On Edges, Templates, Texture

Lecture 9

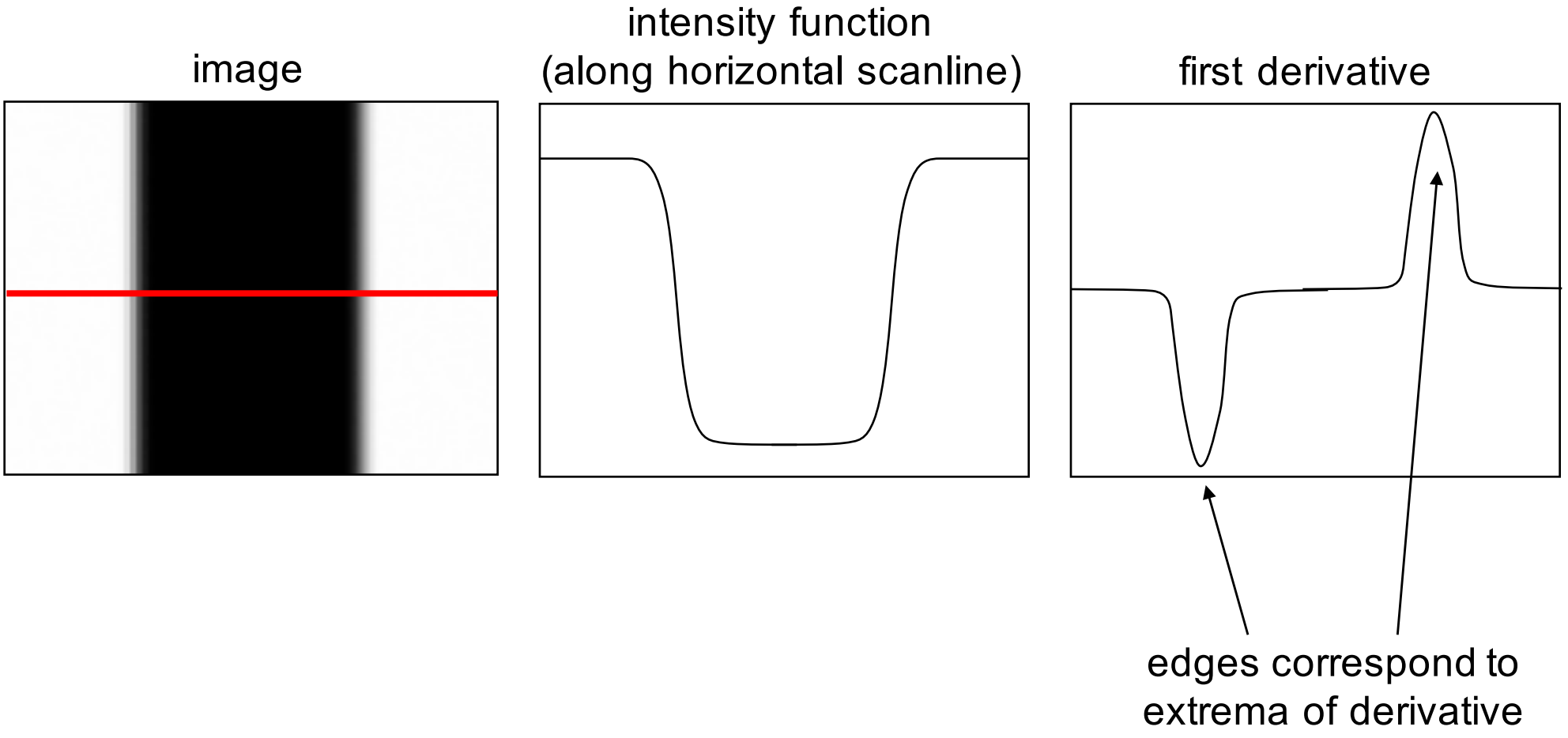
Stella Yu

Slide Credit: Alyosha Efros

Cinque Terre, May 2005

Characterizing edges

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



Taking derivative by convolution

Partial derivatives with convolution

For 2D function f(x,y), the partial derivative is:

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

For discrete data, we can approximate using finite differences:

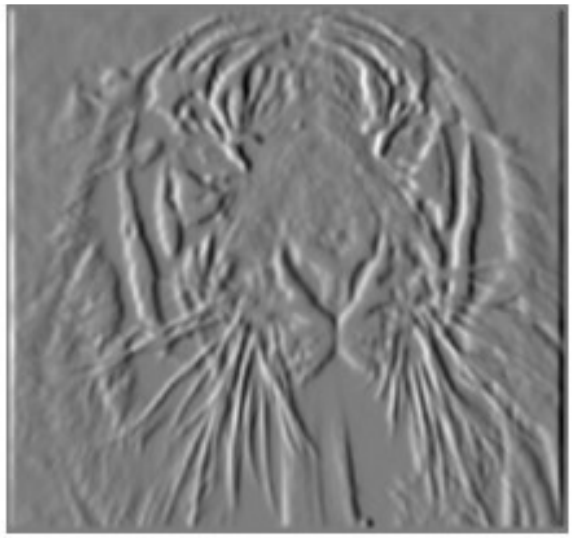
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

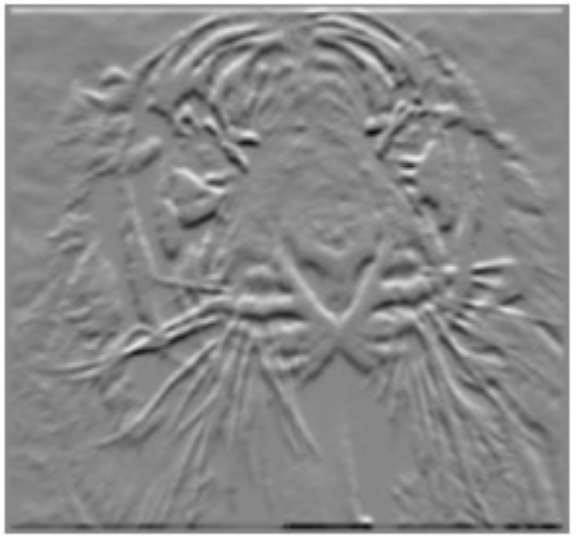
To implement above as convolution, what would be the associated filter?

Source: K. Grauman

Partial derivatives of an image

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





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

-1	1

-1
1

or

1
-1

Which shows changes with respect to x?

Finite difference filters

Other approximations of derivative filters exist:

Prewitt: $M_x =$

-1	0	1
-1	0	1
-1	0	1

; $M_y =$

1	1	1
0	0	0
-1	-1	-1

Sobel: $M_x =$

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

; $M_y =$

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

Roberts: $M_x =$

0	1
-1	0

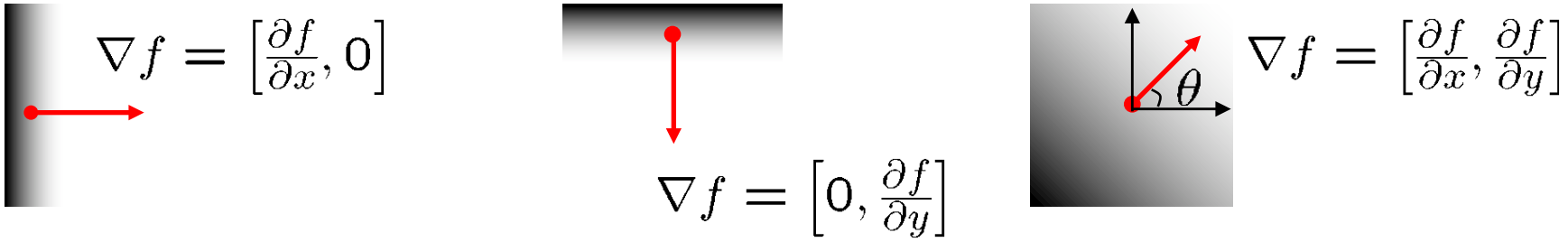
; $M_y =$

1	0
0	-1

Source: K. Grauman

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?

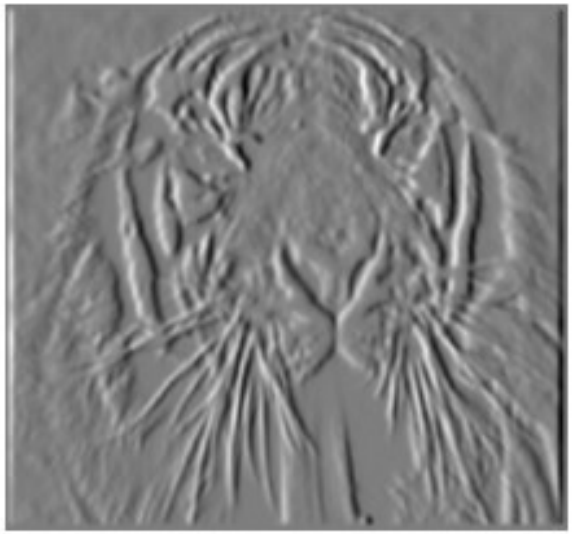
The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

The *edge strength* is given by the gradient magnitude

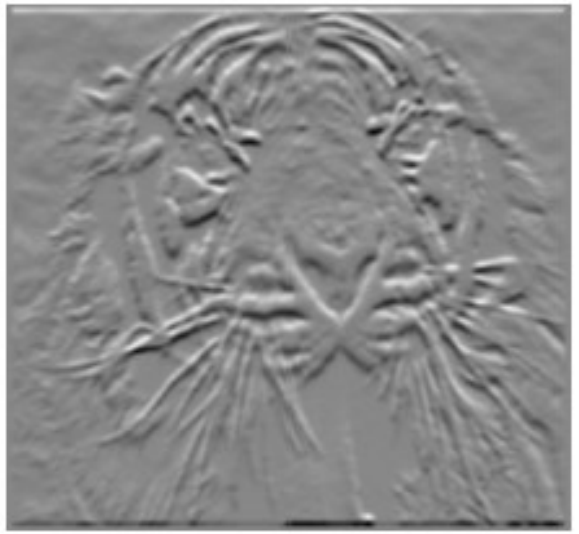
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

Image Gradient



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



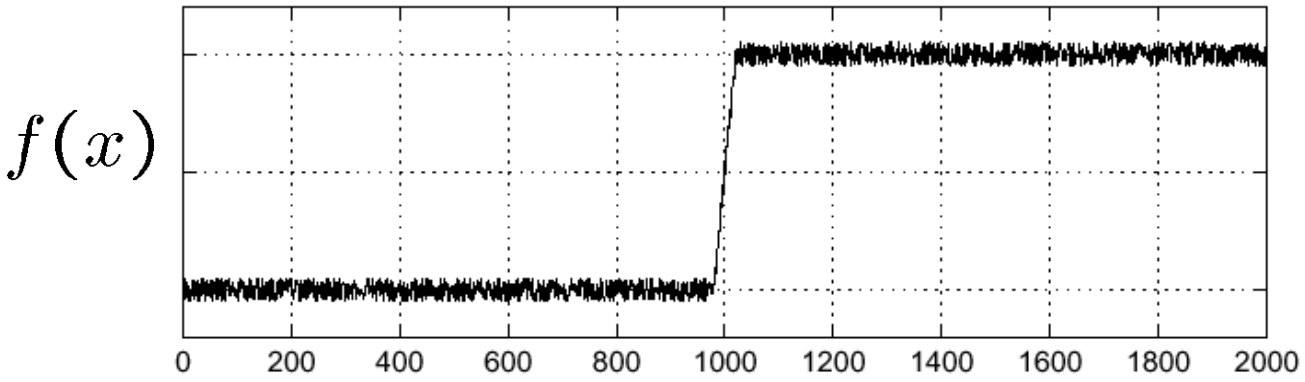
$$\frac{\partial f(x,y)}{\partial y}$$

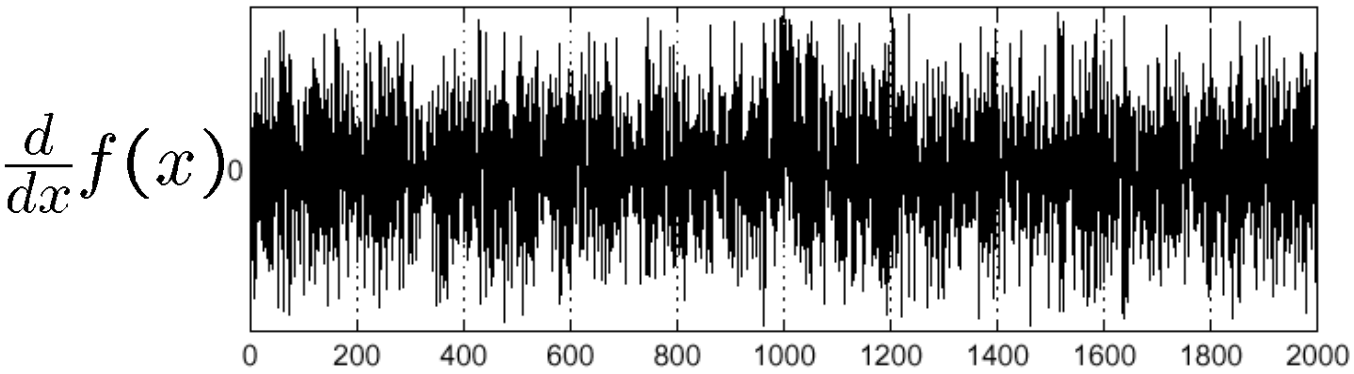
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

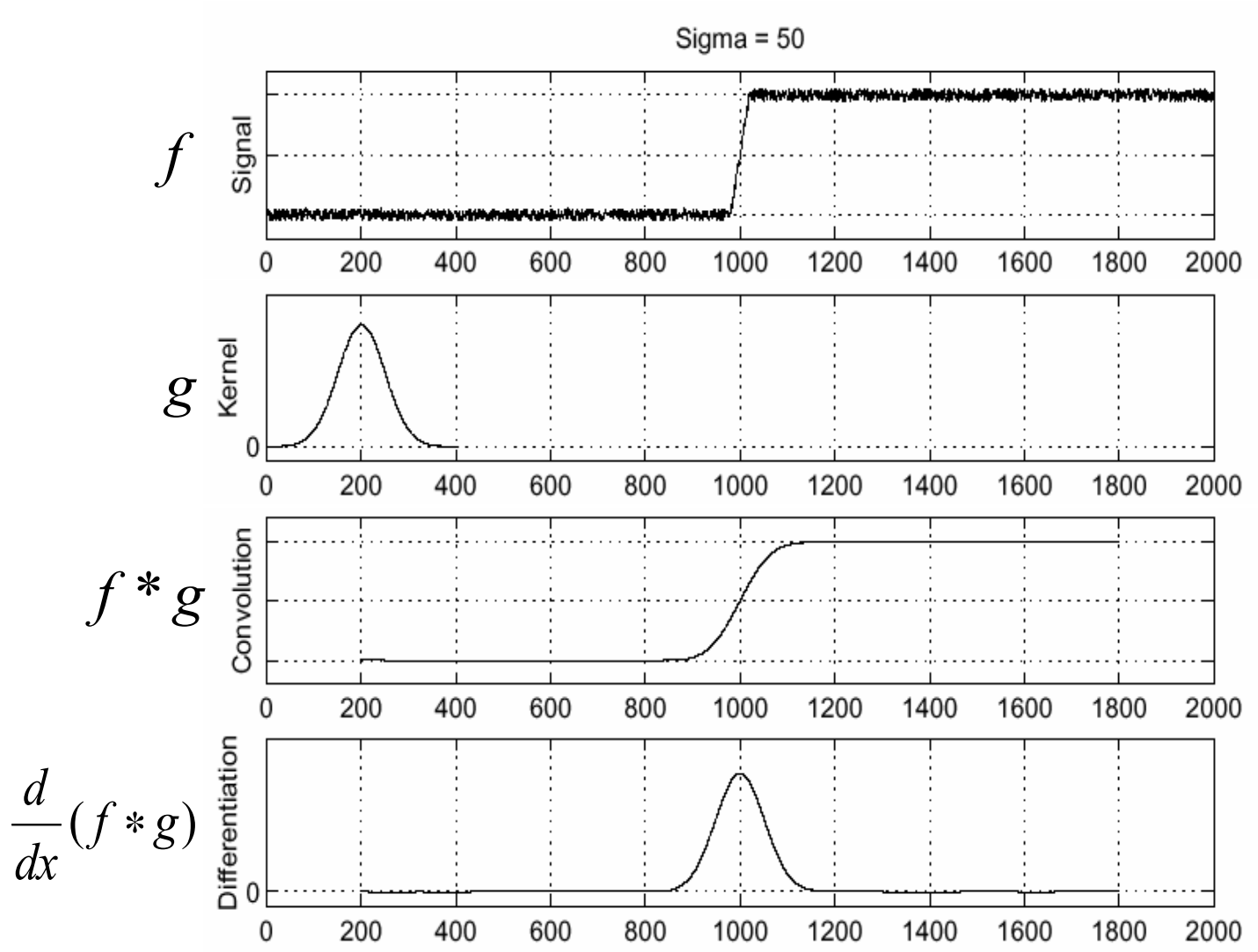




Where is the edge?

