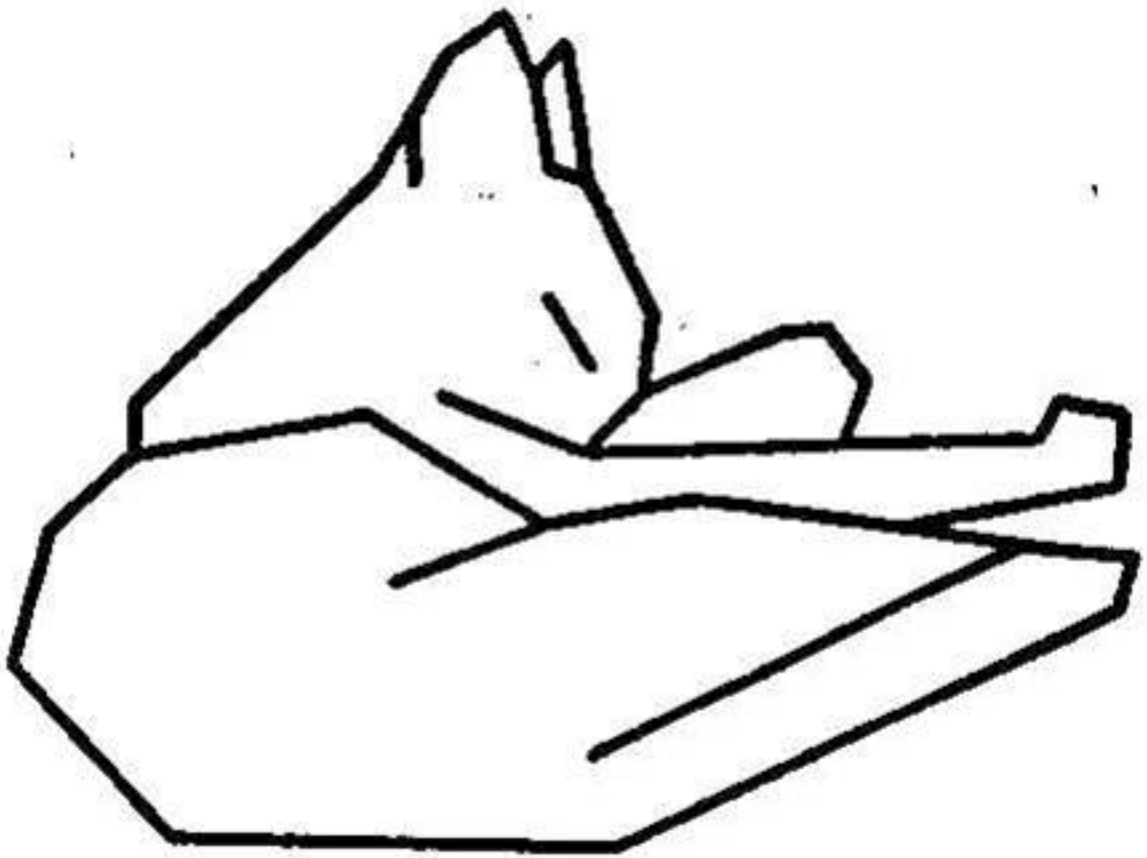
Edge Detection

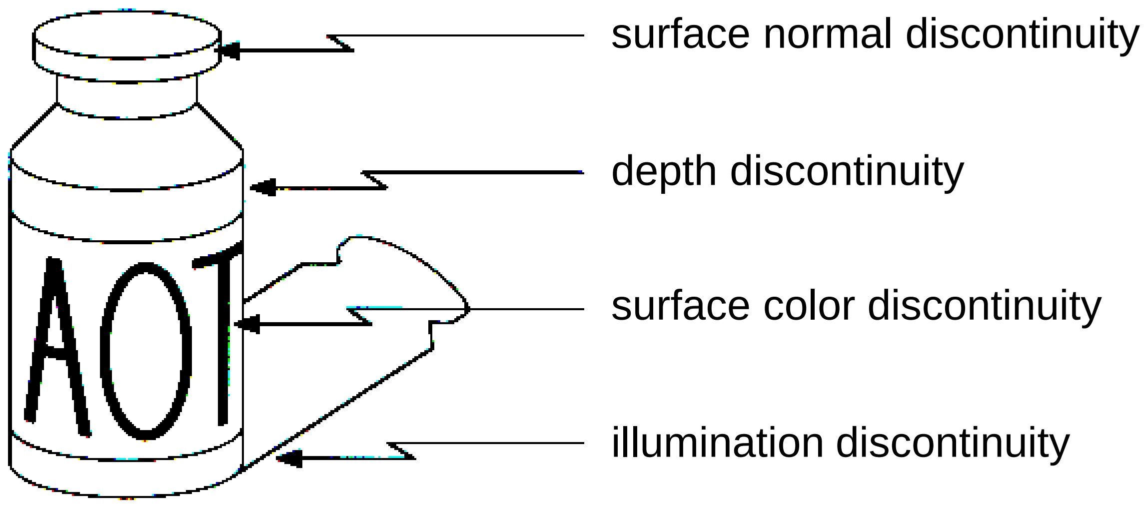
ECE/CSE 576
Linda Shapiro

Edge



Attneave's Cat (1954)

Origin of edges



Edges are caused by a variety of factors.

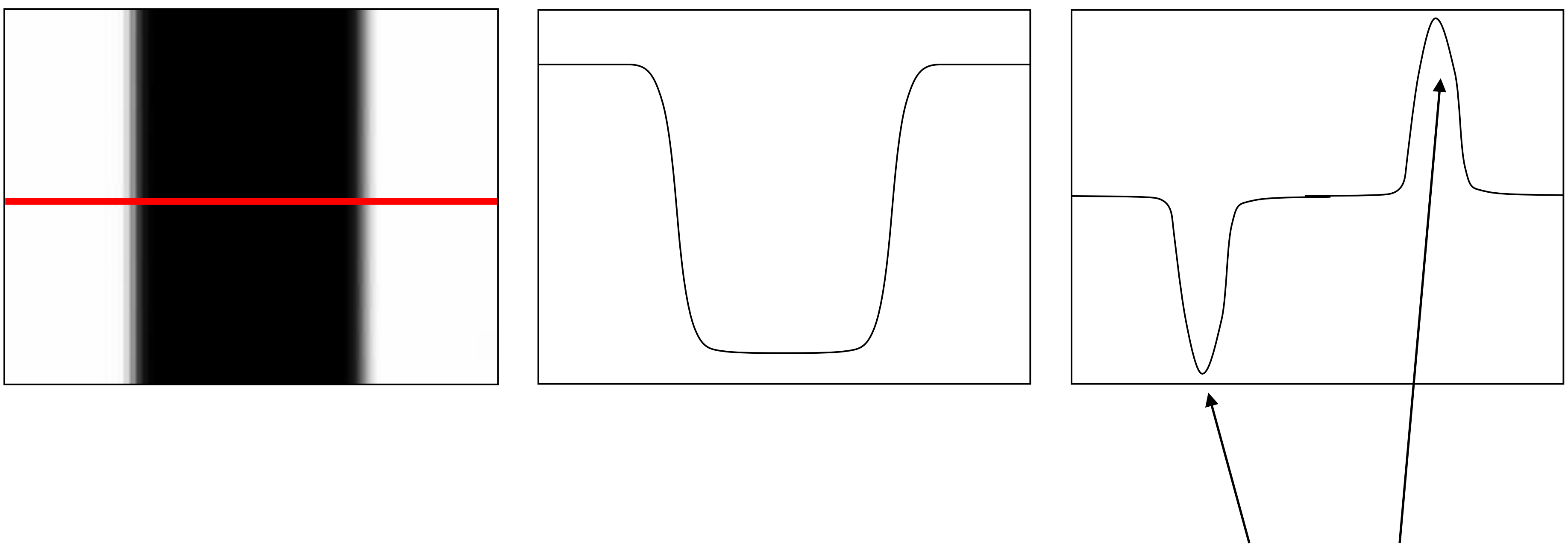
Characterizing edges

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

image

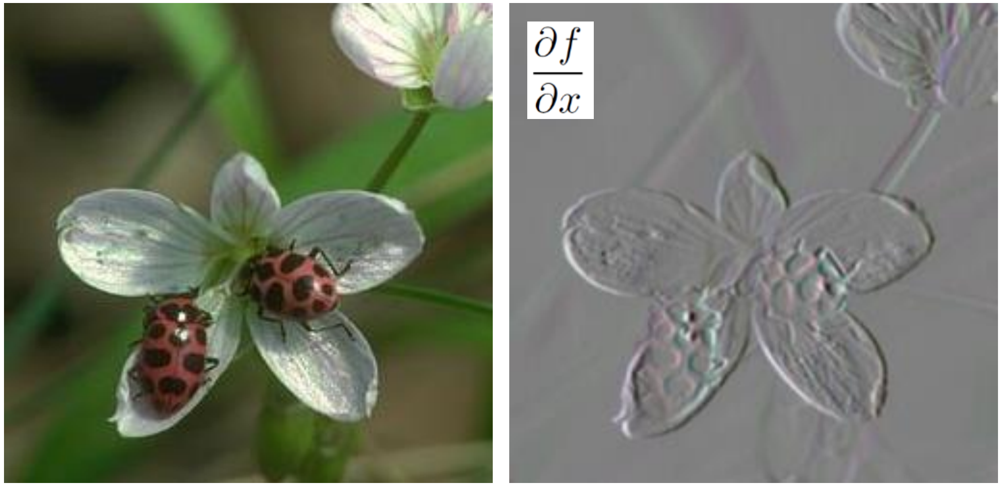
intensity function
(along horizontal scanline)

first derivative



edges correspond to
extrema of derivative

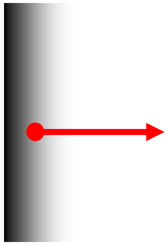
Image gradient



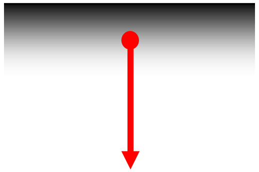
- The gradient of an image:

