Digital Image Processing

Lecture # 7

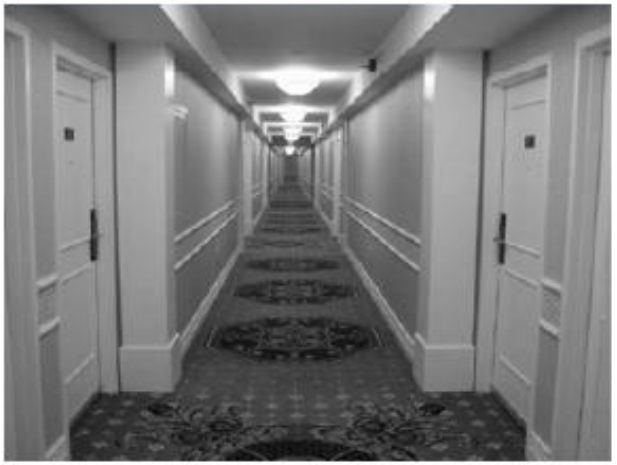
Hough Transform, Segmentation & Morphological Operations

Introduction to Hough transform

- The Hough transform (HT) can be used to detect lines, circles or other parametric curves.
- It was introduced in 1962 (Hough 1962) and first used to find lines in images a decade later (Duda 1972).
- The goal is to find the location of lines in images.
- This problem could be solved by e.g. Morphology and a linear structuring element, or by correlation.
  - Then we would need to handle rotation, zoom, distortions etc.
- Hough transform can detect lines, circles and other structures if their parametric equation is known.
- It can give robust detection under noise and partial occlusion.

An image with linear structures

- Borders between the regions are straight lines.
- These lines separate regions with different grey levels.
- Edge detection is often used as preprocessing to Hough transform.

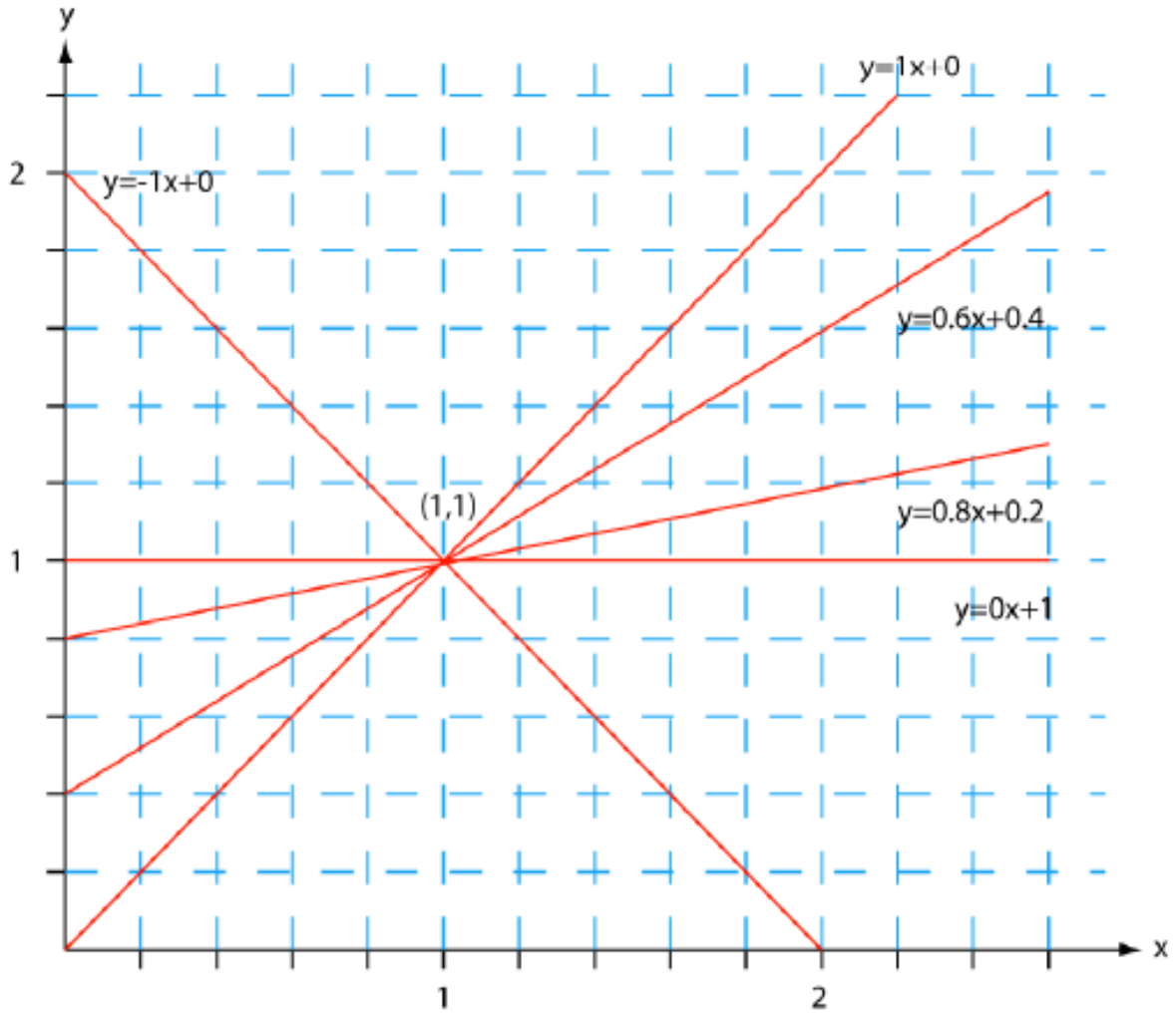


Hough-transform

- Assume that we have performed some edge detection, and a thresholding of the edge magnitude image.
- Thus, we have *n* pixels that may partially describe the boundary of some objects.
- We wish to find sets of pixels that make up straight lines.
- Regard a point (x_i, y_i)

