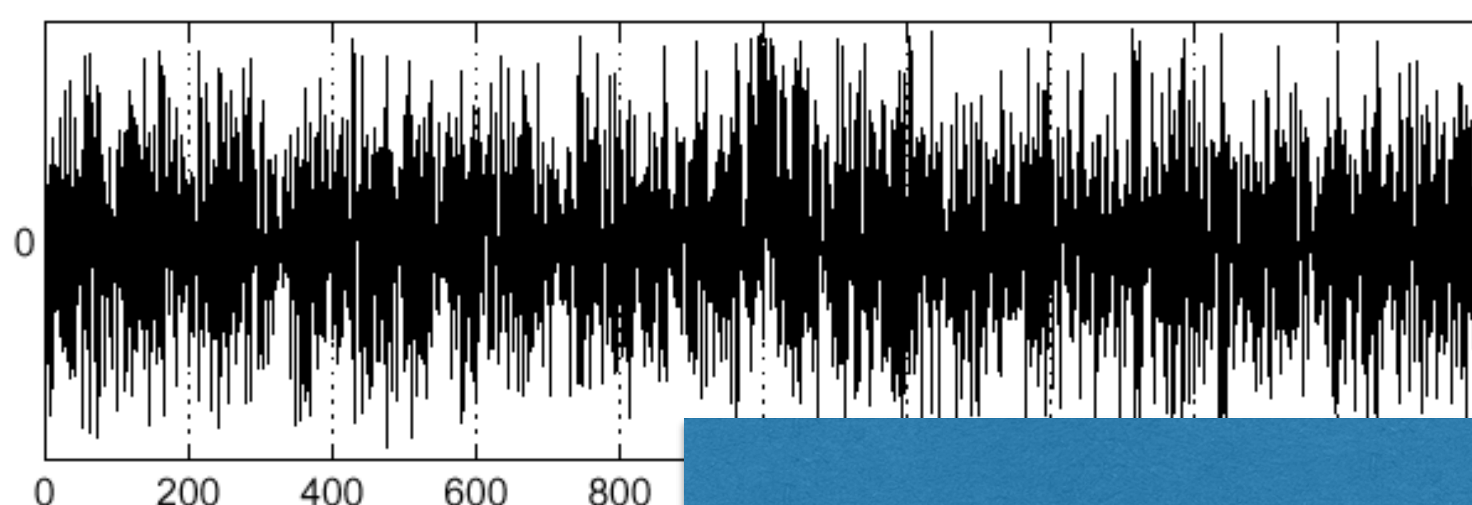
Effects of Noise

- Gradient is highly sensitive to noise

$f(x)$



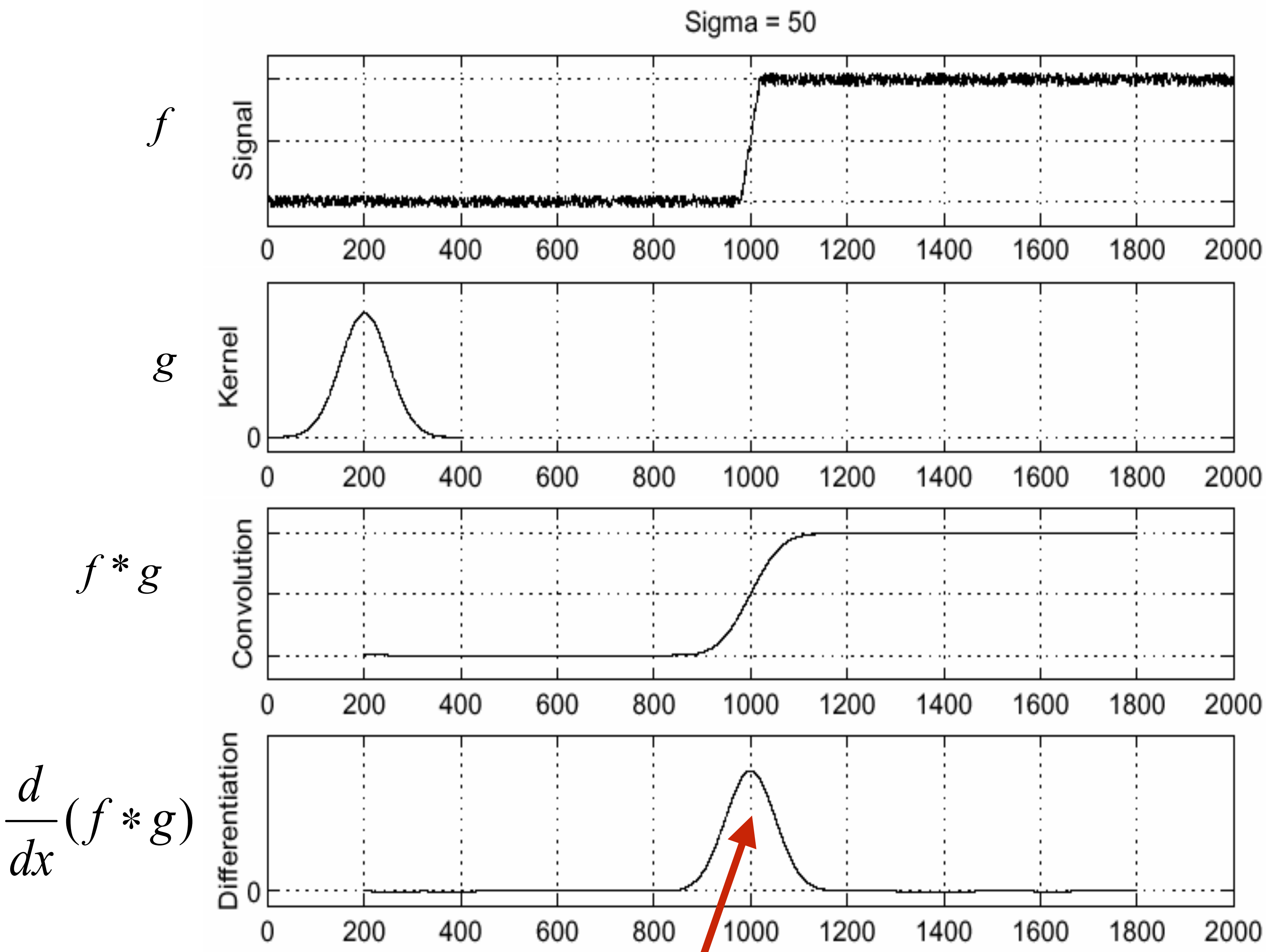
$\frac{d}{dx}f(x)$



Where is the edge?

Effects of Noise

- Gradient is highly sensitive to noise
- Difference filters (that we can use for gradient computation) respond strongly to noise
  - The larger the noise, the stronger the response
- How to handle it?
  - Smooth first. Get rid of high-frequency component.



Peaks in $\frac{d}{dx}(f * g)$ correspond to edges

[Source: S. Seitz]

Differentiation via Convolution

- Convolution is associative

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

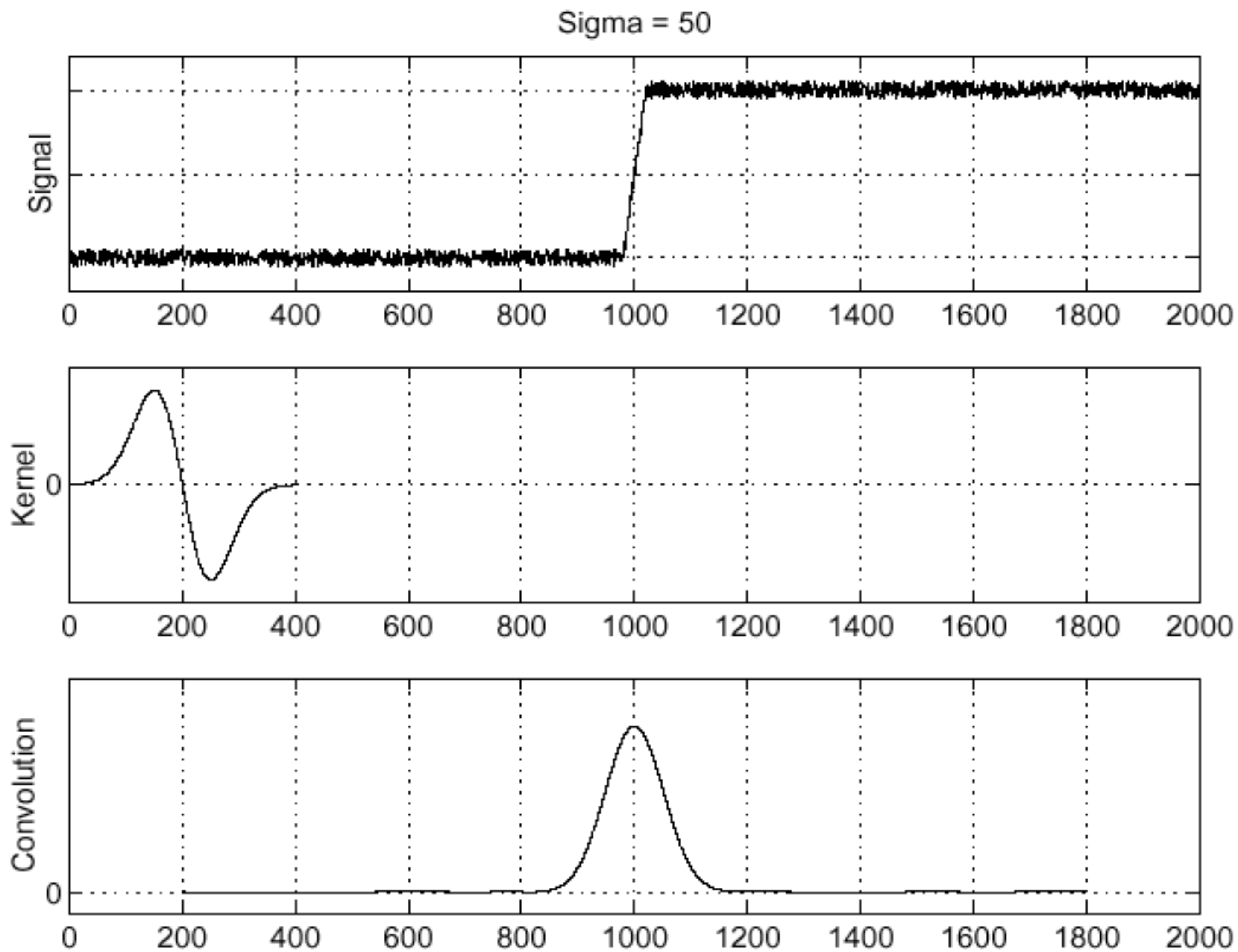
- This saves us one convolution

Differentiation via Convolution

f

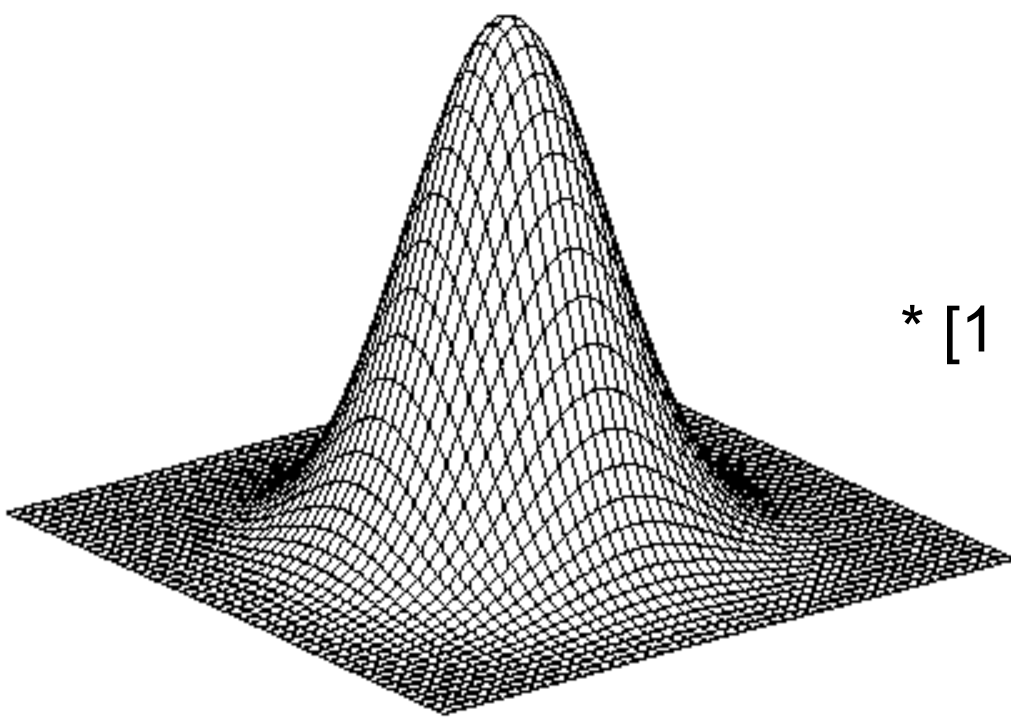
$\frac{d}{dx} g$

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



[Source: S. Seitz]

