Digital Image processing
Chapter 10
Image segmentation

By Lital Badash and Rostislav Pinski

Oct. 2010

Outline

Preview

- Segmentation is to subdivide an image into its component regions or objects.
- Segmentation should stop when the objects of interest in an application have been isolated.

Preview - Example

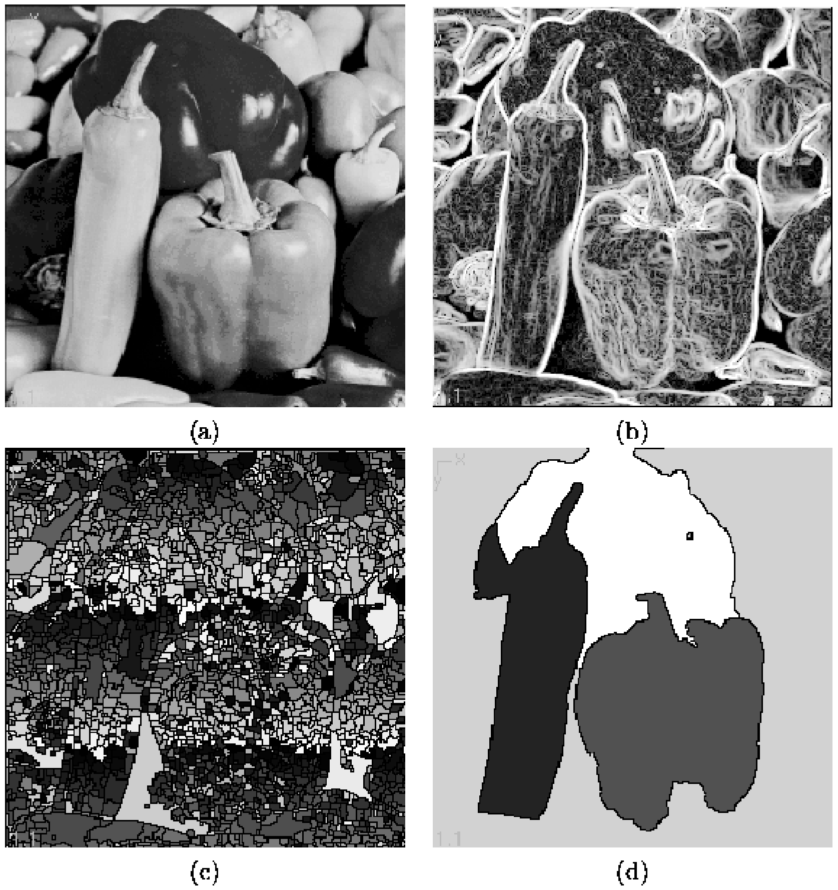


Figure 5.51: *Watershed segmentation: (a) original; (b) gradient image, 3×3 Sobel edge detection, histogram equalized; (c) raw watershed segmentation; (d) watershed segmentation using region markers to control oversegmentation. Courtesy W. Higgins, Penn State University.*

Principal approaches

- Segmentation algorithms generally are based on one of 2 basis properties of intensity values:
  - discontinuity : to partition an image based on sharp changes in intensity
  - similarity : to partition an image into regions that are similar according to a set of predefined criteria.

Fundamentals

A more formal definition – Let R represent the entire image. Segmentation is a process that divides R into n subregions $R_1, R_2, \ldots, R_n$ such that:

1. $\bigcup_{i=1}^{n} R_i = R$.
2. R_i is a connected set for each $i = 1, 2, \ldots, n$.
3. $R_i \cap R_j = \emptyset$ for all i and j, $j \neq i$.
4. $Q(R_i) = TRUE$ for each $i = 1, 2, \ldots, n$.
5. $Q(R_i \cup R_j) = FALSE$ for any adjacent regions R_i and R_j.

Here $Q(R_k)$ is a predicate that indicates some property over the region.

Example

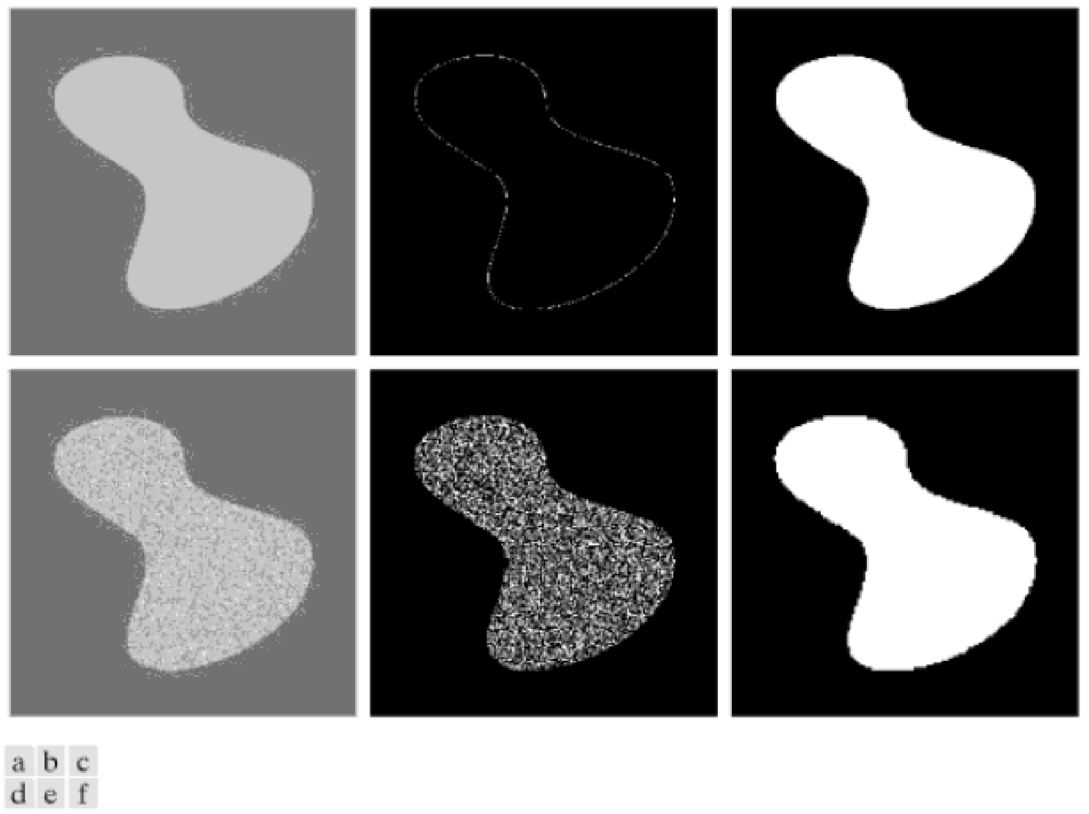
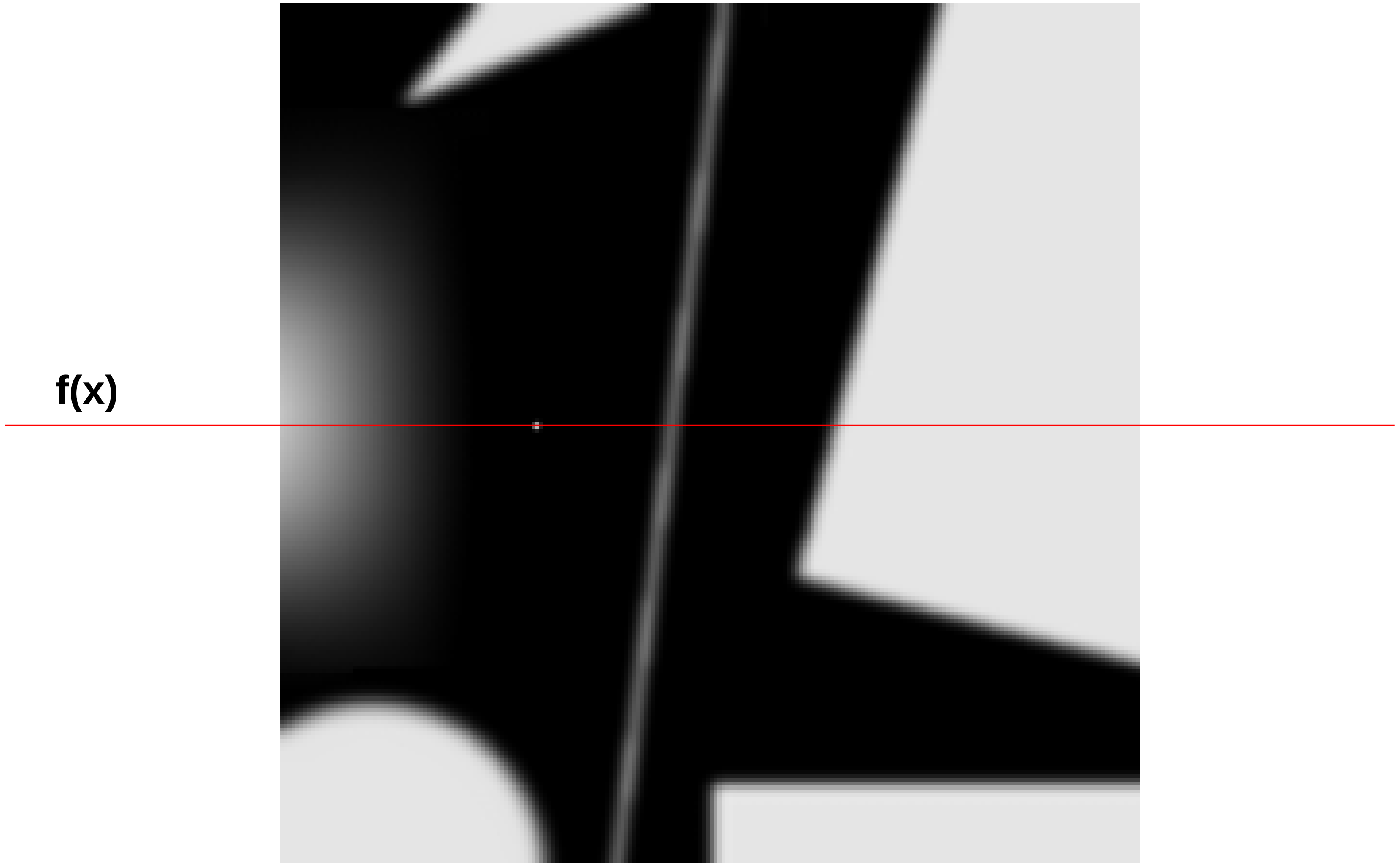


FIGURE 10.1 (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large number of small edges that are connected to the original boundary, making it difficult to find a unique boundary using only edge information. (f) Result of segmentation based on region properties.