Source: S. Seitz

Solution: smooth first

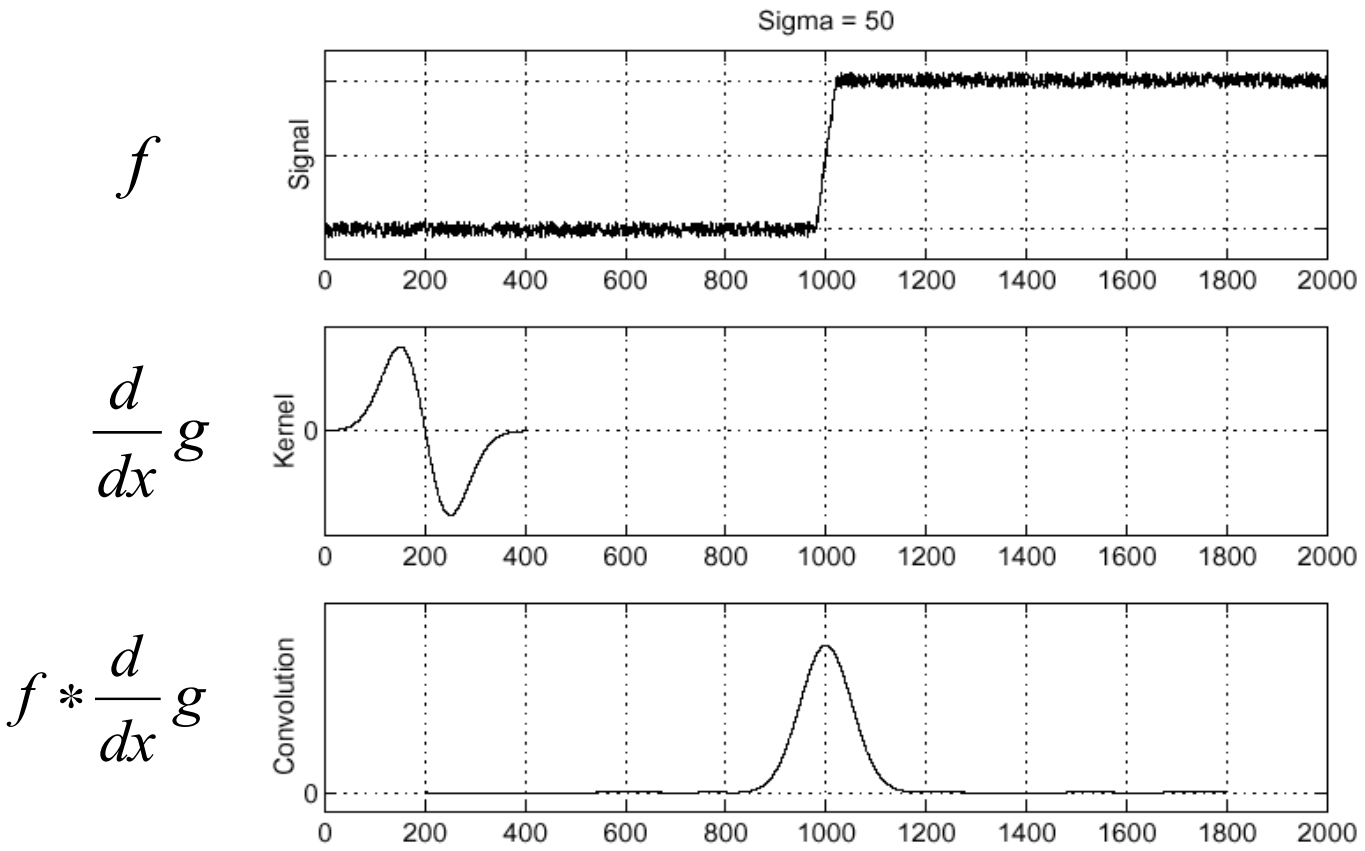


- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

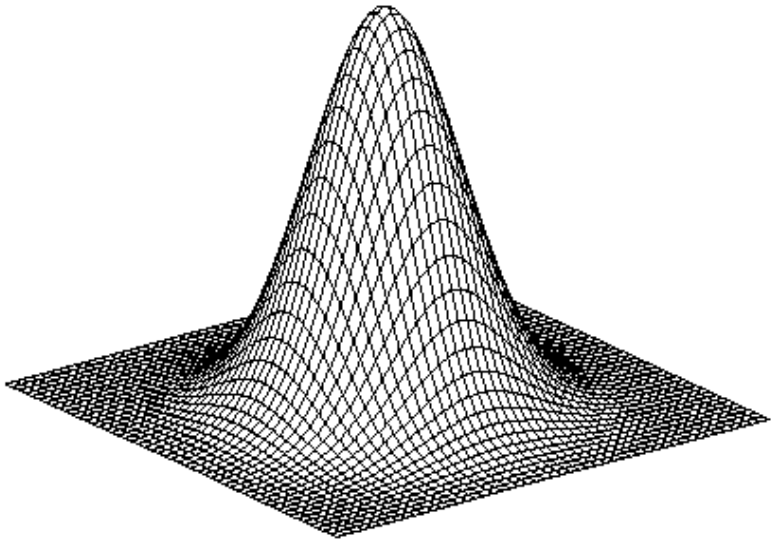
Derivative theorem of convolution

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

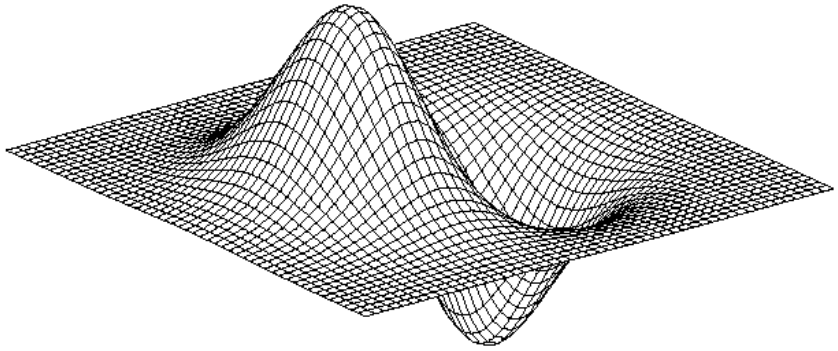


Source: S. Seitz

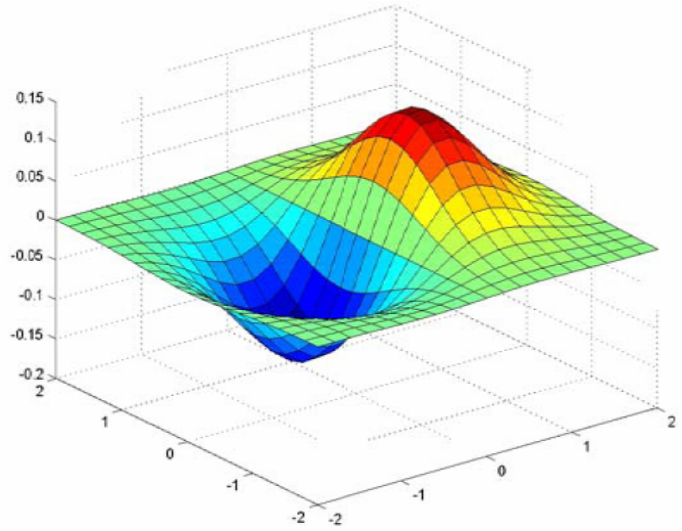
Derivative of Gaussian filter



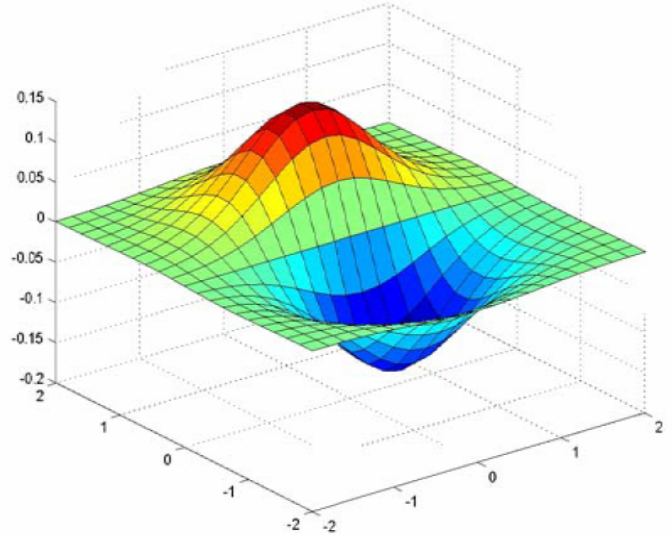
* [1 -1] =



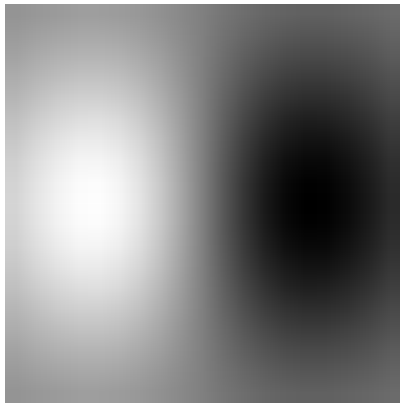
Derivative of Gaussian filter

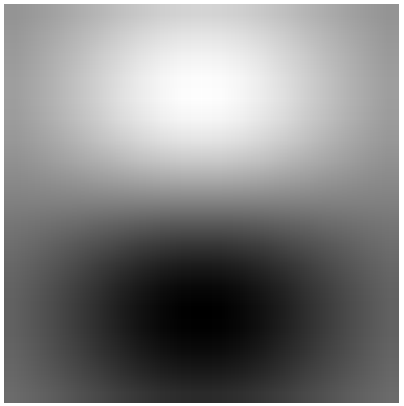


x-direction



y-direction





Which one finds horizontal/vertical edges?

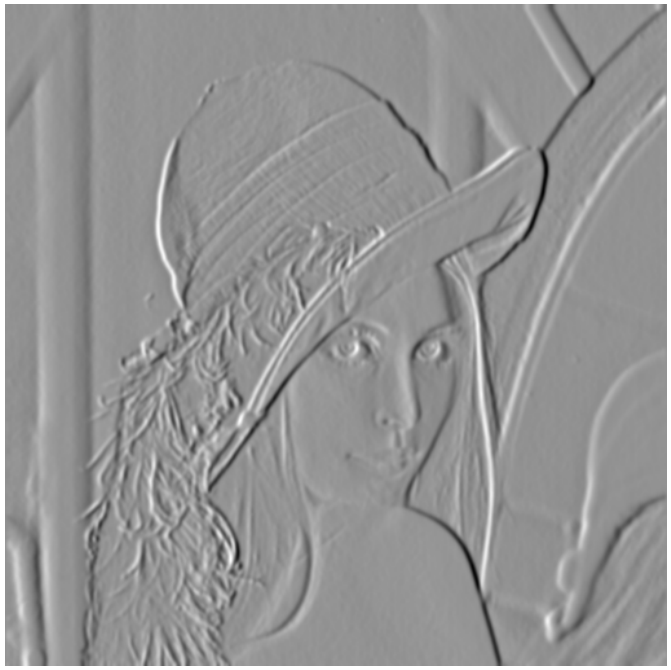
MATLAB code

```
im = im2double(imread(filemane));
g = fspecial('gaussian',15,2);
imagesc(g);
surfl(g);
gim = conv2(im,g,'same');
imagesc(conv2(im,[-1 1],'same'));
imagesc(conv2(gim,[-1 1],'same'));
dx = conv2(g,[-1 1],'same');
surfl(dx);
imagesc(conv2(im,dx,'same'));
```

Example

input image

Compute Gradients (DoG)



X-Derivative of Gaussian

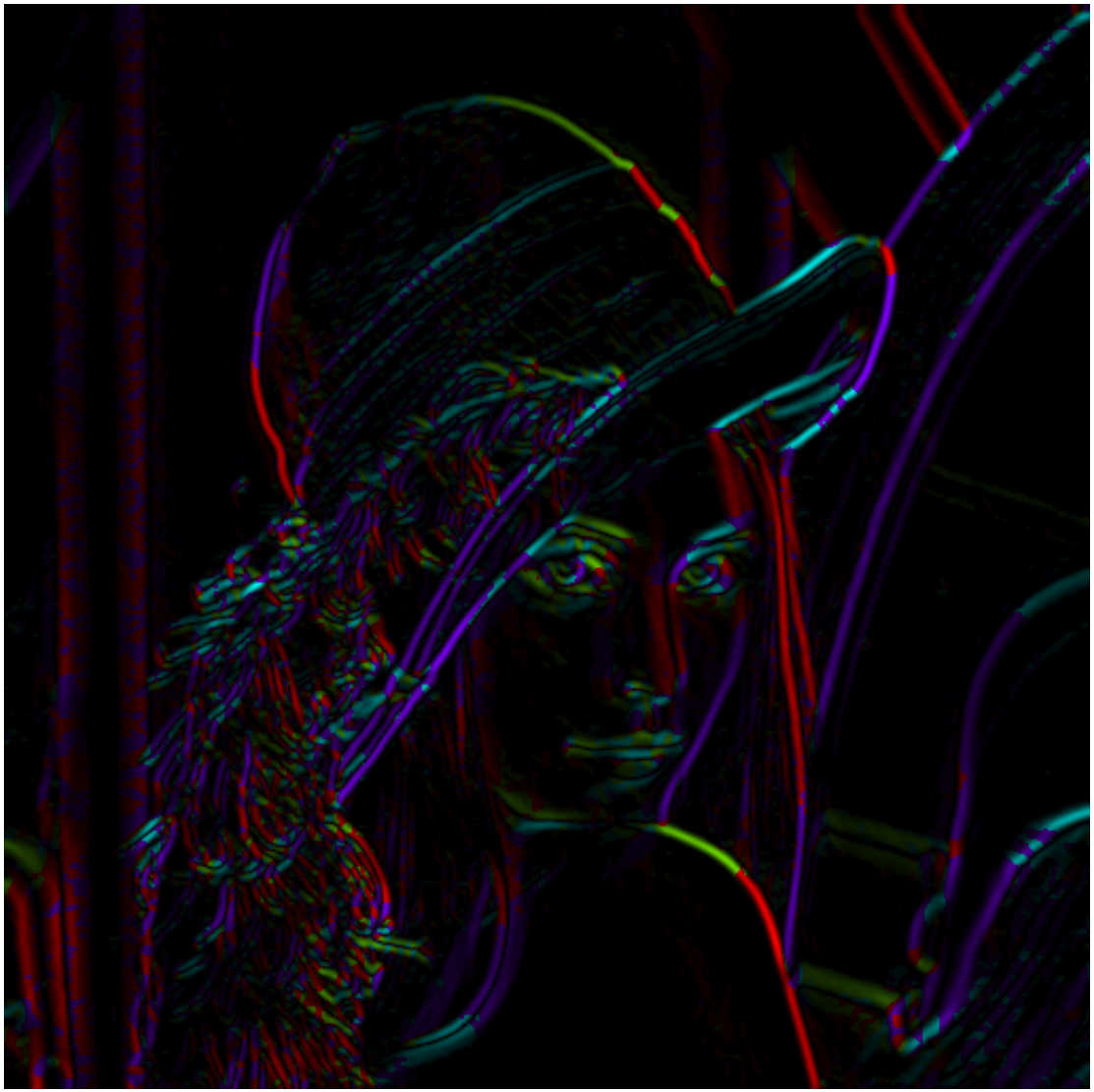
Y-Derivative of Gaussian

Gradient Magnitude

Get Orientation at Each Pixel

Threshold at minimum level

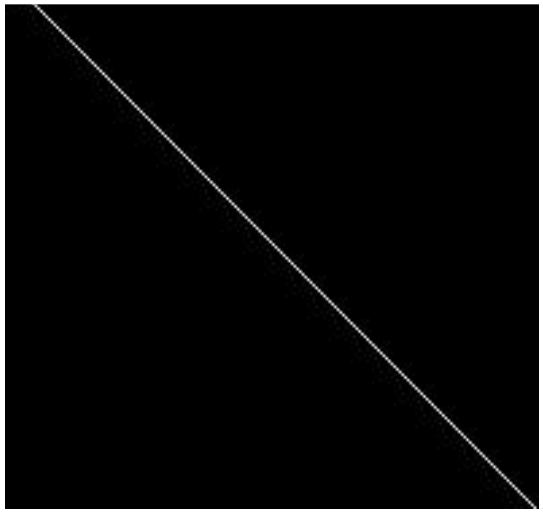
Get orientation



theta = atan2(-gy, gx)

There is ALWAYS a tradeoff between smoothing and good edge localization!