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

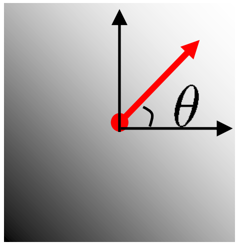
- The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



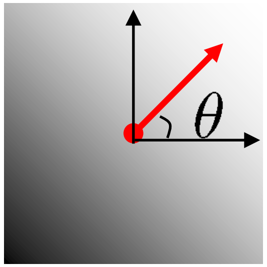
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The discrete gradient

- How can we differentiate a *digital* image F[x,y]?
  - Option 1: reconstruct a continuous image, then take gradient
  - Option 2: take discrete derivative ("finite difference")

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

Simple image gradient



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

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

How would you implement this as a filter?

0	-1	1

The gradient direction is given by:

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

How does this relate to the direction of the edge?

perpendicular

The *edge strength* is given by the gradient magnitude

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

or various simplifications

Sobel operator

Who was Sobel?

In practice, it is common to use:

$g_x =$

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

$g_y =$

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

Magnitude:

$$g = \sqrt{g_x^2 + g_y^2}$$

Orientation:

$$\Theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

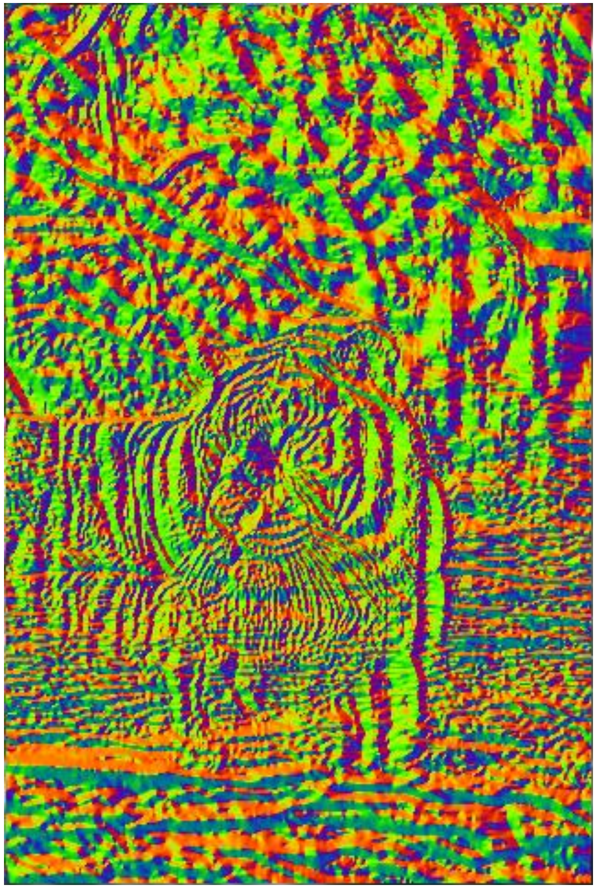
What's the C/C++ function?

Use atan2

Sobel operator

Original

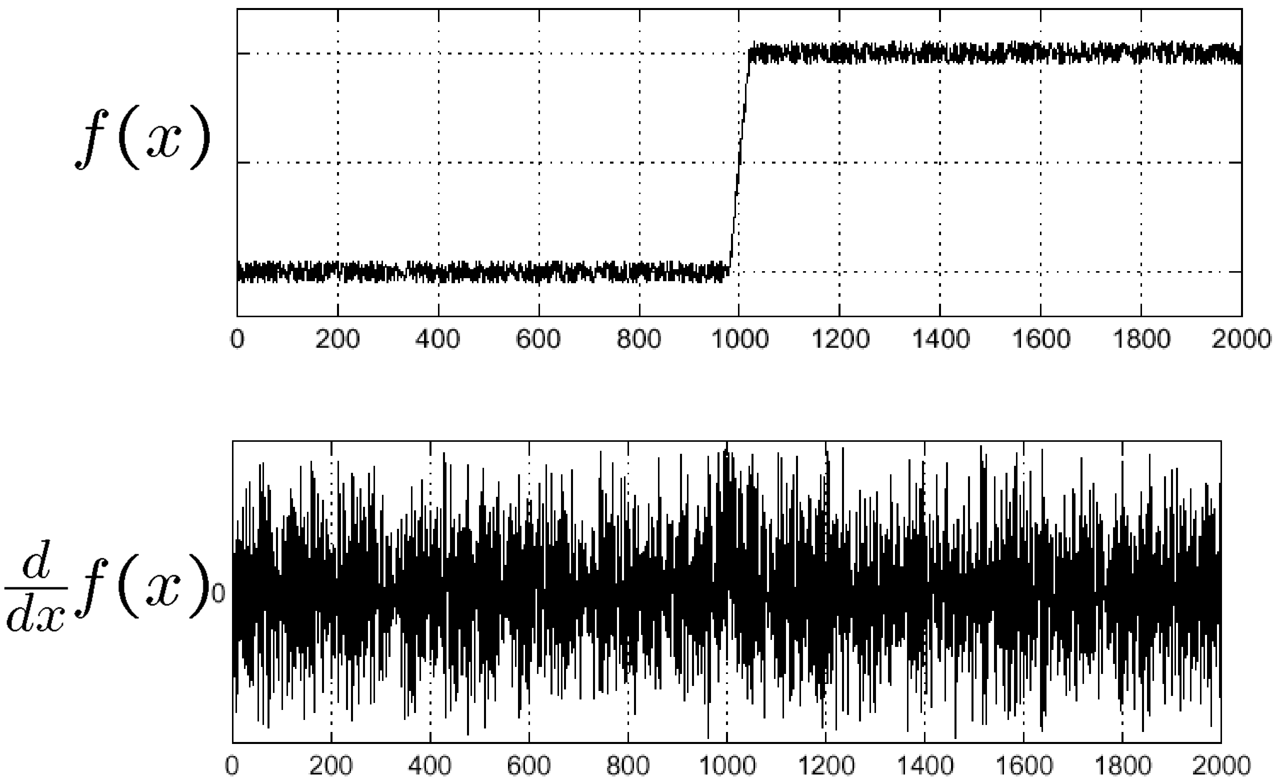
Magnitude



Orientation

Effects of noise

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



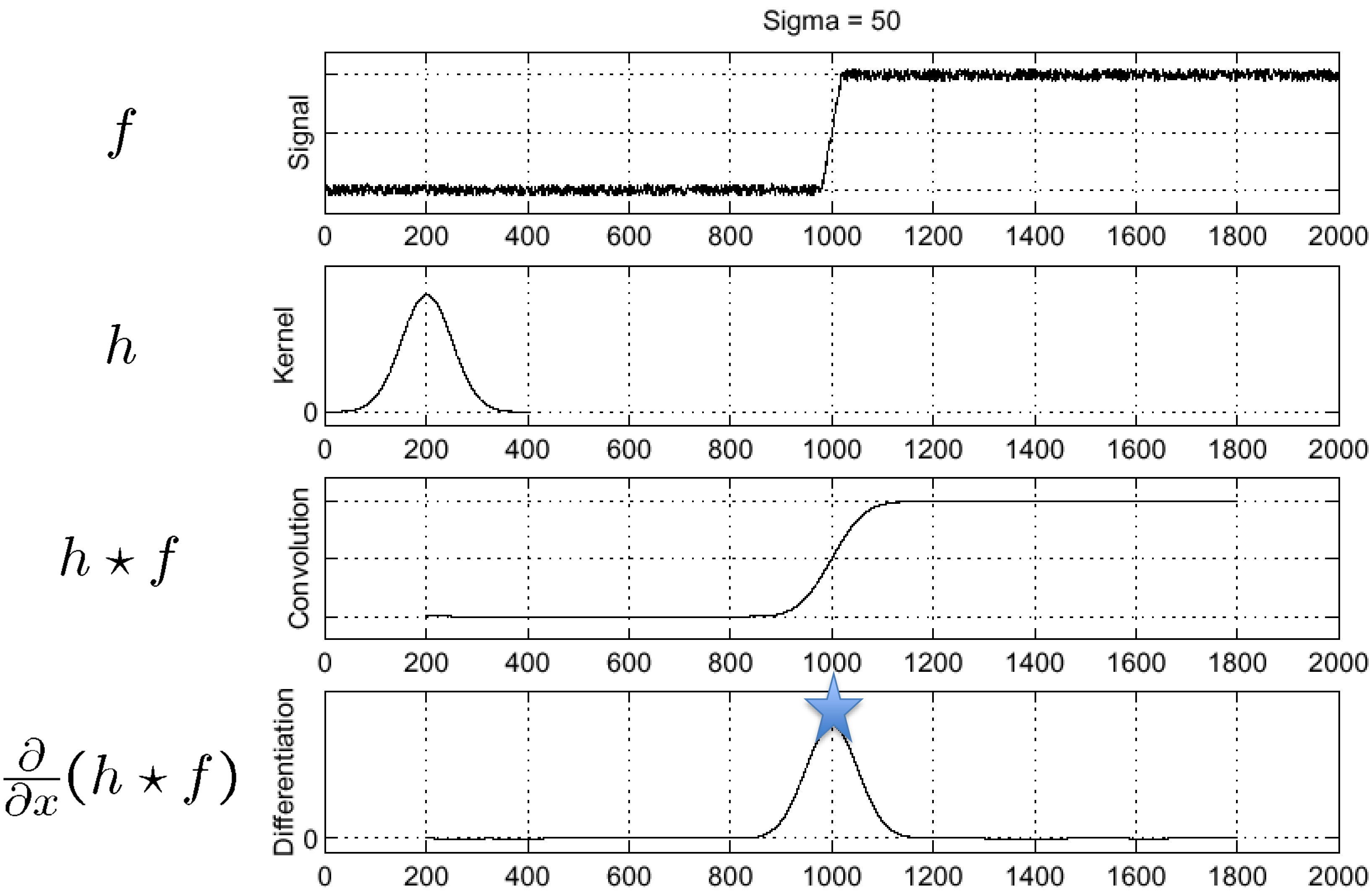
Where is the edge?