Derivative of Gaussian filter



* [1 -1] =

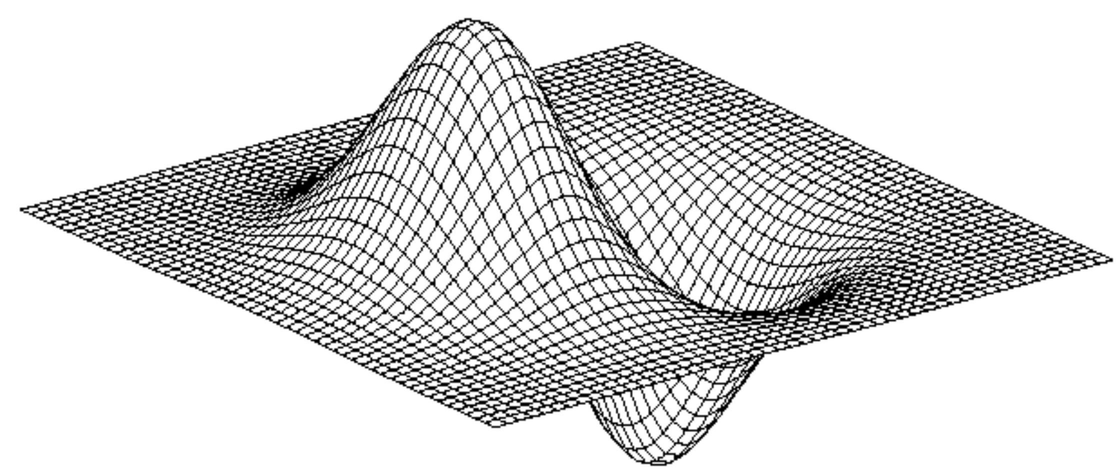
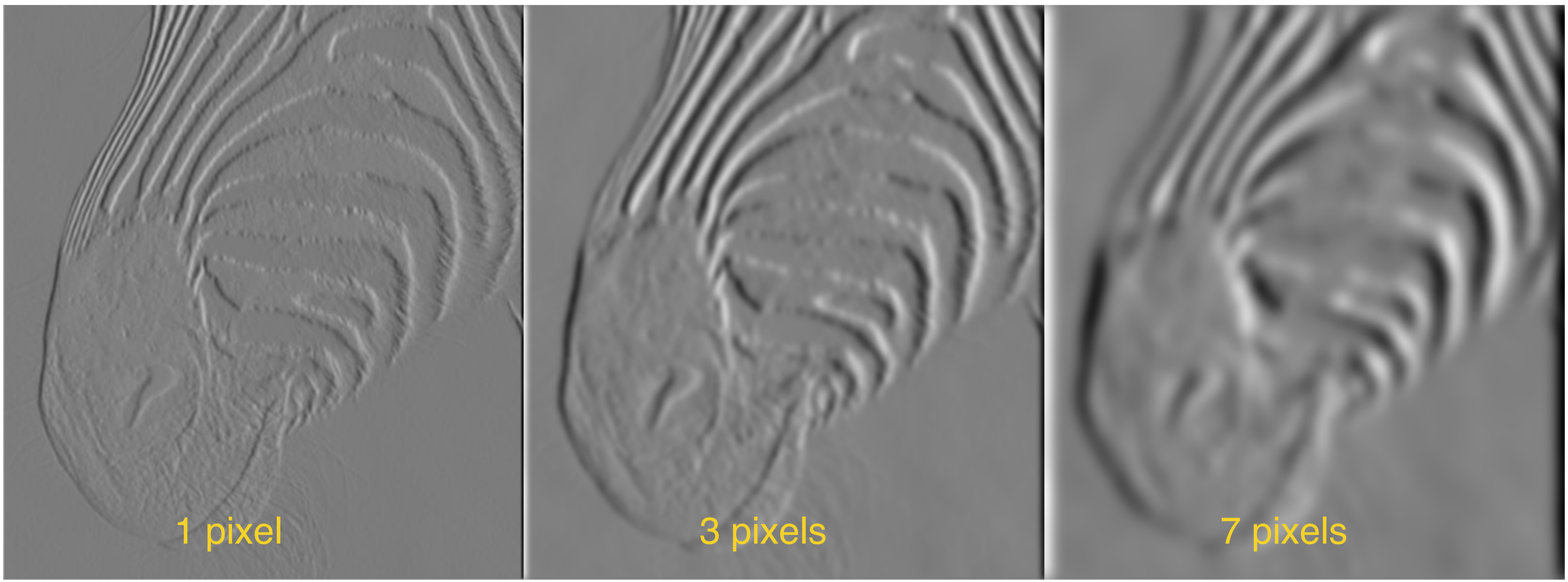


Image Smoothing & Edge Localization

- Smoothed derivative removes noise, but blurs the edges, making edge localization challenging



[Source: D. Forsyth]

Implementation Issues

- 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?

[Source: D. Forsyth]

Designing an edge detector

- Detection: an edge detector should find all “real” edges, ignoring noise and other artifacts
- Localization: the detected edges must be as close to the “real” edges as possible
- The following cues can be leveraged when detecting edges:
  - Differences in intensity, color and texture
  - Continuity and closure
  - High-level knowledge

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

[Source: L. Fei-Fei]

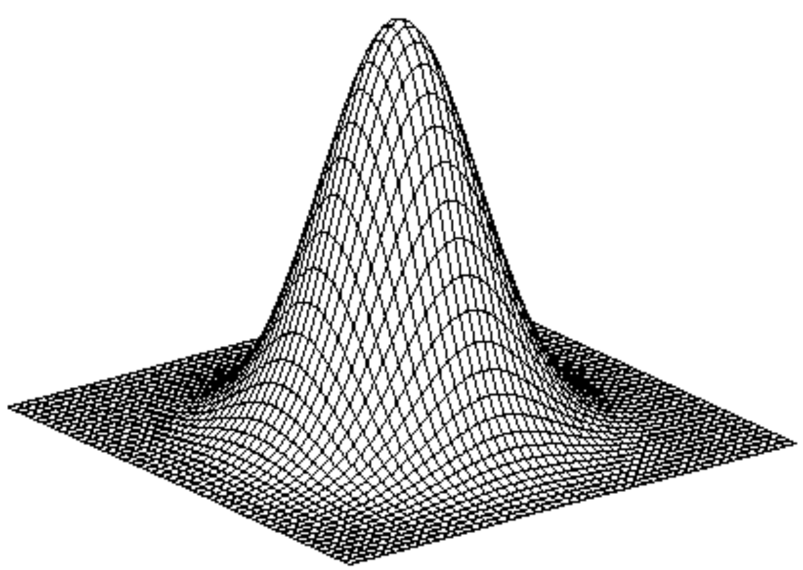
Canny Edge Detector

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

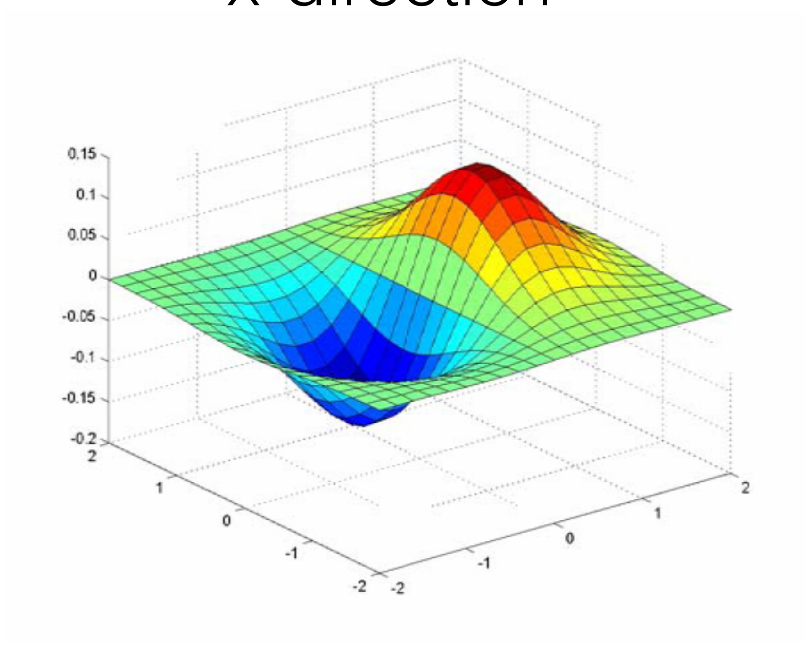
- The most widely used edge detector in computer vision