Image with Edge



Edge Location

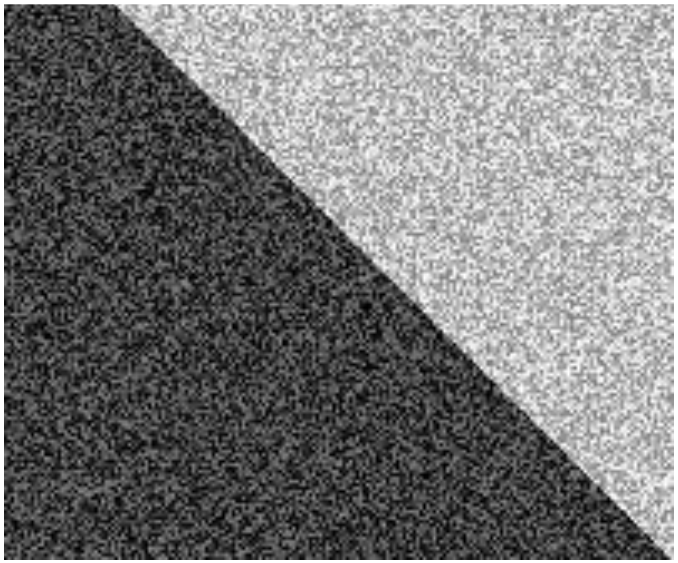
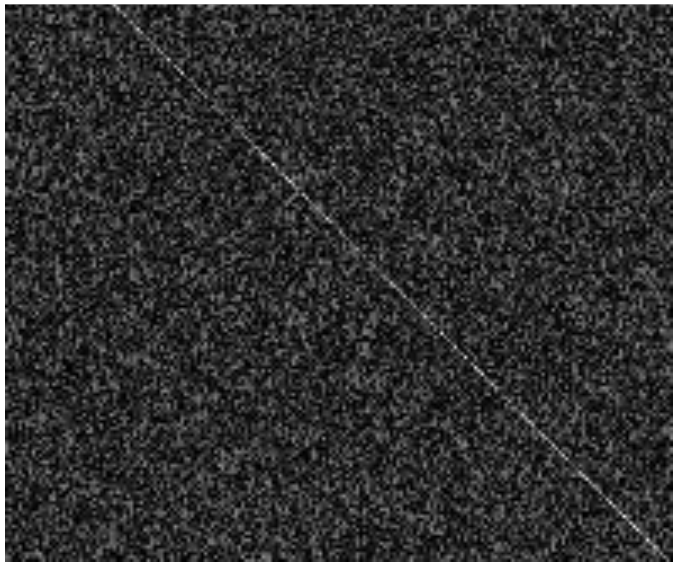
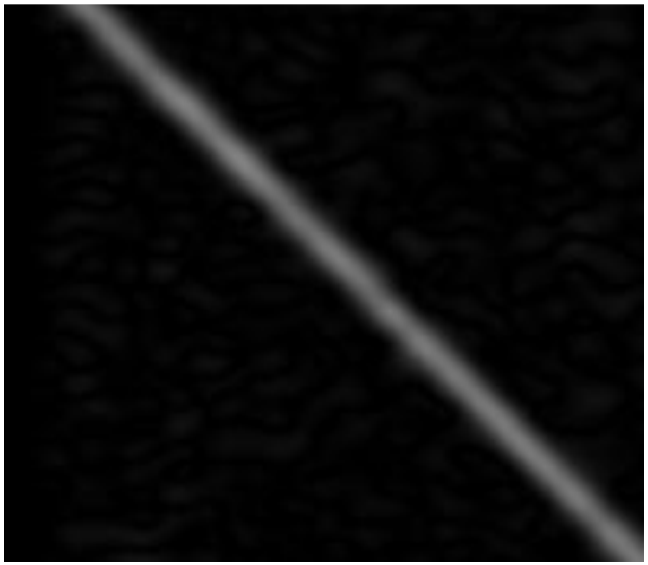


Image + Noise



Derivatives detect edge *and* noise



Smoothed derivative removes noise, but blurs edge

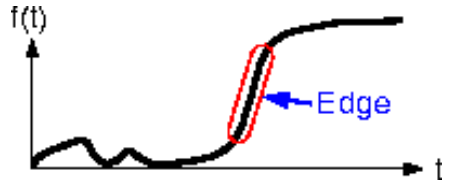
The Canny edge detector

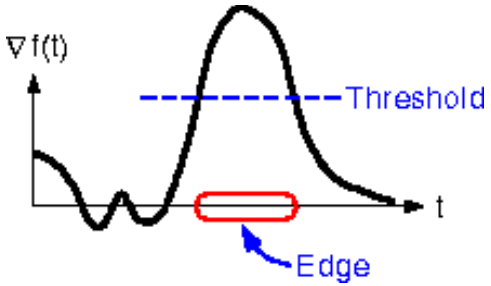
norm of the gradient

The Canny edge detector

thresholding

The Canny edge detector

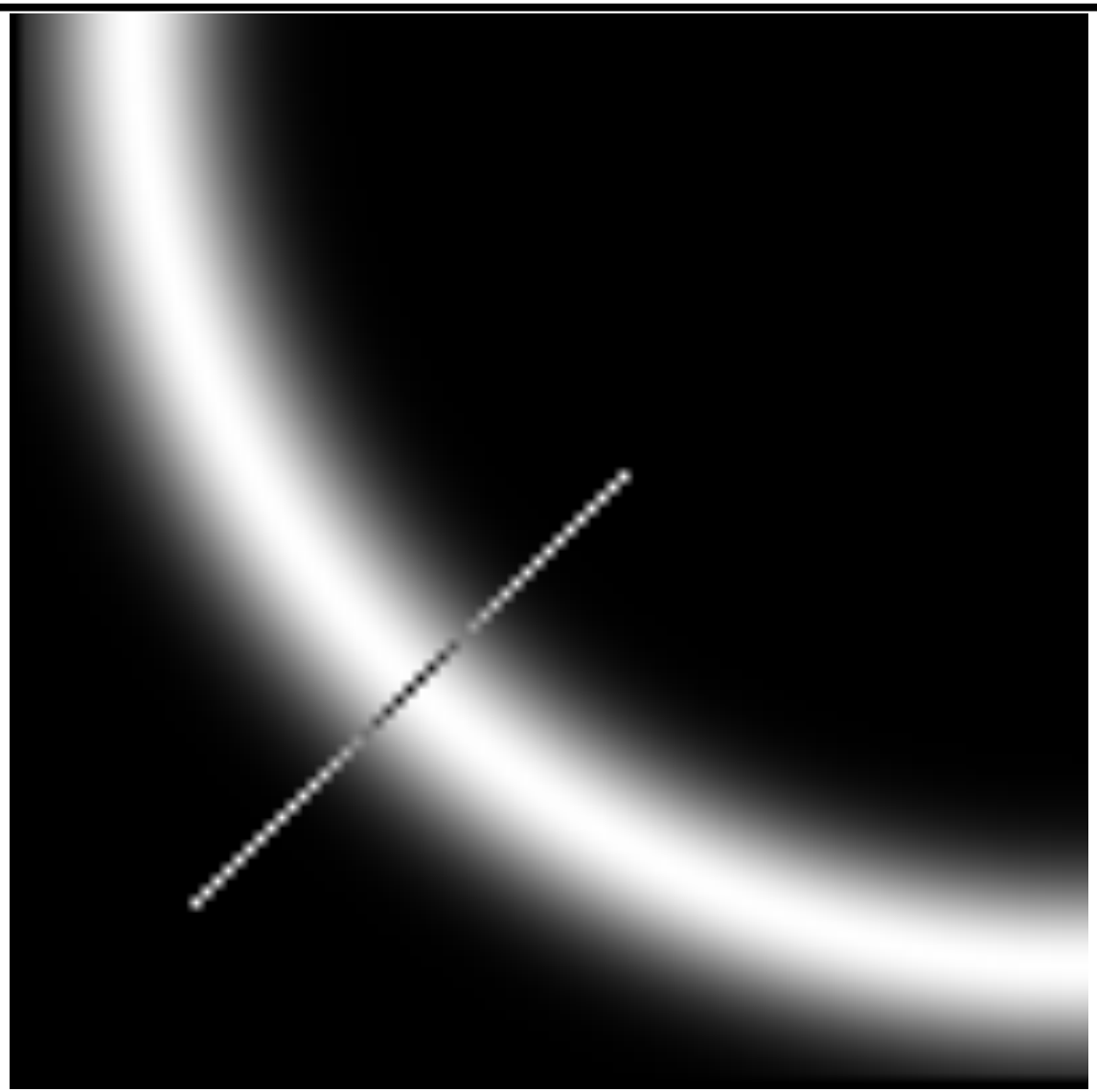




How to turn these thick regions of the gradient into curves?

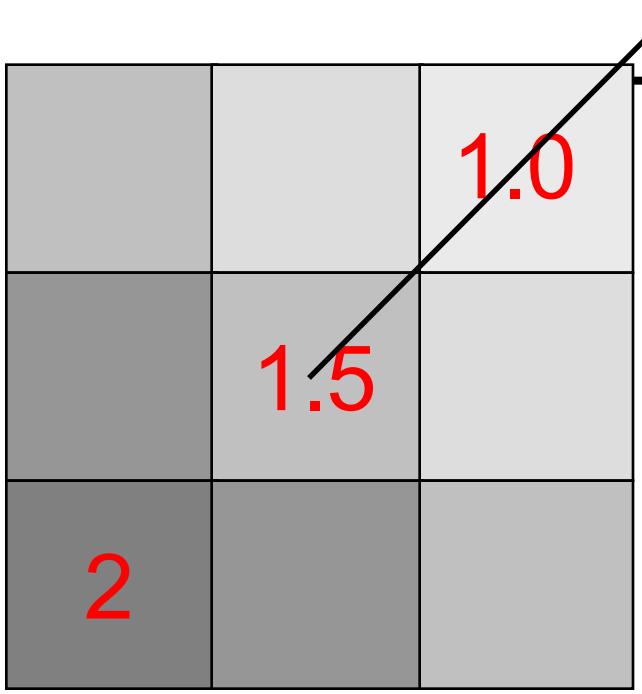
thresholding

Non-maximum suppression



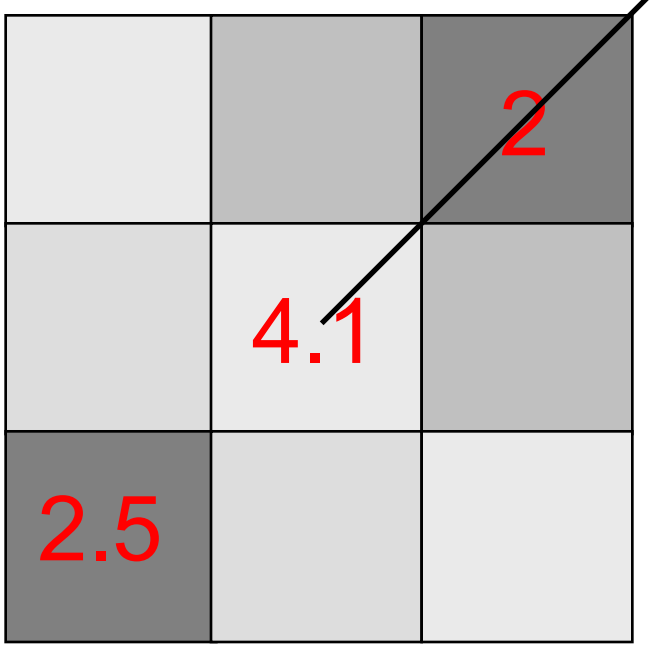
Check if pixel is local maximum along gradient direction, select single max across width of the edge

Non-Local Maxima Suppression



∇I

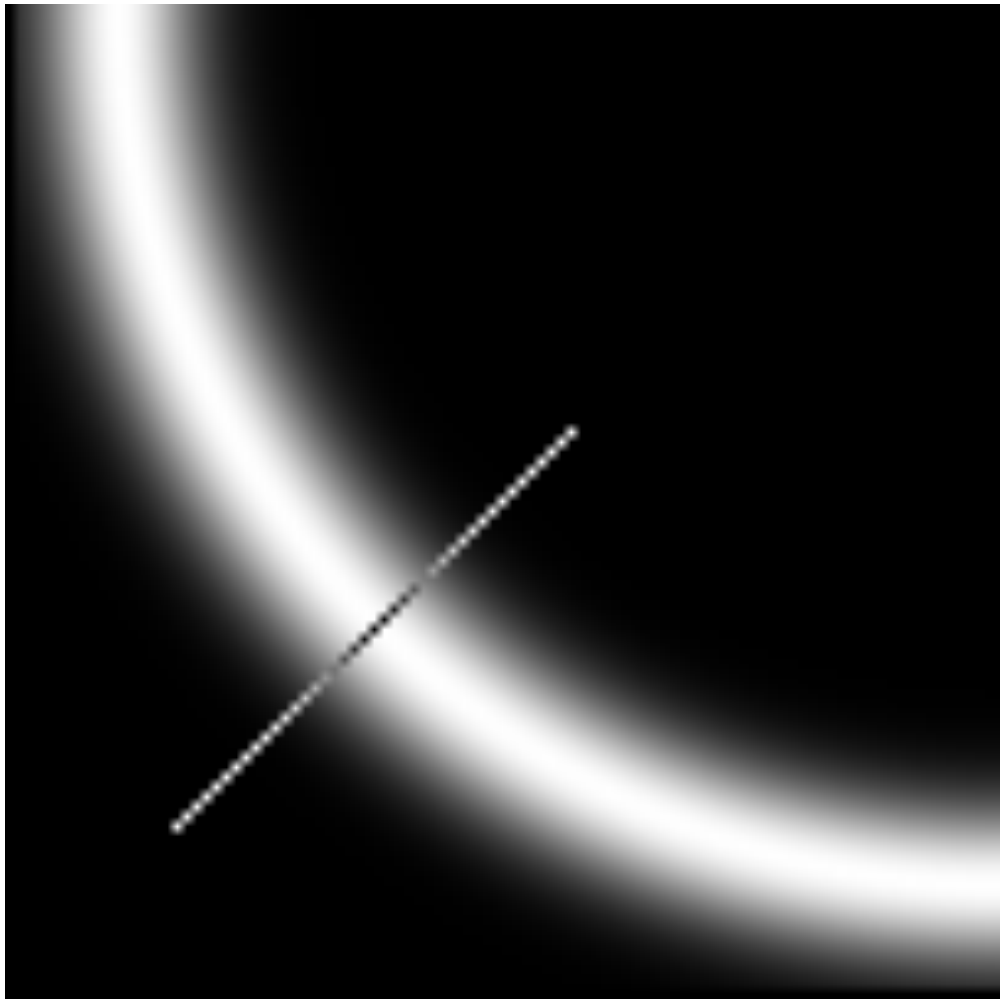
Gradient magnitude at center pixel is lower than the gradient magnitude of a neighbor *in the direction of the gradient*
→ Discard center pixel (set magnitude to 0)

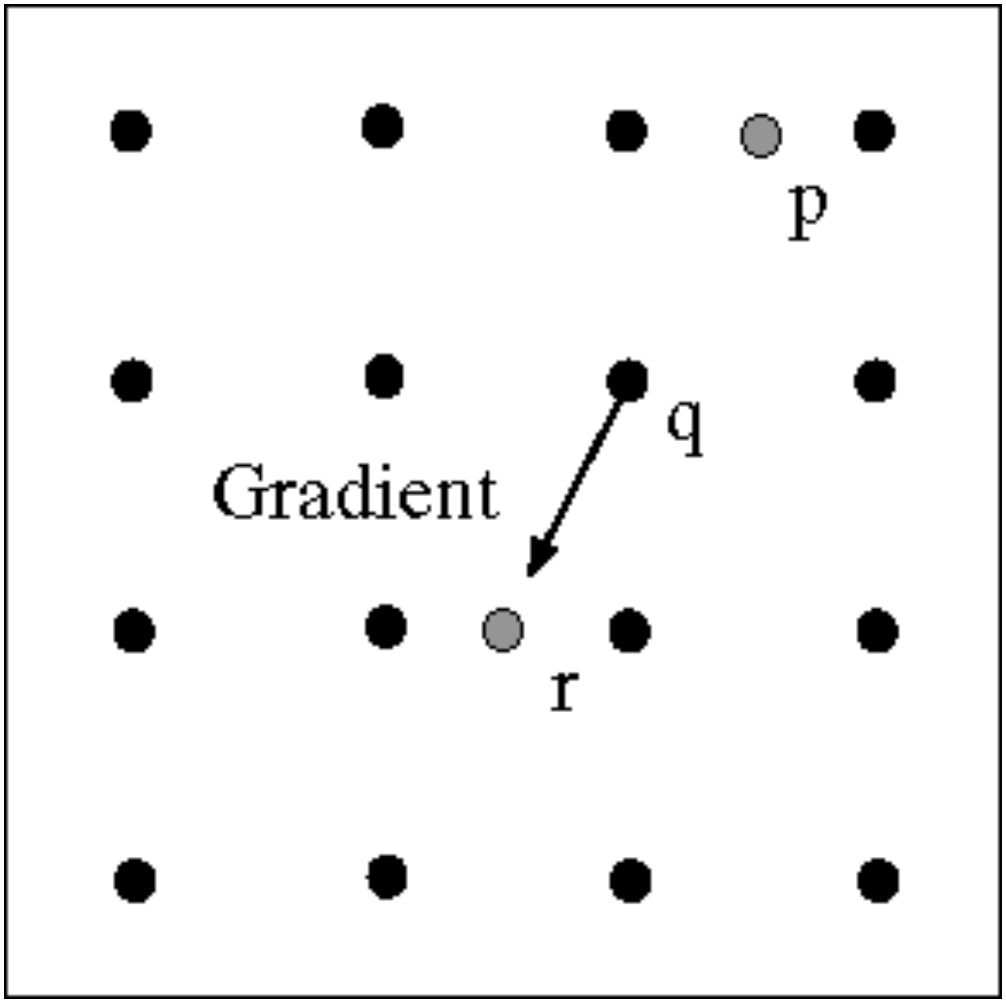


∇I

Gradient magnitude at center pixel is greater than gradient magnitude of all the neighbors *in the direction of the gradient*
→ Keep center pixel unchanged