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



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

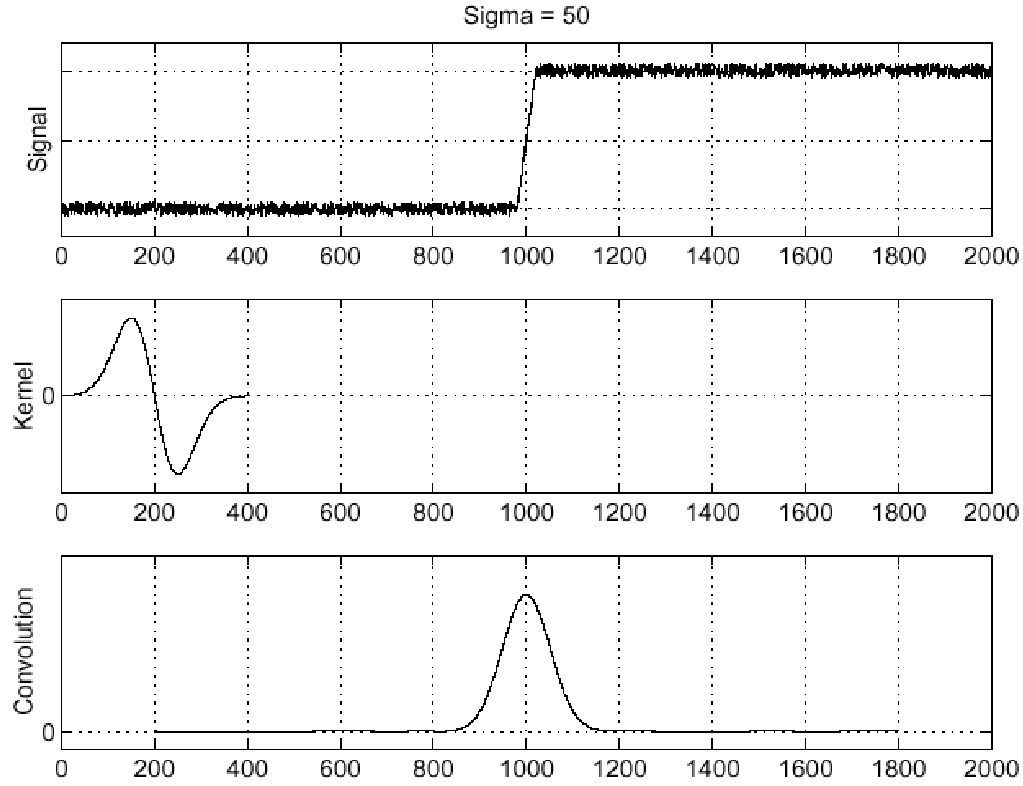
Derivative theorem of convolution

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

f

$\frac{d}{dx} g$

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

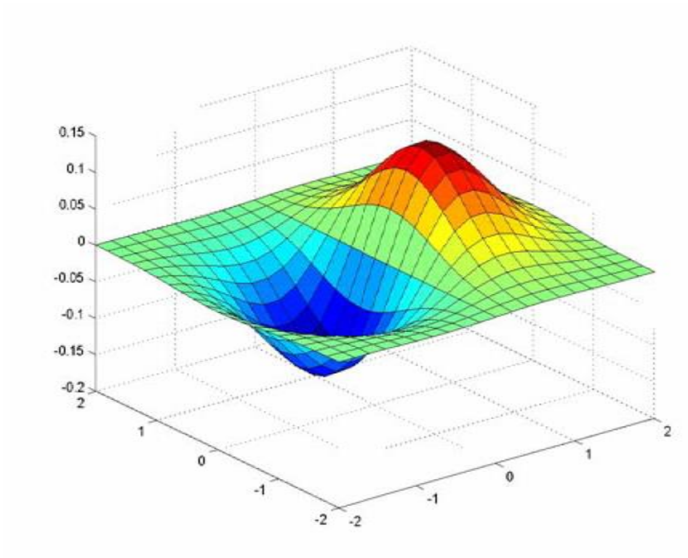


We don't do that.

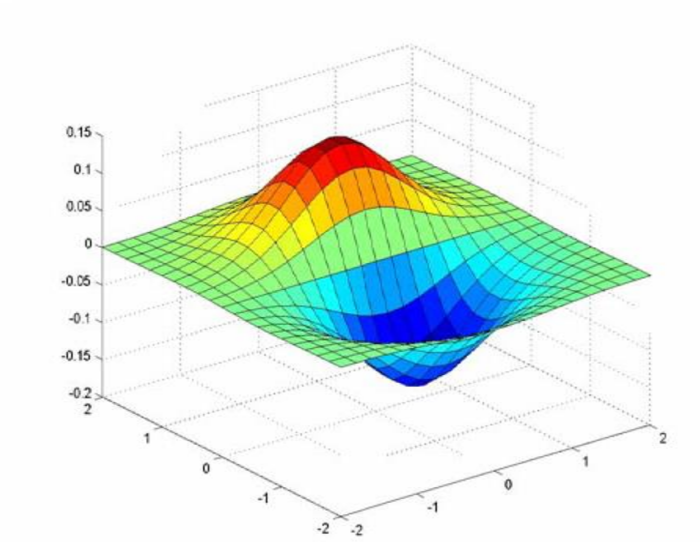
How can we find (local) maxima of a function?