  and a straight line $y_i = ax_i + b$
  - There are many lines passing through the point (x_i, y_i).
  - Common to them is that they satisfy the equation for some set of parameters (a, b).

Hough transform – basic idea



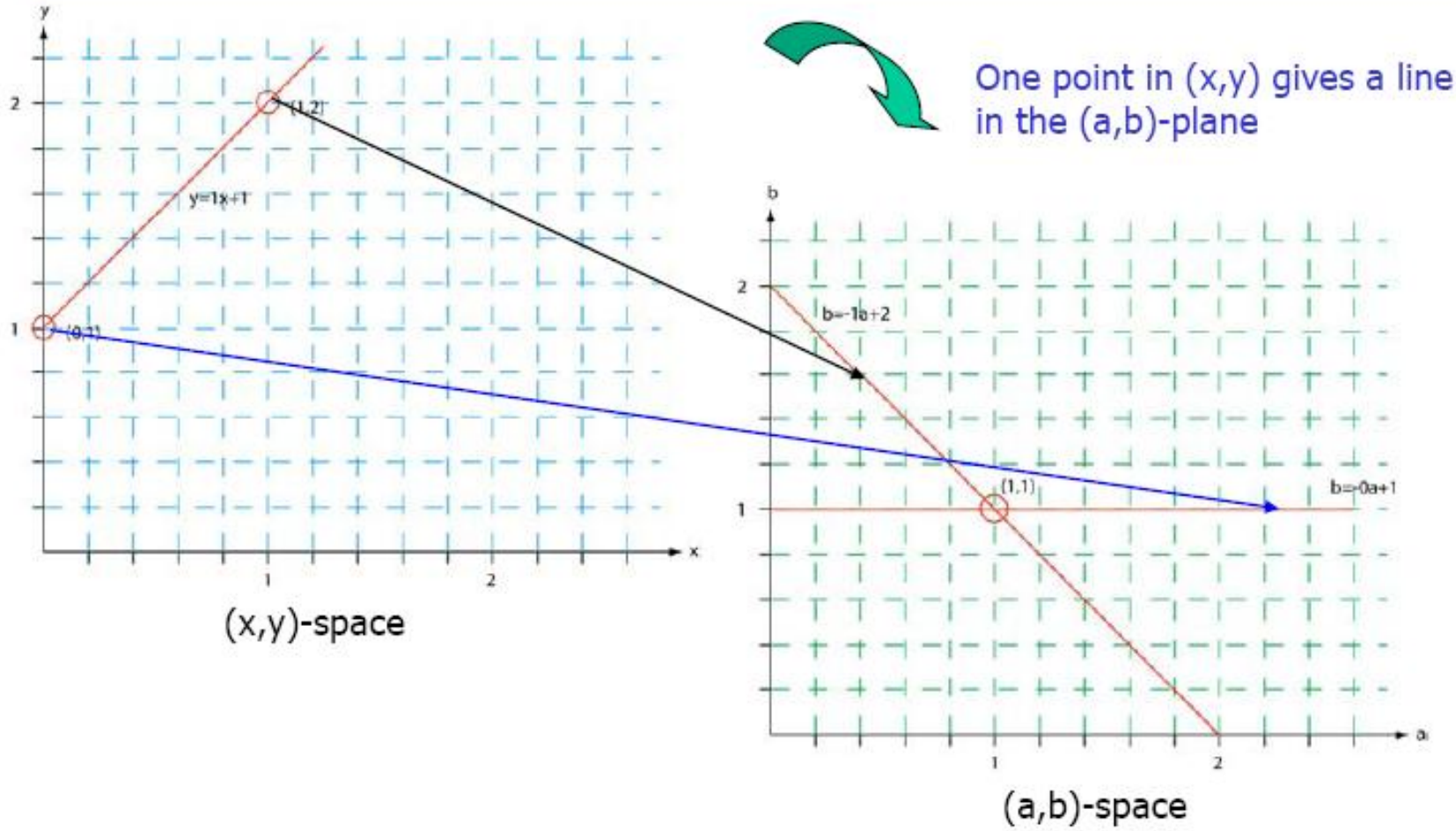
Hough transform – basic idea

- This equation can obviously be rewritten as follows:

$$b = -xa + y$$

- We now consider *x* and *y* as parameters and *a* and *b* as variables.
- This is a line in (*a*,*b*) space parameterized by *x* and *y*.
  - So: a single point in xy-space gives a line in (a,b) space.
- Another point (*x*,*y*) will give rise to another line in (*a*,*b*) space.