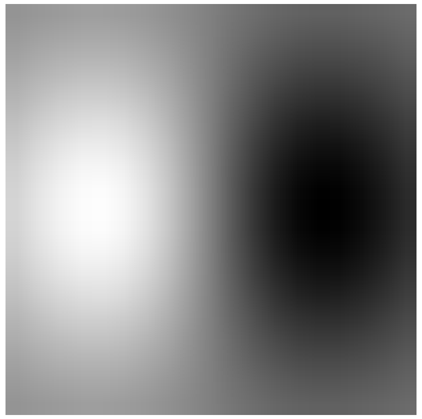
Derivatives of a Gaussian Filter

Gaussian

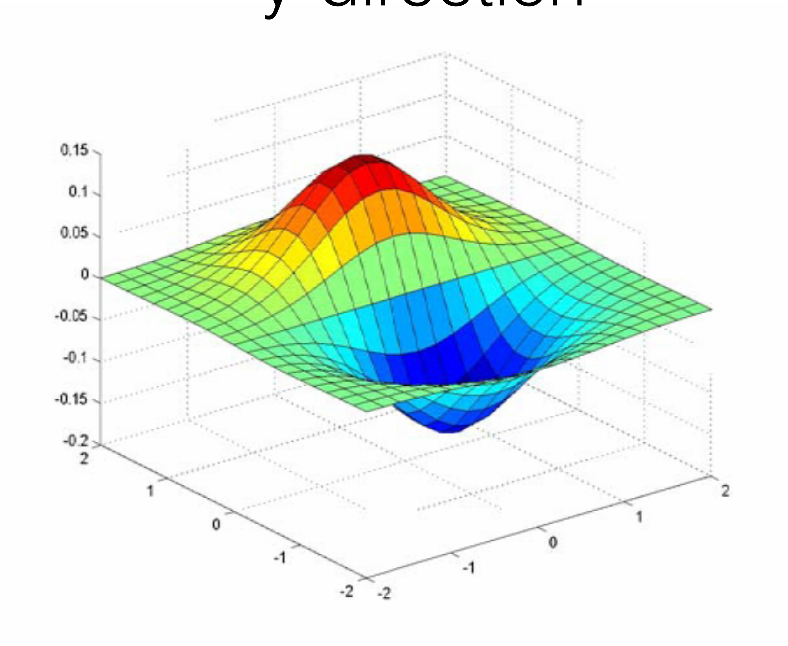


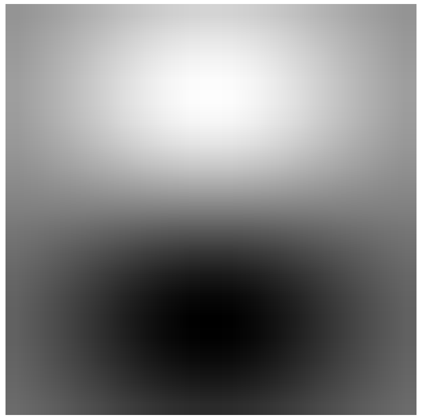
x-direction





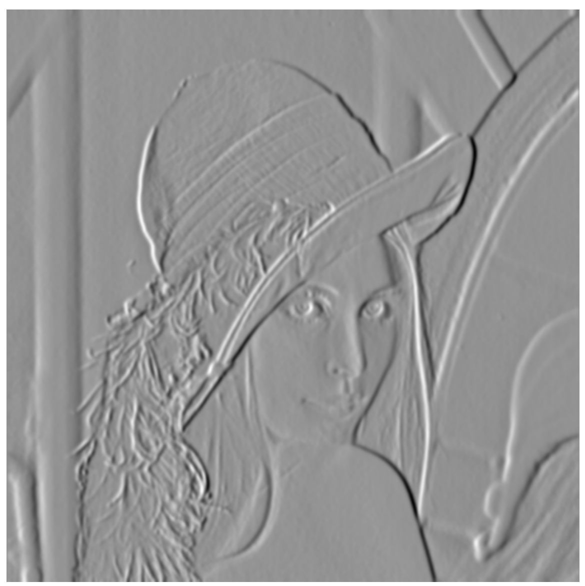
y-direction





Canny Edge Detector

- Compute image gradient in x and y directions



I_x

x-derivative of Gaussian

I_y

y-derivative of Gaussian

$\sqrt{I_x^2 + I_y^2}$

gradient magnitude

- We can use Difference-of-Gaussians for this purpose

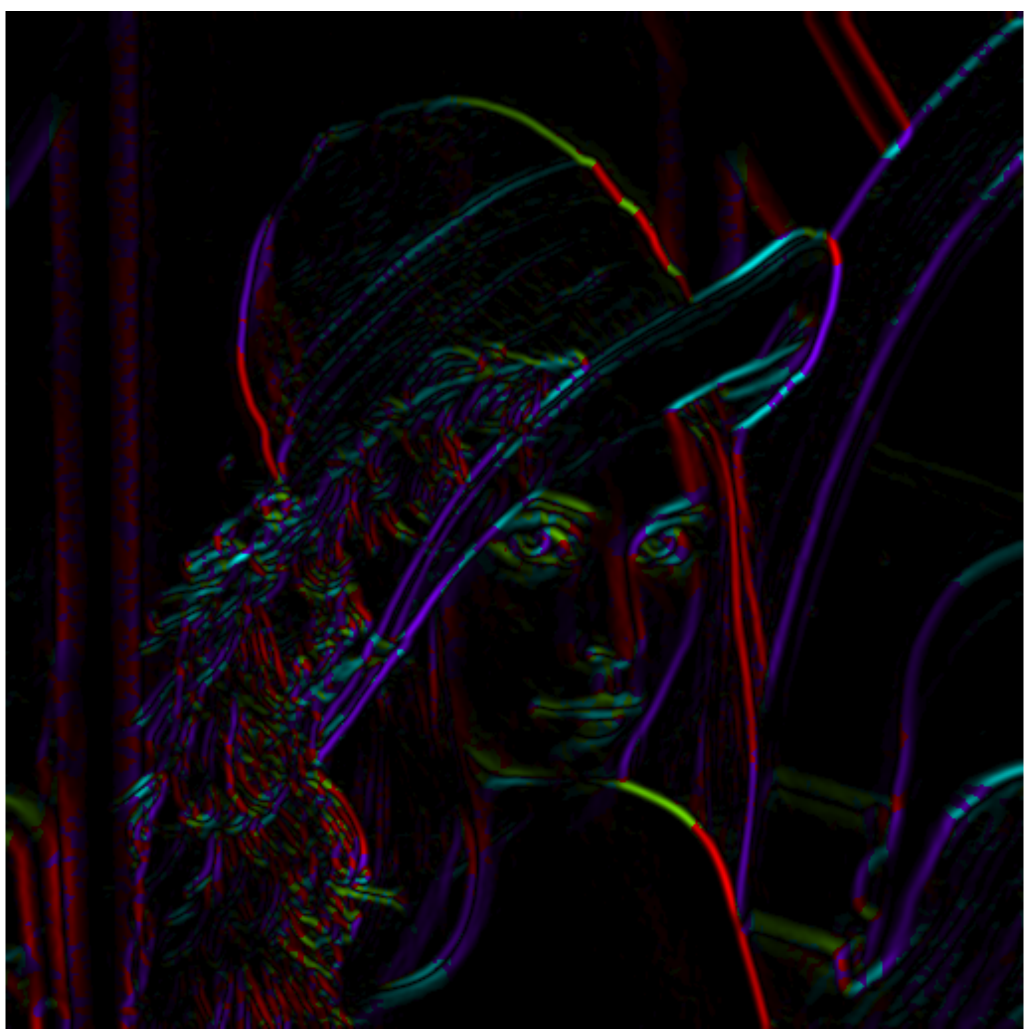
Difference of Gaussian

- DoG can be used to approximate smoothed gradient of an image

$$\frac{\partial (G_\sigma * I)}{\partial x} \approx (G_{\sigma_1} * I) - (G_{\sigma_2} * I)$$

Canny Edge Detector

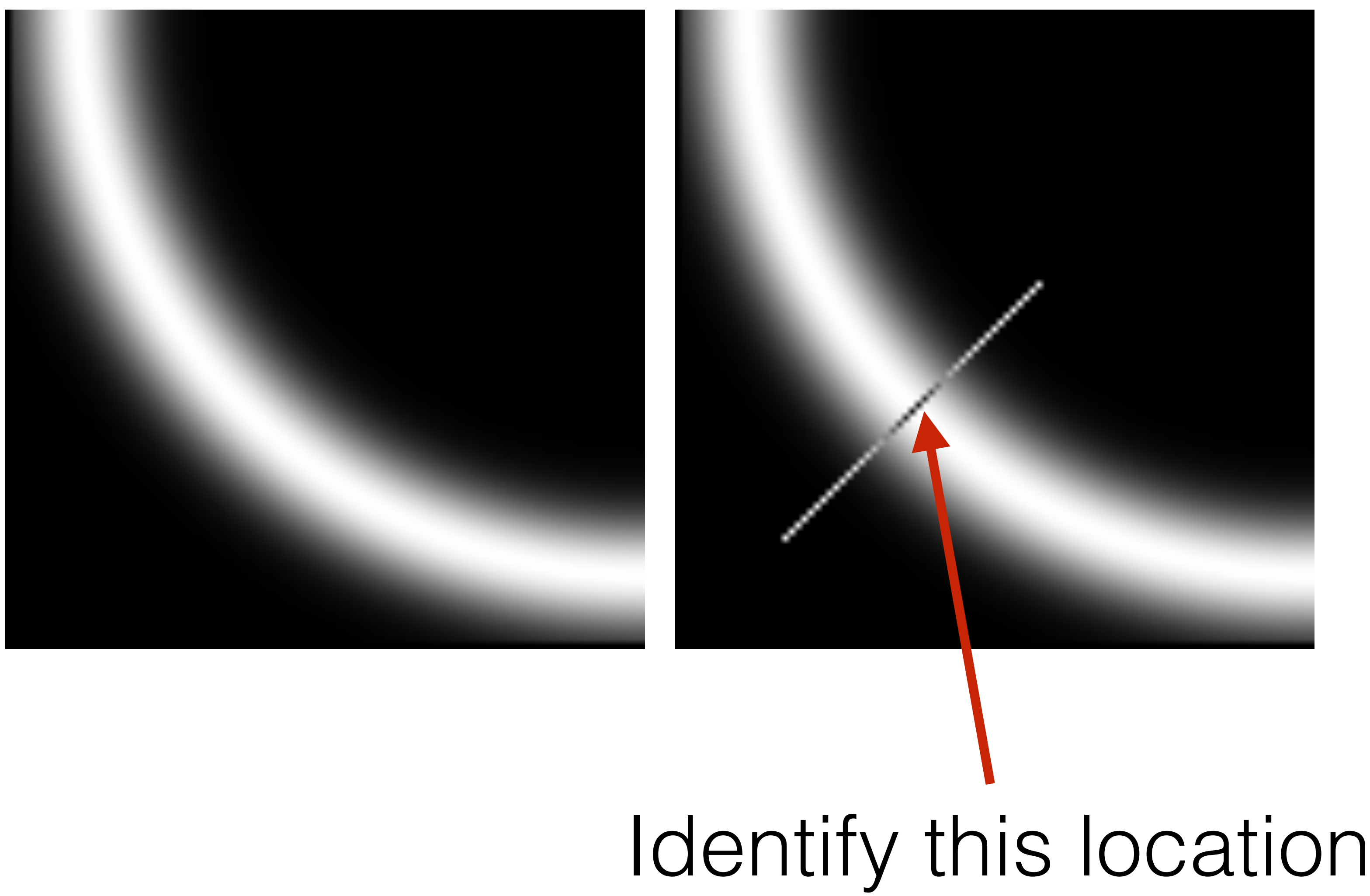
- Threshold pixels (on gradient magnitude)
- Compute edge orientation $\theta = \arctan 2(\nabla I_x, \nabla I_y)$



[Source: J. Hays]