Source: S. Seitz

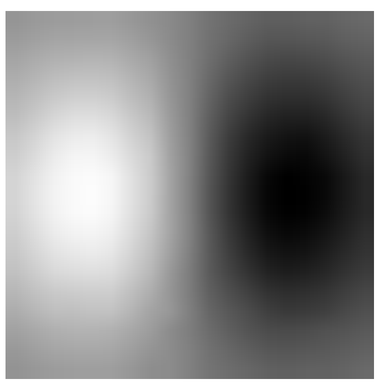
Remember:
Derivative of Gaussian filter

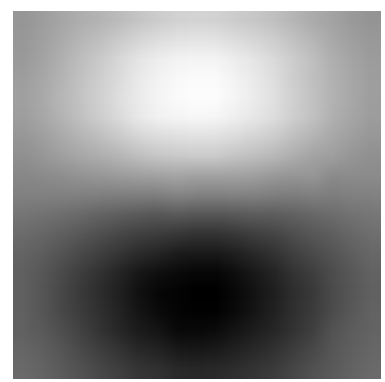


x-direction



y-direction





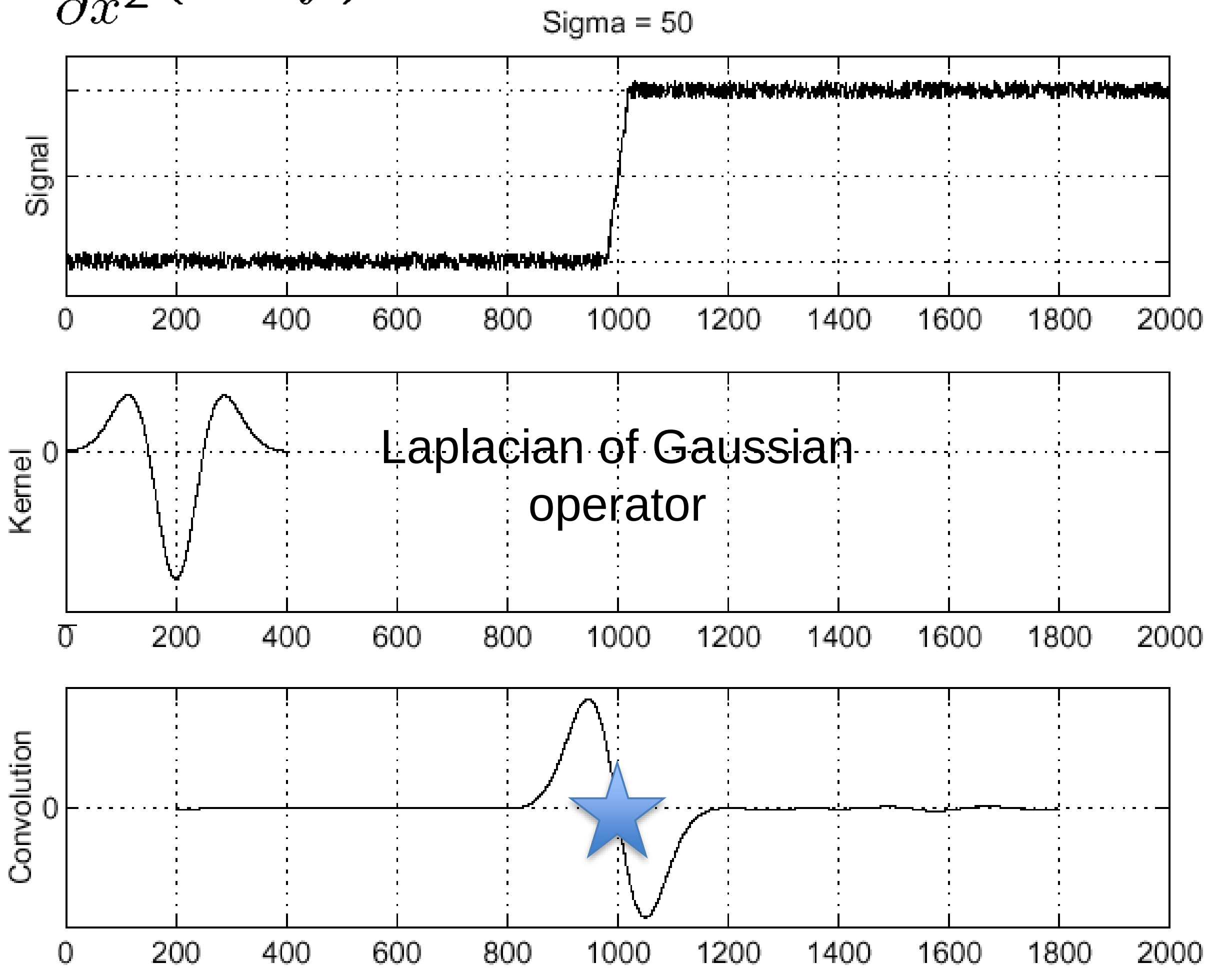
Laplacian of Gaussian

- Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

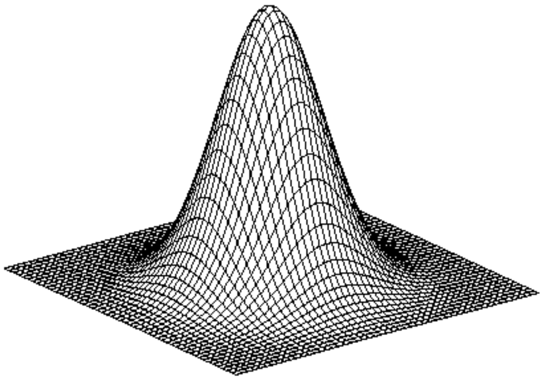
$(\frac{\partial^2}{\partial x^2}h) \star f$



Where is the edge?

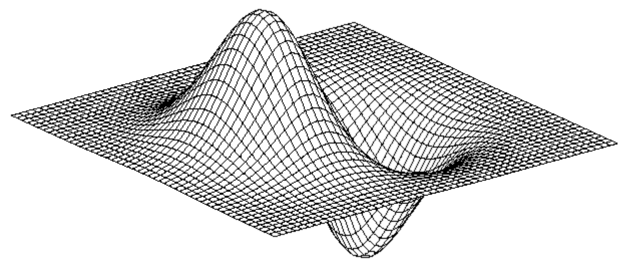
Zero-crossings of bottom graph

2D edge detection filters



Gaussian

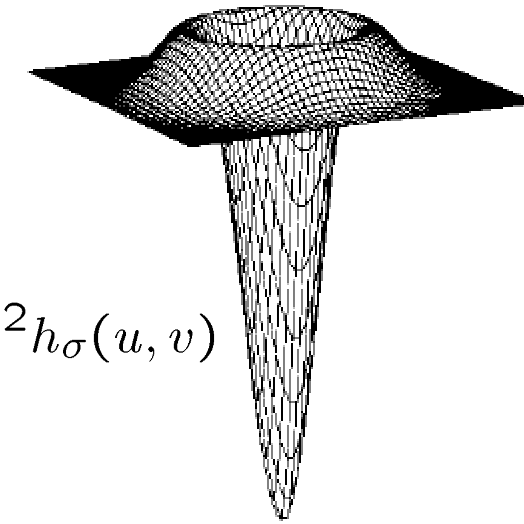
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Edge detection by subtraction

original

Edge detection by subtraction

smoothed (5x5 Gaussian)

Edge detection by subtraction

smoothed – original
(scaled by 4, offset +128)

Using the LoG Function (Laplacian of Gaussian)

- The LoG function will be
  - Zero far away from the edge
  - Positive on one side
  - Negative on the other side
  - Zero just at the edge
- It has simple digital mask implementation(s)
- So it can be used as an edge operator
- BUT, THERE'S SOMETHING BETTER

Canny edge detector

- This is probably the most widely used edge detector in computer vision

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

Source: L. Fei-Fei

The Canny edge detector

Note:
I hate the
Lena images.

- original image (Lena)

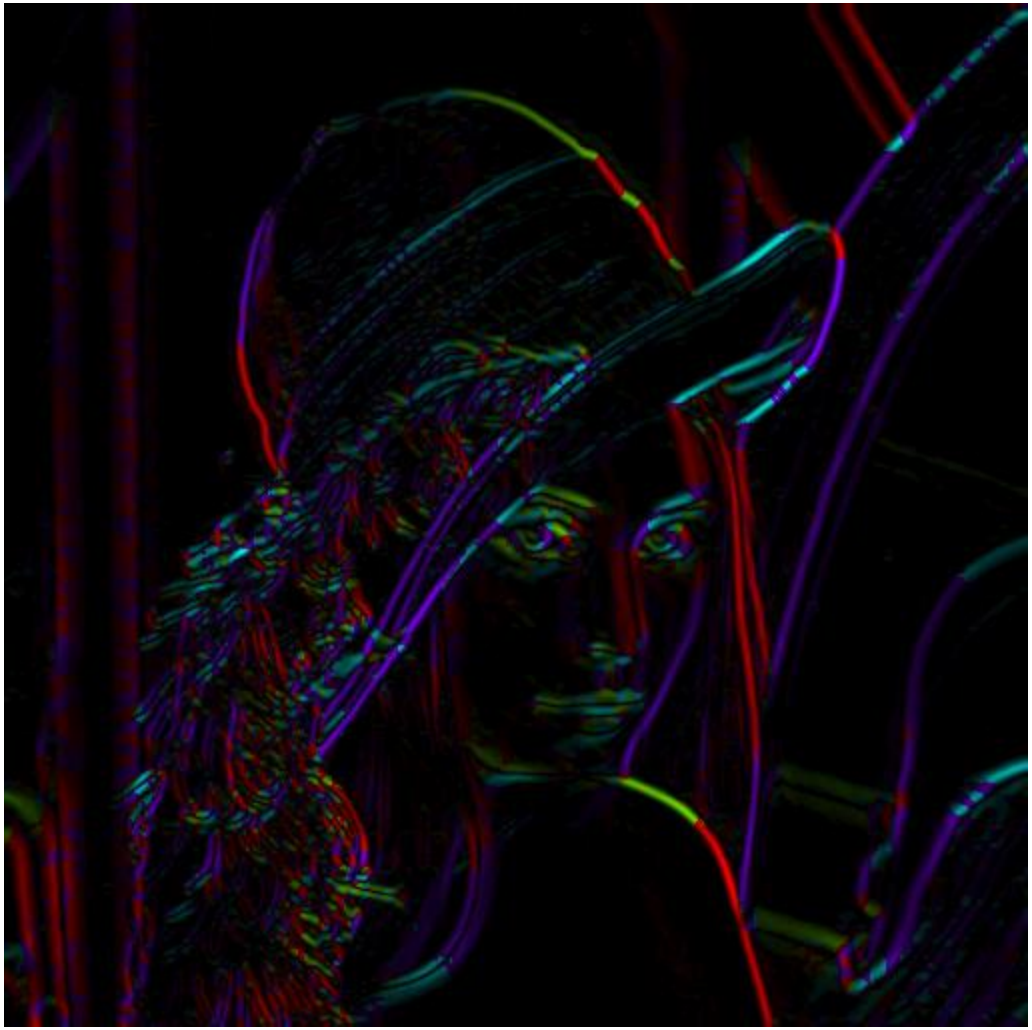
The Canny edge detector

norm of the gradient

The Canny edge detector

thresholding

Get Orientation at Each Pixel



theta = atan2(-gy, gx)