Hough transform – basic idea

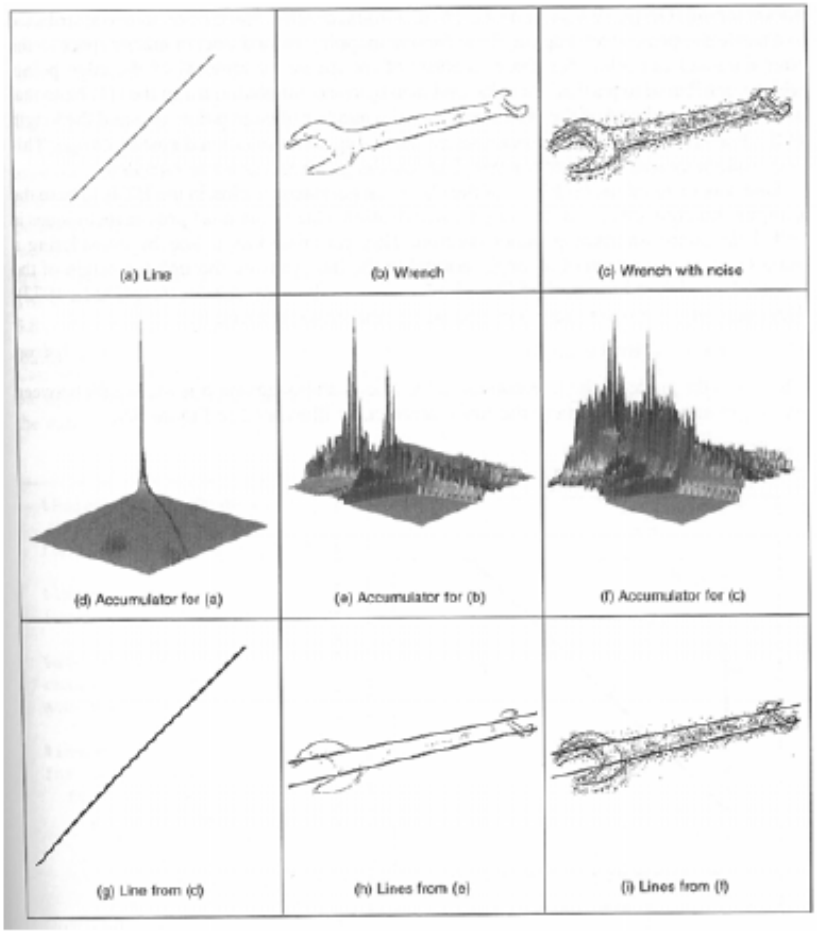


Example – images and accumulator space

Thresholded edge images

Visualizing the accumulator space
The height of the peak will be defined by the number of pixels in the line.