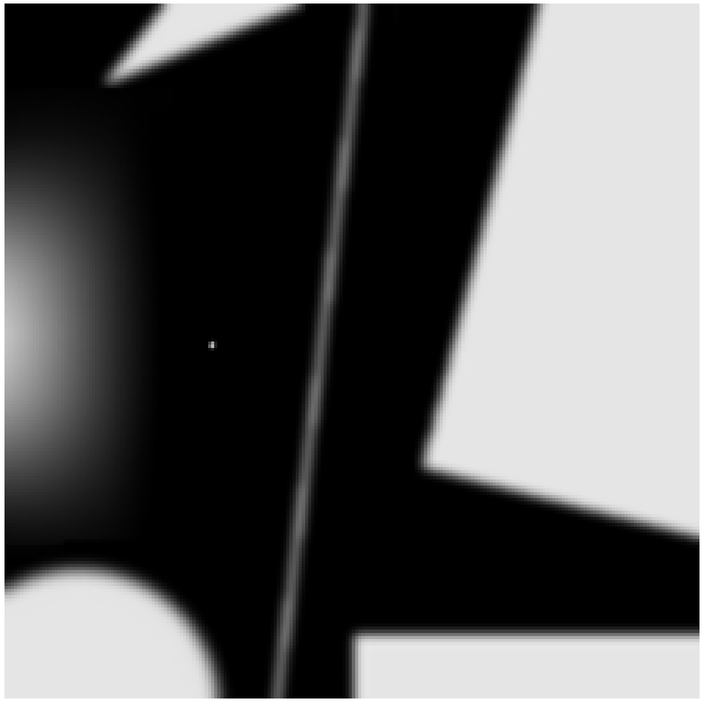
Outline

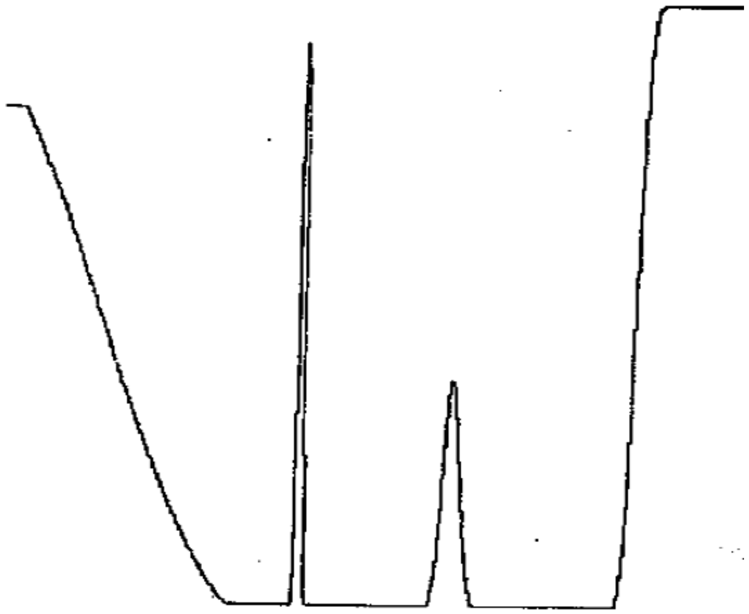
- Introduction
- Mathematical background
- Detection of isolated point
- Line detection
- Edge Models
- Basic edge detection
- Advanced edge detection
- Edge linking and Boundary detection

Example



Example





Derivatives

- We will use derivatives (first and second) in order to discover discontinuities.
- If we observe a line within an image we can consider it as a one-dimensional function f(x). We will now define the first derivative simply as the difference between two adjacent pixels.

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

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x+1) - 2f(x) + f(x-1)$$

Derivatives Properties

- First derivative generally produce thicker edges in an image
- Second derivative has a very strong response to fine details and noise
- Second derivative sign can be used to determine transition direction.

Derivatives - Computation

- The derivative is a linear operator ant therefore can be computed using a mask
- For example, the mask for the second derivative on the x direction would be:

0	0	0
-1	2	-1
0	0	0

Outline

Detection of isolated point

- Based on the fact that a second order derivative is very sensitive to sudden changes we will use it to detect an isolated point.

- We will use a Laplacian which is the second order derivative over a two dimensional function.

Detection of isolated point

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

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) - 2f(x,y) + f(x-1,y)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) - 2f(x,y) + f(x,y-1)$$

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Detection of isolated point

- The mask for the Laplacian is:

0	-1	0
-1	4	-1
0	-1	0

- We will now decide about a threshold T, and any pixel whose value is larger than T after being masked will be considered as an isolated point.

Example

−1	−1	−1
−1	8	−1
−1	−1	−1

a
b c d

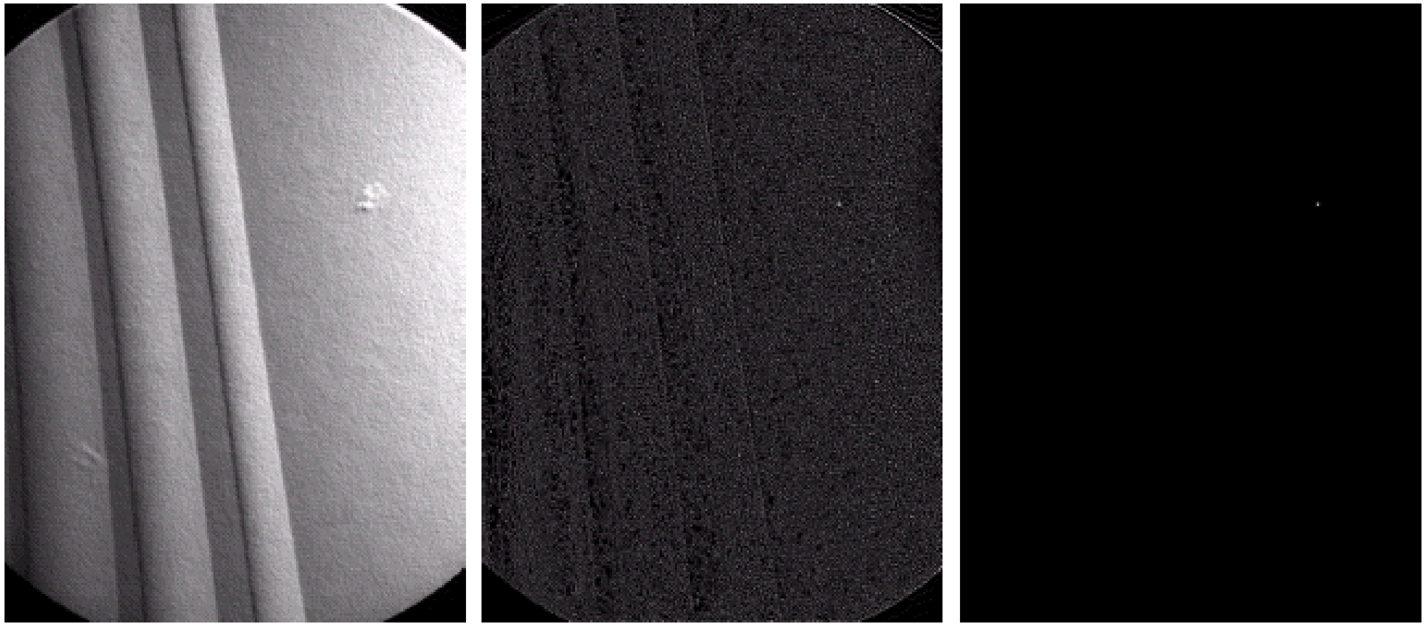


FIGURE 10.2
(a) Point detection mask.
(b) X-ray image of a turbine blade with a porosity.
(c) Result of point detection.
(d) Result of using Eq. (10.1-2).
(Original image courtesy of X-TEK Systems Ltd.)

Outline

- Introduction
- Mathematical background
- Detection of isolated point
- **Line detection**
- Edge Models
- Basic edge detection
- Advanced edge detection
- Edge linking and Boundary detection

Line Detection

- We can use the Laplacian also for detection of line, since it is sensitive to sudden changes and thin lines.
- We must note that since the second derivative changes its sign on a line it creates a “double line effect” and it must be handled.
- Second derivative can have negative results and we need to scale the results

Double Edge Effect - Example

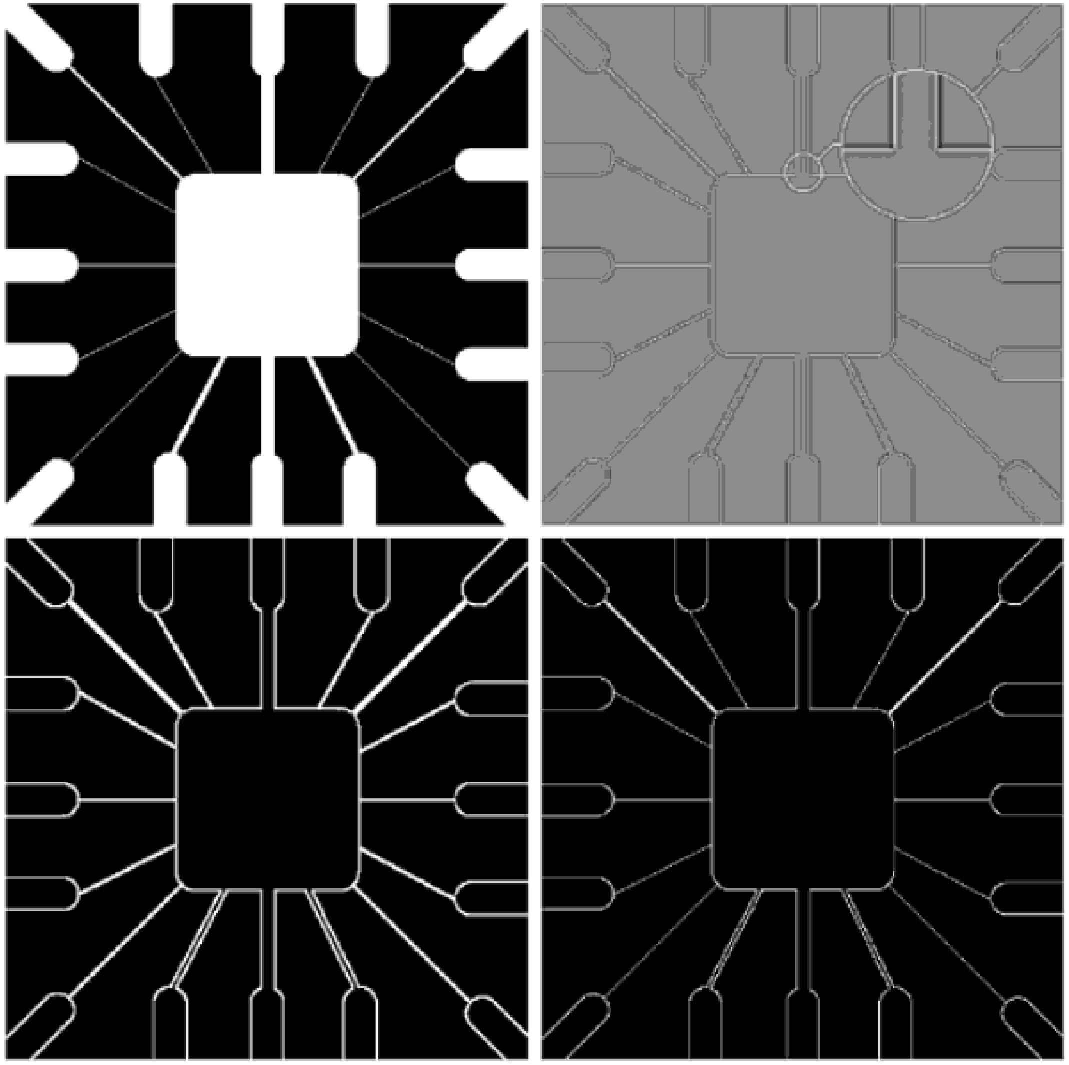


FIGURE 10.5 (a) Original image. (b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian. (c) Absolute value of the Laplacian. (d) Positive values of the Laplacian.