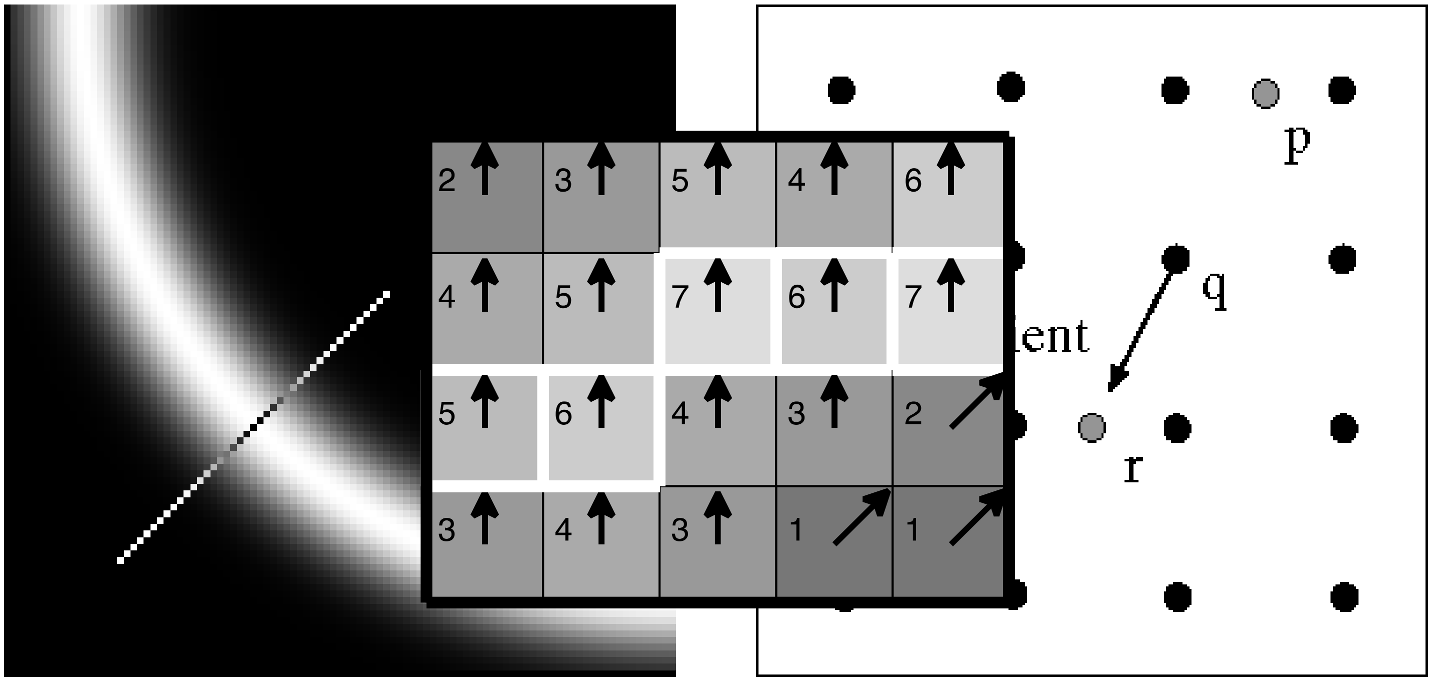
The Canny edge detector

The Canny edge detector

thinning

(non-maximum suppression)

Non-maximum suppression



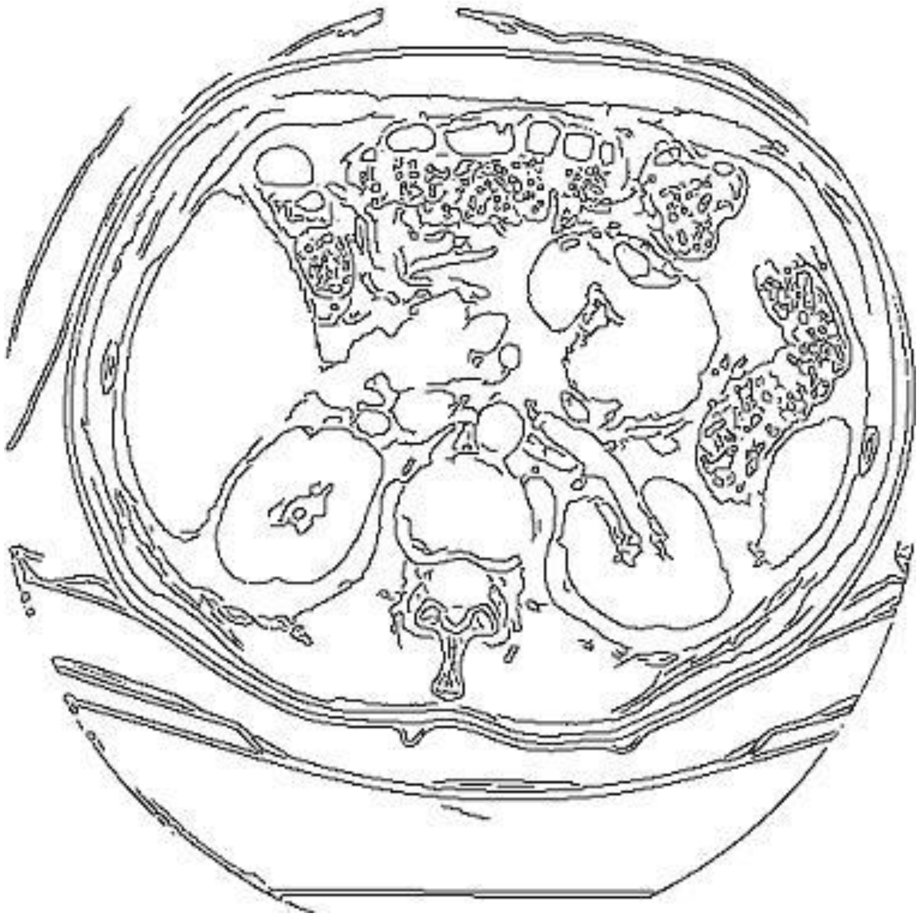
- Check if pixel is local maximum along gradient direction
- It also keeps only pixels part of small curves or lines

Picture from Prem K Kalra

Canny Edges

Canny on Kidney



Canny Characteristics

- The Canny operator gives single-pixel-wide images with good continuation between adjacent pixels
- It is the most widely used edge operator today; no one has done better since it came out in the late 80s. Many implementations are available.
- It is very sensitive to its parameters, which need to be adjusted for different application domains.

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

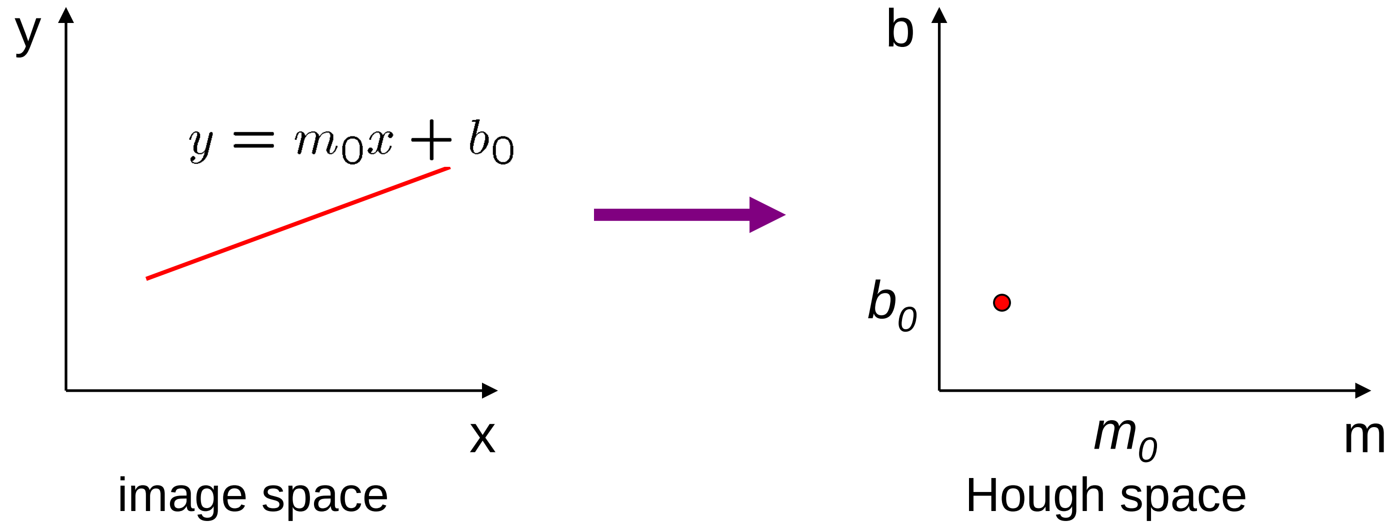
An edge is not a line...

How can we detect ***lines*** ?