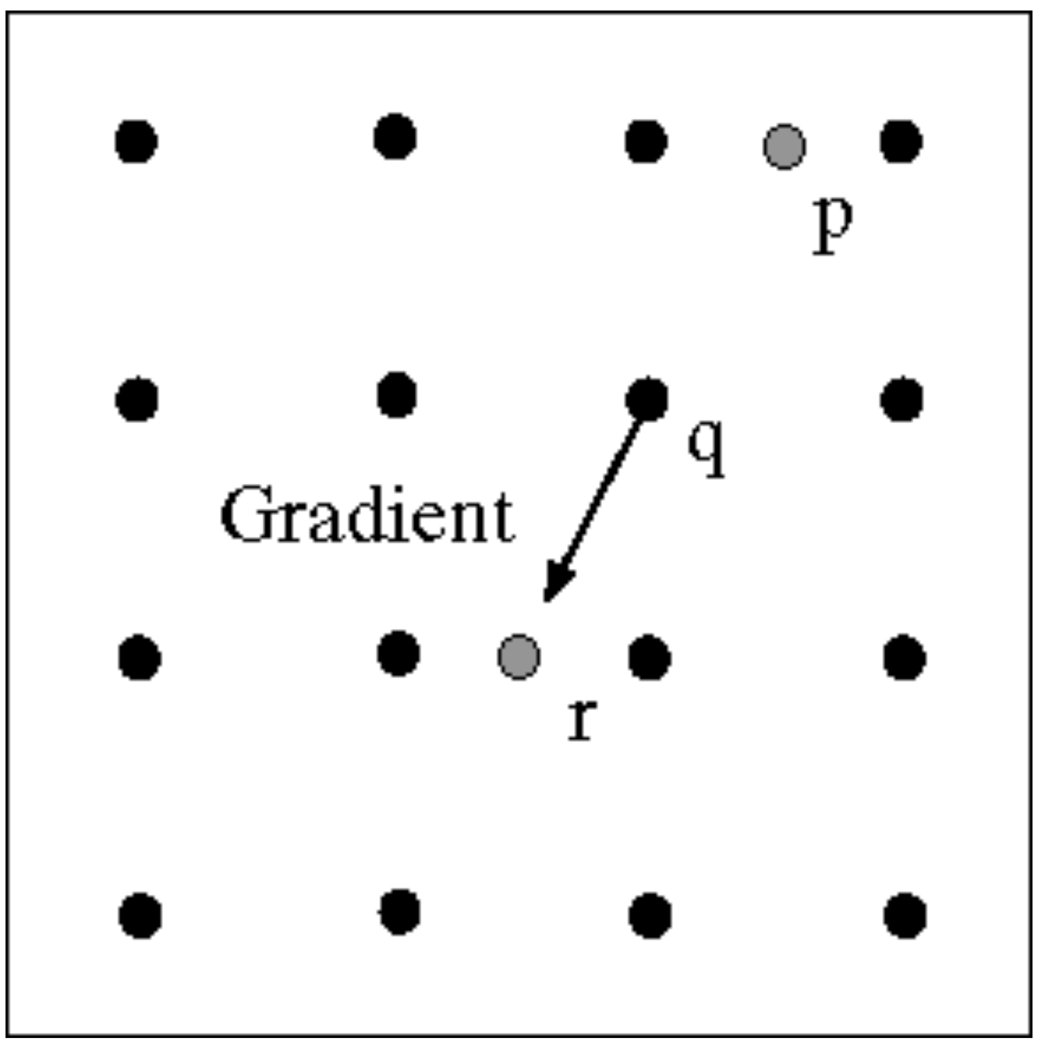
Canny Edge Detector

- Edge localization via non-maximum suppression for each orientation



Canny Edge Detector

- Edge localization via non-maximum suppression for each orientation

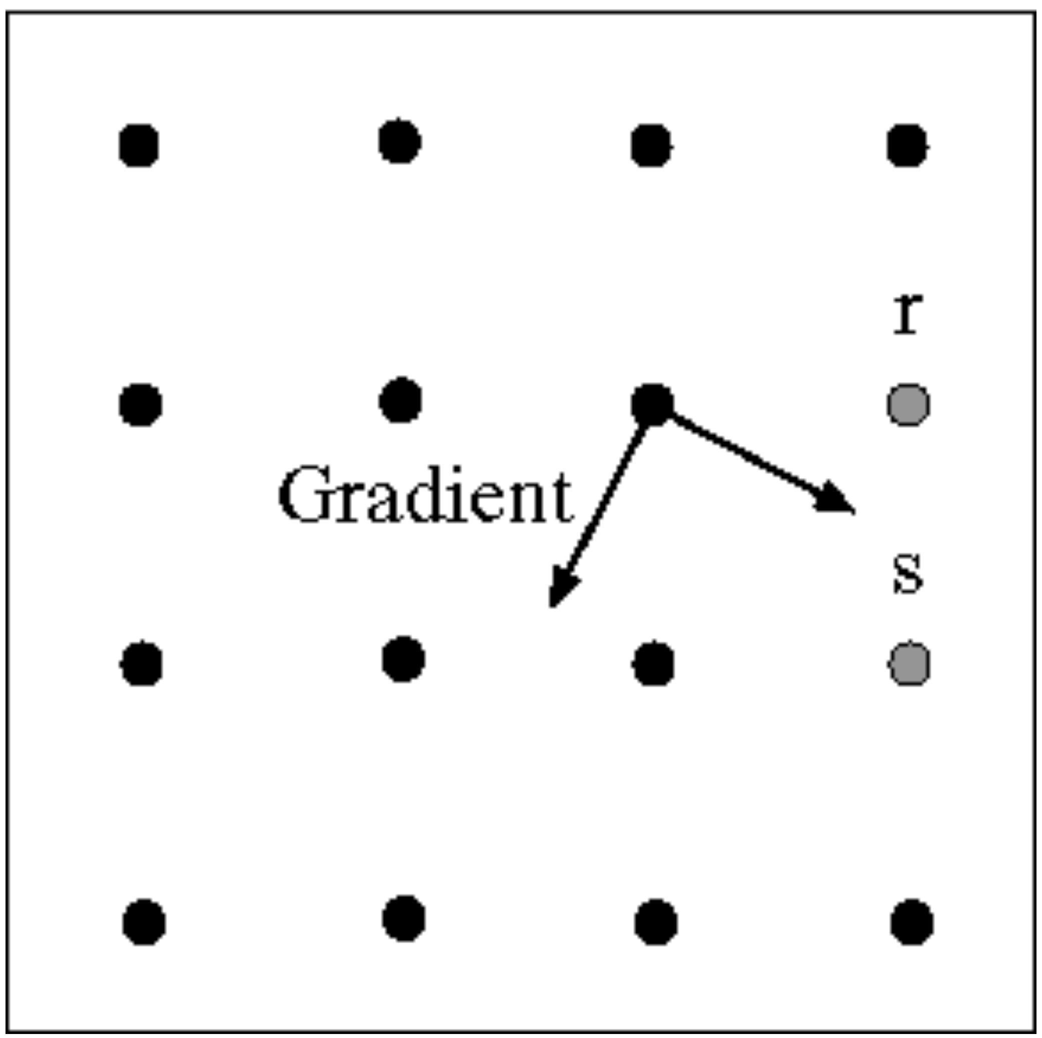


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

[Source: D. Forsyth]

Canny Edge Detector

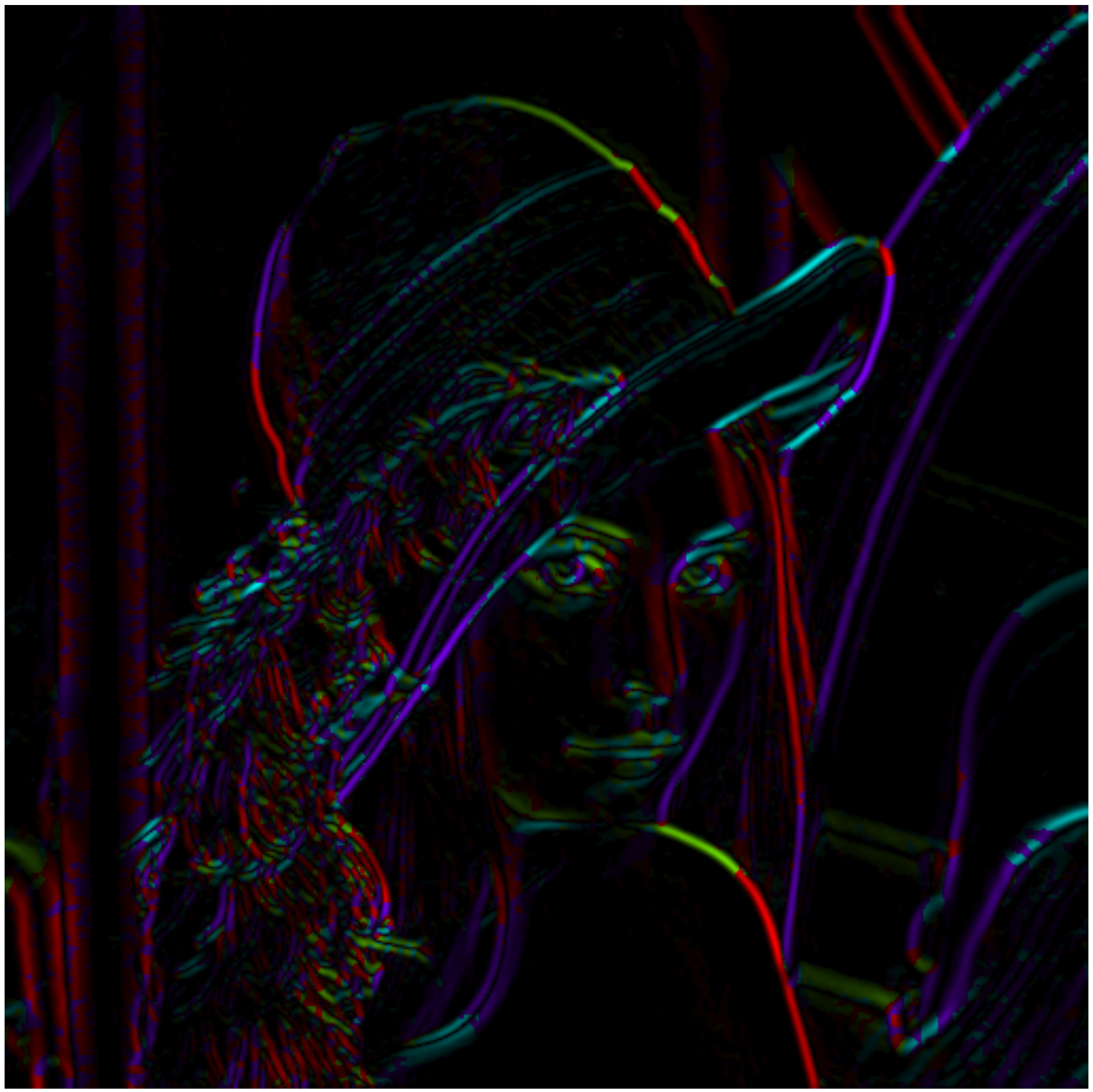
- Edge linking



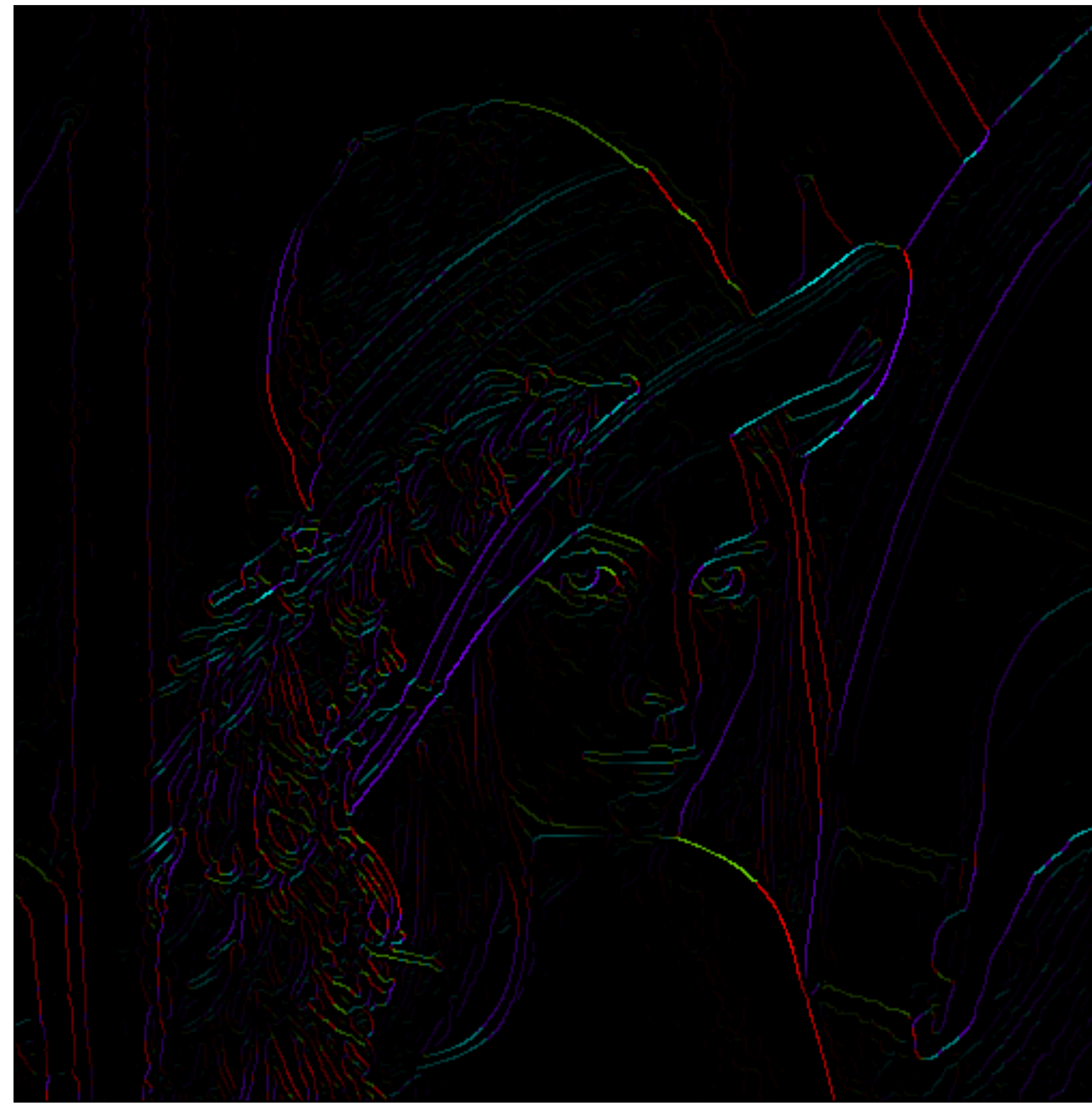
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