Thresholding the accumulator space and superimposing this onto the edge image

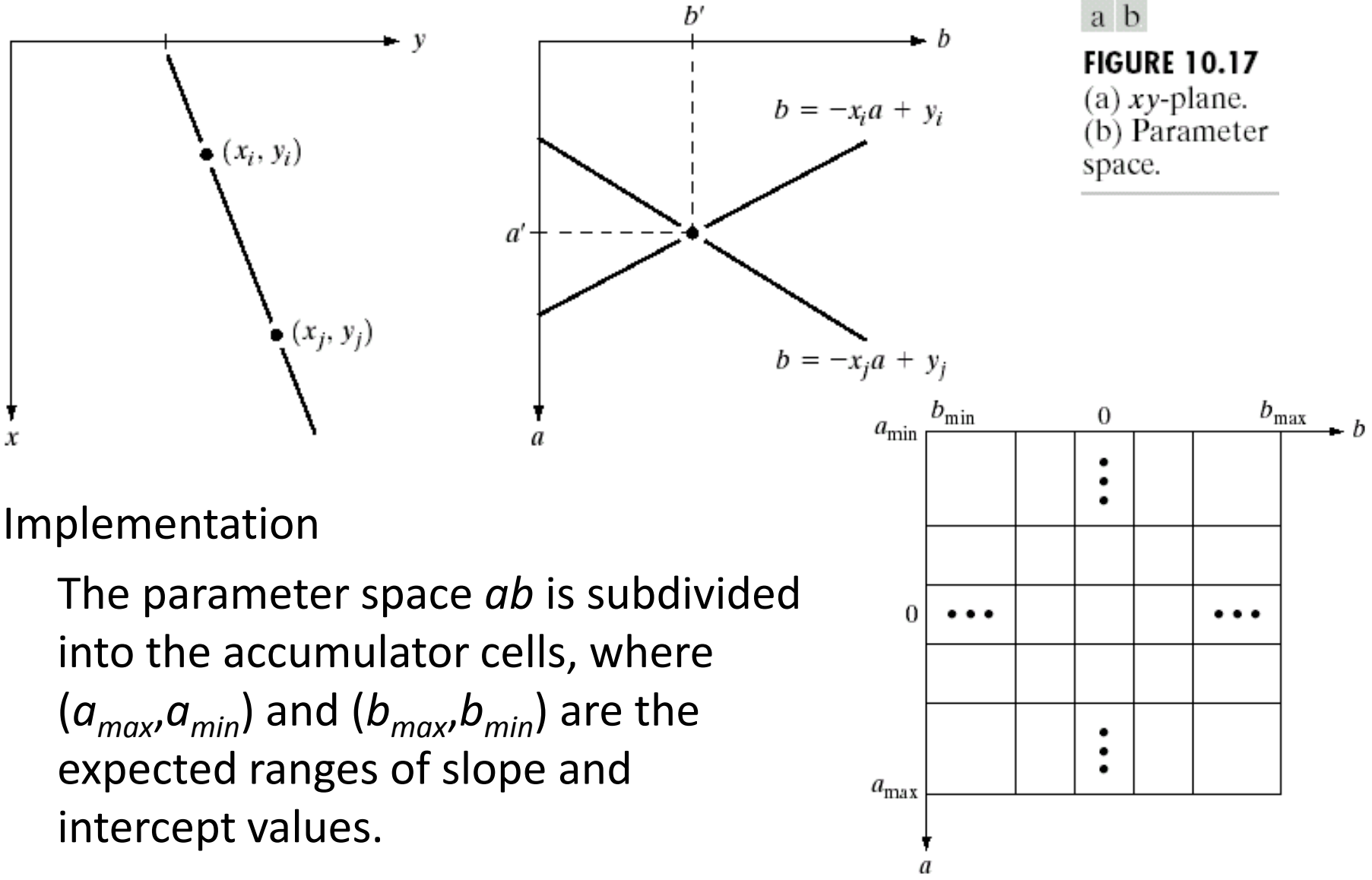


Note how noise effects the accumulator. Still with noise, the largest peaks correspond to the major lines.

Hough Transform

- The edge points are linked by determining if they lie on a curve of specified shape.
- Let us take the case of straight lines: we want to link points if they lie on a straight line.
- Consider a point (x_i,y_i), the equation of any line passing through this point is given by: $y_i = a\,x_i + b$, which can be written as: $b = -\,x_i a + y_i$. Therefore every point in xy plane corresponds to a straight line in ab plane.
- If there is a second point (x_j,y_j), another line with equation: $b = -\,x_j a + y_j$ is drawn in the ab plane.
- The intersection of the two lines in ab space give the values of (a',b'), which define a line passing through both points (x_i,y_i) and (x_j,y_j).

Hough Transform

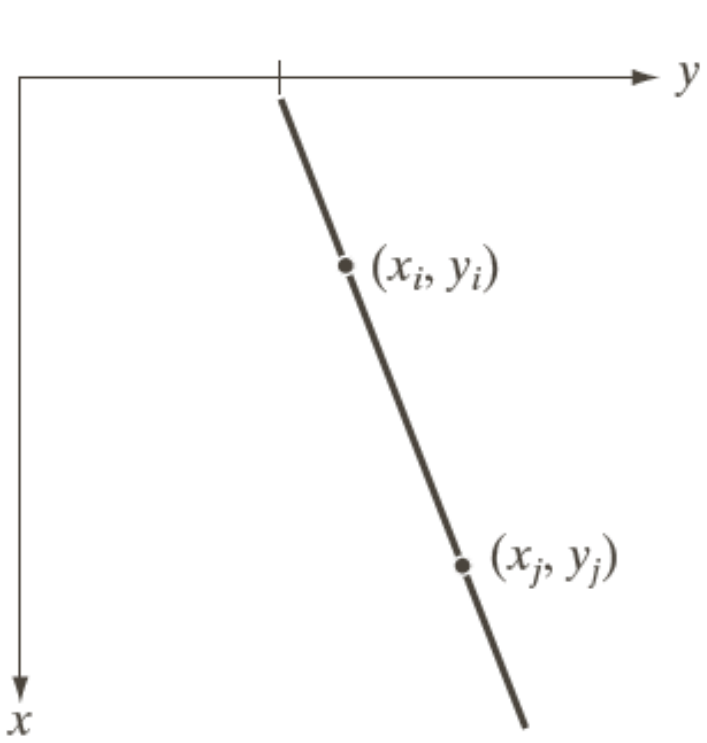


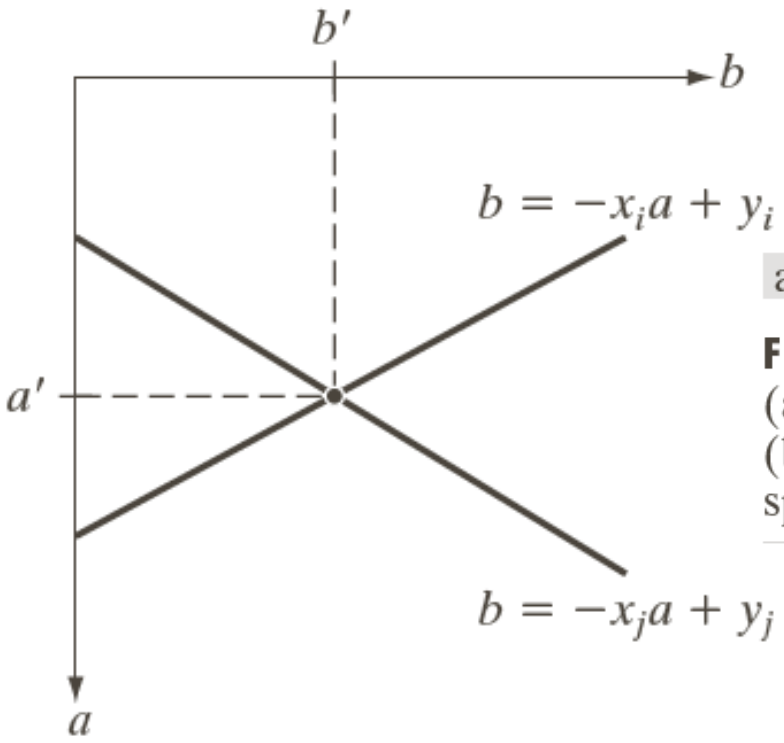
a b

FIGURE 10.17
(a) *xy*-plane.
(b) Parameter space.