Line Detection - Orientation

- The Laplacian is isotropic, i.e. independent of direction.
- If we would like to detect lines on a certain direction only we might want to use masks that would emphasize a certain direction and be less sensitive to other directions.

FIGURE 10.3 Line masks.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

$+45^\circ$

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

-45°

Example

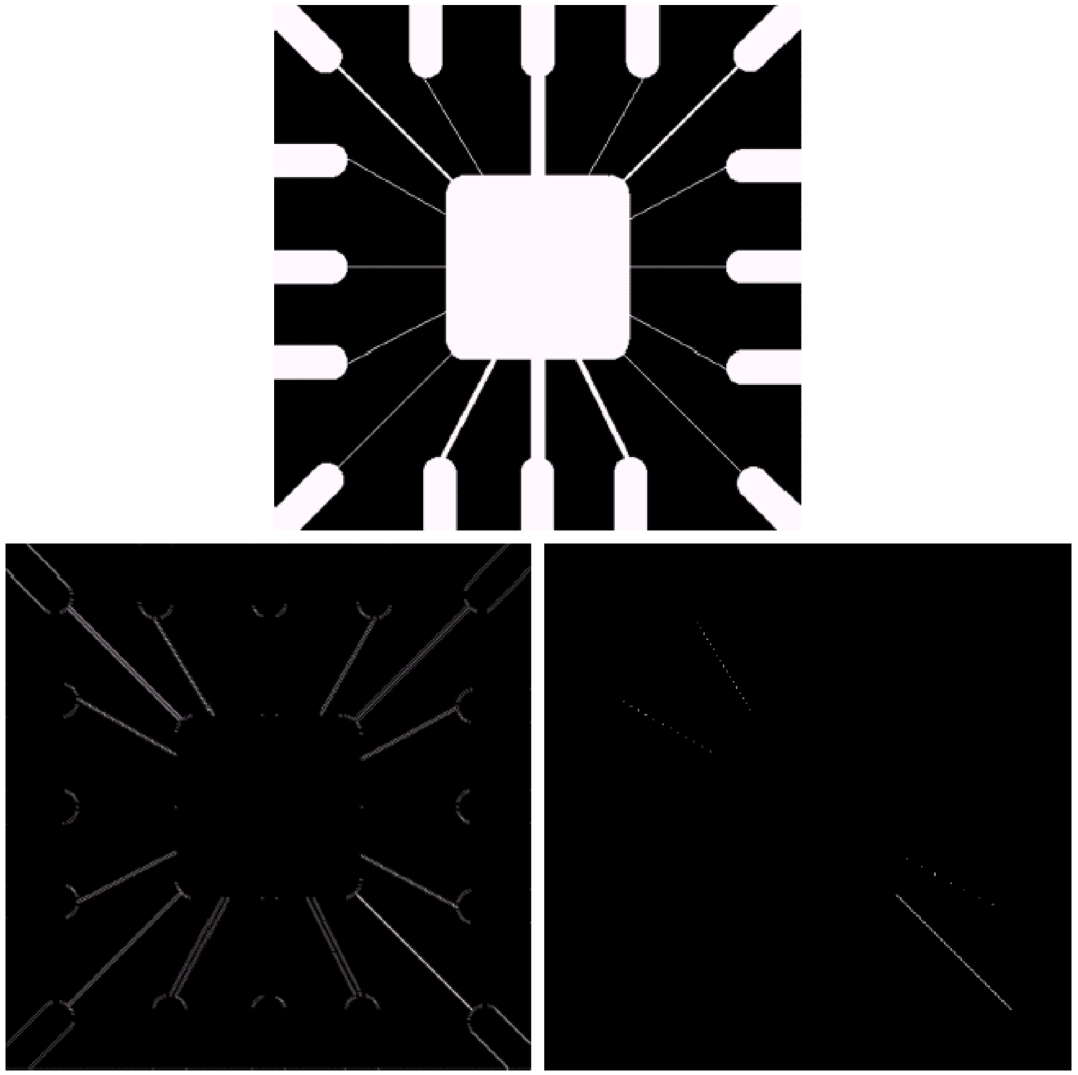


FIGURE 10.4 Illustration of line detection. (a) Binary wire-bond mask. (b) Absolute value of result after processing with −45° line detector. (c) Result of thresholding image (b).

Outline

- Introduction
- Mathematical background
- Detection of isolated point
- Line detection
- **Edge Models**
- Basic edge detection
- Advanced edge detection
- Edge linking and Boundary detection

Edge models

- 3 differentt edge types are observed:
  - Step edge – Transition of intensity level over 1 pixel only in ideal, or few pixels on a more practical use
  - Ramp edge – A slow and graduate transition
  - Roof edge – A transition to a different intensity and back. Some kind of spread line.

Edge models

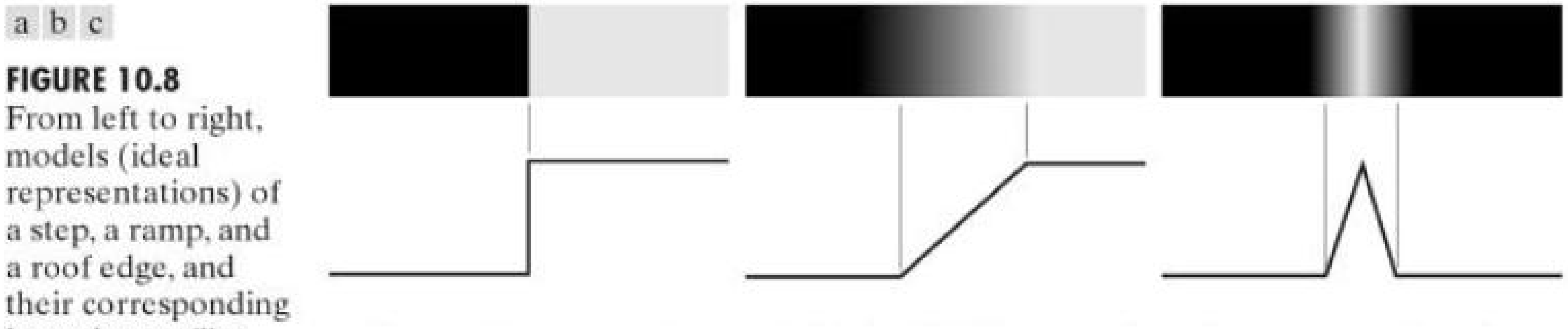
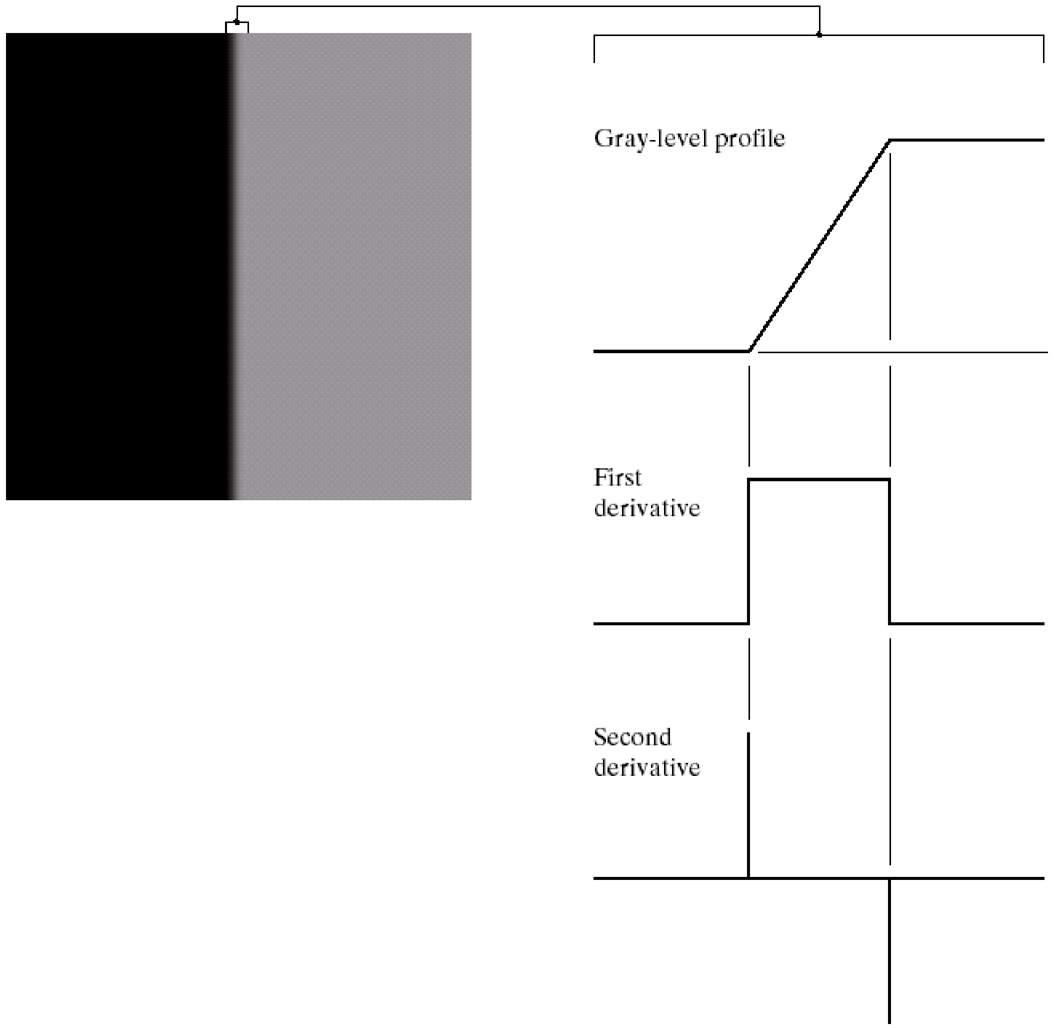


FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

Edge models



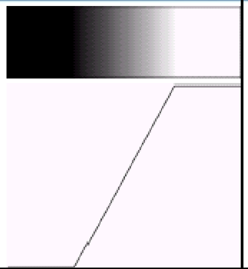
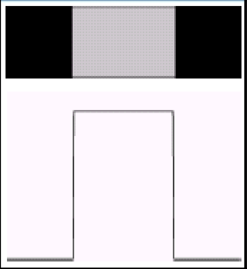
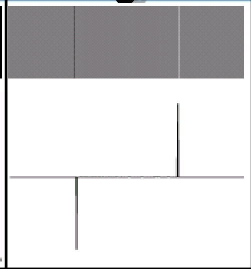
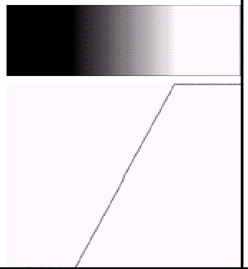
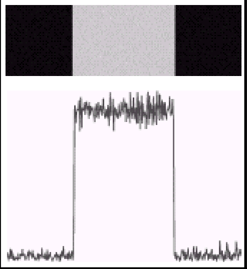
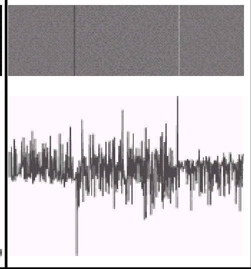
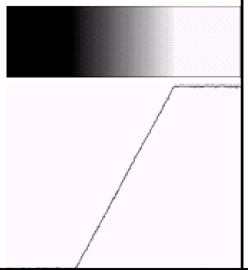
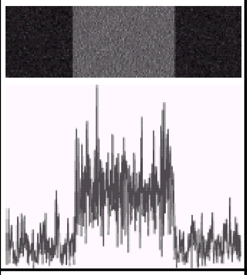
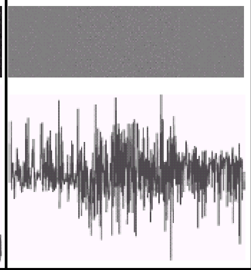
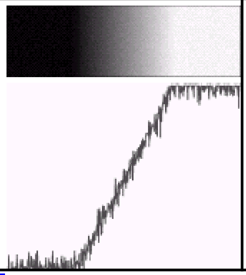
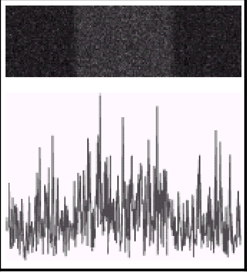
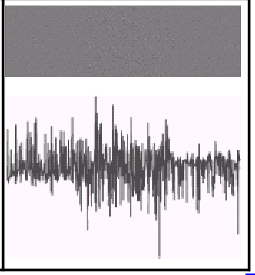
a b

FIGURE 10.6 (a) Two regions separated by a vertical edge. (b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.

Edge models - Derivatives

- Magnitude of the first derivative can be used to detect an edge
- The sign of the second derivative indicates its direction (black to white or white to black)

Edge models - Sensitivity

Clear Image			
A few noise			
Some more noise			
Visible (to eye) noise			

Edge models – General algorithm

- It is a good practice to smooth the image before edge detection to reduce noise.
- Detect edge point – detect the points that may be part of an edge
- Select the true edge members and compose them to an edge

Outline

- Introduction
- Mathematical background
- Detection of isolated point
- Line detection
- Edge Models
- **Basic edge detection**
- Advanced edge detection
- Edge linking and Boundary detection

Image gradient

The tool of choice for finding edge strength and direction at location (*x,y*) of an image, *f*, is the gradient

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The vector has the important geometrical property that it points in the direction of the greatest rate of change of *f* at location (*x,y*).

Gradient Properties

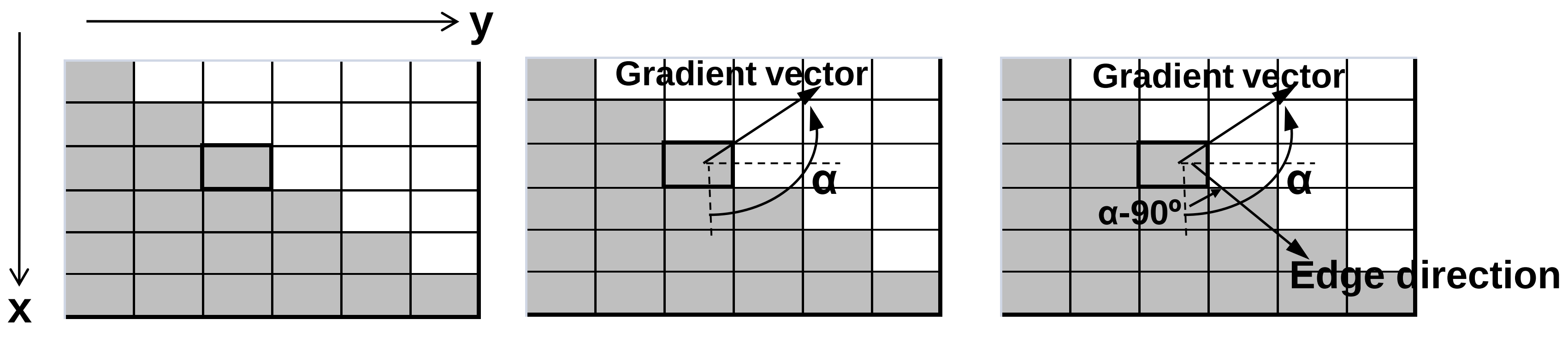
The magnitude of the gradient is the value of the rate of change in the direction of the gradient vector.

$$M(x,y) = mag(\nabla f) = \sqrt{g_x{}^2 + g_y{}^2}$$

The direction of the gradient with respect to the x-axis :

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

Example – Fig. 10.12

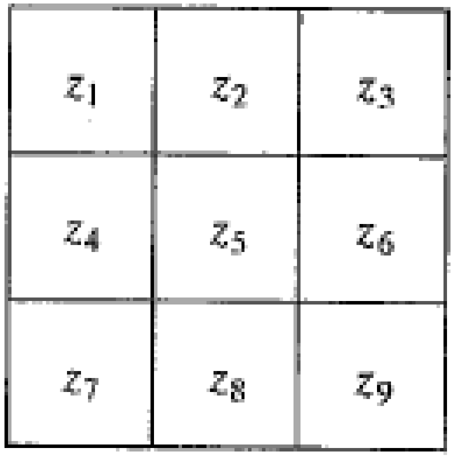


Gradient operators

- Instead of computing the partial derivatives at every pixel location in the image, we will use approximation.
- Using 3x3 neighborhood centered about a point.
  - **For g_y** : Subtract the pixel in the left column from the pixel in the right column.
  - **For g_x** : Subtract the pixel in the top row from the pixel in the bottom row.

Two dimensional mask

For diagonal edge direction we will use a 2-D mask. Consider the 3x3 region where Z*'s* are intensity values and we want to compute Z_5 gradient.



z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Mask of 2x2 are simple but they are not as useful for computing edge direction as masks that are symmetric about the center point. Carry more information regarding the direction of an edge.

Prewitt operators

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

Columns subtract. Y-direction

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

Rows subtract. X-direction

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

Sobel operators

Using a 2 in the center location provides image smoothing

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

Sobel Vs. Prewitt

Prewitt	Sobel
Simpler to implement	Better noise suppression (smoothing)
Give isotropic results only for vertical and horizontal edges	Give isotropic results only for vertical and horizontal edges
	Preferred because noise suppression is an important issue when dealing with derivatives .

The coefficients of all mask sum to zero. That gives a response of zero to an areas of constant intensity.

Computing the magnitude

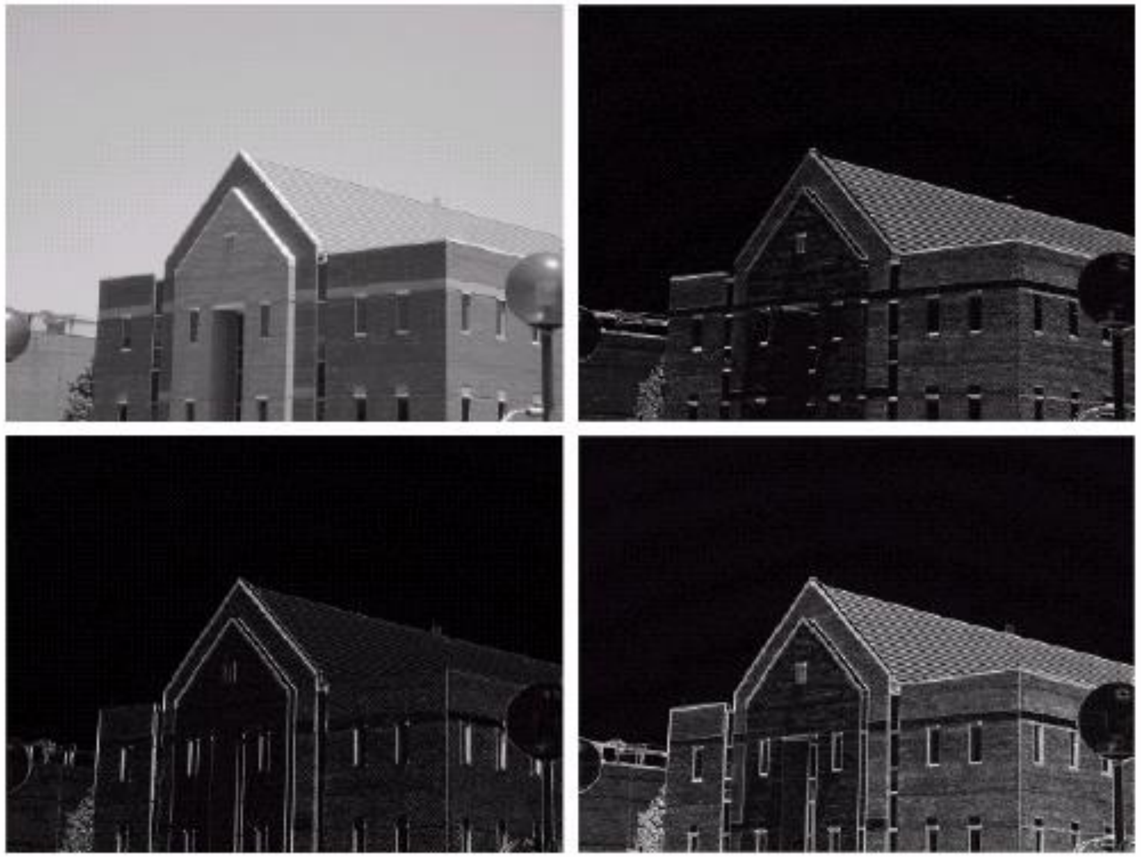
Requires that g_x and g_y be combined. However, it is hard to compute thus we will use approximate value.

$$M(x,y) \approx |g_x| + |g_y|$$

2-D Gradient Magnitude and Angle

a b
c d

FIGURE 10.10 (a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.