[Source: D. Forsyth]

Before and After Non-Maxima Suppression



Before



After

Before and After Non-Maxima Suppression

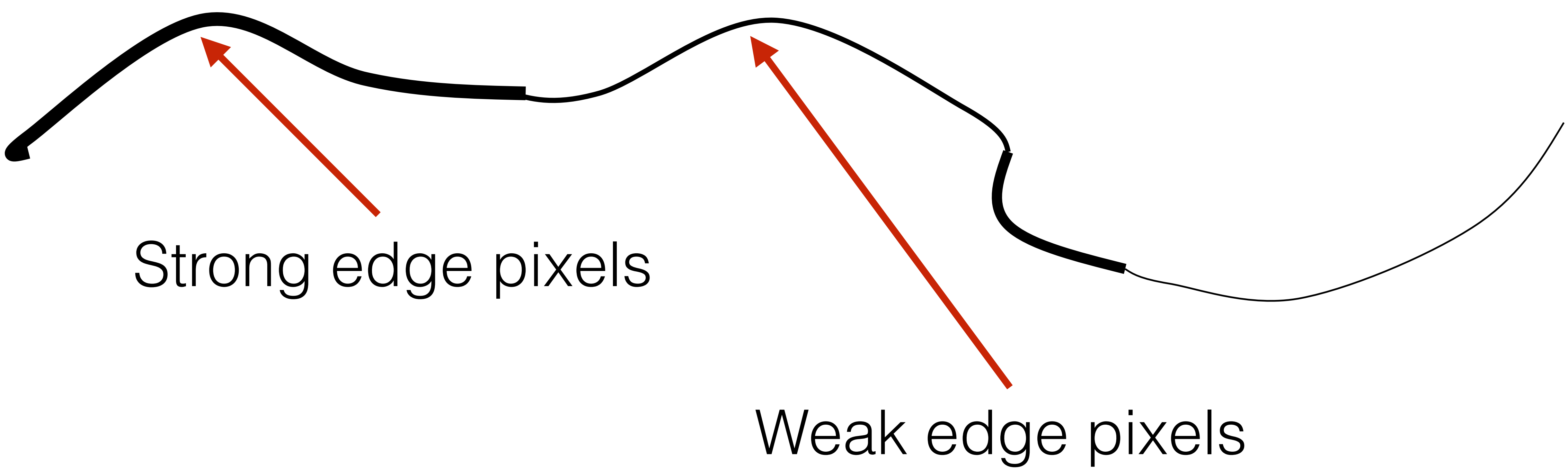
Before

After

Canny Edge Detection

- Hysteresis thresholding (on gradient magnitude values)

  - Use a high threshold to start an edge, low threshold to continue and edge

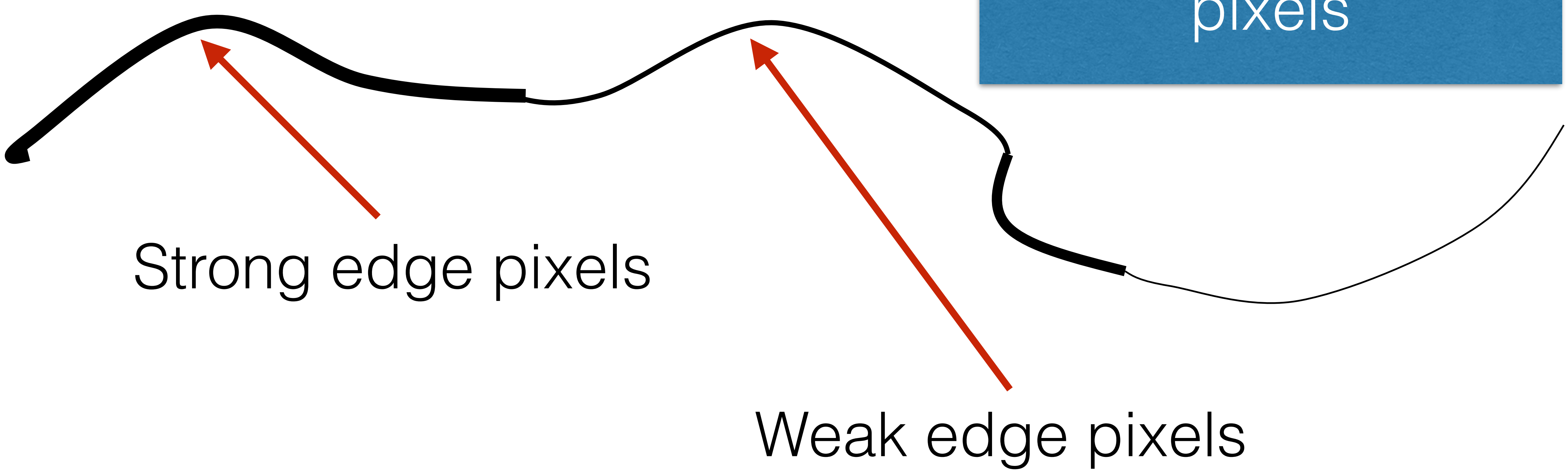


[Source: S. Seitz]

Canny Edge Detection

- Hysteresis thresholding (on gradient magnitude values)
  - Use a high threshold to start ar
    threshold to continue and edge

Do connected components starting from strong edge pixels



[Source: S. Seitz]

Canny Edge Detection

Before hysteresis thresholding

After hysteresis thresholding

Canny Edge Detector Recipe

- Filter image with x and y Gaussian derivatives
- Compute gradient magnitude and orientation at each pixel
- Perform non-maxima suppression: reduce thick multi-pixel wide ridges to thin single pixel lines (contours)
- Hysteresis thresholding and linking: use high-threshold to start an edge and a low-threshold to continue

[Source: D. Lowe, L. Fei-Fei]

Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

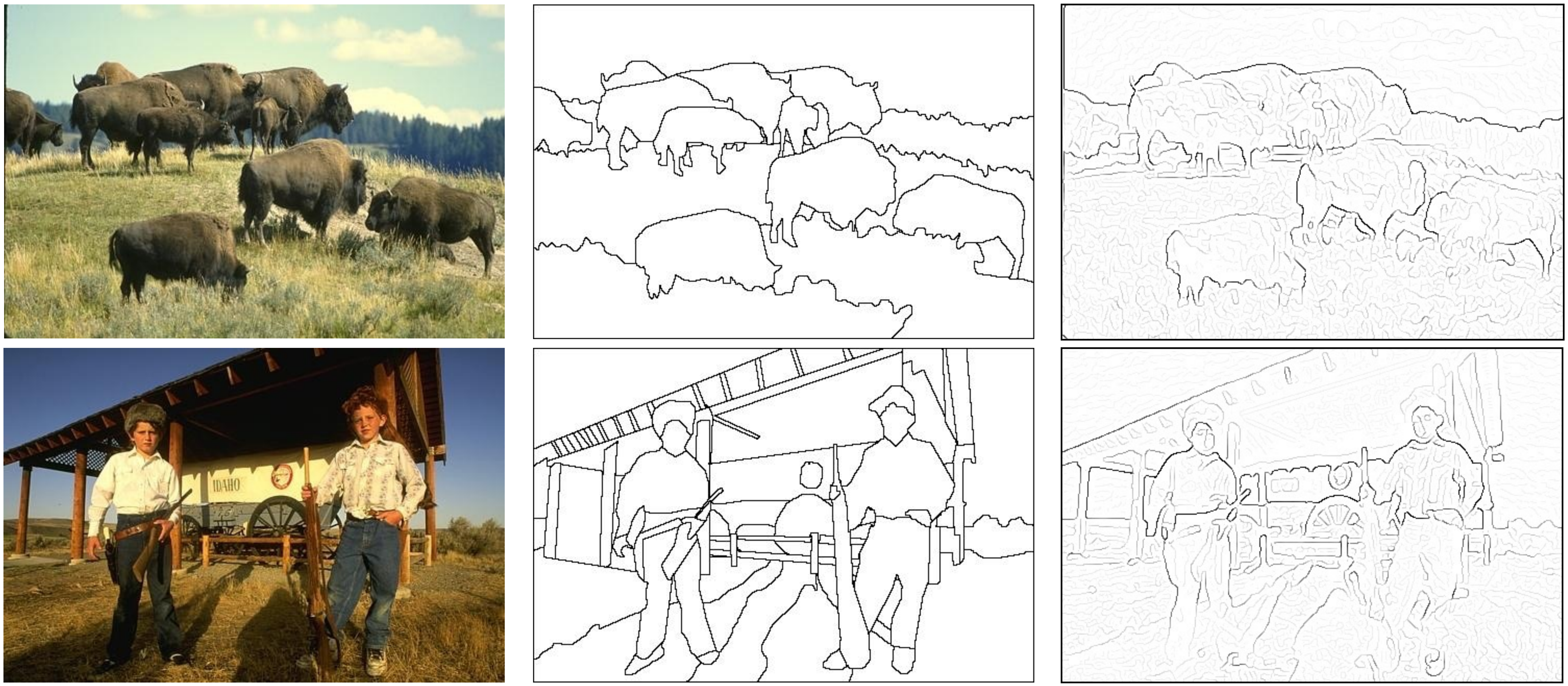
The choice of σ depends on desired behavior

- 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/

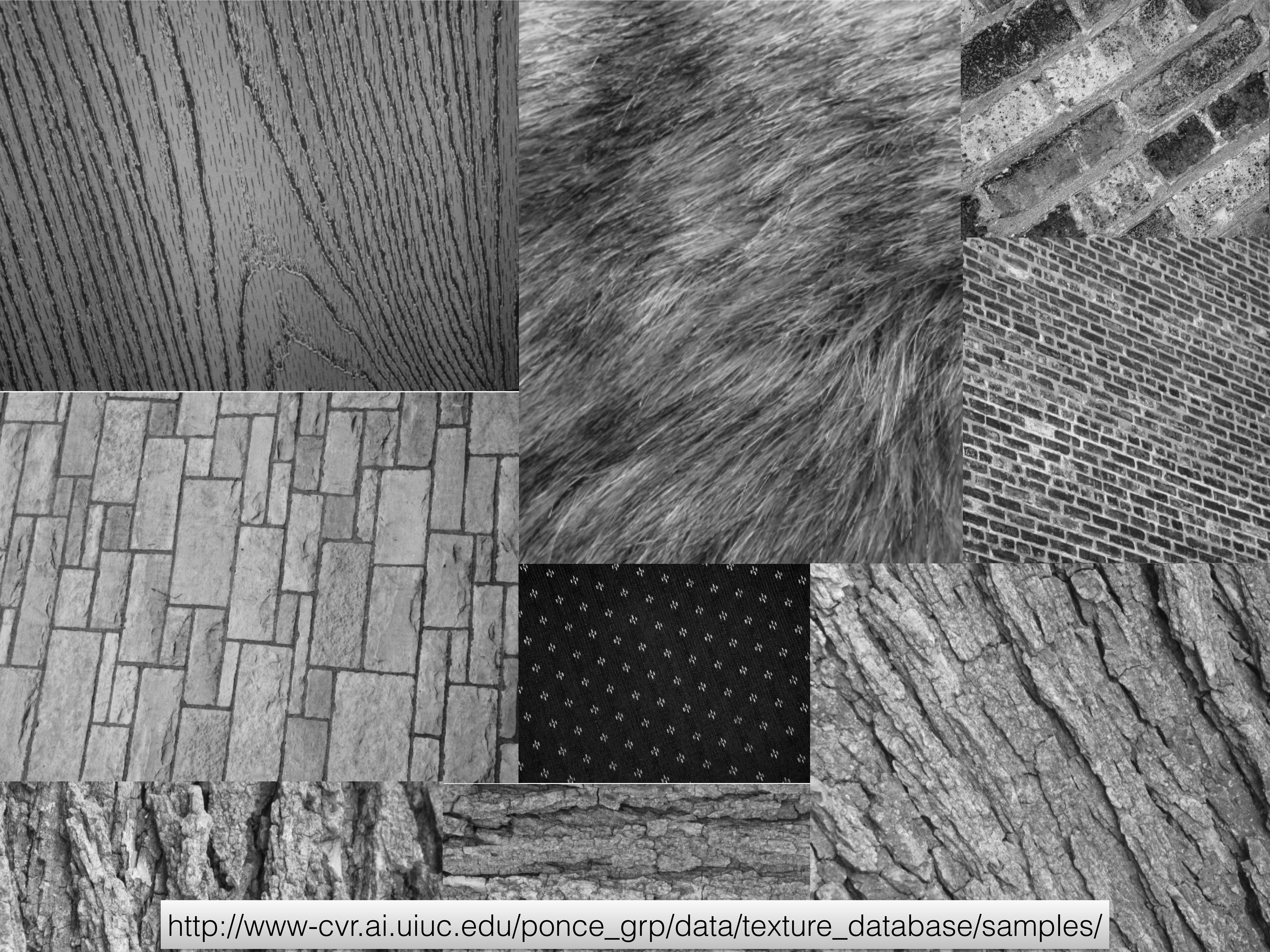
[Source: James Hays]