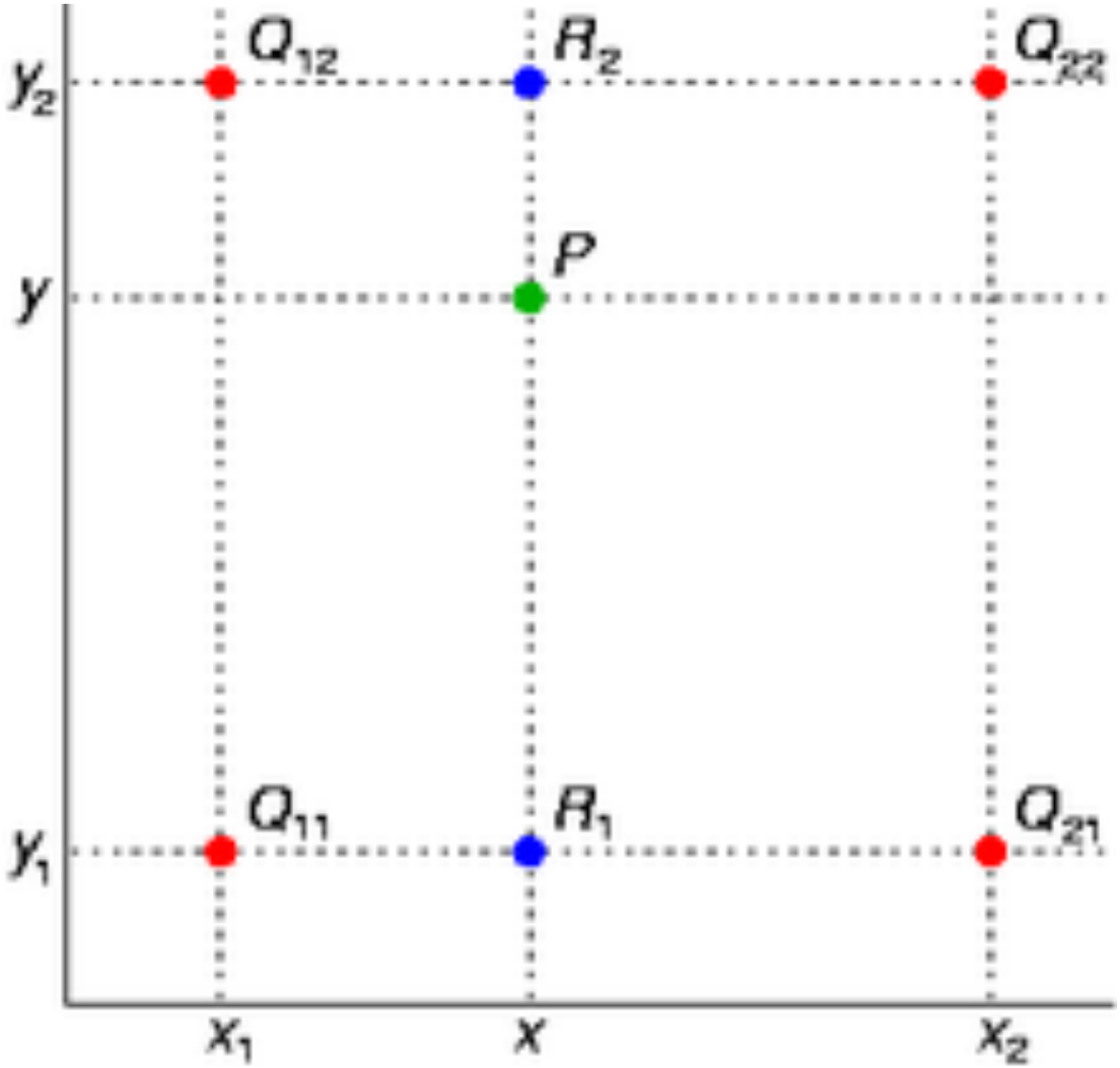
Non-maximum suppression





requires checking interpolated pixels p and r

Bilinear Interpolation



http://en.wikipedia.org/wiki/Bilinear_interpolation

```
Help interp2
```

The Canny edge detector

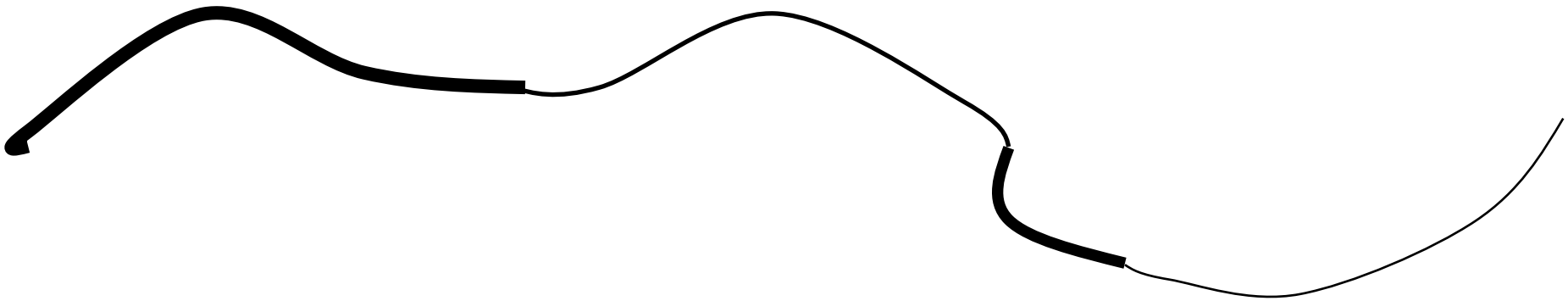
Problem: pixels along this edge didn't survive the thresholding

thinning

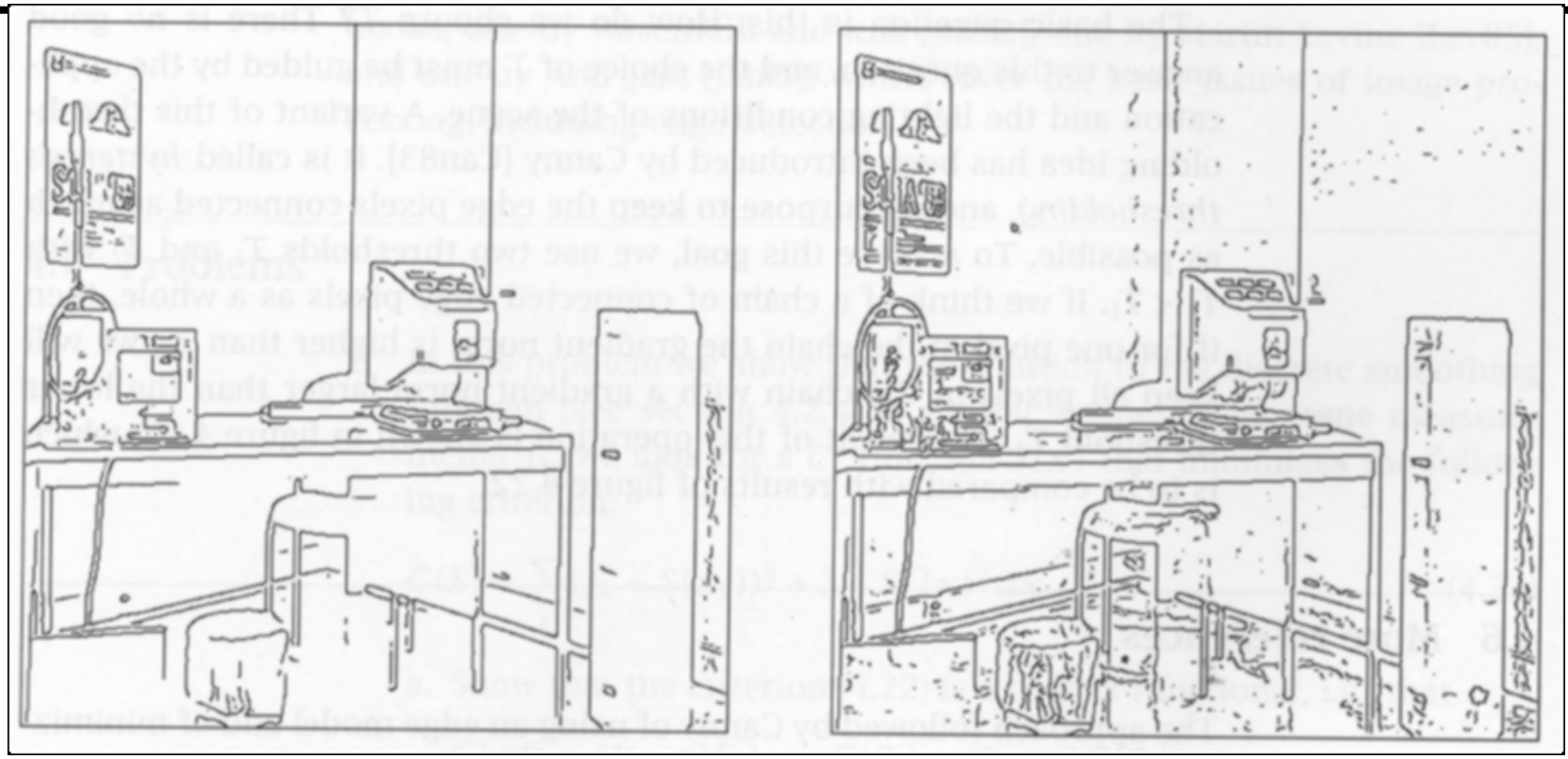
(non-maximum suppression)

Hysteresis thresholding

Use a high threshold to start edge curves, and a low threshold to continue them.



Source: Steve Seitz

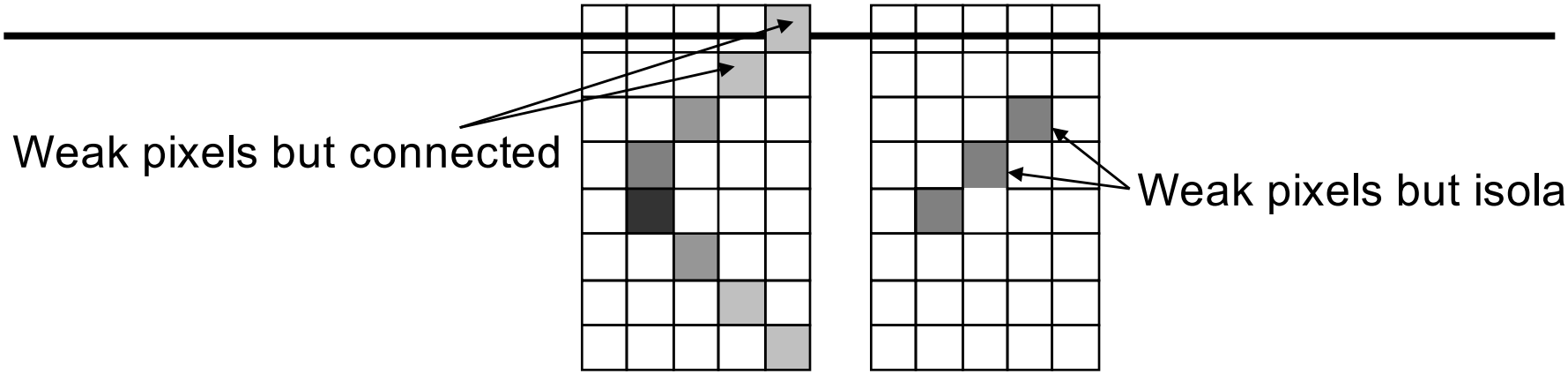


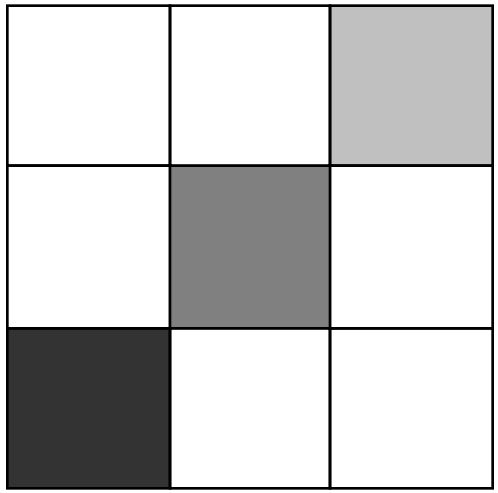
T = 15

T = 5

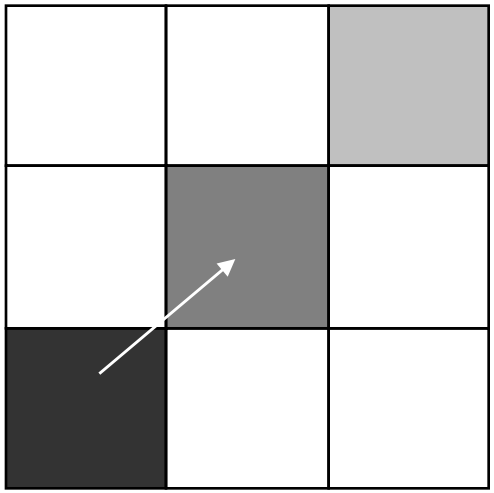
Two thresholds applied to gradient magnitude

Hysteresis Thresholding

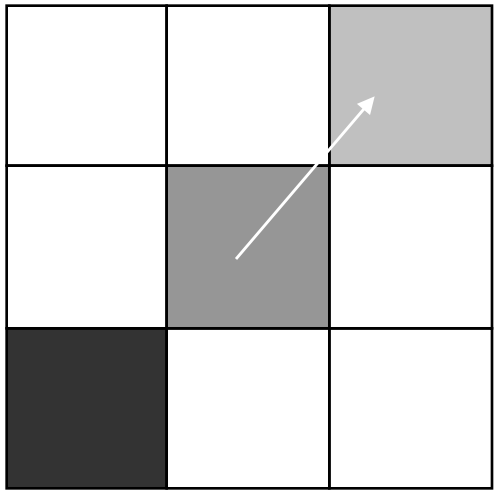




Very strong edge response. Let's start here



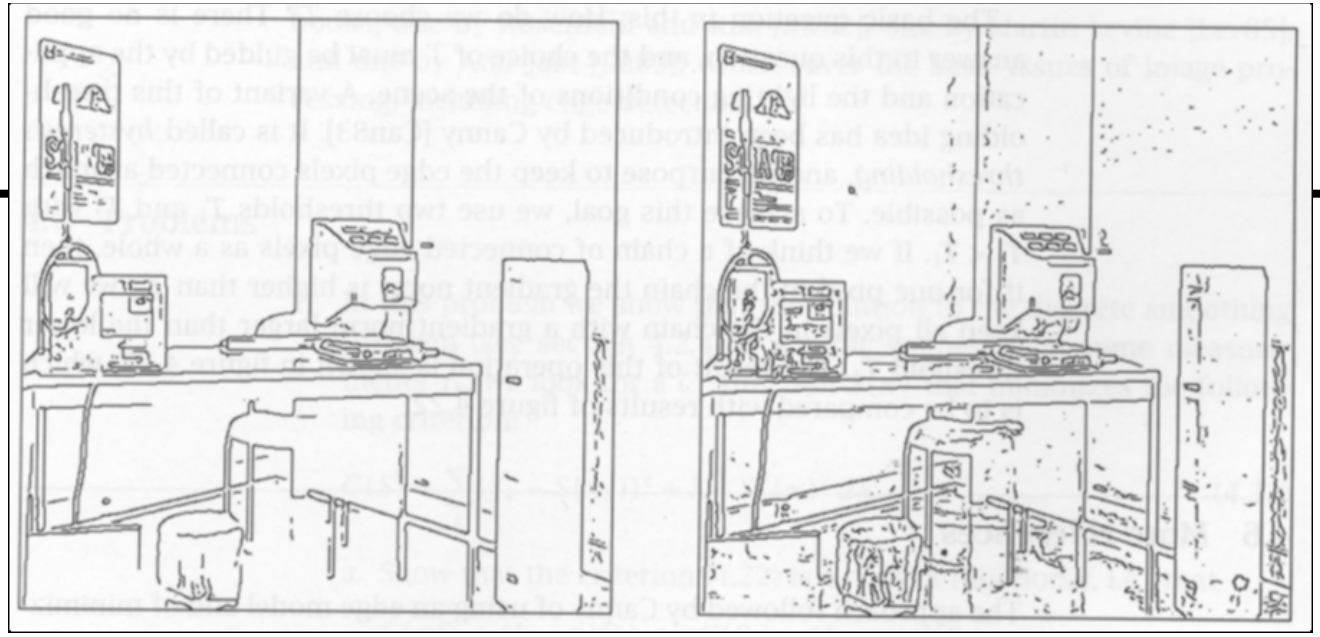
Weaker response but it is connected to a confirmed edge point. Let's keep it.



Continue….

Note: Darker squares illustrate stronger edge response (larger *M*)

T=15



T=5

Hysteresis
T_h=15 T_l = 5

Hysteresis thresholding

Recap: Canny edge detector

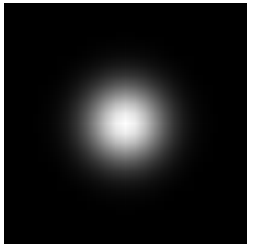
1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
   - Thin wide "ridges" down to single pixel width
4. **Linking and thresholding** (**hysteresis**):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

MATLAB: `edge(image, 'canny');`

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

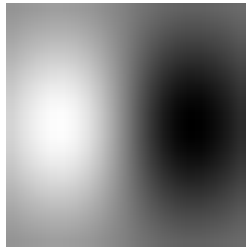
Review: Smoothing vs. derivative filters

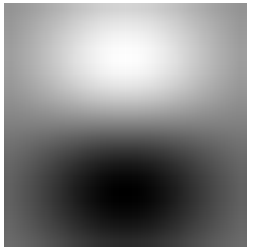
1. Smoothing filters



- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
  - **One:** constant regions are not affected by the filter

2. Derivative filters



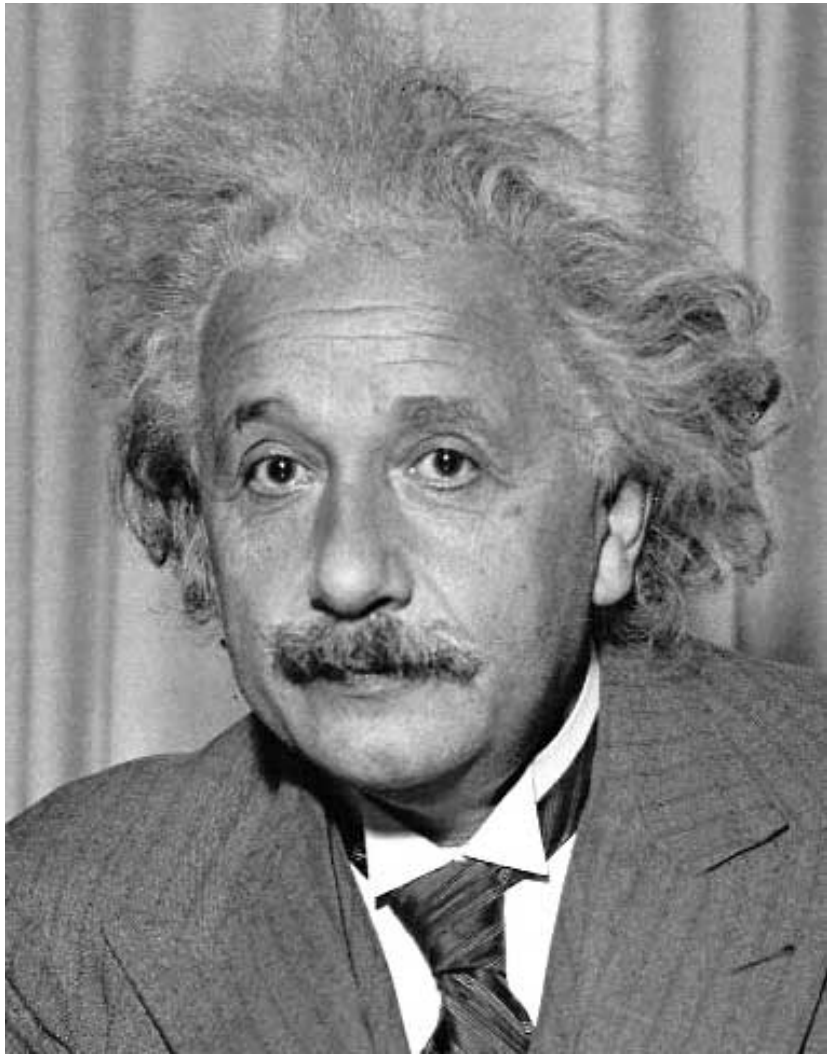


- E.g. Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
  - **Zero:** no response in constant regions
- High absolute value at points of high contrast

3. Filtering as template matching

- Goal: find  in image

- Main challenge: What is a good similarity or distance measure between two patches?