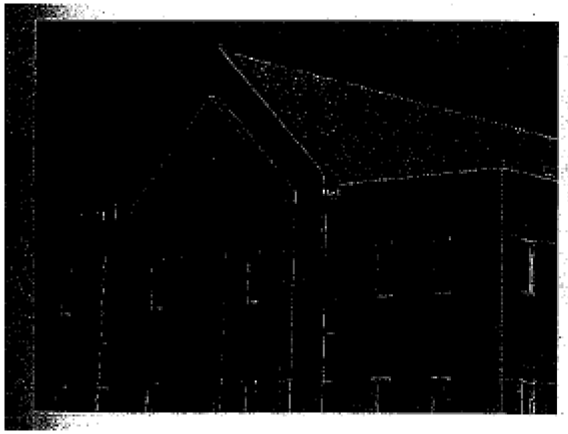


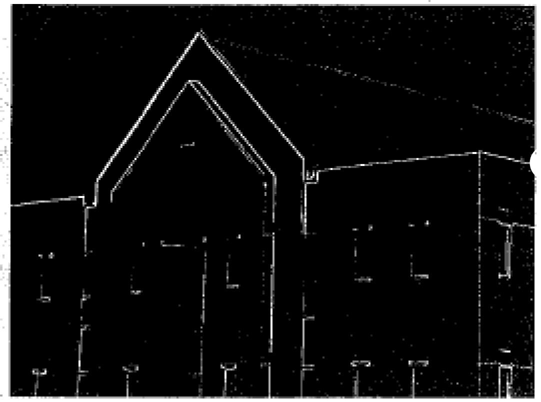
Gradient with threshold

sometimes we would use threshold on the gradient image to achieve the same results.

Pixels with values greater than threshold are shown white and the other are shown black.

Threshold the gradient image





Smooth and threshold

Outline

- Introduction
- Mathematical background
- Detection of isolated point
- Line detection
- Edge Models
- Basic edge detection
- **Advanced edge detection**
- Edge linking and Boundary detection

Advanced Edge detection

- The basic edge detection method is based on simple filtering without taking note of image characteristics and other information.
- More advanced techniques make attempt to improve the simple detection by taking into account factors such as noise, scaling etc.
- We introduce 2 techniques:
  - Marr-Hildreth [1980]
  - Canny [1986]

Marr-Hildreth edge detector [1980]

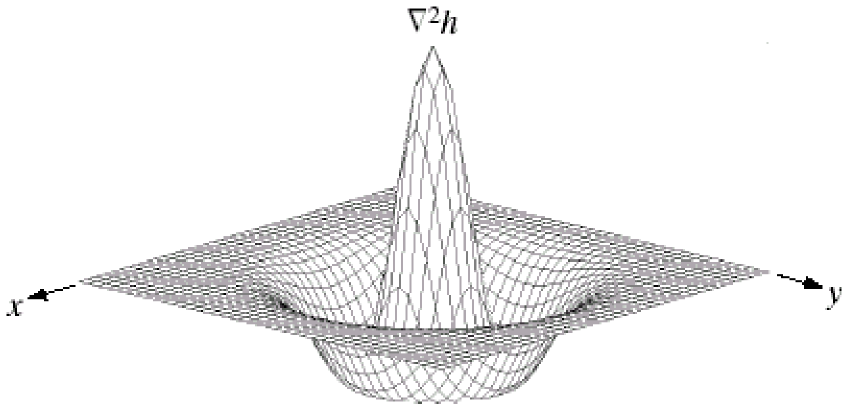
- Marr and Hildreth argued that:
  - Intensity of changes is not independent of image scale
  - Sudden intensity change will cause a zero-crossing of the second derivative
- Therefore, an edge detection operator should:
  - Be capable of being tuned to any scale
  - Be capable of computing the first and second derivatives

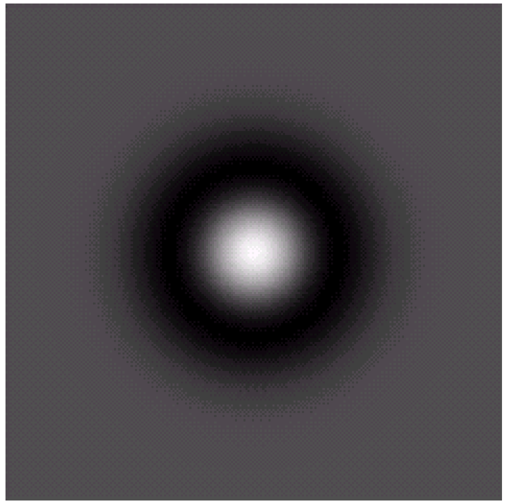
Marr-Hildreth edge detector [1980]

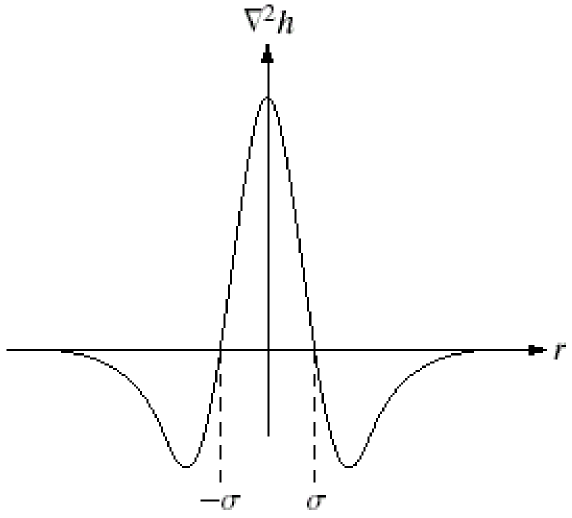
They used the operator $\nabla^2 G$ where ∇^2 is the Laplacian and G is the 2-D Gaussian function $G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$ with standard deviation σ.

And we get $\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$, also called *Laplacian of Gaussian* (LoG).

-LoG







0	0	–1	0	0
0	–1	–2	–1	0
–1	–2	16	–2	–1
0	–1	–2	–1	0
0	0	–1	0	0

a	b
c	d

FIGURE 10.14 Laplacian of a Gaussian (LoG). (a) 3-D plot. (b) Image (black is negative, gray is the zero plane, and white is positive). (c) Cross section showing zero crossings. (d) 5 × 5 mask approximation to the shape of (a).

What does it do?

- The Gaussian blurs the image by reducing the intensity of structures (such as noise) at scales much lower than σ.
- The Laplacian part is responsible for detecting the edges due to the sensitivity of second derivative.
- Since filter is linear action these two filters can be applied separately, thus allowing us to use different sized filters for each of the actions.

The algorithm

- The algorithm is:
  - Filter image with nxn Gaussian filter
  - Compute the Laplacian using for example a 3x3 mask.
  - Find the zero crossings
- To find a zero crossing it is possible to use 3x3 mask that checks sign changes around a pixel.
- Sometimes it is suggested to use the algorithm with different σ and then to combine the results.

Canny edge detector

- Three basic objectives:
  - Low error rate – Edge detected must be as close as possible to the true edge.
  - Single edge point response – The detector should return only one point for each true edge point.

Step 1 – Smooth image

- Create a smoothed image from the original one using Gaussian function.

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$f_s(x,y) = G(x,y) \star f(x,y)$$

Step 2 – Compute Gradient

- We located the edges using direction of the point with the same method as basic edge detection – gradient:

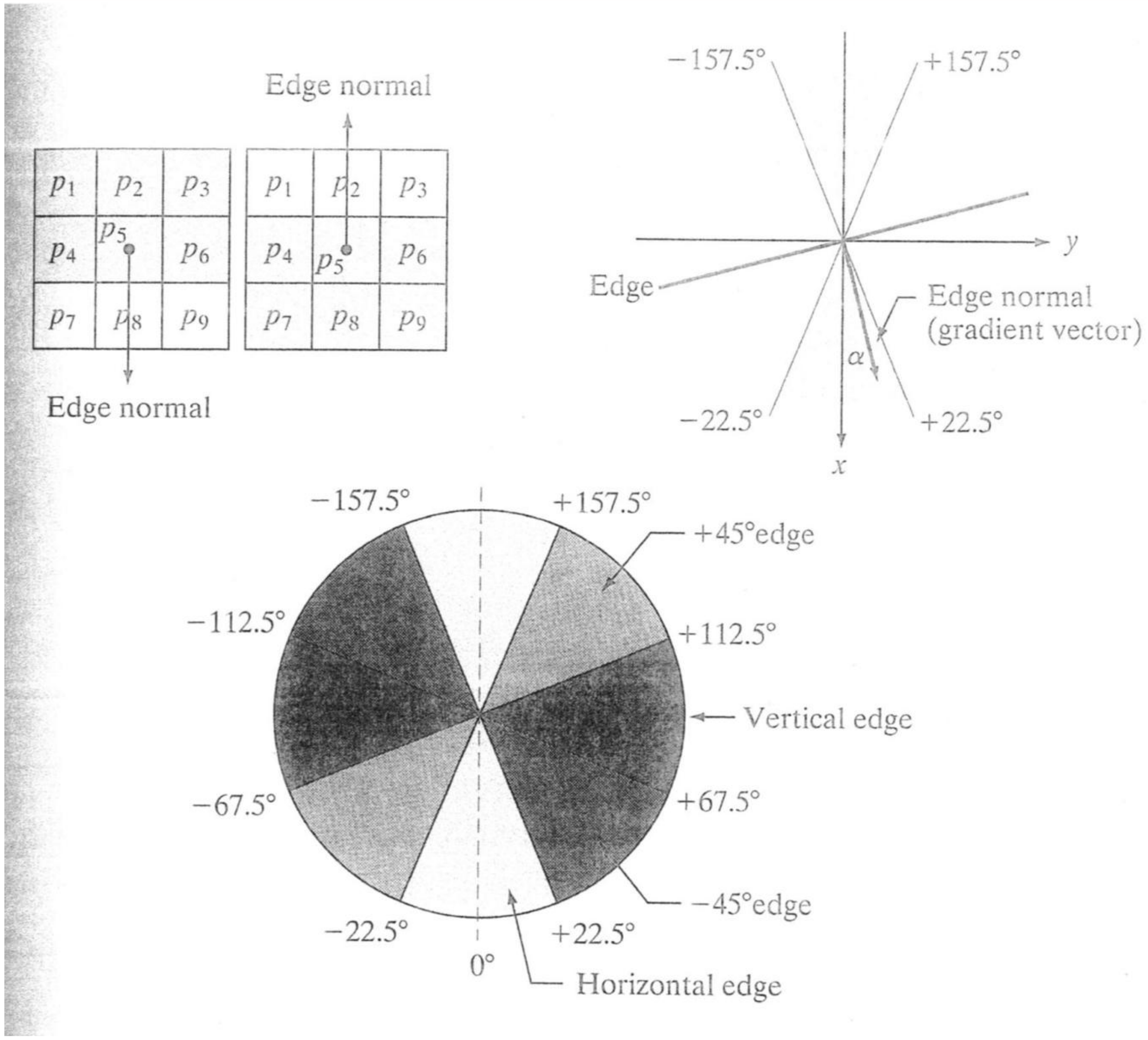
$$g_x = \frac{\partial f_s}{\partial x} \qquad g_y = \frac{\partial f_s}{\partial y}$$