Edge Detection — Summary

- Canny edge detector
- Effect of noise on edge detector
- Linear filters and edge detection

Texture

[Source: Forsyth]



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


http://www.ultimateupcycle.com/

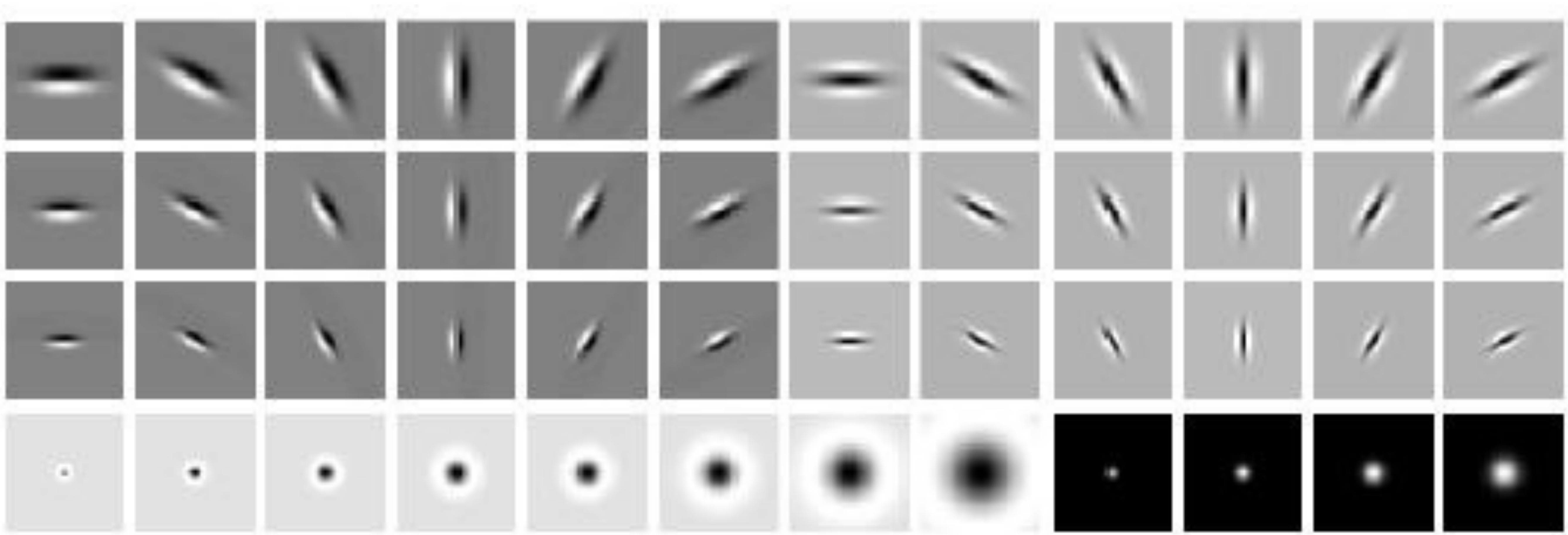
Texture Representation

- Textures are regular or stochastic pattern caused by bumps, grooves, color changes

- How do we represent texture?

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

Filter Banks

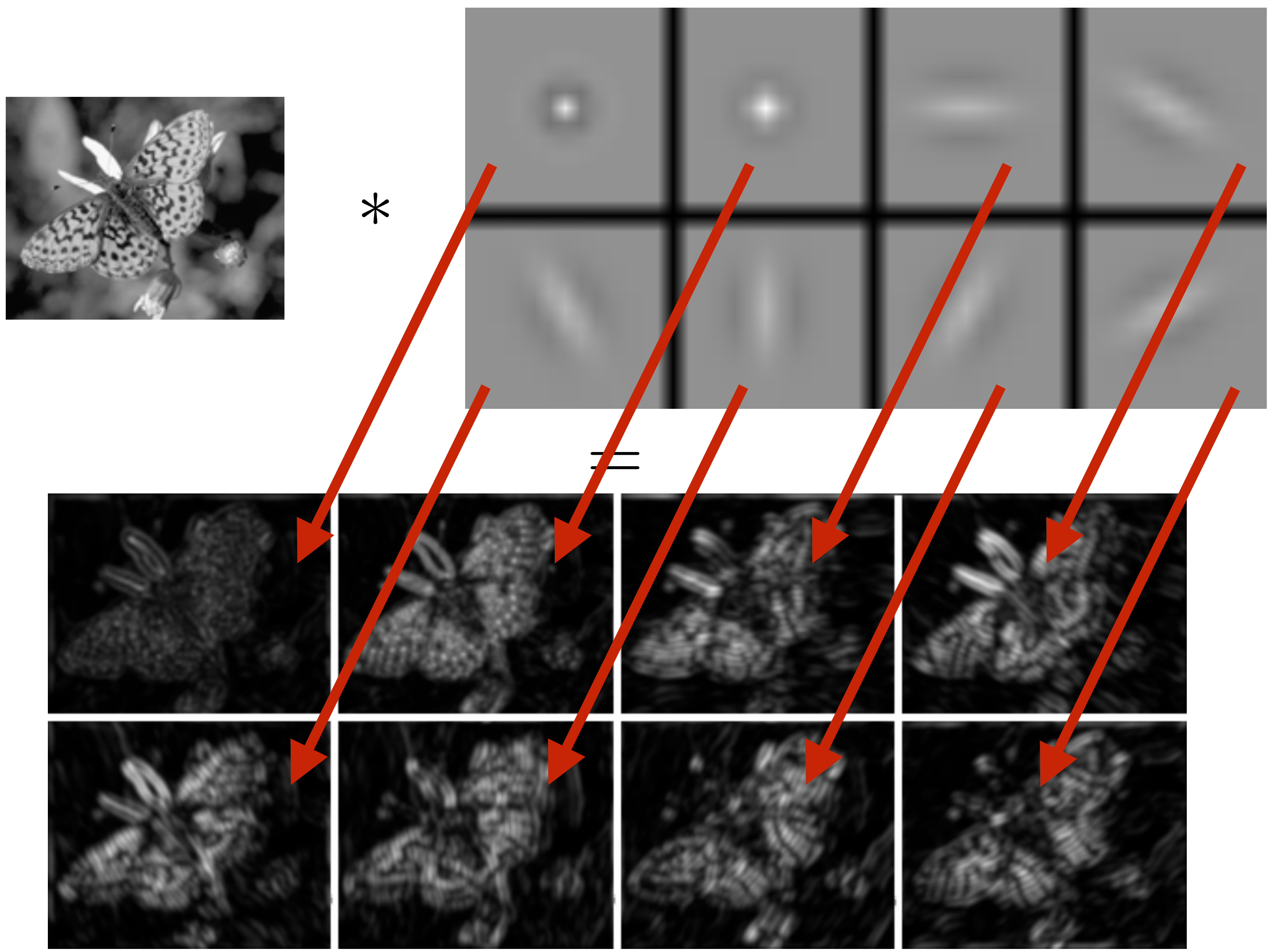


LM Filter Bank

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

Using Filter Banks

- Convolve image with each filter and store the responses (convolution results)



[Source: James Hays]

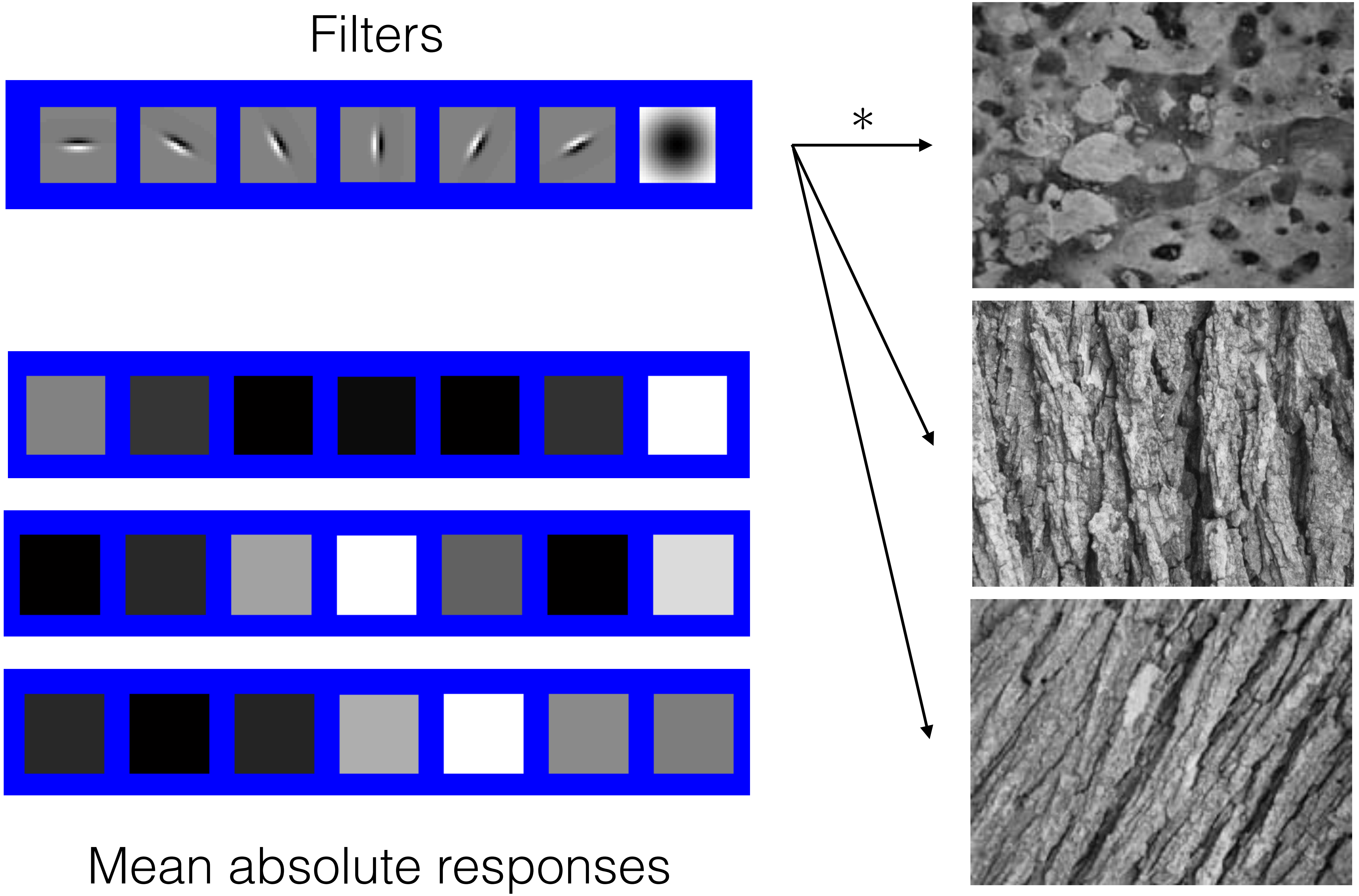
Texture Representation

- Textures are regular or stochastic pattern caused by bumps, grooves, color changes

- How do we represent texture?