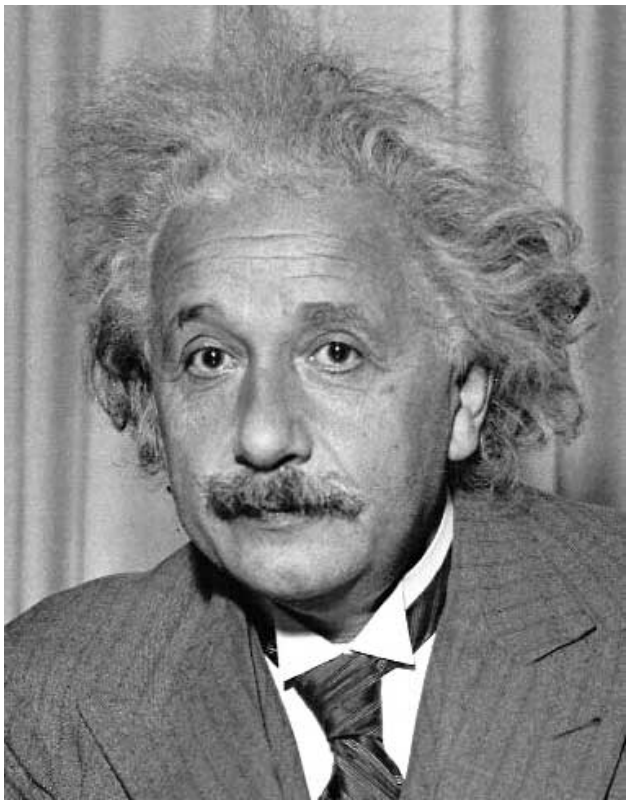
Slide by Derek Hoiem

Matching with filters

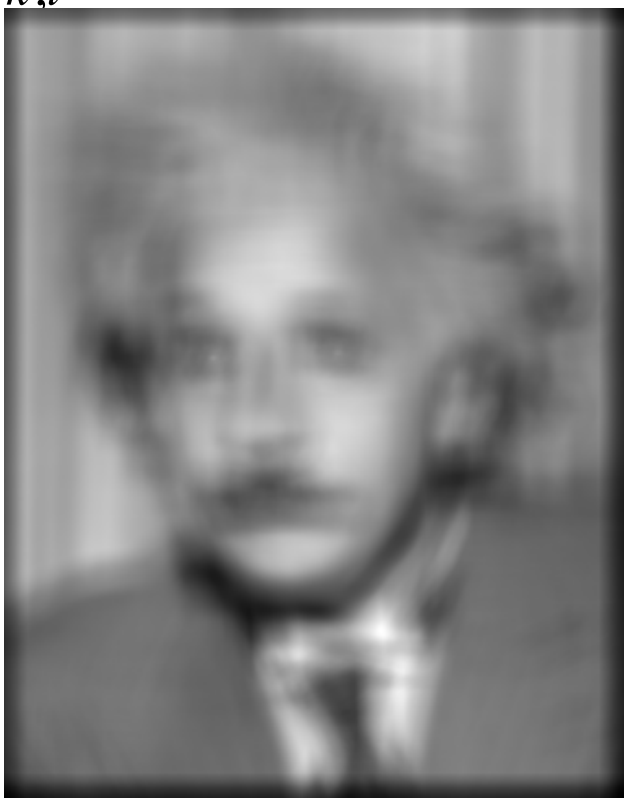
- Goal: find  in image
- Method 0: filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

f = image
g = filter



Input



Filtered Image

What went wrong?

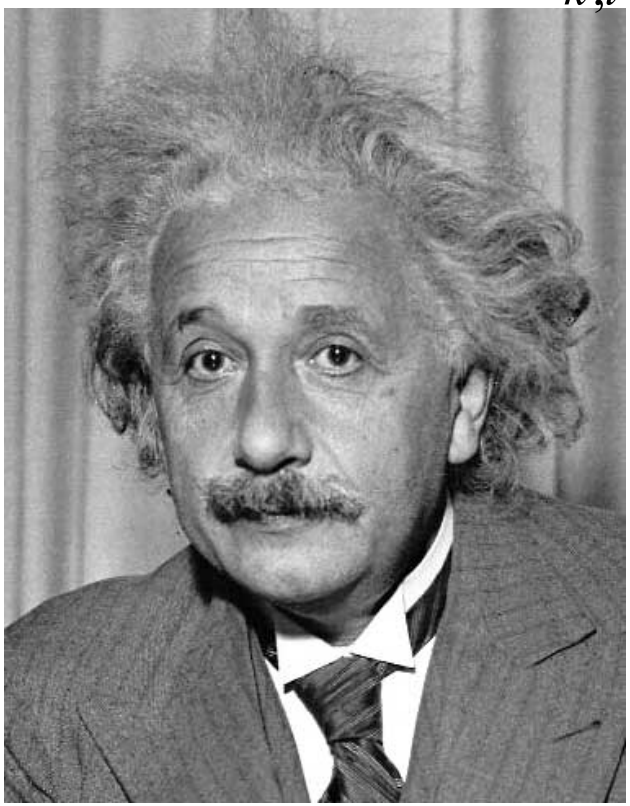
Slide by Derek Hoiem

Matching with filters

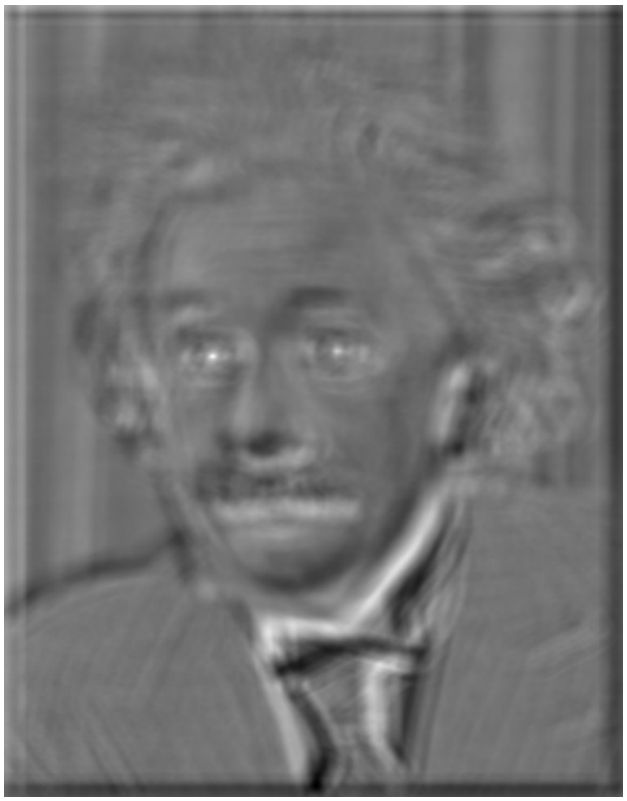
- Goal: find  in image
- Method 1: filter the image with zero-mean eye

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g})\,(f[m+k,n+l])$$

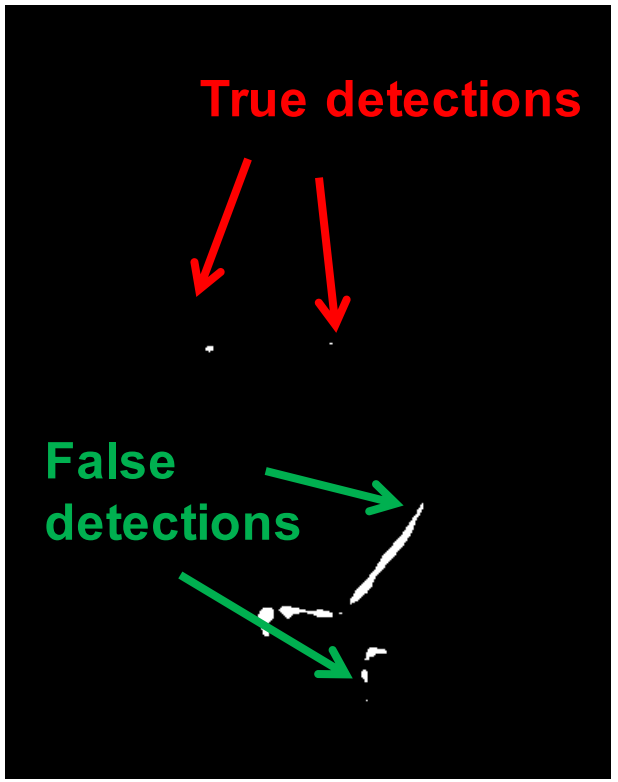
mean of template g



Input



Filtered Image (scaled)

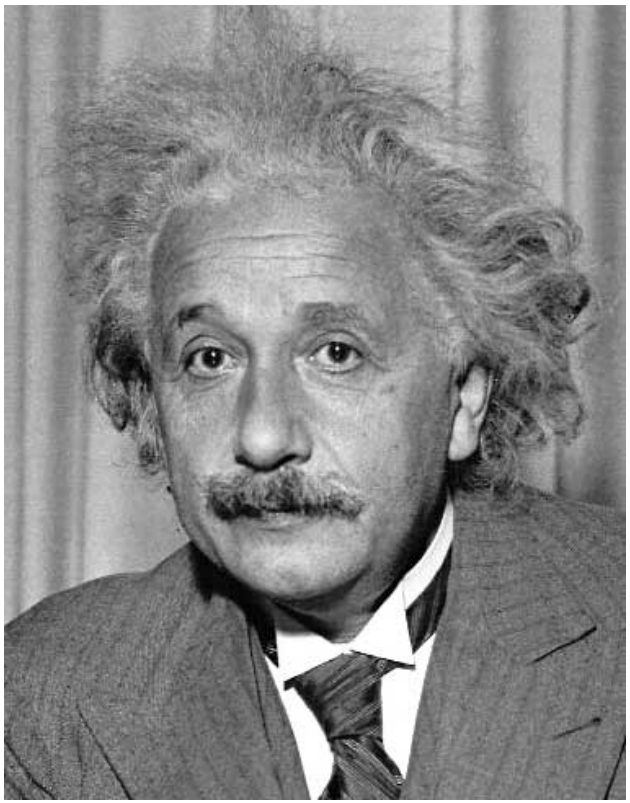


Thresholded Image

Matching with filters

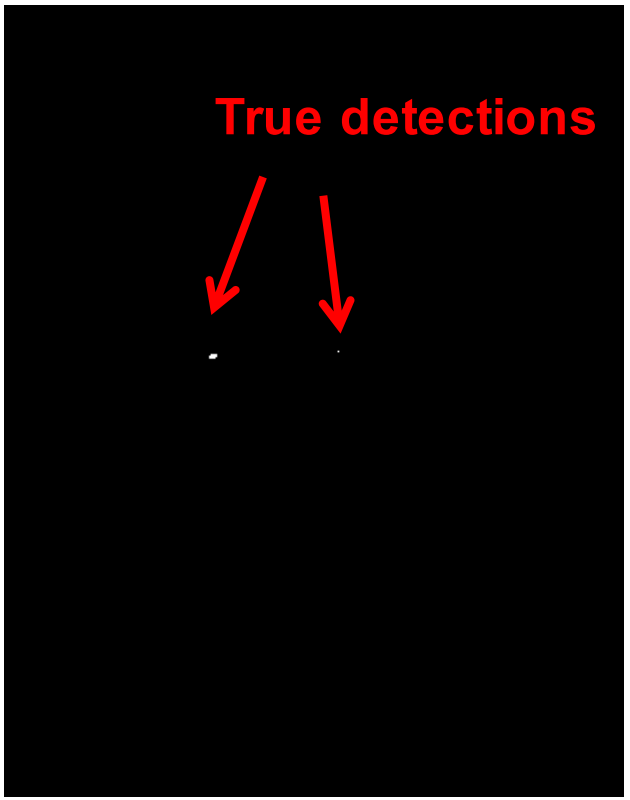
- Goal: find  in image
- Method 2: SSD (L2-norm)

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input

1- sqrt(SSD)



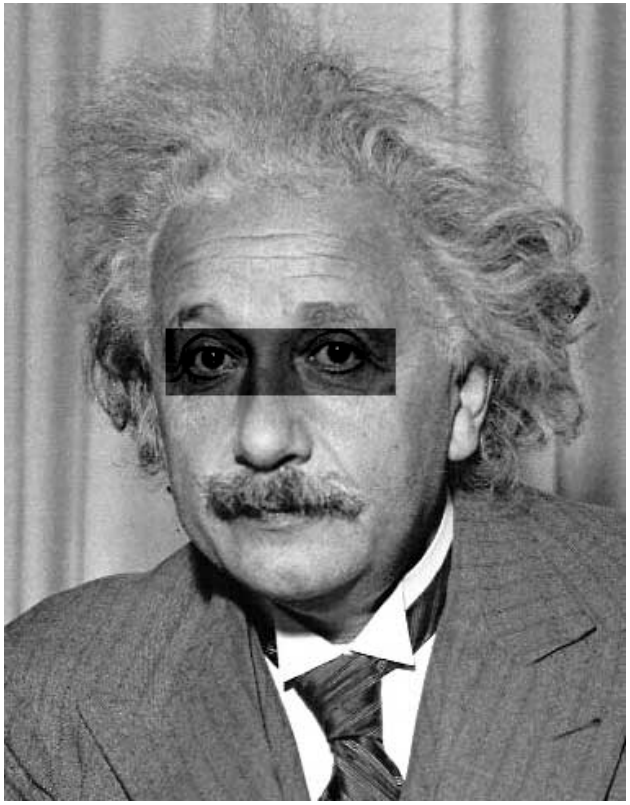
Thresholded Image

Matching with filters

- Goal: find  in image
- Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$

What's the potential downside of SSD?