Finding lines in an image

- Option 1:
  - Search for the line at every possible position/orientation
  - What is the cost of this operation?
- Option 2:
  - Use a voting scheme: Hough transform

Finding lines in an image



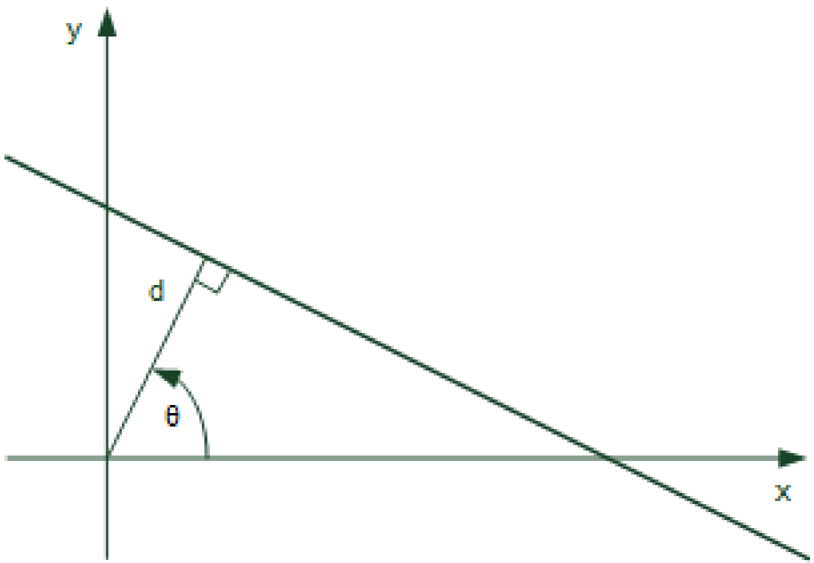
- Connection between image (x,y) and Hough (m,b) spaces
  - A line in the image corresponds to a point in Hough space
  - To go from image space to Hough space:
    - given a set of points (x,y), find all (m,b) such that y = mx + b

Hough transform algorithm

- Typically use a different parameterization

$$d = x cos\theta + y sin\theta$$

  - d is the perpendicular distance from the line to the origin
  - θ is the angle of this perpendicular with the horizontal.



Hough transform algorithm

Array H

d

θ

- Basic Hough transform algorithm
  1. Initialize H[d, θ]=0
  2. for each edge point I[x,y] in the image

     for θ = 0 to 180

     $$d = x\cos\theta + y\sin\theta$$

     H[d, θ] += 1
  3. Find the value(s) of (d, θ) where H[d, θ] is maximum
  4. The detected line in the image is given by $d = x\cos\theta + y\sin\theta$
- What's the running time (measured in # votes)?
  1. How big is the array H?
  2. Do we need to try all θ?

Example

gray-tone image

0	0	0	100	100
0	0	0	100	100
0	0	0	100	100
100	100	100	100	100
100	100	100	100	100

DQ

-	-	3	3	-
-	-	3	3	-
3	3	3	3	-
3	3	3	3	-
-	-	-	-	-

THETAQ

-	-	0	0	-
-	-	0	0	-
90	90	40	20	-
90	90	90	40	-
-	-	-	-	-

Accumulator H

distance							
360	-	-	-	-	-	-	-
.	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-
3	4	-	1	-	2	-	5
0	-	-	-	-	-	-	-

angle 0 10 20 30 40 …90

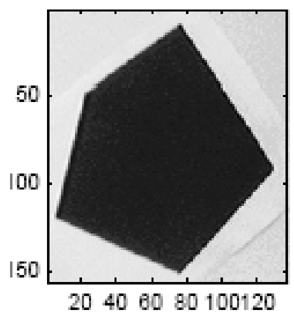
PTLIST

360	-	-	-	-	-	-	-
.	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-
3	*	-	*	-	*	-	*
0	-	-	-	-	-	-	-

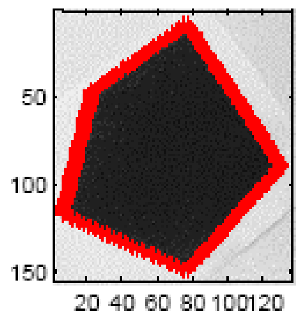
(1,3)(1,4)(2,3)(2,4)

(3,1)
(3,2)
(4,1)
(4,2)
(4,3)

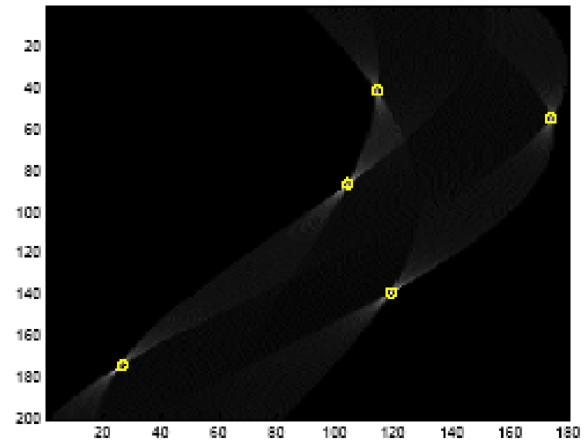
Examples



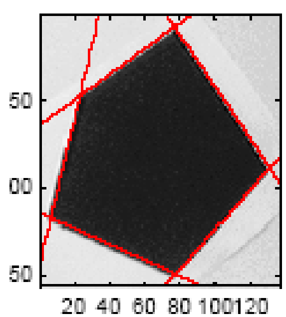
Original Image



Detected Edges



The vote histogram with the detected lines marked with 'o'



Detected lines

Examples cont

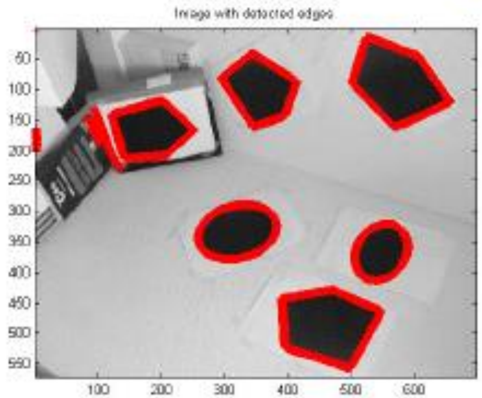
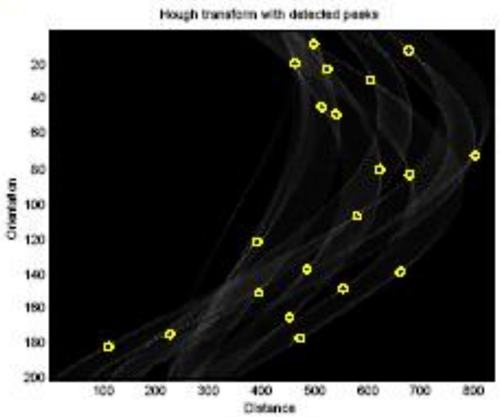


Image with detected edges



The vote histogram with the selected lines

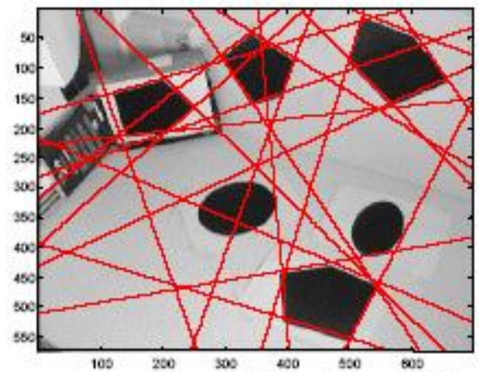


Image with the detected lines

How do you extract the line segments from the accumulators?

pick the bin of H with highest value V
while V > value_threshold {

- order the corresponding pointlist from PTLIST
- merge in high gradient neighbors within 10 degrees
- create line segment from final point list
- zero out that bin of H
- pick the bin of H with highest value V }

Line segments from Hough Transform

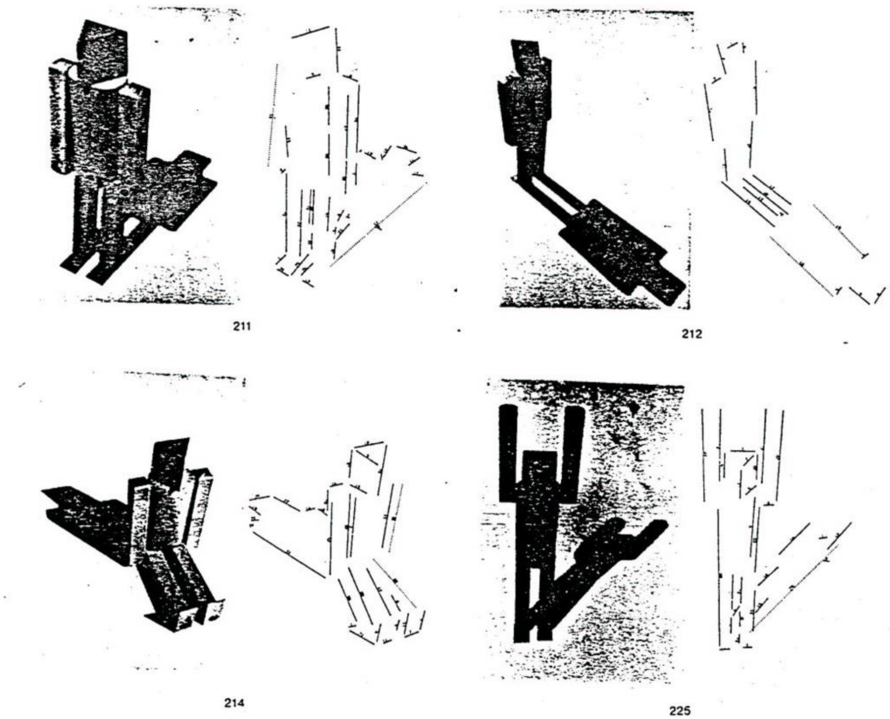


Fig.7. Puppet scenes 211, 212, 214, 225 and the edges recovered by the algorithm.