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

Record simple statistics, mean, standard deviation of absolute responses



[Source: James Hays]

Texture Representation

- Textures are regular or stochastic pattern caused by bumps, grooves, color changes

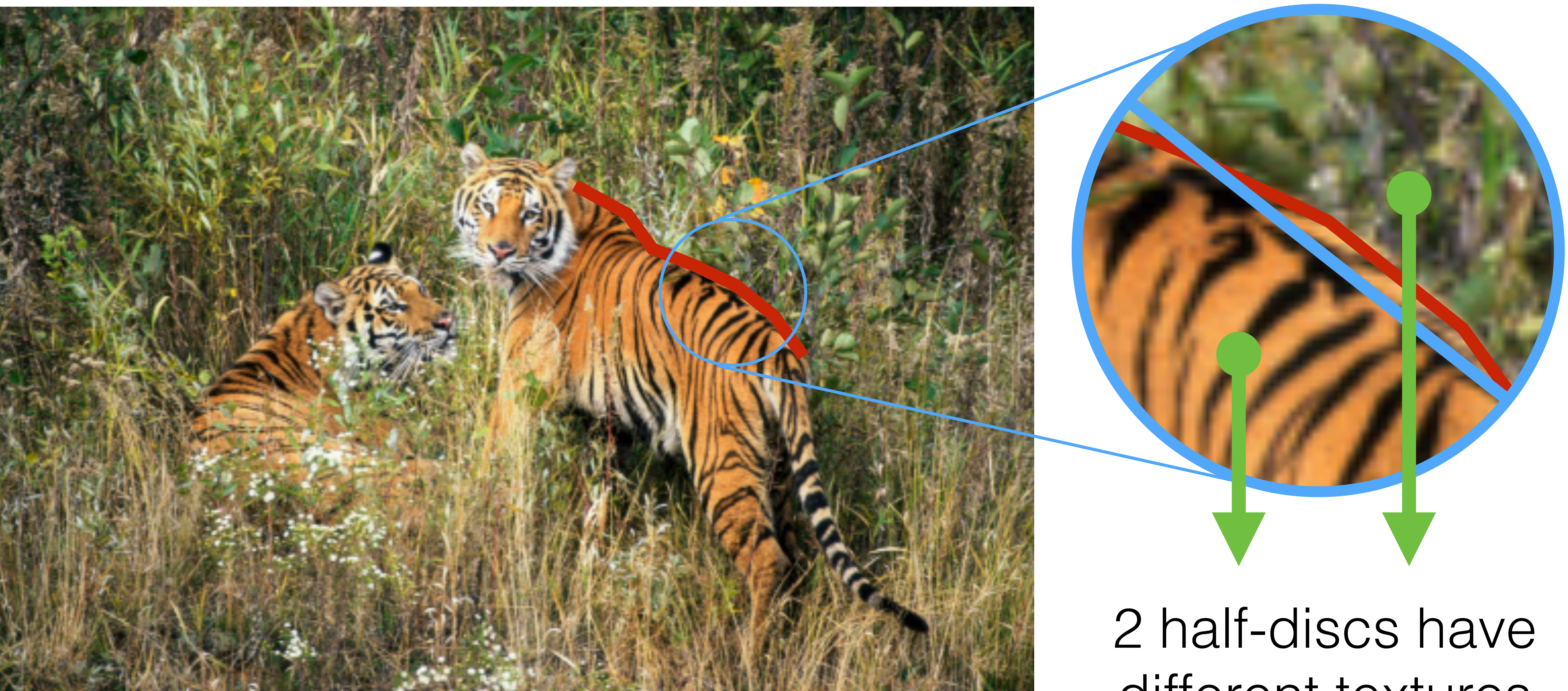
- How do we represent texture?

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

Take vectors of filter responses, cluster them, construct histograms

Boundary Detections: the interplay of textures

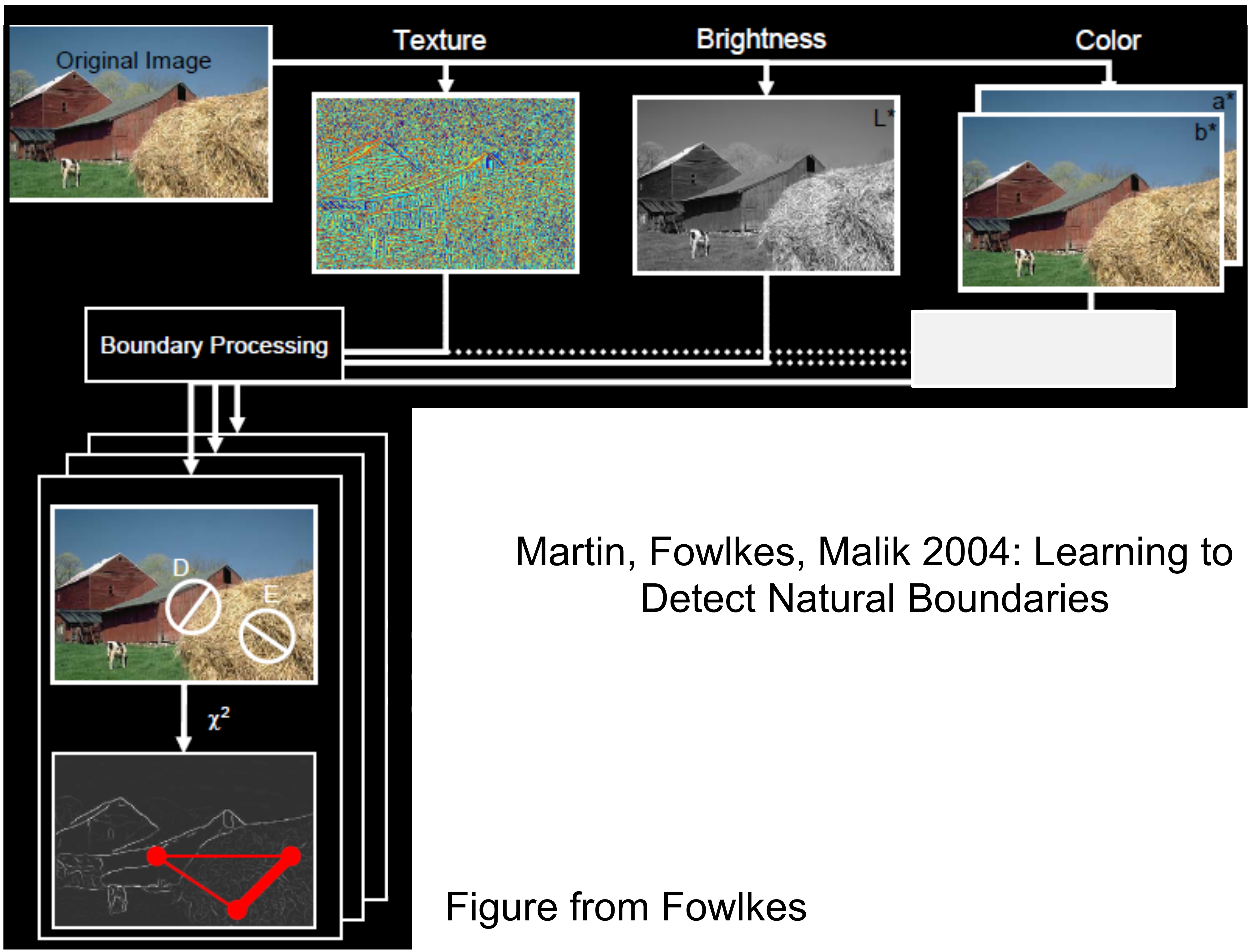
Boundary (Edge) Detection



2 half-discs have different textures

Picture from http://www.gettyimages.ca/detail/photo/siberian-tigers-playing-royalty-free-image/120220173