Input

1- sqrt(SSD)

Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation

mean template

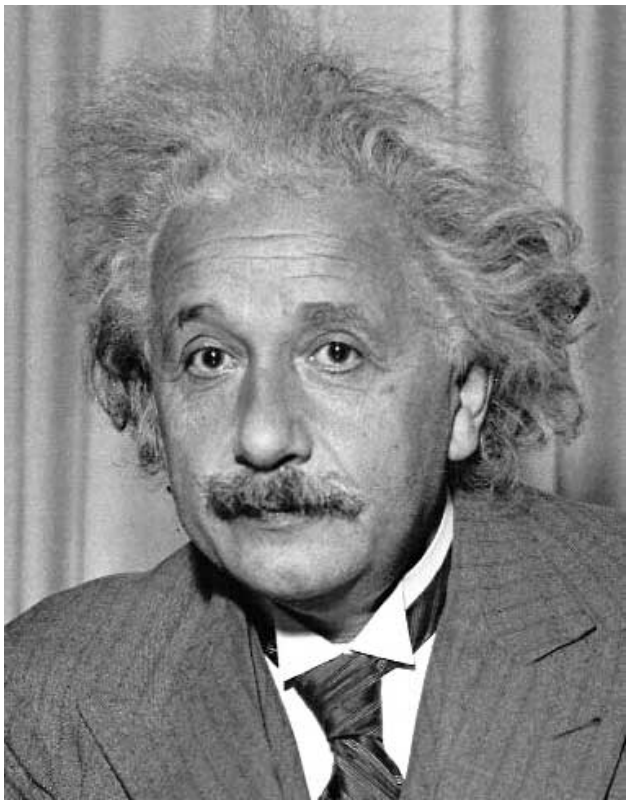
mean image patch

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m+k,n+l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Matlab: `normxcorr2(template, im)`

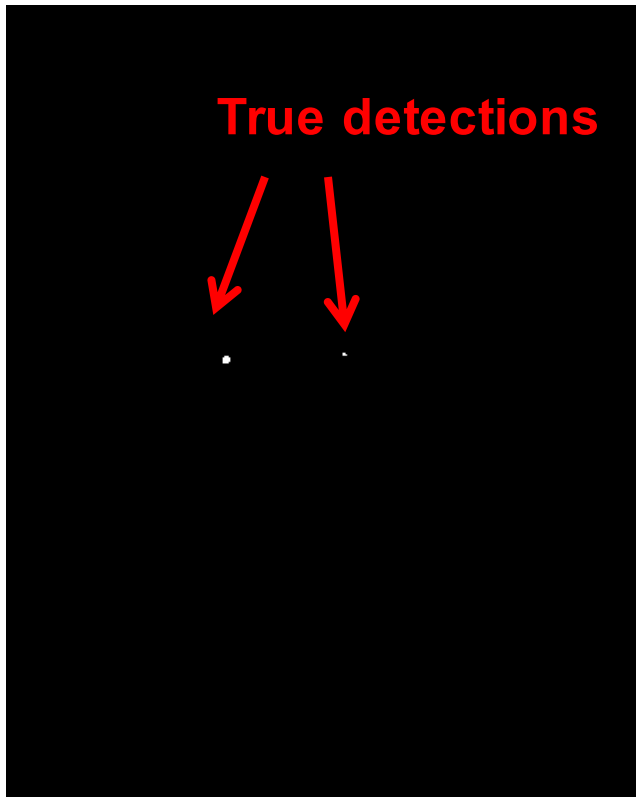
Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation



Input

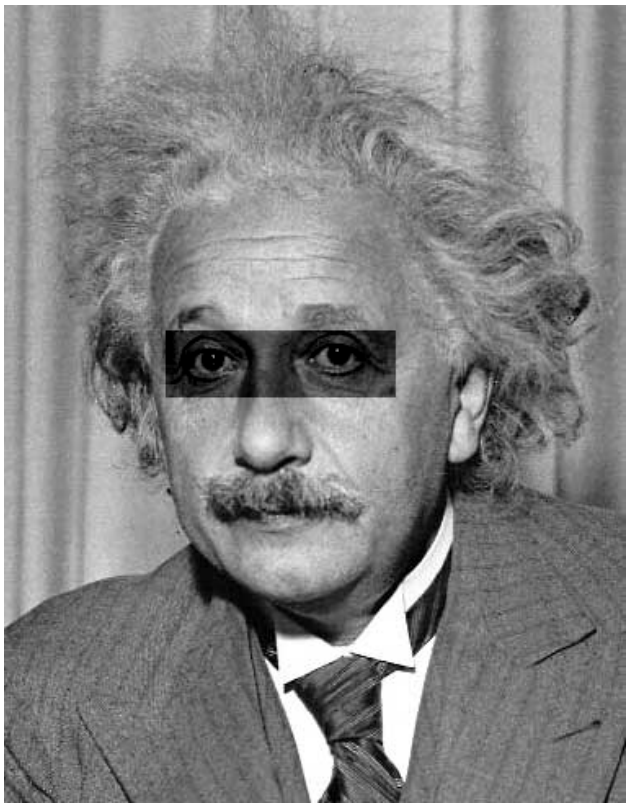
Normalized X-Correlation



Thresholded Image

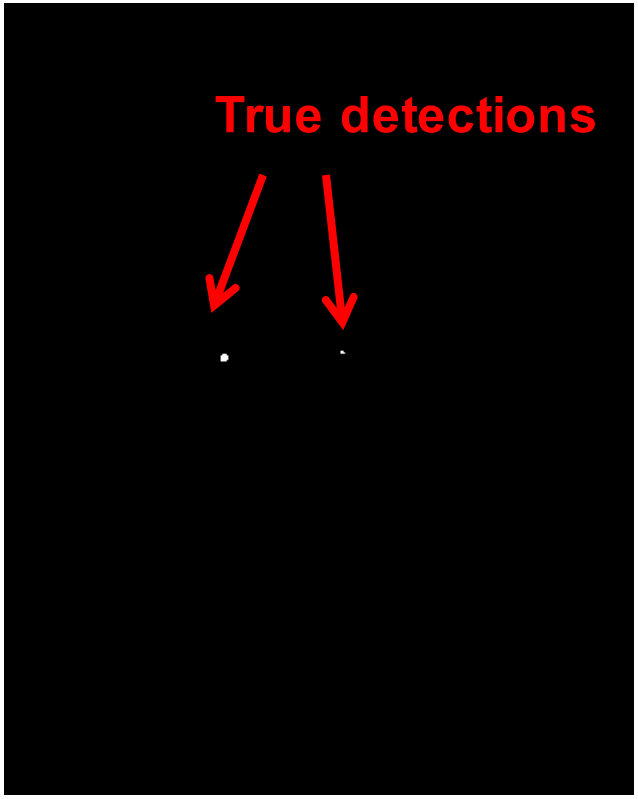
Matching with filters

- Goal: find  in image
- Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



Thresholded Image

Q: What is the best method to use?

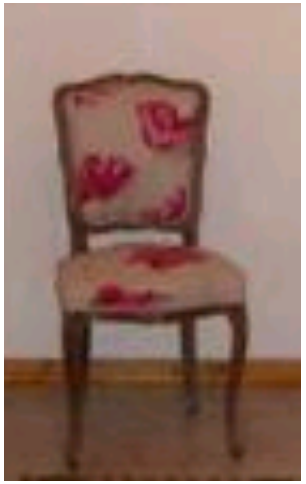
A: Depends

- Zero-mean filter: fastest but not a great matcher
- SSD: next fastest, sensitive to overall intensity
- Normalized cross-correlation: slowest, invariant to local average intensity and contrast

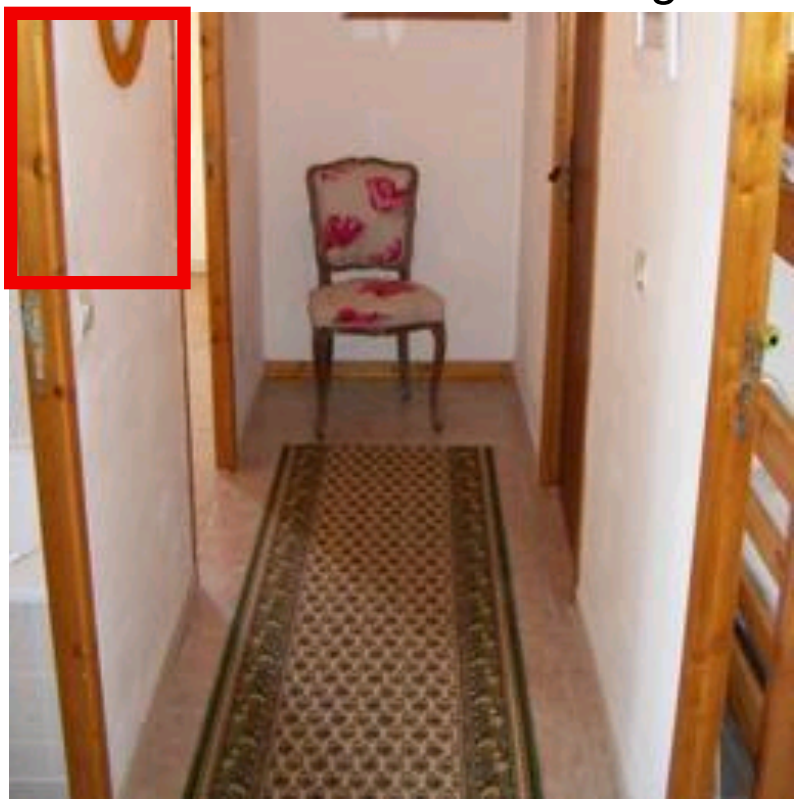
Slide by Derek Hoiem

Object recognition
Is it really so hard?

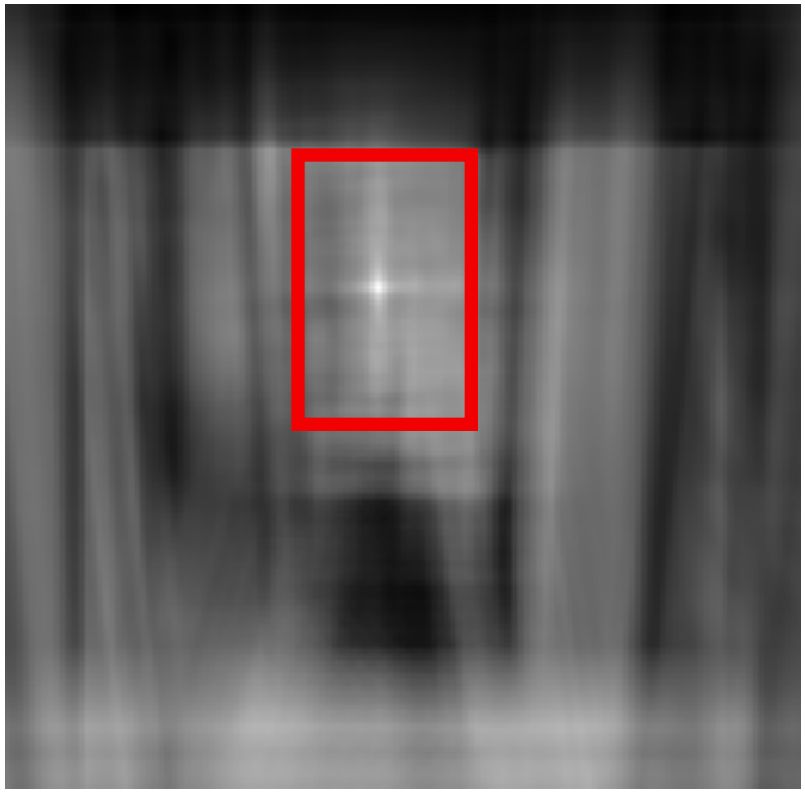
This is a chair



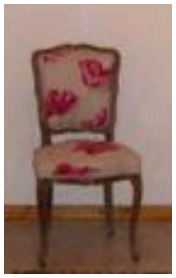
Find the chair in this image



Output of normalized correlation

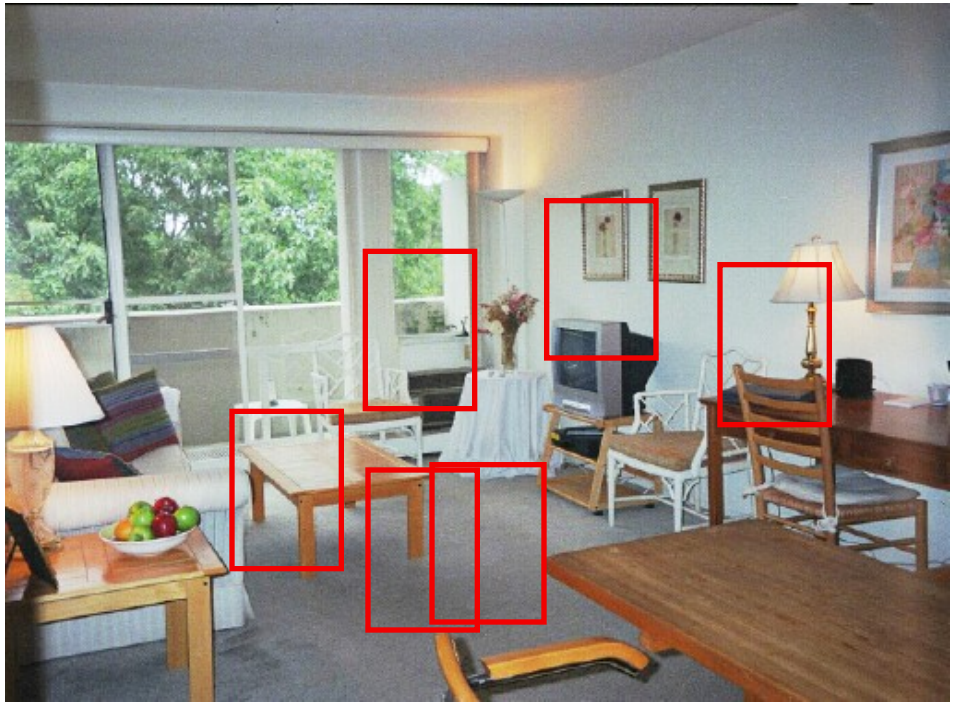


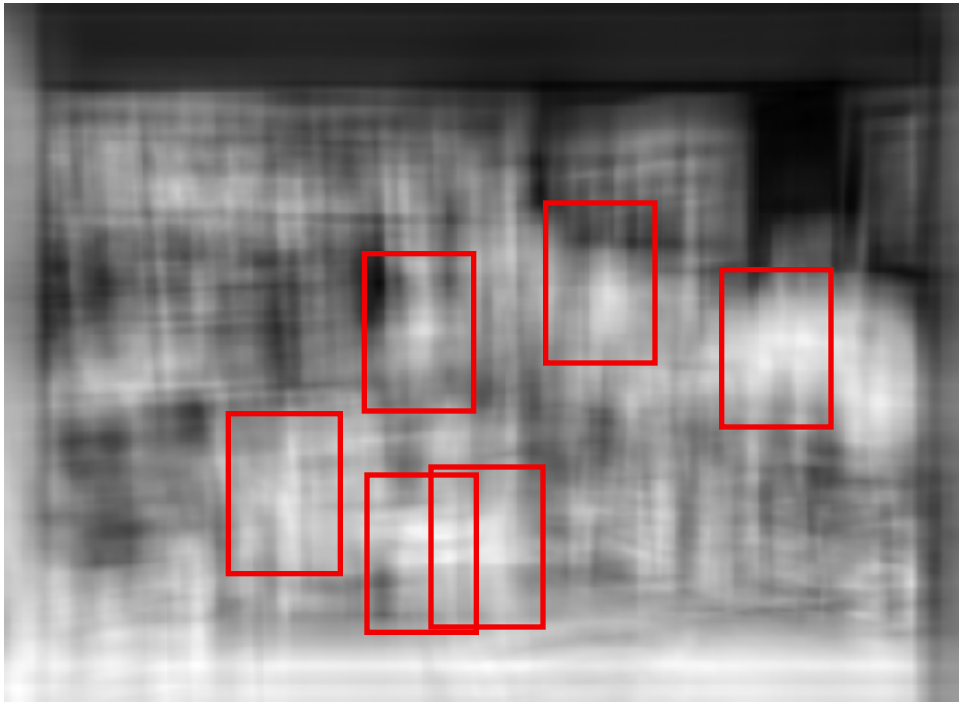
Slide by Antonio Torralba



Object recognition
Is it really so hard?

Find the chair in this image





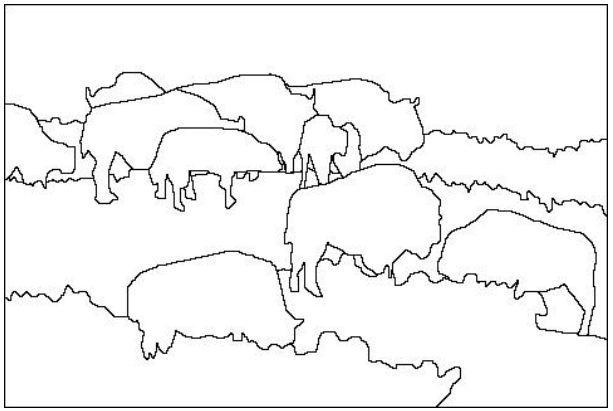
Slide by Antonio Torralba

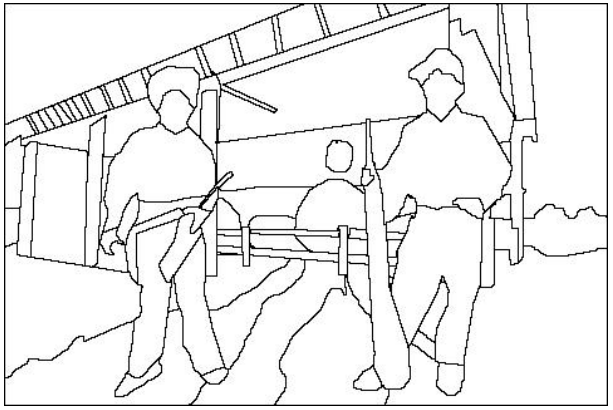
Recognition: Instance vs. Category

- Instance recognition:
  - "Find me this particular chair again"
  - Often simple template matching works OK
    - Even better with many small templates, a.k.a. feature descriptors (later in the course)
- Category recognition:
  - "find me all chairs"
  - Templates don't work. Why?
  - Focus on things that might be invariant across the category
  - Hugely dependent on human vision

Edge detection is just the beginning…

image | human segmentation | gradient magnitude

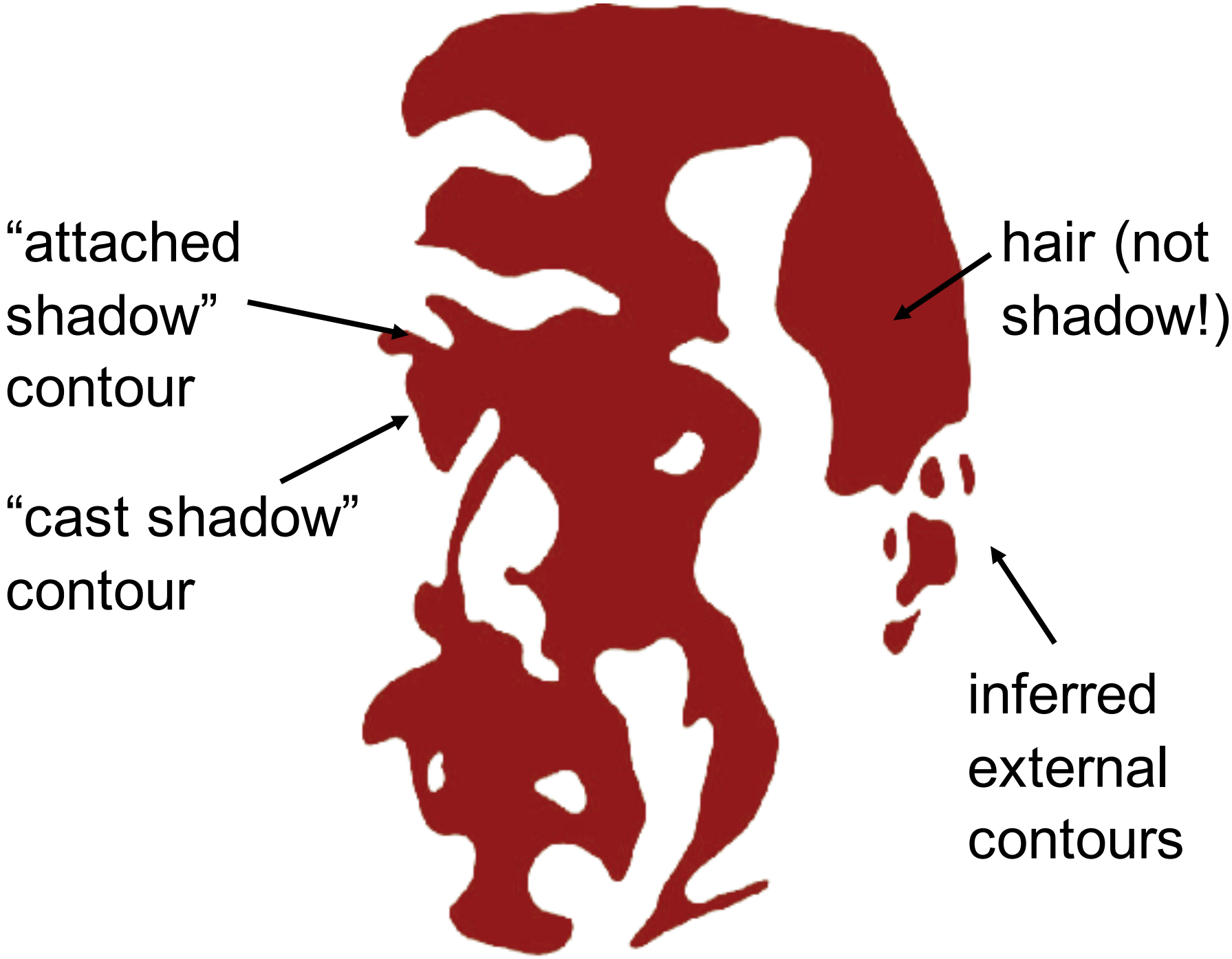






Berkeley segmentation database:

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

two-tone images


“attached shadow” contour
“cast shadow” contour
hair (not shadow!)
inferred external contours

What is Texture?