Extensions

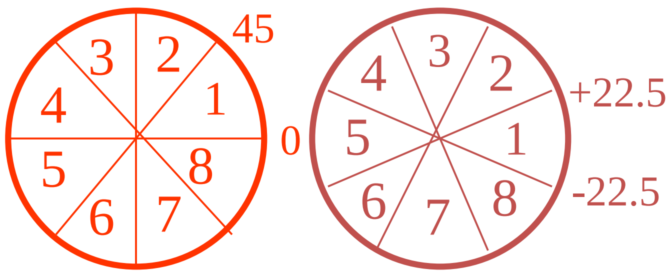
- Extension 1: Use the image gradient
  1. same
  2. for each edge point I[x,y] in the image

     compute unique (d, θ) based on image gradient at (x,y)

     $H[d, \theta]$ += 1
  3. same
  4. same
- What's the running time measured in votes?

- Extension 2
  - give more votes for stronger edges
- Extension 3
  - change the sampling of (d, θ) to give more/less resolution
- Extension 4
  - The same procedure can be used with circles, squares, or any other shape, How?
- Extension 5; the Burns procedure. Uses only angle, two different quantifications, and connected components with votes for larger one.

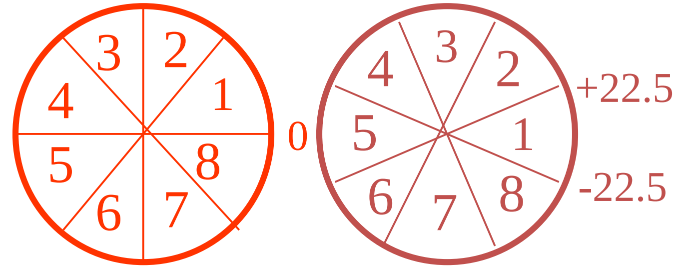
A Nice Hough Variant
The Burns Line Finder

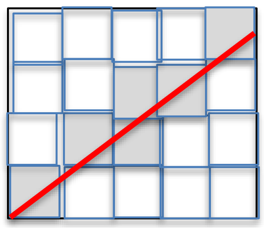


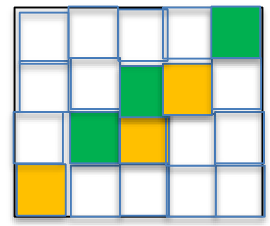
1. Compute gradient magnitude and direction at each pixel.
2. For high gradient magnitude points, assign direction labels to two symbolic images for two different quantizations.
3. Find connected components of each symbolic image.

- Each pixel belongs to 2 components, one for each symbolic image.
- Each pixel votes for its longer component.
- Each component receives a count of pixels who voted for it.
- The components that receive majority support are selected.

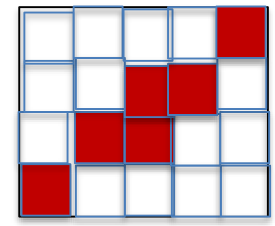
Example







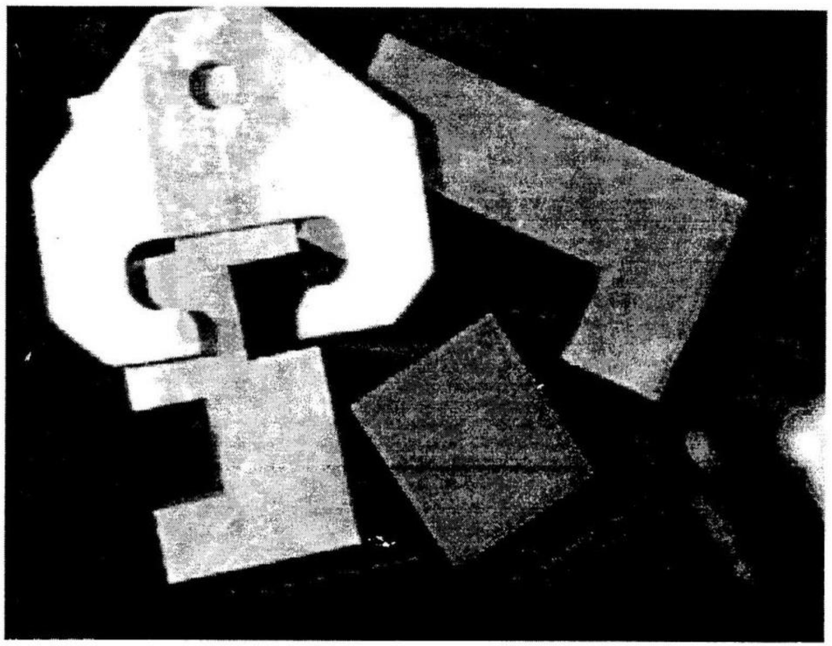
Quantization 1

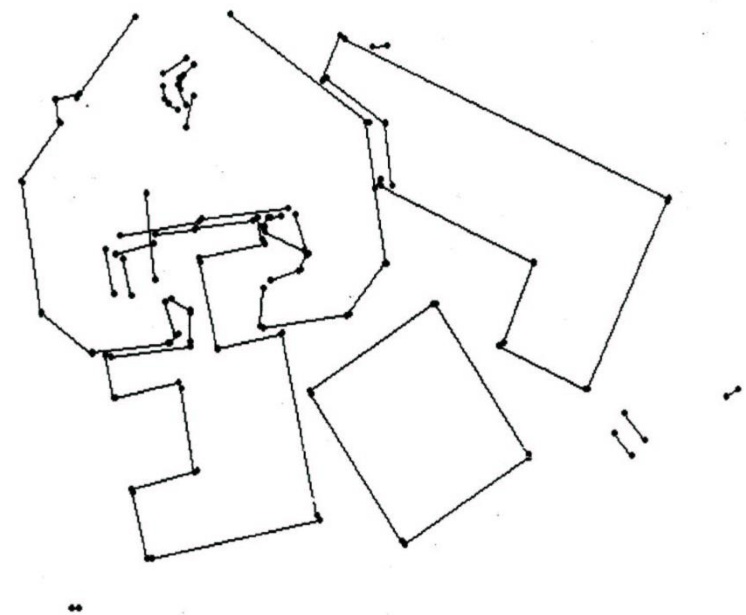


Quantization 2

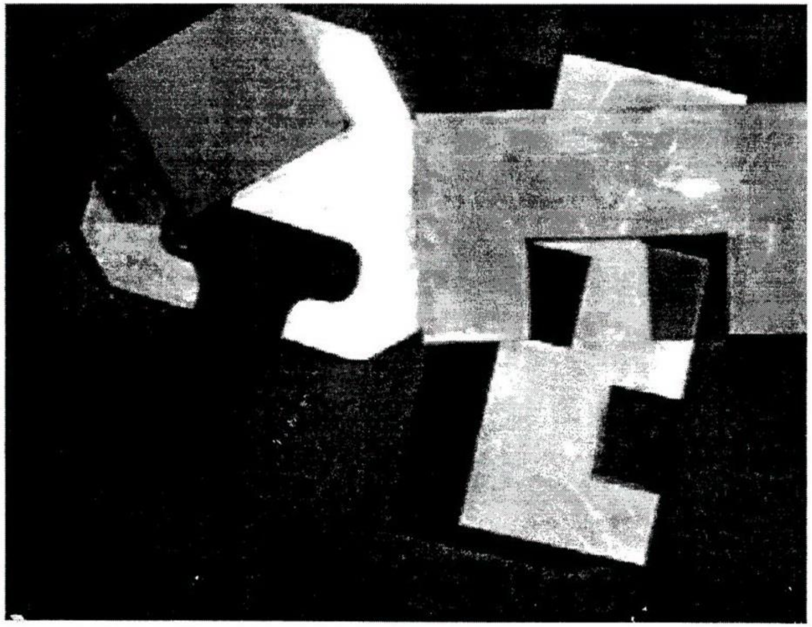
- Quantization 1 leads to 2 yellow components and 2 green.
- Quantization 2 leads to 1 BIG red component.
- All the pixels on the line vote for their Quantization 2 component. It becomes the basis for the line.