$$M(x,y) = \sqrt{{g^2}_x + {g^2}_y} \qquad \propto (x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

Step 3 – Nonmaxima suppression

- Edges generated using gradient typically contain wide ridges around local maxima.
- We will locate the local maxima using non-maxima suppression method.
- We will define a number of discrete orientations. For example in a 3x3 region 4 direction can be defined.

Step 3 – Nonmaxima suppression



Step 3 – Nonmaxima suppression

- We will now generate a local maxima function

For each point:

1. Find the direction d_k that is closest to $\propto (x, y)$
2. If the value of $M(x, y)$ is less than at least one of two neighbors along d_k, $g_N(x, y) = 0$ otherwise $g_N(x, y) = M(x, y)$.

In the figure in the previous slide we would compare p_5 to p_2 and p_8.

Step 4 – Double Thresholding

- The received image may still contain false edge points.
- We will reduce them using *hysteresis* (or double) thresholding.

Let T_L , T_H be low and high thresholds. The suggested ratio is 1:3 or 2:3. We will define $g_{NH}(x,y)$ and $g_{NL}(x,y)$ to be $g_N(x,y)$ after threshloding it with T_L , T_H.
It is clear that $g_{NL}(x,y)$ contains all the point located in $g_{NH}(x,y)$.
We will separate them by substruction:
$$g_{NL}(x,y) = g_{NH}(x,y) - g_{NL}(x,y)$$
Now $g_{NL}(x,y)$ and $g_{NH}(x,y)$ are the “weak” and the “strong” edge pixels.

Step 4 – Double Thresholding

- The algorithm for building the edges is:
  1. Locate the next unvisited pixel p in $g_{NH}(x, y)$.
  2. Mark as valid pixels all the weak pixels in $g_{NL}(x, y)$ that are connected to p
  3. If not all non-zero pixels in $g_{NH}(x, y)$ have been visited return to step 1.
  4. Set all the non-marked pixels in $g_{NL}(x, y)$ to 0.
  5. Return $g_{NH}(x, y) + g_{NL}(x, y)$.

Outline

- Introduction
- Mathematical background
- Detection of isolated point
- Line detection
- Edge Models
- Basic edge detection
- Advanced edge detection
- **Edge linking and Boundary detection**

Edge Linking introduction

Ideally, edge detection should yield sets of pixels lying only on edges. In practice, these pixels seldom characterize edges completely because of noise or breaks in the edges.

Therefore, edge detection typically is followed by linking algorithms designed to assemble edge pixels into meaningful edges and/or region boundaries.

We will introduce two linking techniques.

Local processing

All points that are similar according to predefined criteria are linked, forming an edge of pixels that share common properties.

Similarity according to:

1. Strength (magnitude)
2. Direction of the gradient vector

Local processing

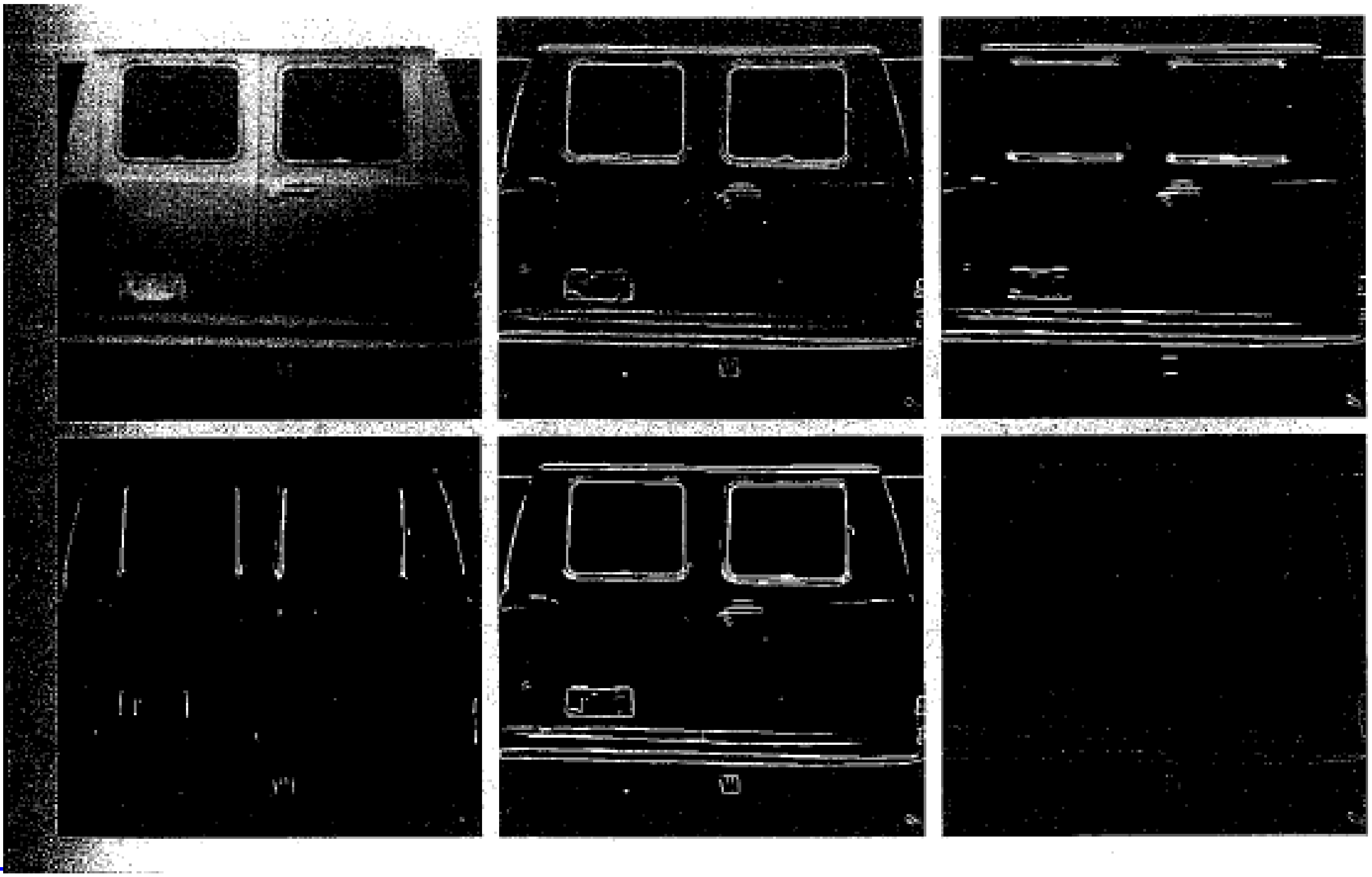
1. **1. Compute the gradient magnitude and angle arrays, *M*(x,y) and α(x,y), of the input image, *f*(x,y) .**
2. **Form a binary image, g, whose value at any pair of coordinates (x,y) is given by:**

$$g(x,y)=\begin{cases}1 & if\ M(x,y)>T_M\ AND\ \alpha(x,y)=A\pm T_A \\ 0 & otherwise\end{cases}$$

where T_M is a threshold, A is a specified angle direction, and $\pm T_A$ defines a "band" of acceptable directions about A

3. **Scan the rows of g and fill (set to 1) all gaps (sets of 0s) in each row that do not exceed a specified length, *K*. Note that, by definition, a gap is bounded at both ends by one or more 1s. The rows are processed individually with no memory between them.**
4. **To detect gaps in any other direction, *θ*, rotate g by this angle and apply the horizontal scanning procedure in step 3. Rotate the result back by -*θ*.**

Example



Global processing

- Good for unstructured environments.
- All pixels are candidate for linking.
- Need for predefined global properties.

- What we are looking for?
- Decide whether sets of pixels lie on curves of a specified shape.

Trivial application

- Given n points.
- We want to find subset of n that lie on straight lines:
  - Find all lines determined by every pair of points
  - Find all subset of points that are close to particular lines.