- Texture depicts spatially repeating patterns
- Many natural phenomena are textures

radishes

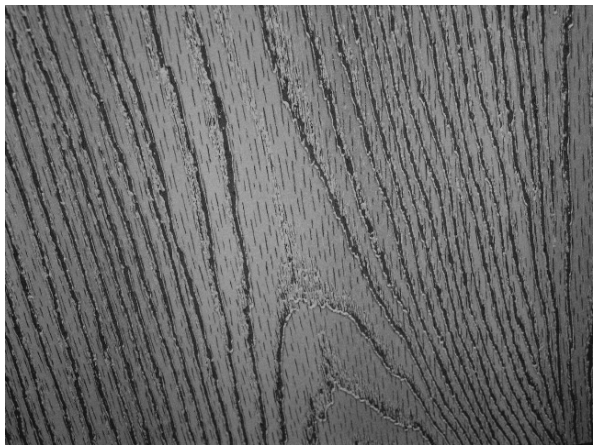
rocks

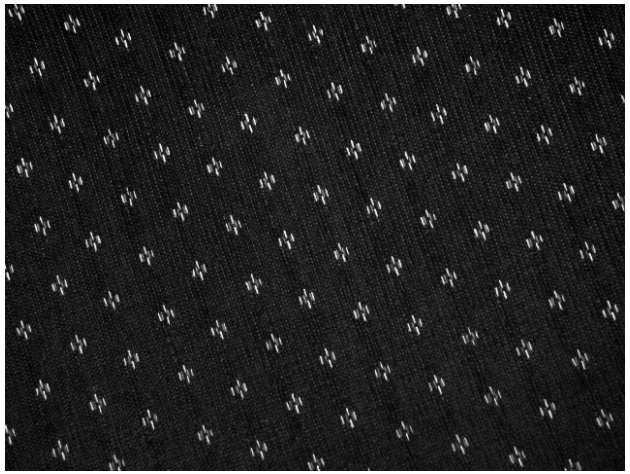
yogurt

Texture as “stuff”

Source: Forsyth

Texture and Material





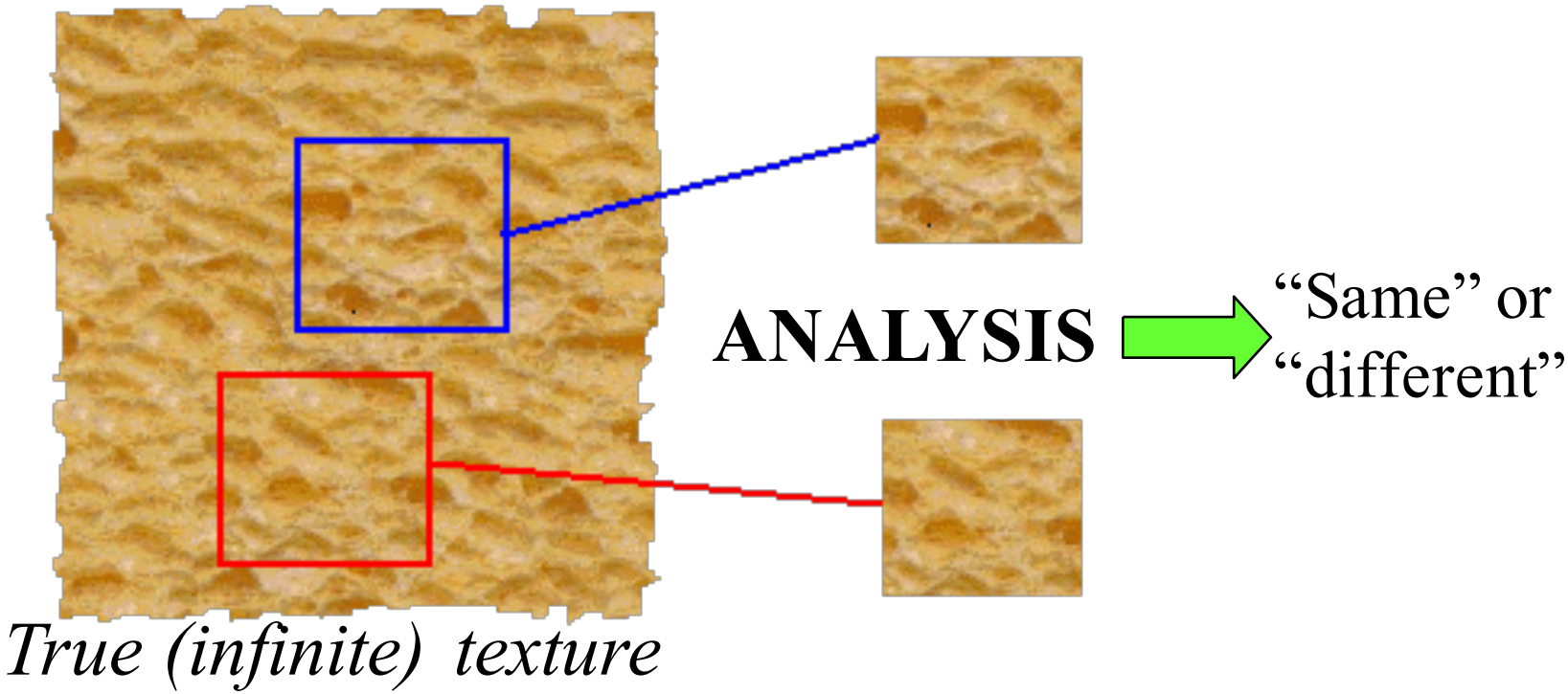
http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

When are two textures similar?



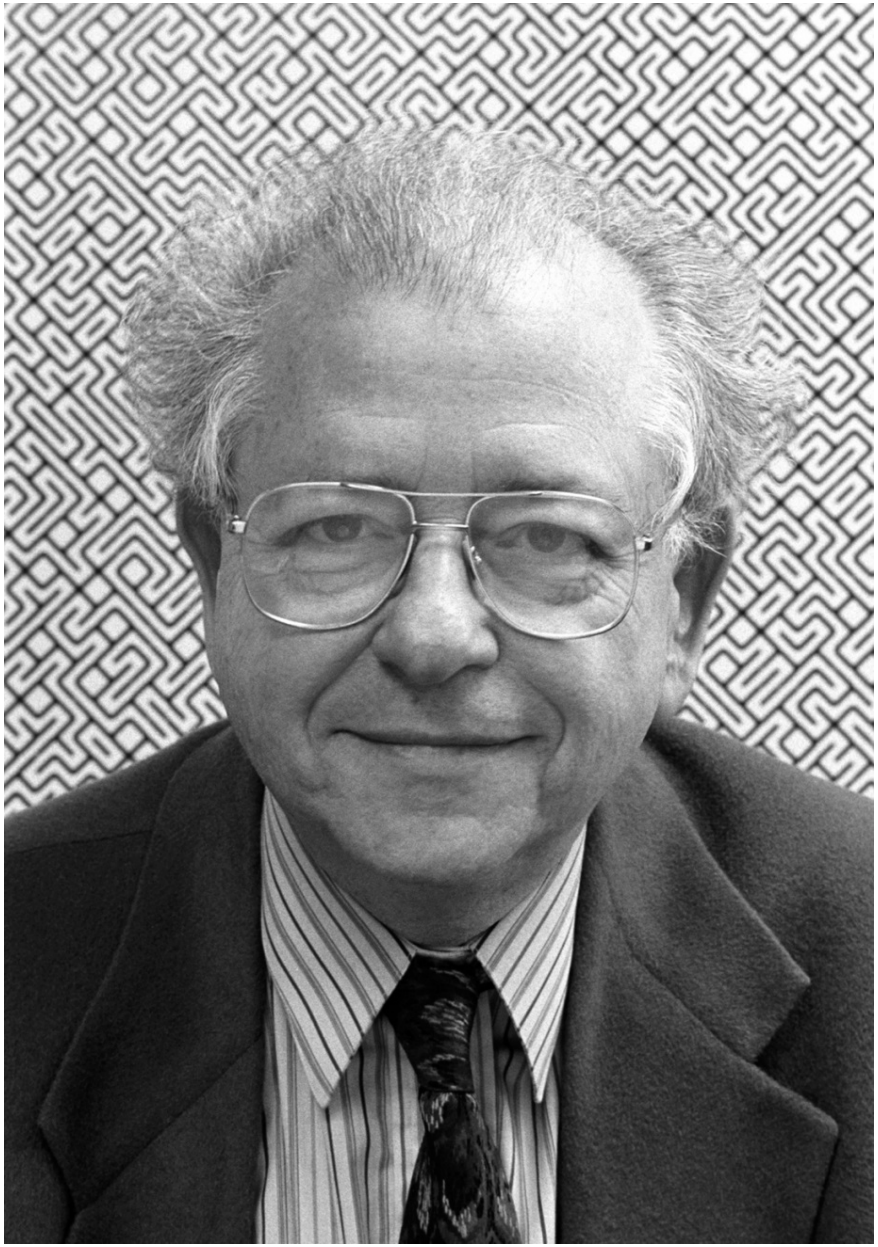


Texture Analysis

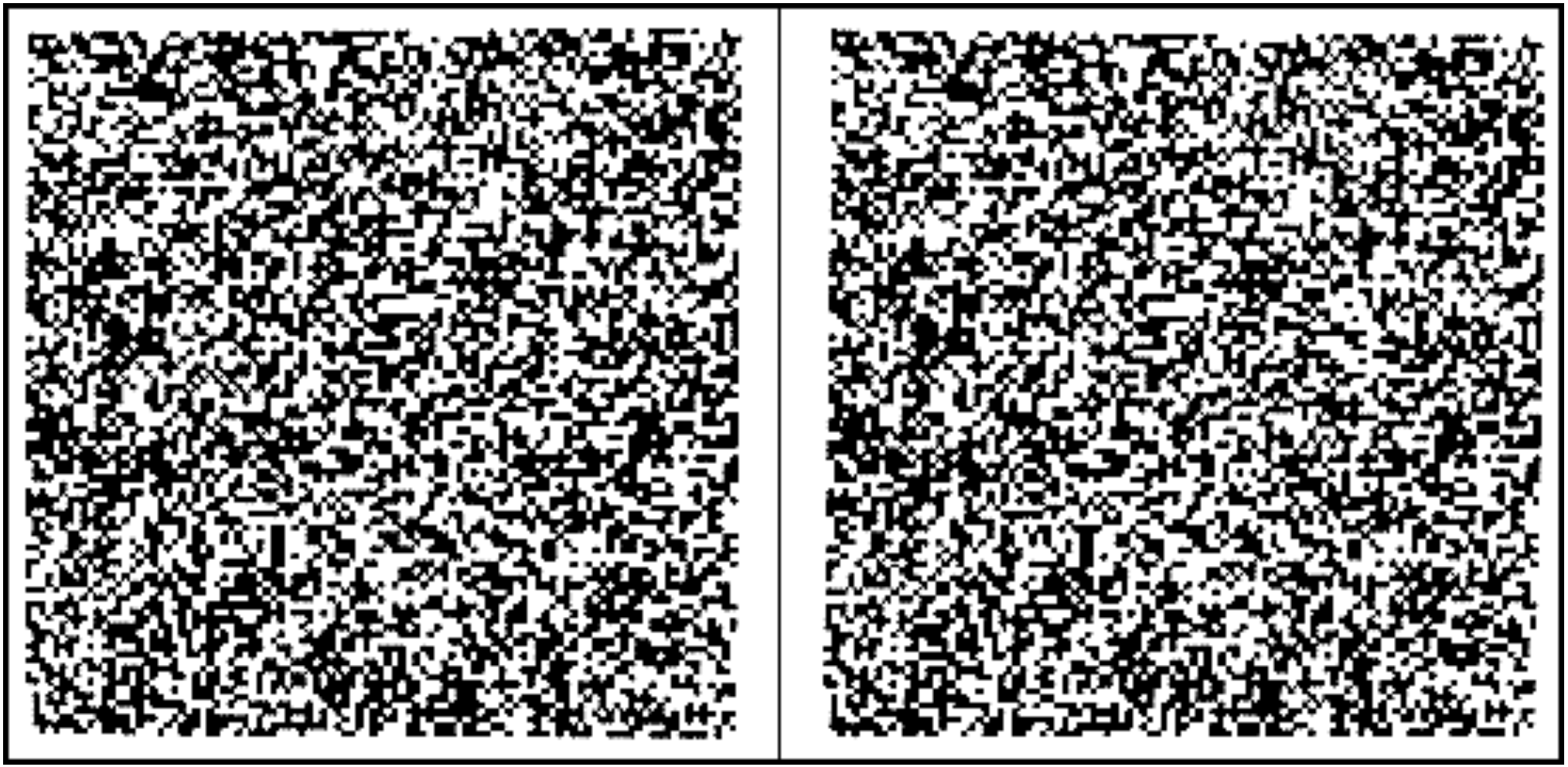


Compare textures and decide if they're made of the same "stuff".