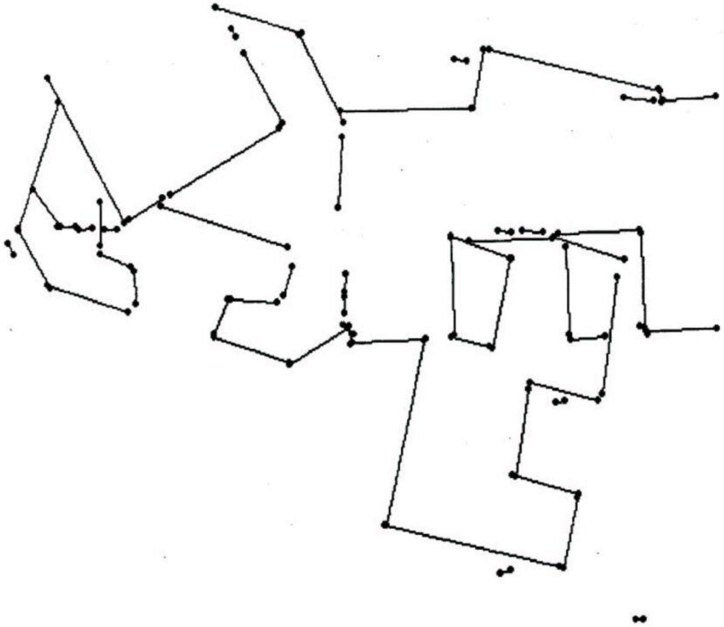
Burns Example 1





Burns Example 2





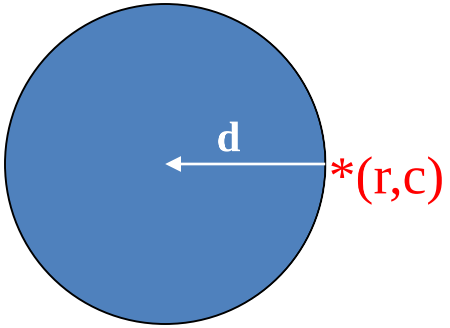
Hough Transform for Finding Circles

Equations:

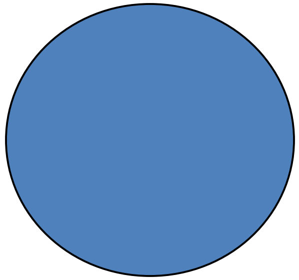
$$\boxed{\begin{aligned} r &= r0 + d \sin \theta \\ c &= c0 - d \cos \theta \end{aligned}}$$

r, c, d are parameters

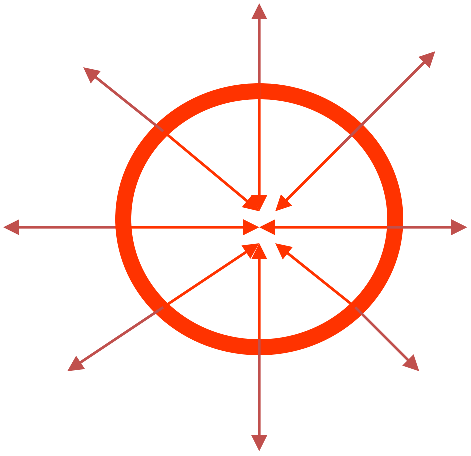
Main idea: The gradient vector at an edge pixel points to the center of the circle.



Why it works



Filled Circle:
Outer points of circle have gradient direction pointing to center.



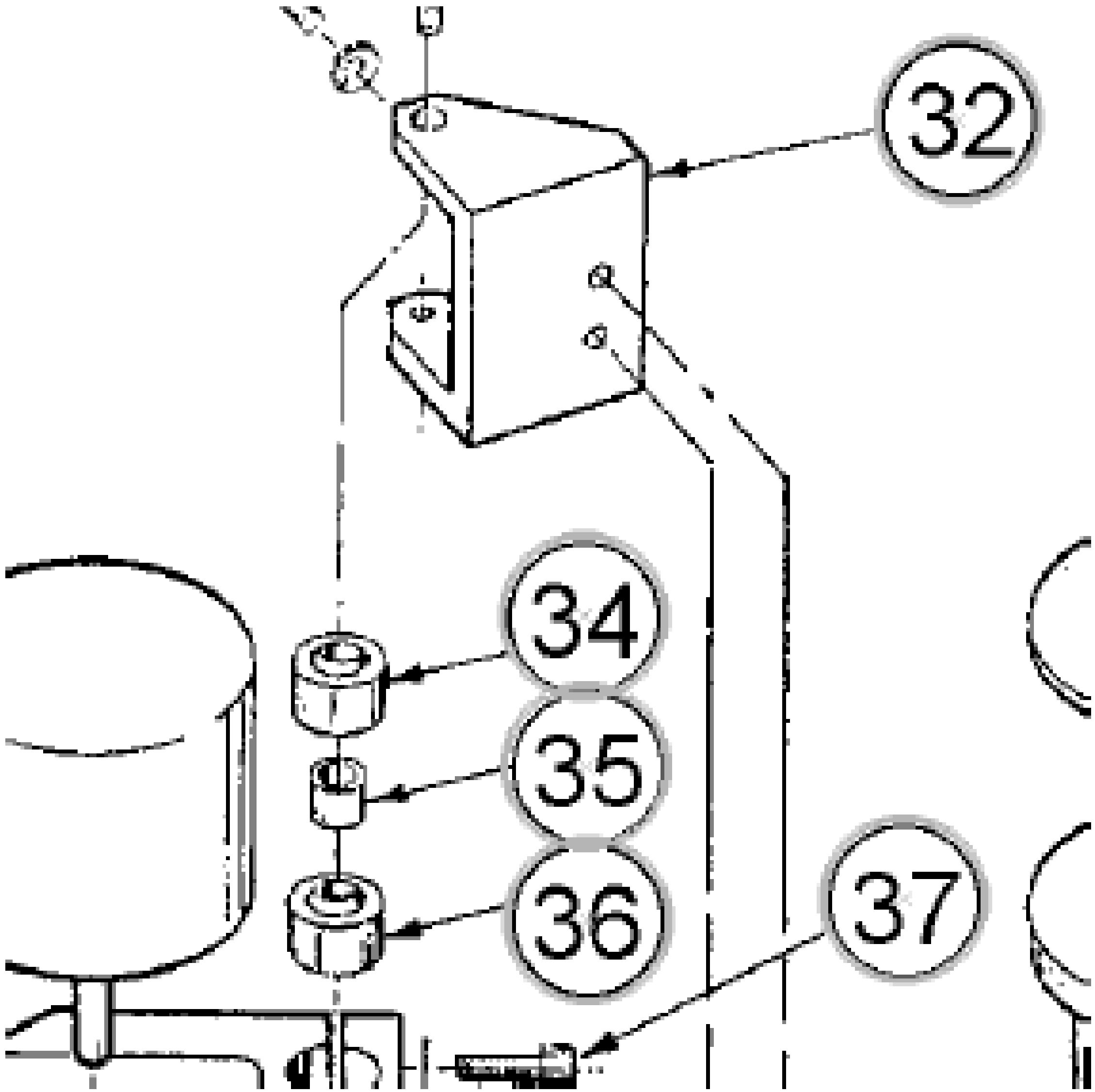
Circular Ring:
Outer points gradient towards center.
Inner points gradient away from center.

The points in the away direction don't accumulate in one bin!

Procedure to Accumulate Circles

- Set accumulator array A to all zero.
  Set point list array PTLIST to all NIL.

- For each pixel (R,C) in the image {
  For each possible value of D {
  - compute gradient magnitude GMAG
  - if GMAG > gradient_threshold {
    . Compute THETA(R,C,D)
    . R0 := R - D*sin(THETA)
    . C0 := C + D*cos(THETA)
    . increment A(R0,C0,D)
    . update PTLIST(R0,C0,D) }}


32
34
35
36
37

Finding lung nodules (Kimme & Ballard)

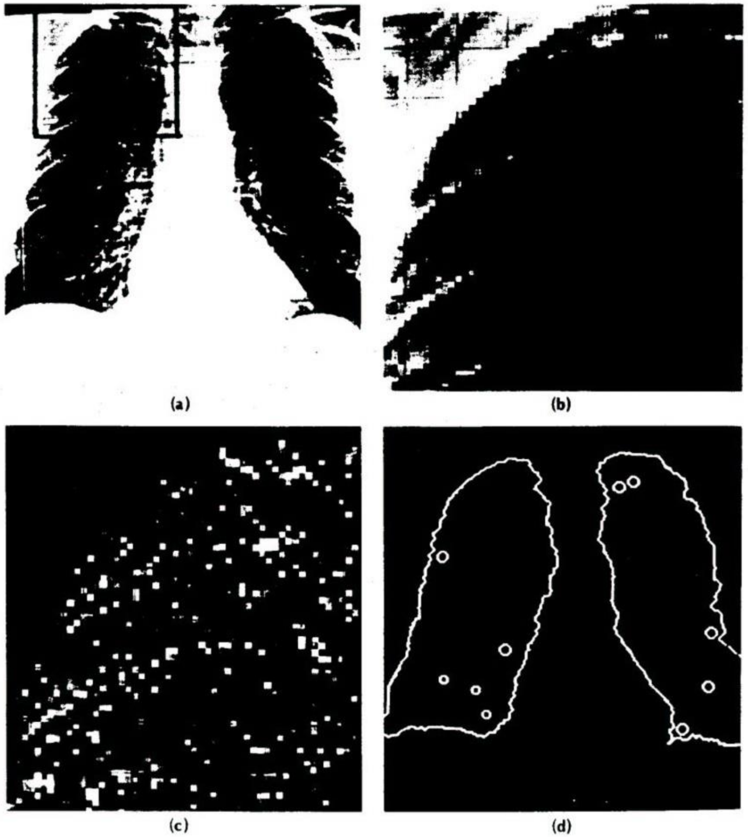


Fig. 4.7 Using the Hough technique for circular shapes. (a) Radiograph. (b) Window. (c) Accumulator array for $r = 3$. (d) Results of maxima detection.

Finale

- Edge operators are based on estimating derivatives.
- While first derivatives show approximately where the edges are, zero crossings of second derivatives were shown to be better.
- Ignoring that entirely, Canny developed his own edge detector that everyone uses now.
- After finding good edges, we have to group them into lines, circles, curves, etc. to use further.
- The Hough transform for circles works well, but for lines the performance can be poor. The Burns operator or some tracking operators (old ORT pkg) work better.