$n*(n-1)/2 + n*(n*(n-1))/2 \sim n^2+ n^3.$

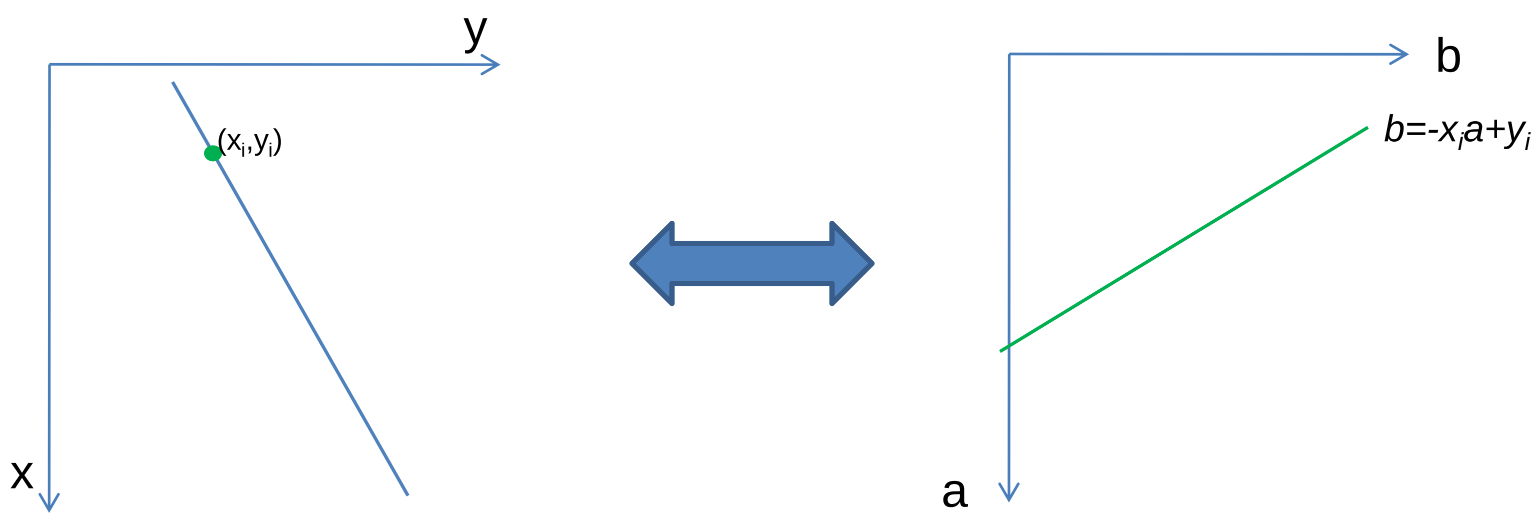
Too hard to compute.

Hough transform

- The *xy-plane*:
- Consider a point (x_i,y_i). The general equation of a straight line : $y_i = ax_i + b$.
- Many lines pass through (x_i,y_i). They all satisfy : $y_i = ax_i + b$ for varying values of a and b.

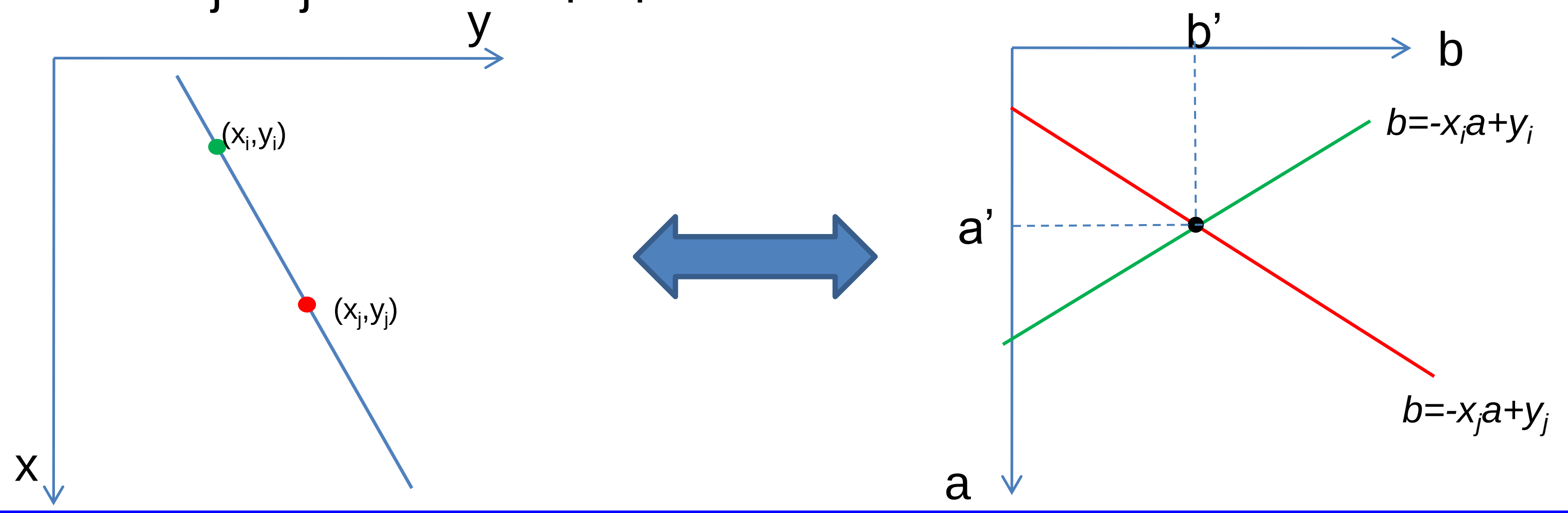
Hough transform

- The *ab-plane*: (Parameter space)
- Consider a point (x_i,y_i). The general equation of a straight line : $b=-x_ia+y_i$.
- Single line for a fixed pair (x_i,y_i). **–line 1**



Hough transform

- (x_j, y_j) also has a line in parameter space associated with it. **– line 2**.
- Unless line1 and line 2 are parallel they intersects at some point (a’, b’) where a’ is the slope and b’ the intercept of the line containing both (x_j, y_j) and (x_i, y_i).



Hough transform

- Parameter – space lines corresponding to all points (x_k,y_k) in the *xy-plane* could be plotted and the principal lines in that plane could be found by identifying points in parameter space where large numbers of lines intersect.

Hough transform

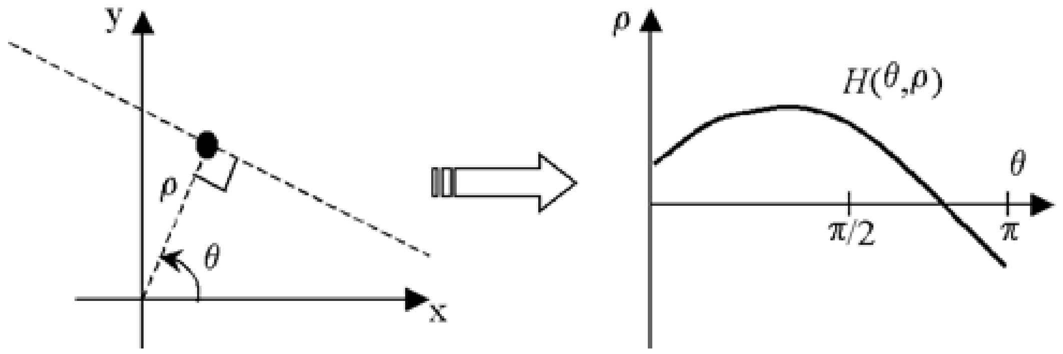
- Using $y_i = ax_i + b$ approaches infinity when lines approach the vertical direction.

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

Normal representation of a line.

Hough transform

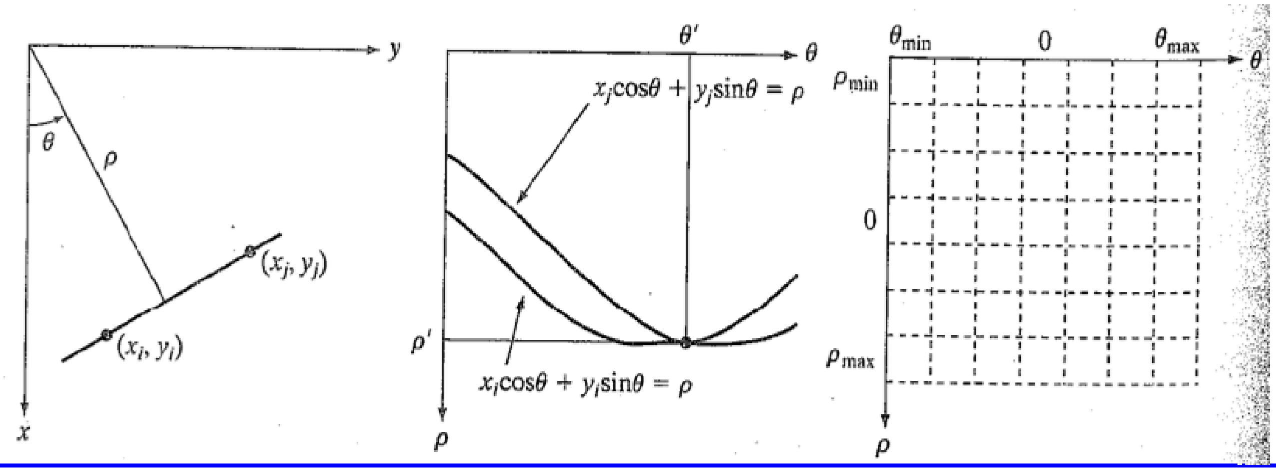
- Each line is transformed into a sinusoidal in the *ρθ-plane, where*
  - *ρ* – The distance between the line and the origin *.*
  - *θ* – The angle between the distance vector and the positive *x-axis .*



- eventually all sinusoidals that intersect at a specific point *(ρ',θ')* transform back to specific points on the same line in the *xy-plane*

Hough transform

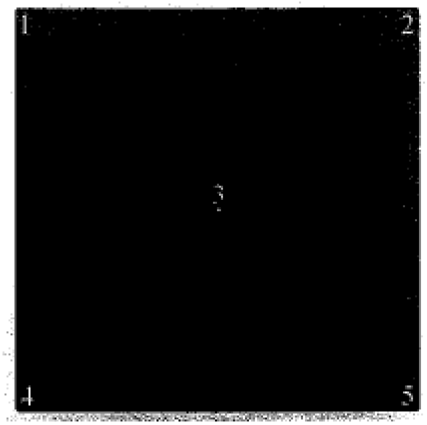
- We divide the $\rho\theta$-plane into *accumulator cells* between the minimum and maximum values of ρ and θ.
- Each cell has a value of zero .
- For every point (x_k, y_k) in the *xy*-plane we calculate ρ's for all allowed values of θ.
- For every result (ρ_p, θ_q) we raise the value of the corresponding cell (p,q) by 1.