Implementation

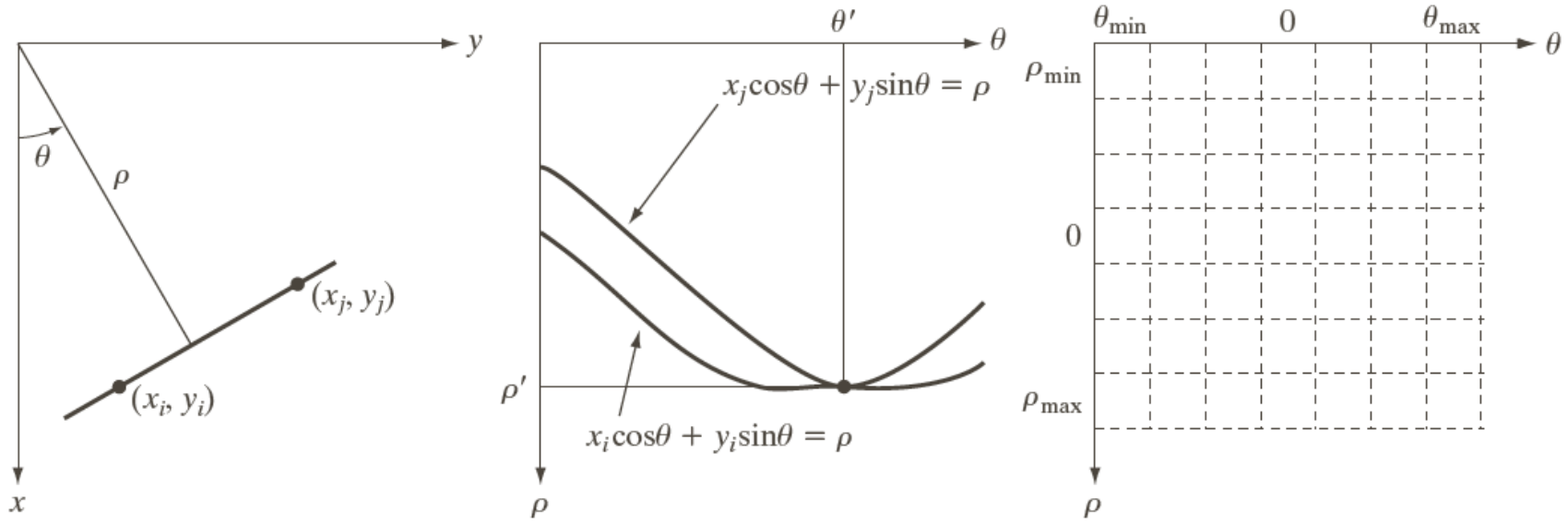
The parameter space *ab* is subdivided into the accumulator cells, where (a_{max},a_{min}) and (b_{max},b_{min}) are the expected ranges of slope and intercept values.





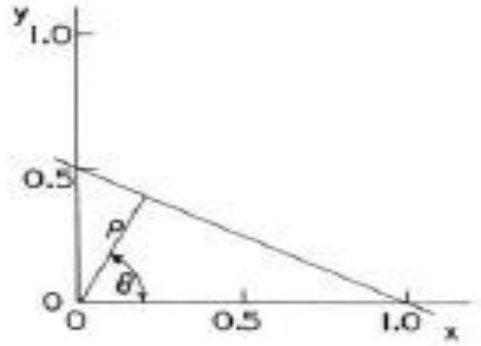
a b

FIGURE 10.31
(a) *xy*-plane.
(b) Parameter space.

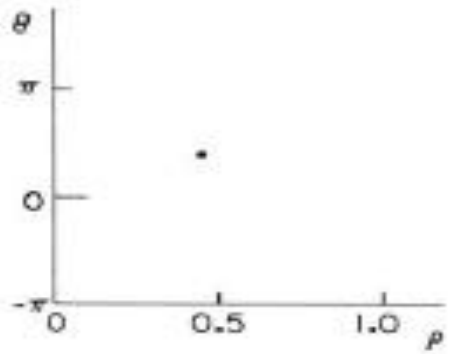


a b c

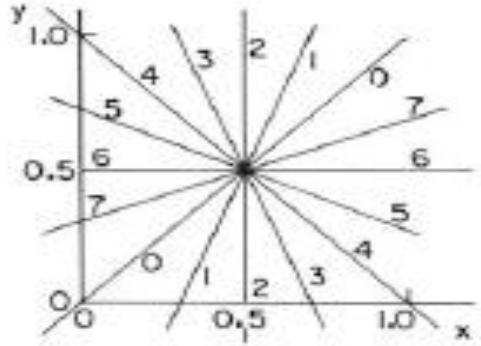
FIGURE 10.32 (a) (ρ, θ) parameterization of line in the xy-plane. (b) Sinusoidal curves in the $\rho\theta$-plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy-plane. (c) Division of the $\rho\theta$-plane into accumulator cells.