pB Boundary Detector



Martin, Fowlkes, Malik 2004: Learning to Detect Natural Boundaries

Figure from Fowlkes

pB Boundary Detector

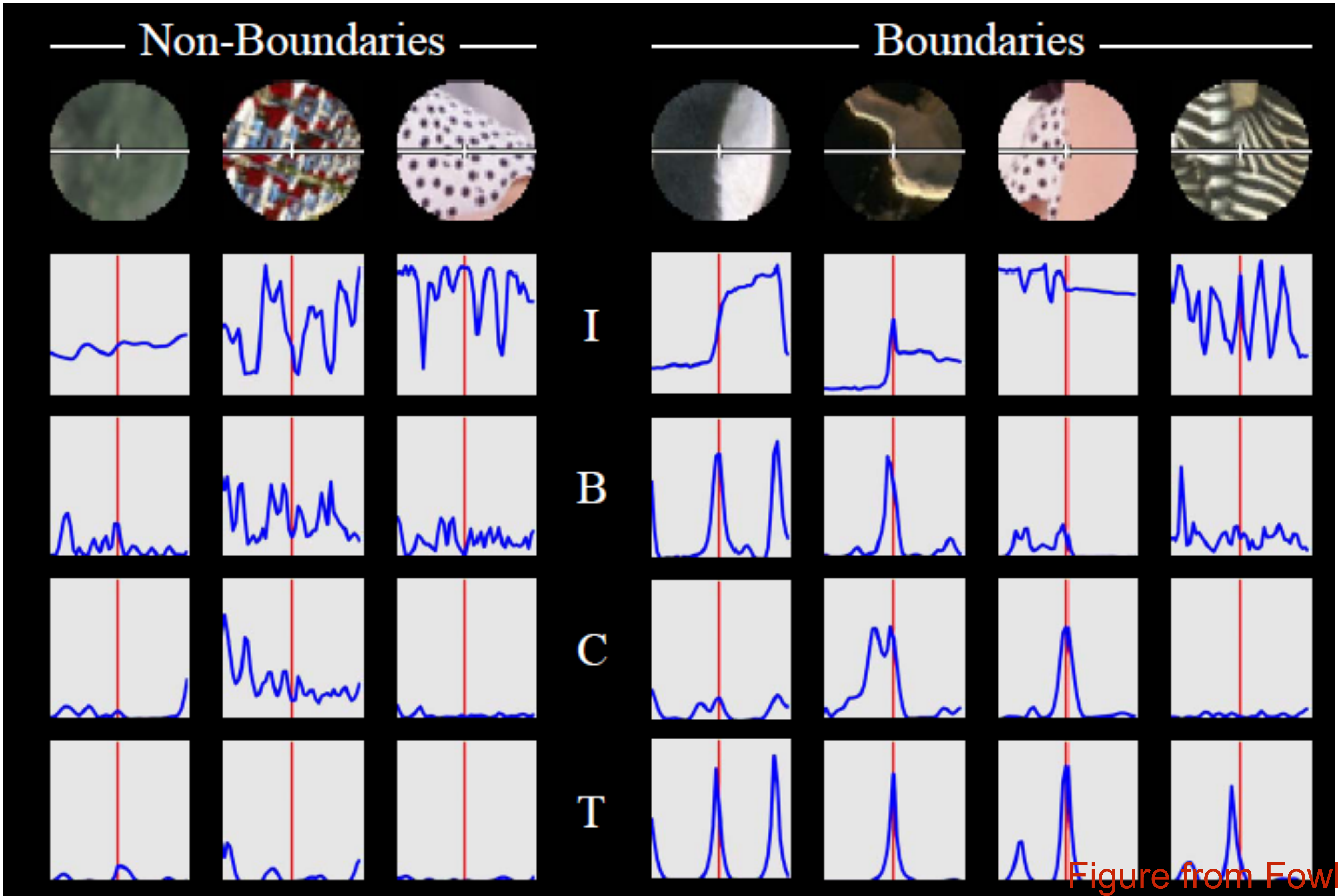


Figure from Fowlkes


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

Global pB Boundary Detector

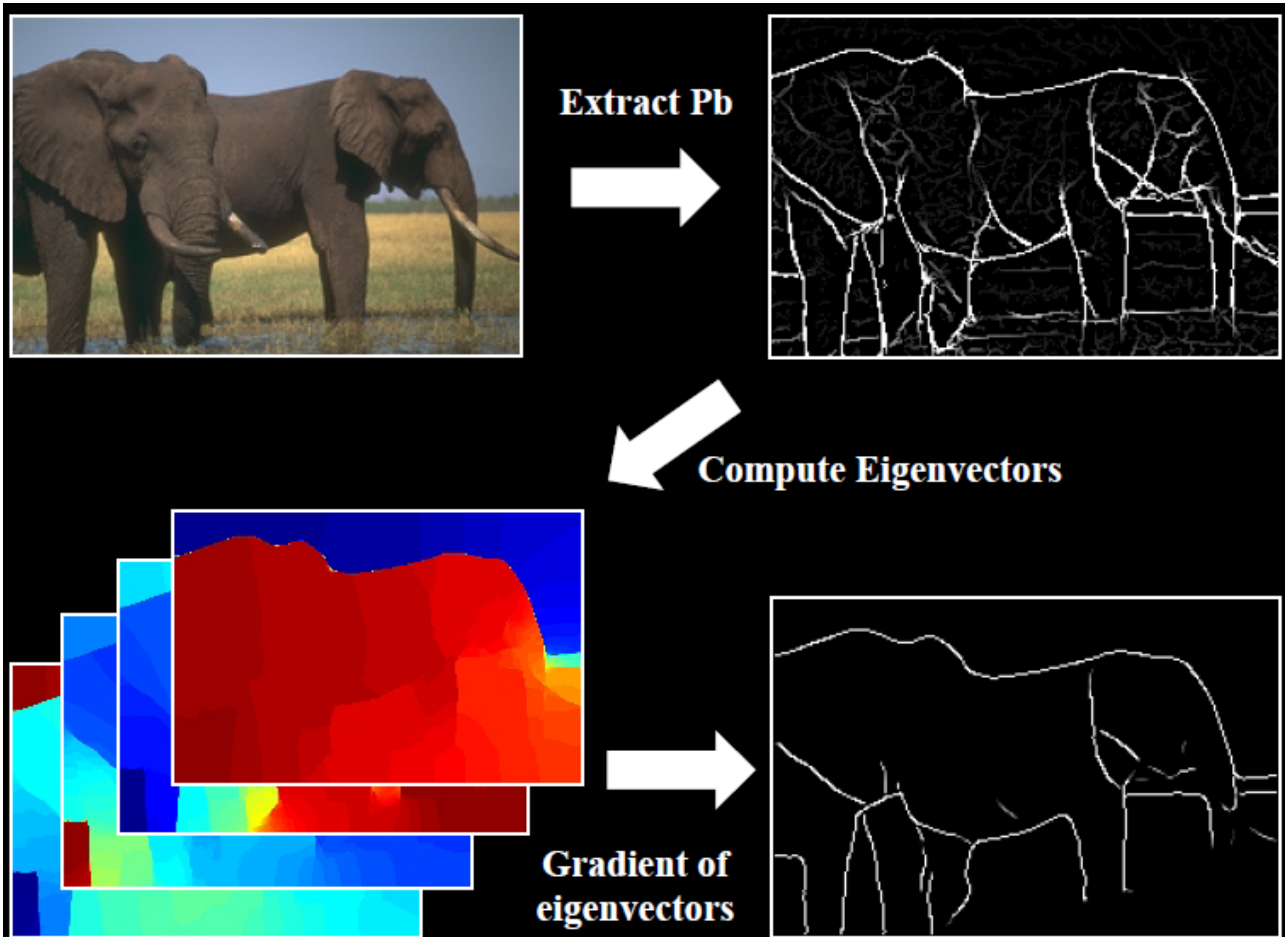
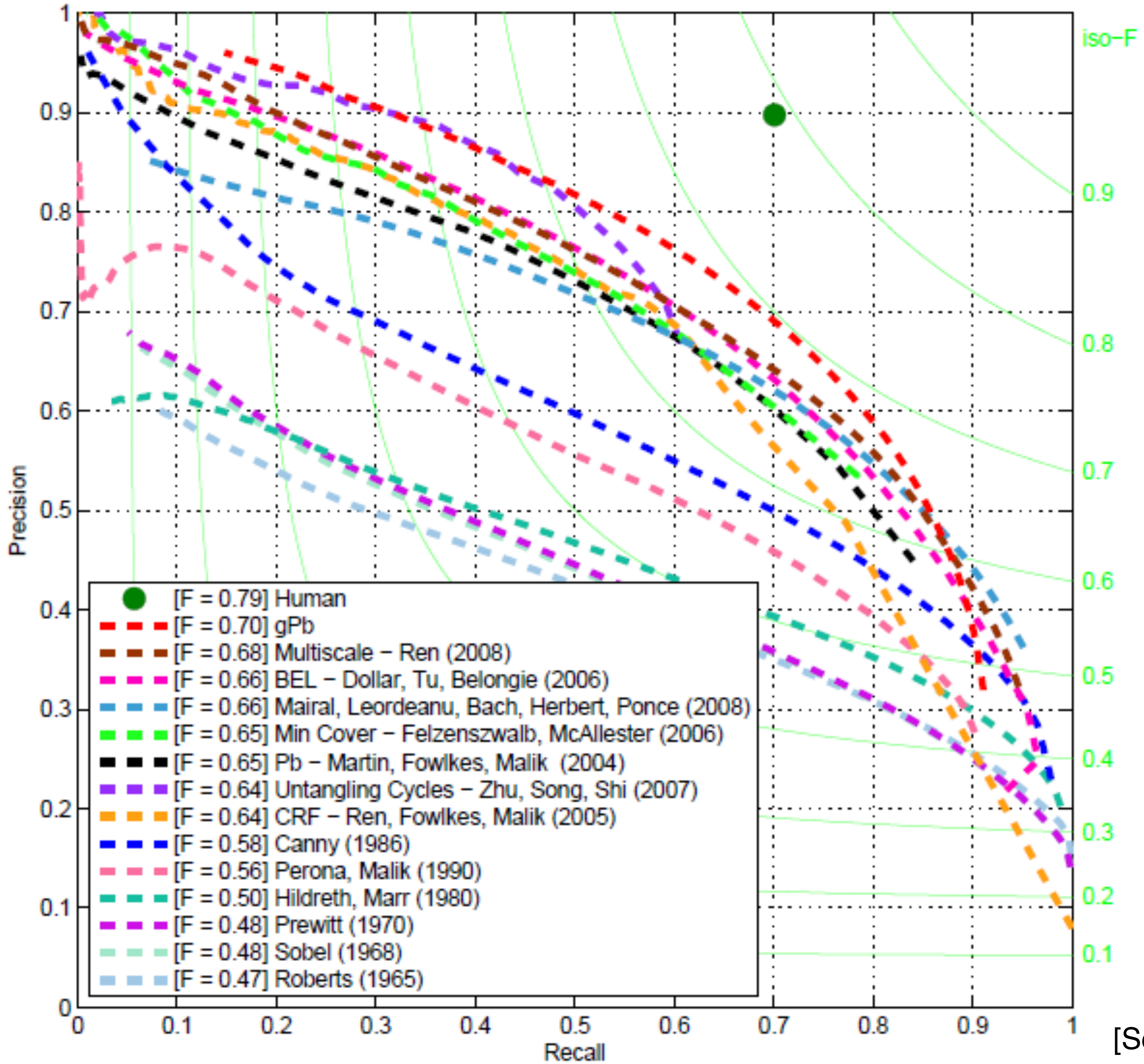


Figure from Fowlkes

45 Years of Boundary Detection



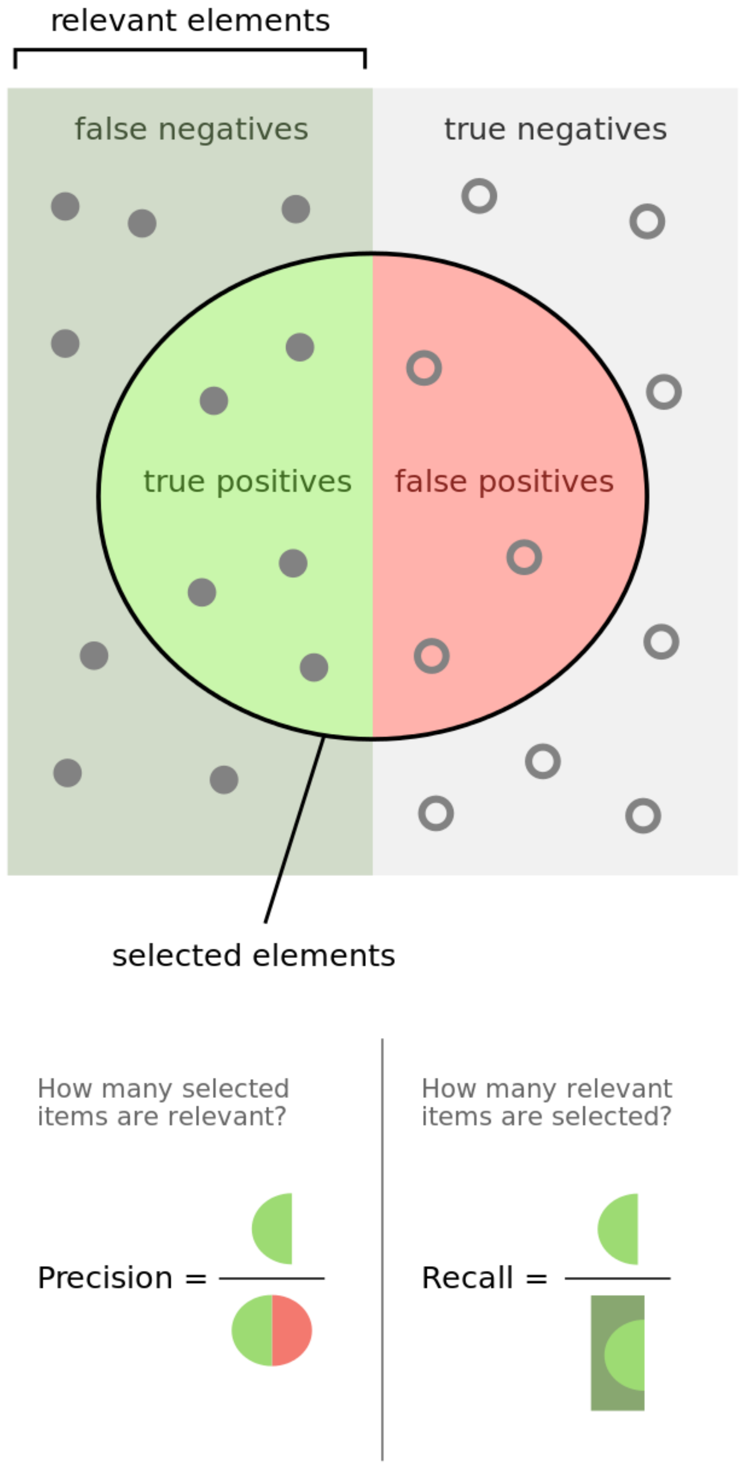
[Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011]

Precision and Recall

- Precision is the fraction of relevant instances among the retrieved instances

- Recall is the fraction of relevant instances that have been retrieved over the total number of relevant instances

- High precision means that an algorithm returned substantially more relevant results than irrelevant ones, while high recall means that an algorithm returned most of the relevant results.



Courtesy: https://en.wikipedia.org/wiki/Precision_and_recall

Precision and Recall

- Precision (often called positive predictive value) and recall (often called true positive rate or sensitivity)

$$\text{Precision} = TP / (TP + FN)$$
$$\text{Recall} = TP / (TP + FP)$$

Precision and Recall

Data	Ground Truth	Algorithm 1	Algorithm 2	Algorithm 3
1	F	T	**F**	**F**
2	T	**T**	F	F
3	T	**T**	F	**T**
4	F	T	**F**	T
7	T	**T**	F	**T**
6	F	T	**F**	T
7	F	T	**F**	T

	Algorithm 1	Algorithm 2	Algorithm 3
True Positive	3	0	2
True Negative	0	4	1
False Positive	4	0	3
False Negative	0	3	1

Specificity and Accuracy

- True negative rates (often called specificity) and accuracy

$$\text{True negative rate} = TN / (TN + FP)$$

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Summary

- Edge detection
- Canny edge detector
- Texture analysis
- pB Boundary Detector

Self-Study

- Experiment with Canny Edge Detector
- Use filter banks to describe various textures

Project Ideas

- Project Idea #1 — Implement pB Boundary Detector
- Project Idea #2 — Implement a texture classification system