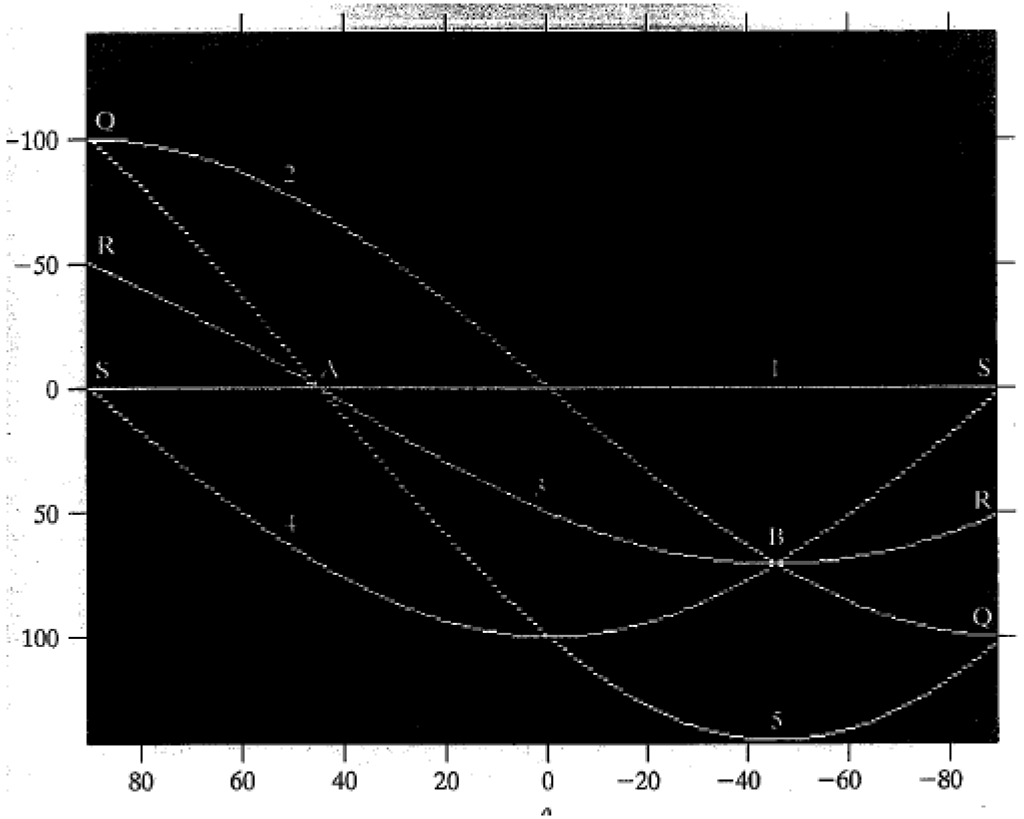


Hough transform

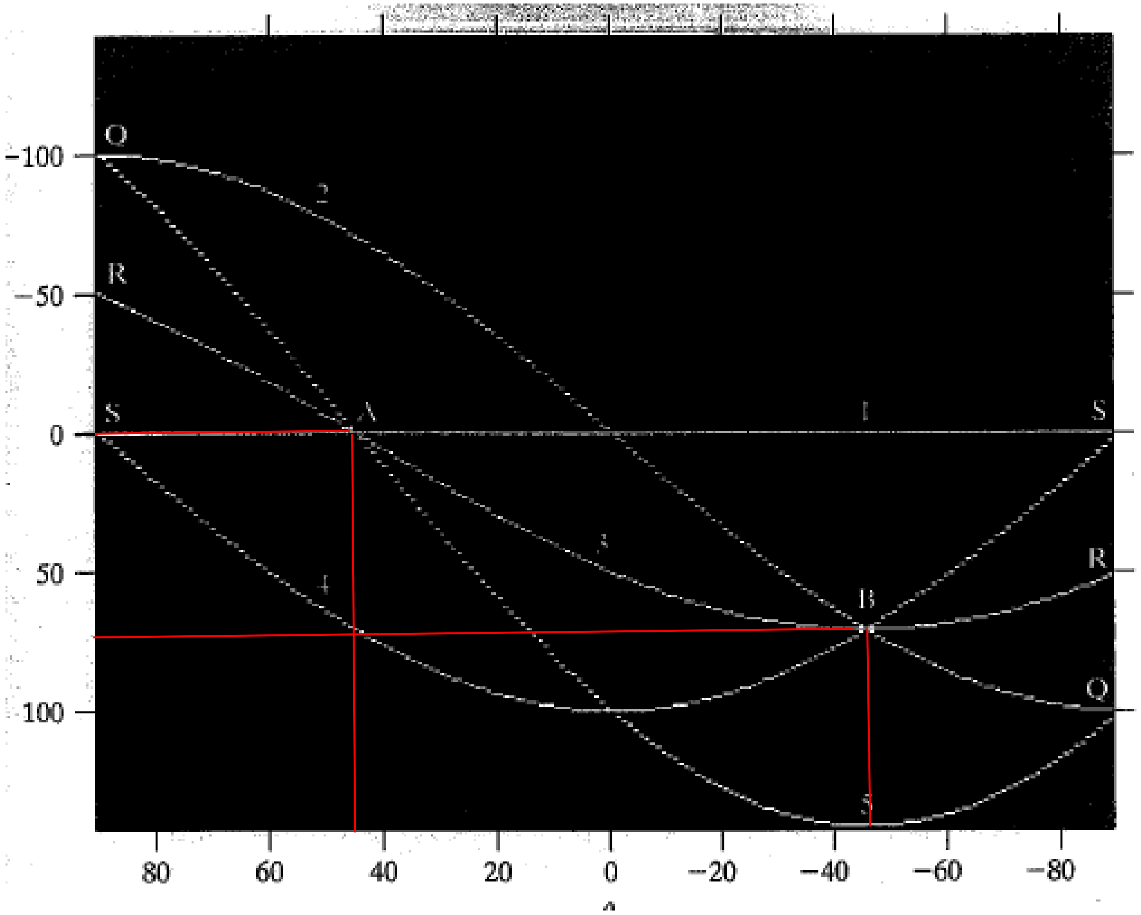
- At the end of the process, the value *P* of the cell A(*i*,*j*) represents a number of *P* points in the *xy*-plane that lie on the $x\cos\theta j + y\sin\theta j = \rho i$ line.
- The number of sub-divisions determines the accuracy of the co-linearity of the points.

Hough transform





Hough transform



0

71

45deg

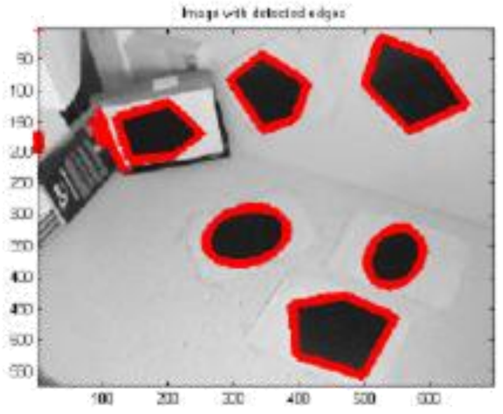
-45deg

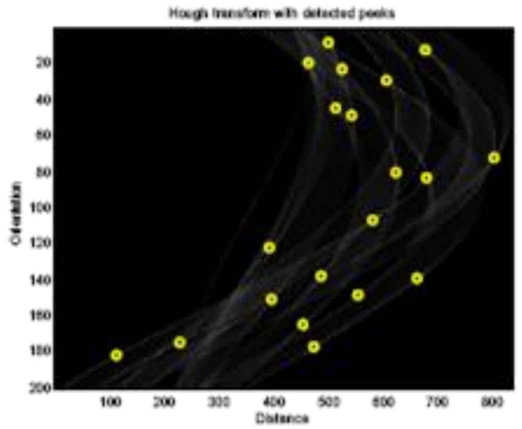
Hough transform – Edge linking

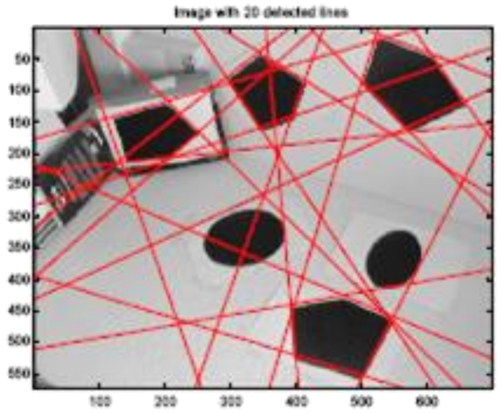
Algorithm :

1. Obtain a *binary* edge image using any of the techniques discussed earlier in this section
2. Specify subdivisions in the $\rho\theta$ –plane.
3. Examine the counts of the accumulator cells for high pixel concentrations .
4. Examine the relationships (principally for continuity) between pixels in a chosen cell .

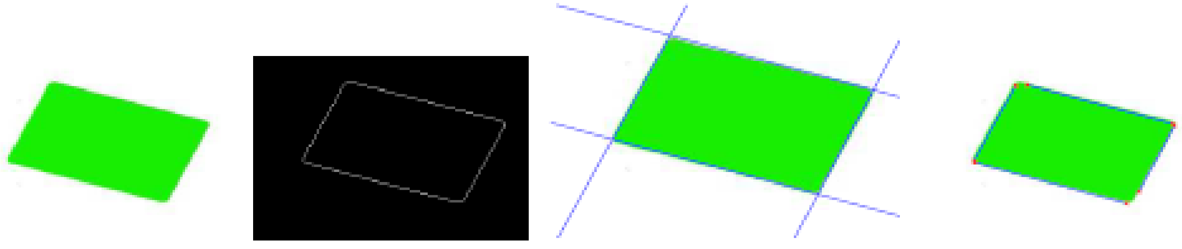
Hough transform - Examples

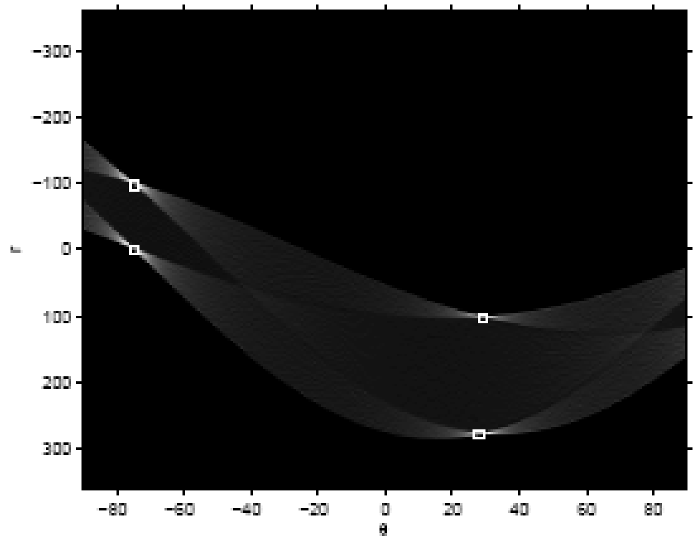






Hough transform - Examples


(a) Original (b) Edges (c) Infinite Lines (d) Finite Lines



(e) Hough Transform

Questions?

Thank You!