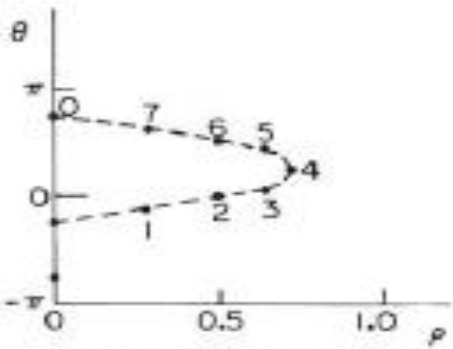
(a) Parametric line



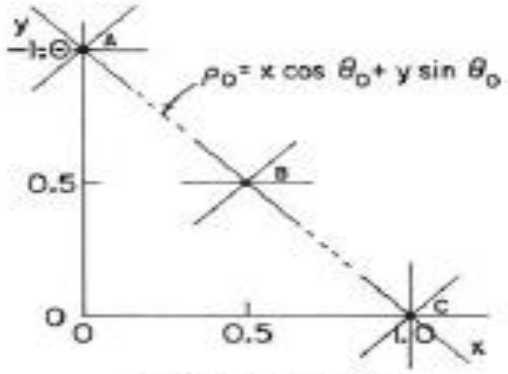
(b) Hough transform of (a)



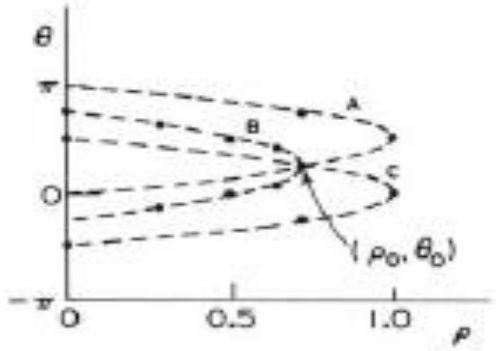
(c) Family of lines, common point



(d) Hough transform of (c)



(e) Colinear points



(f) Hough transform of (e)