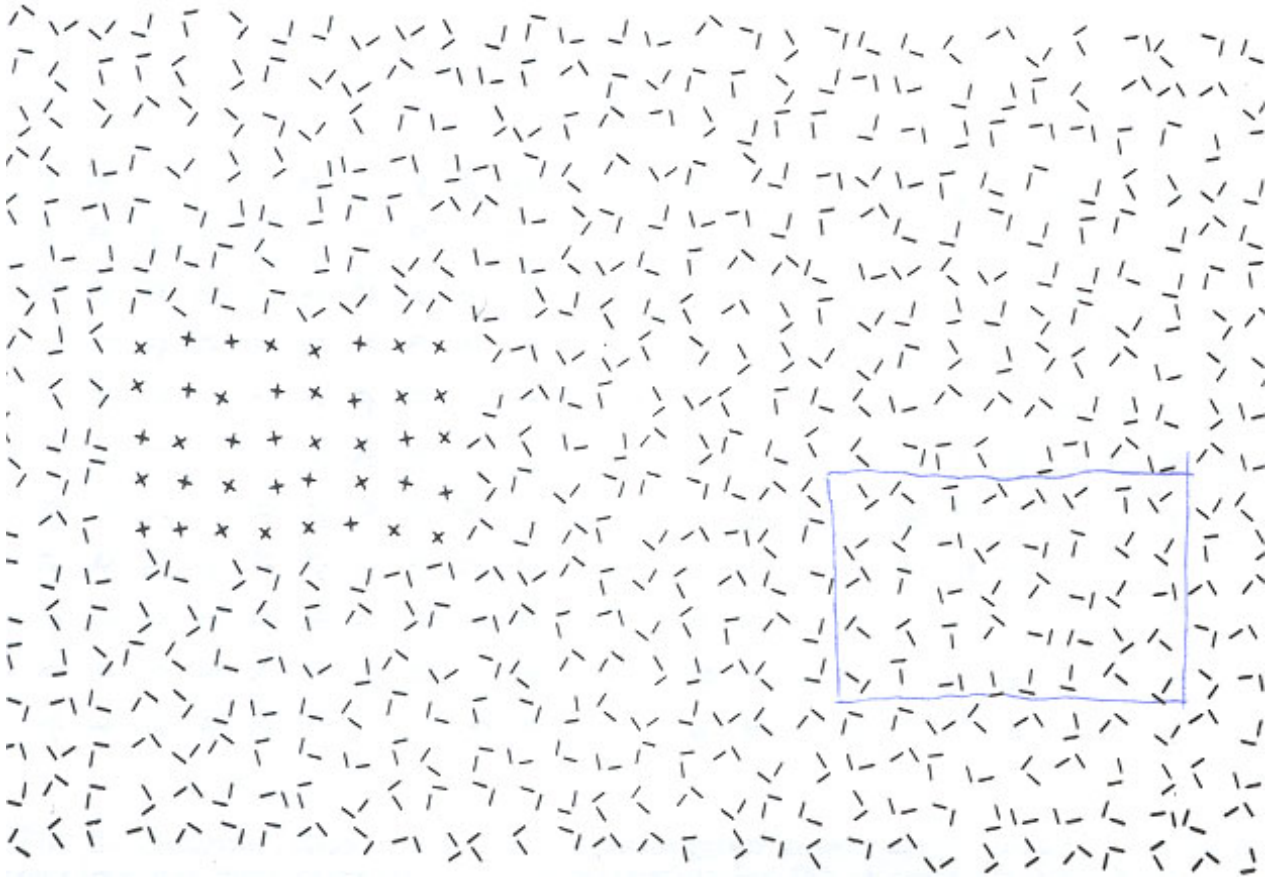
Béla Julesz, father of texture



Random Dot Stereograms



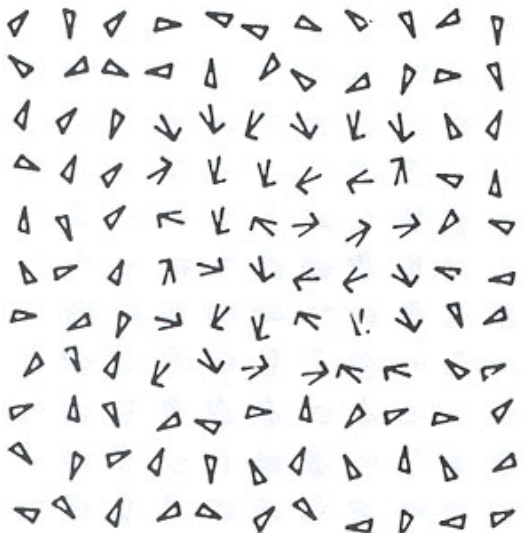
Texton Discrimination (Julesz)

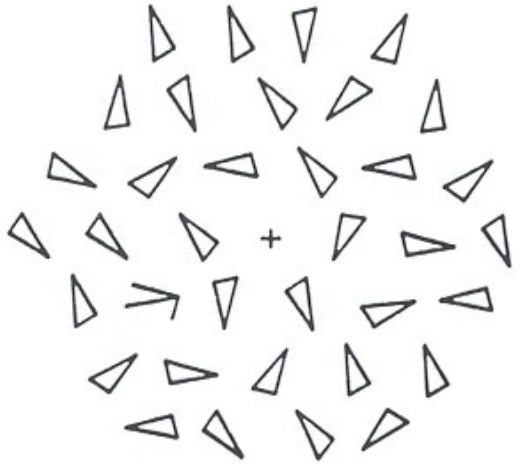


Human vision is sensitive to the difference of some types of elements and appears to be “numb” on other types of differences.

Search Experiment I

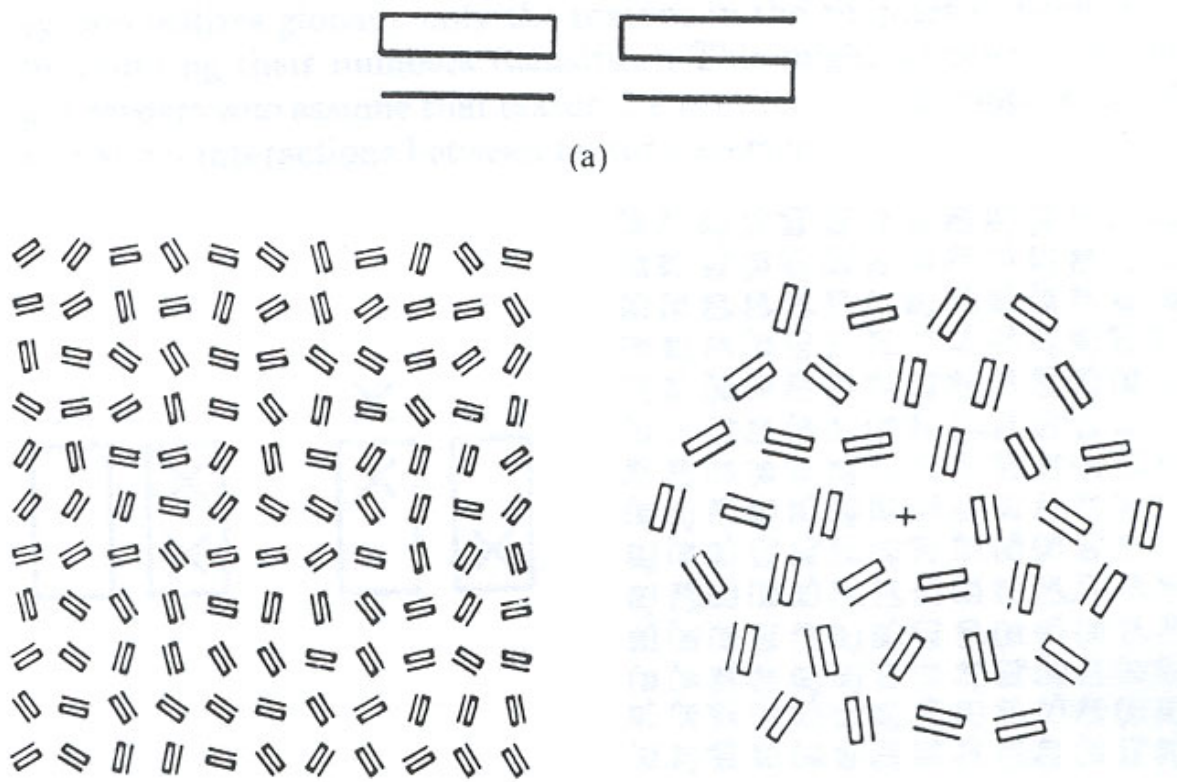
(a)





The subject is told to detect a target element in a number of background elements.
In this example, the detection time is independent of the number of background elements.

Search Experiment II



In this example, the detection time is proportional to the number of background elements, And thus suggests that the subject is doing element-by-element scrutiny.

Heuristic (Axiom) I

Julesz then conjectured the following axiom:

Human vision operates in two distinct modes:

1. Preattentive vision

   parallel, instantaneous *(~100--200ms)*, without scrutiny,
   independent of the number of patterns, covering a large visual field.

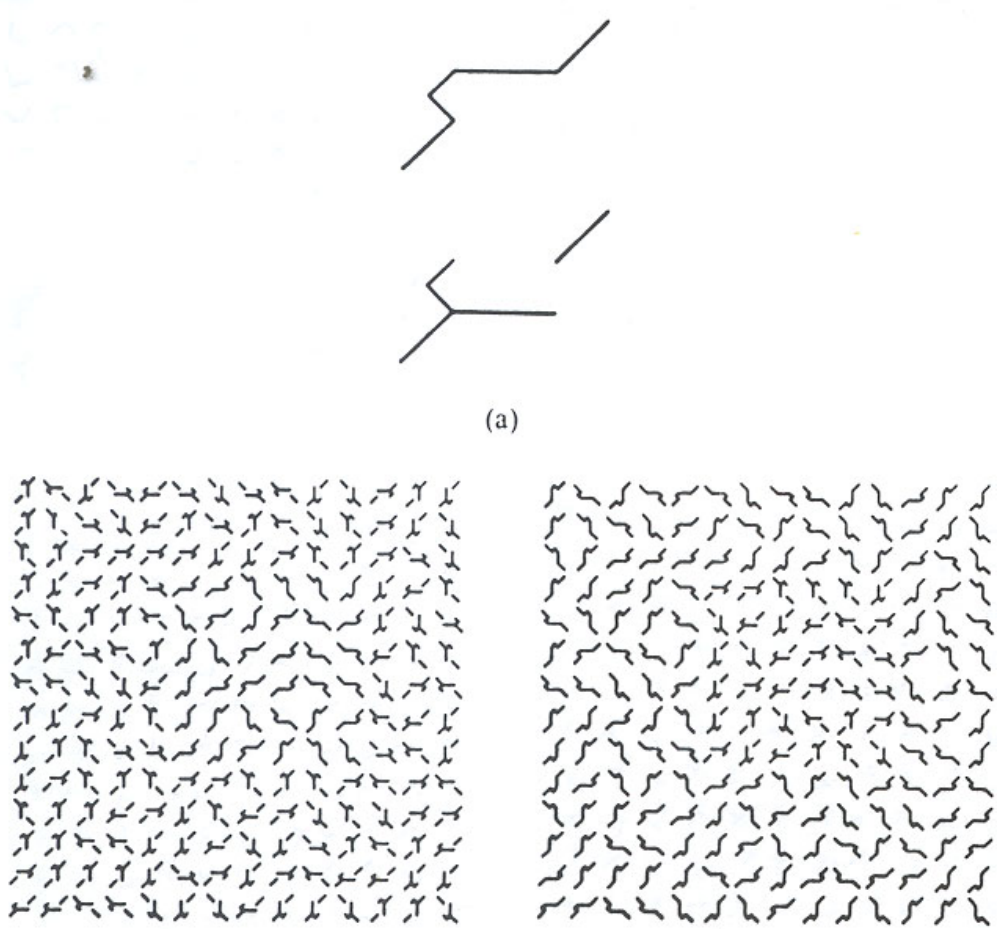
2. Attentive vision

   serial search by focal attention in 50ms steps limited to small aperture.

Examples

Pre-attentive vision is sensitive to size/width, orientation changes

Examples



Sensitive to number of terminators

Left: fore-back
Right: back-fore

See previous examples
For cross and terminators

Heuristic (Axiom) II

Textons are the fundamental elements in preattentive vision, including

1. Elongated blobs

   rectangles, ellipses, line segments with attributes

   color, orientation, width, length, flicker rate.

2. Terminators

   ends of line segments.

3. Crossings of line segments.

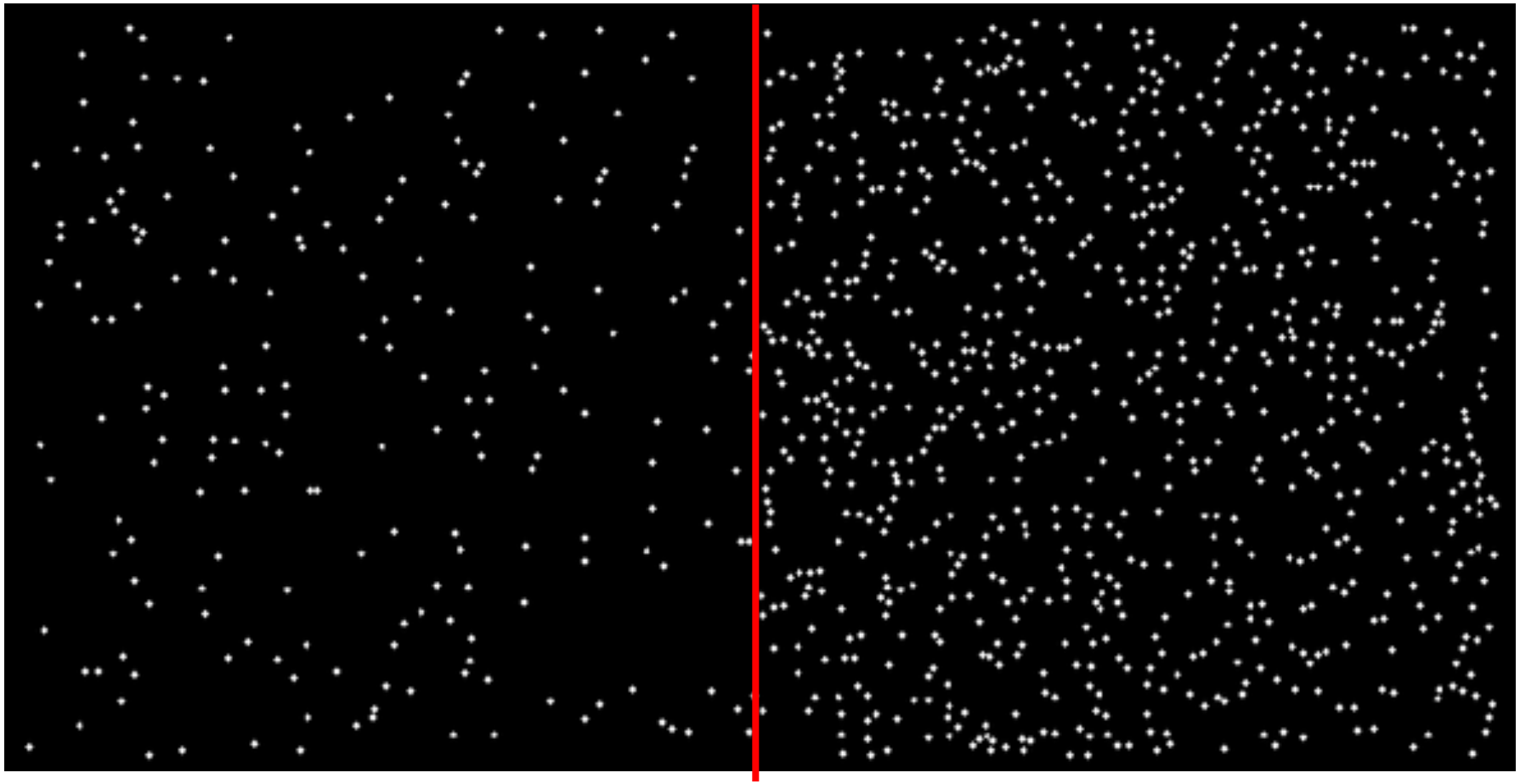
But it is worth noting that Julesz's conclusions are largely based on ensemble of artificial texture patterns. It was infeasible to synthesize natural textures for controlled experiments at that time.

Julesz Conjecture

Textures cannot be spontaneously discriminated if they have the same first-order and second-order statistics and differ only in their third-order or higher-order statistics.

(later proved wrong)

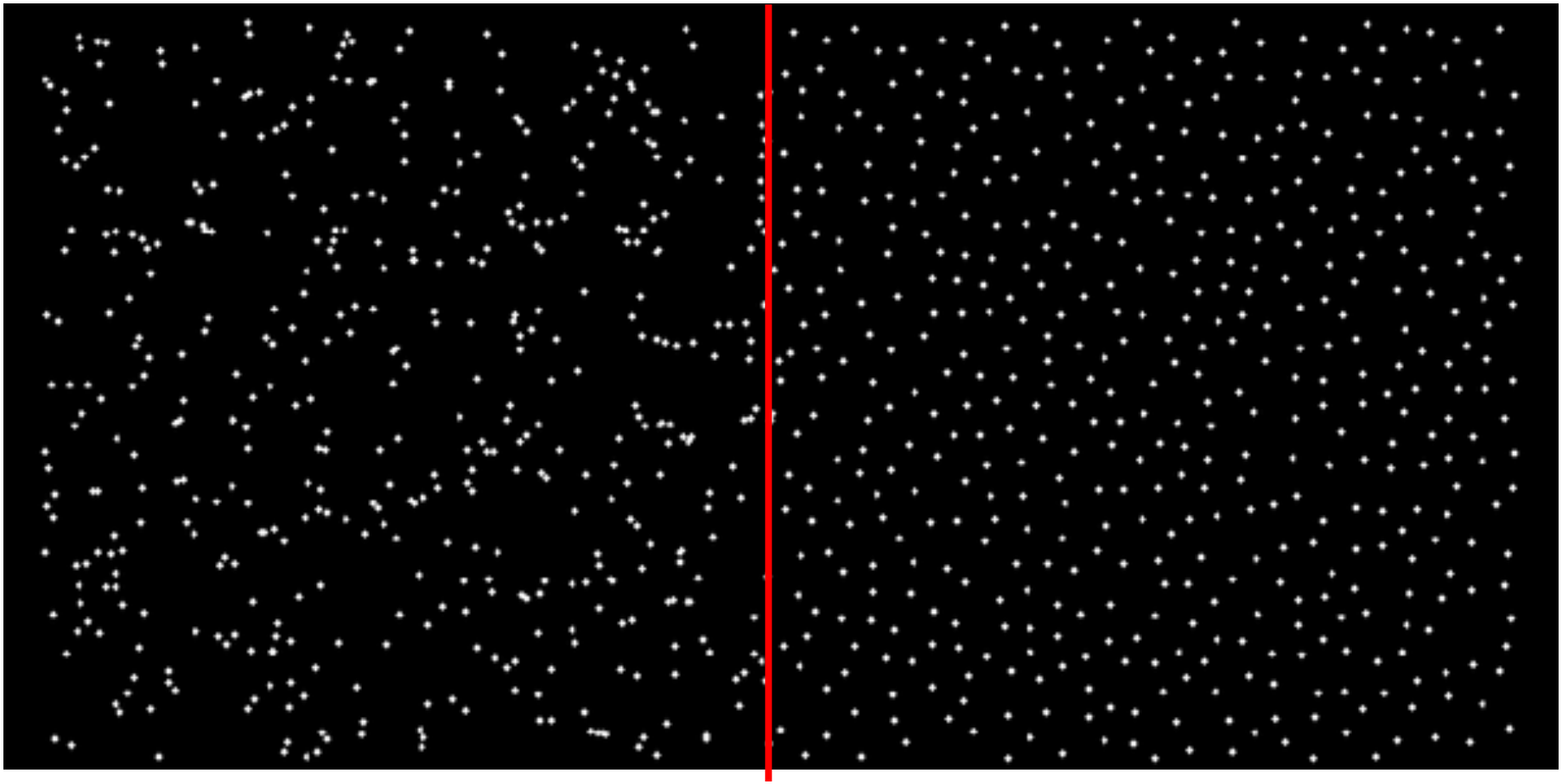
1st Order Statistics



5% white

20% white

2nd Order Statistics



10% white