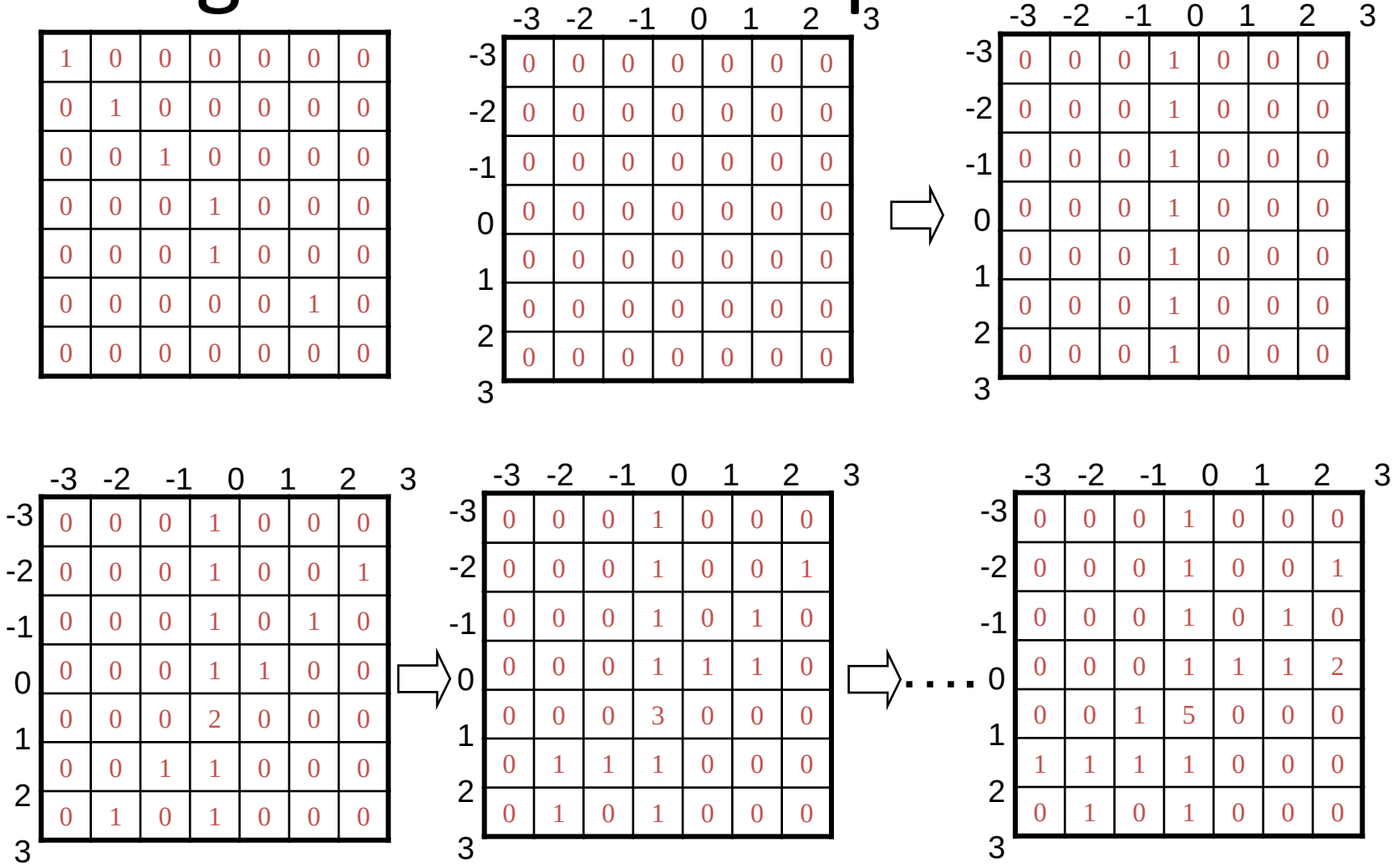
Hough Transform

Implementation steps:

- Initially the accumulator cells are set to zero.
- Then for every point (x_k,y_k) in the image, a is varied over the allowed subdivision values and the corresponding b values are calculated using $b = -x_k a + y_k$.
- The resulting b are then rounded off to the allowed values of b.
- If a value of a_p results in solution b_q, we let the corresponding accumulator value $A(p,q) = A(p,q) +1$.
- In the end the value of Q in $A(i,j)$ corresponds to Q points on the line $y = -a_i x + b_j$.

  Note: The number of subdivisions in the ab plane determines the accuracy of the colinearity of these points.

Hough Transform Implementation

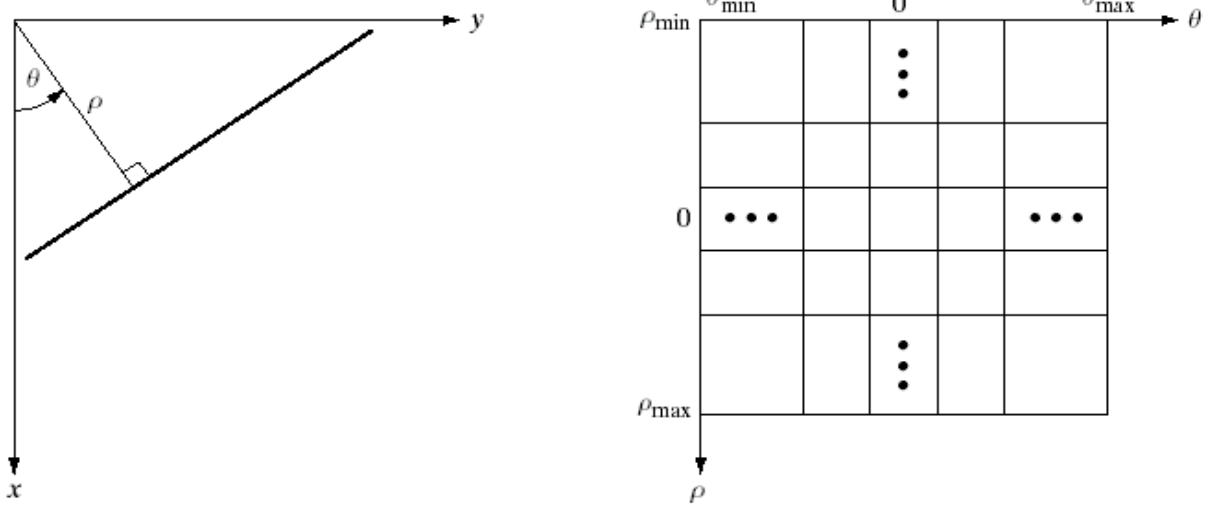


Parameters a and b can take values between –infinity to +infinity

Hough Transform

- Limitation in using $y_i = a\,x_i + b$, as the representation of straight line is that slope approaches to infinity as the line approaches to be vertical.
- Therefore usually Hough Transform is implemented using the polar equation of straight line, i.e. $x\cos\theta + y\sin\theta = \rho$

  and instead of ab-plane $\rho\theta$-plane is used.
- Every point in image gives a sinusoidal curve in the $\rho\theta$-plane.

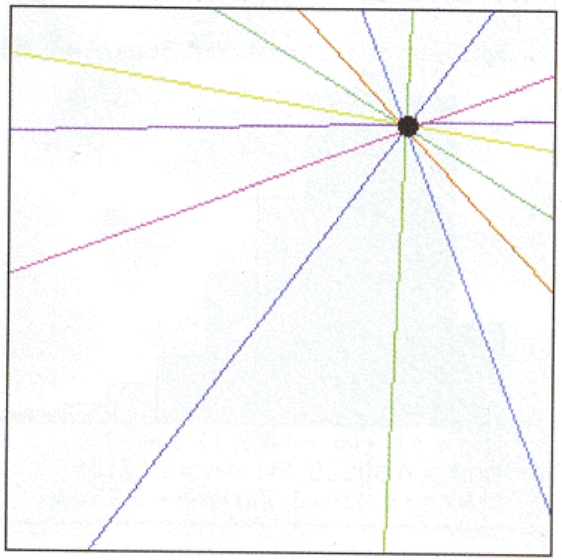


a b

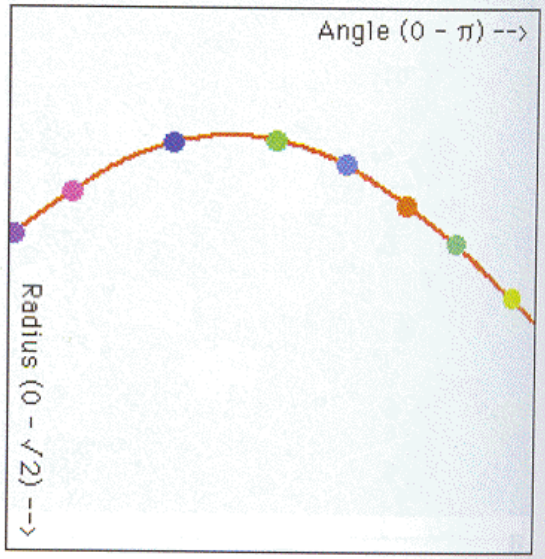
FIGURE 10.19
(a) Normal representation of a line.
(b) Subdivision of the $\rho\theta$-plane into cells.

Hough Transform

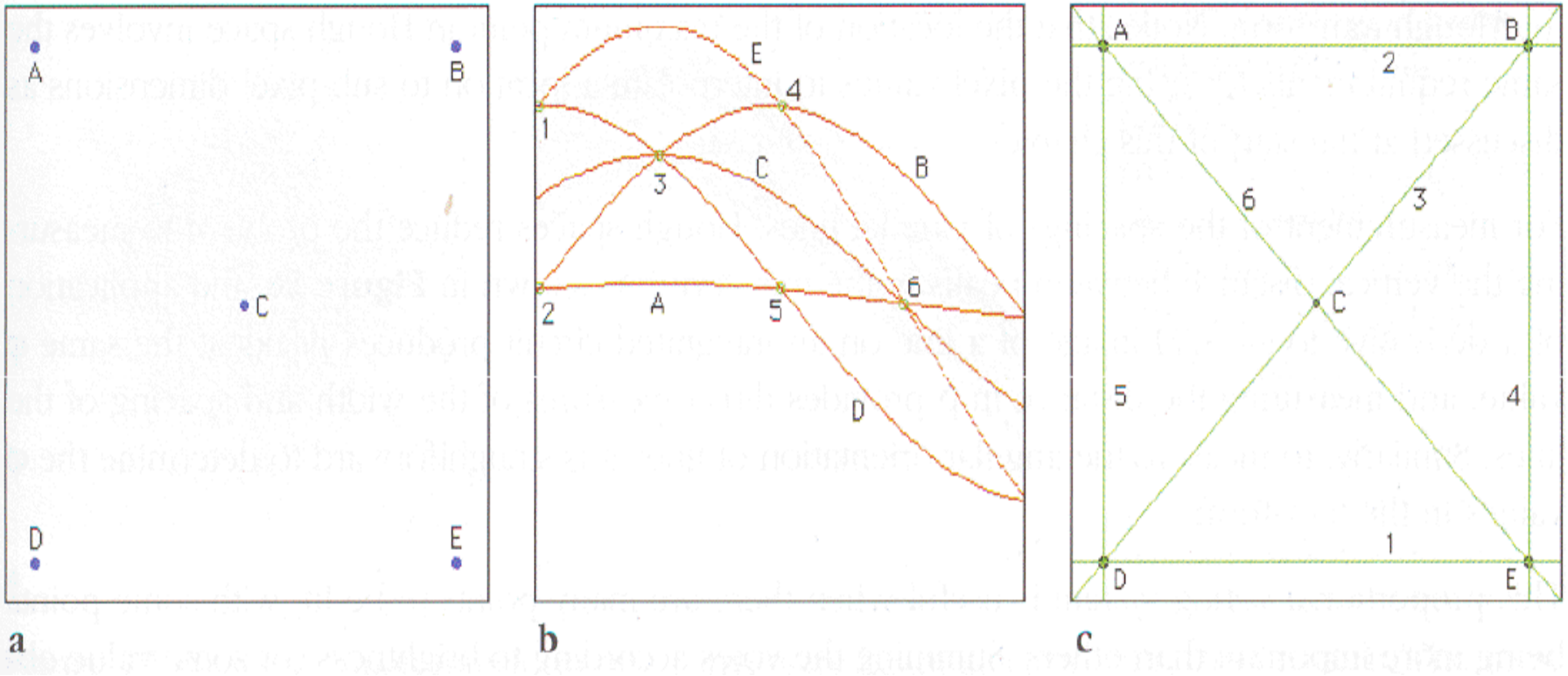
Principle of the Hough transform. *Each point in the real-space image* ***(a)*** *produces a sinusoidal line in Hough space* ***(b)*** *representing all possible lines that can be drawn through it. Each point in Hough space corresponds to a line in real space. The real-space lines corresponding to a few of the points along the sinusoid are shown, with color coding to match them to the points.*

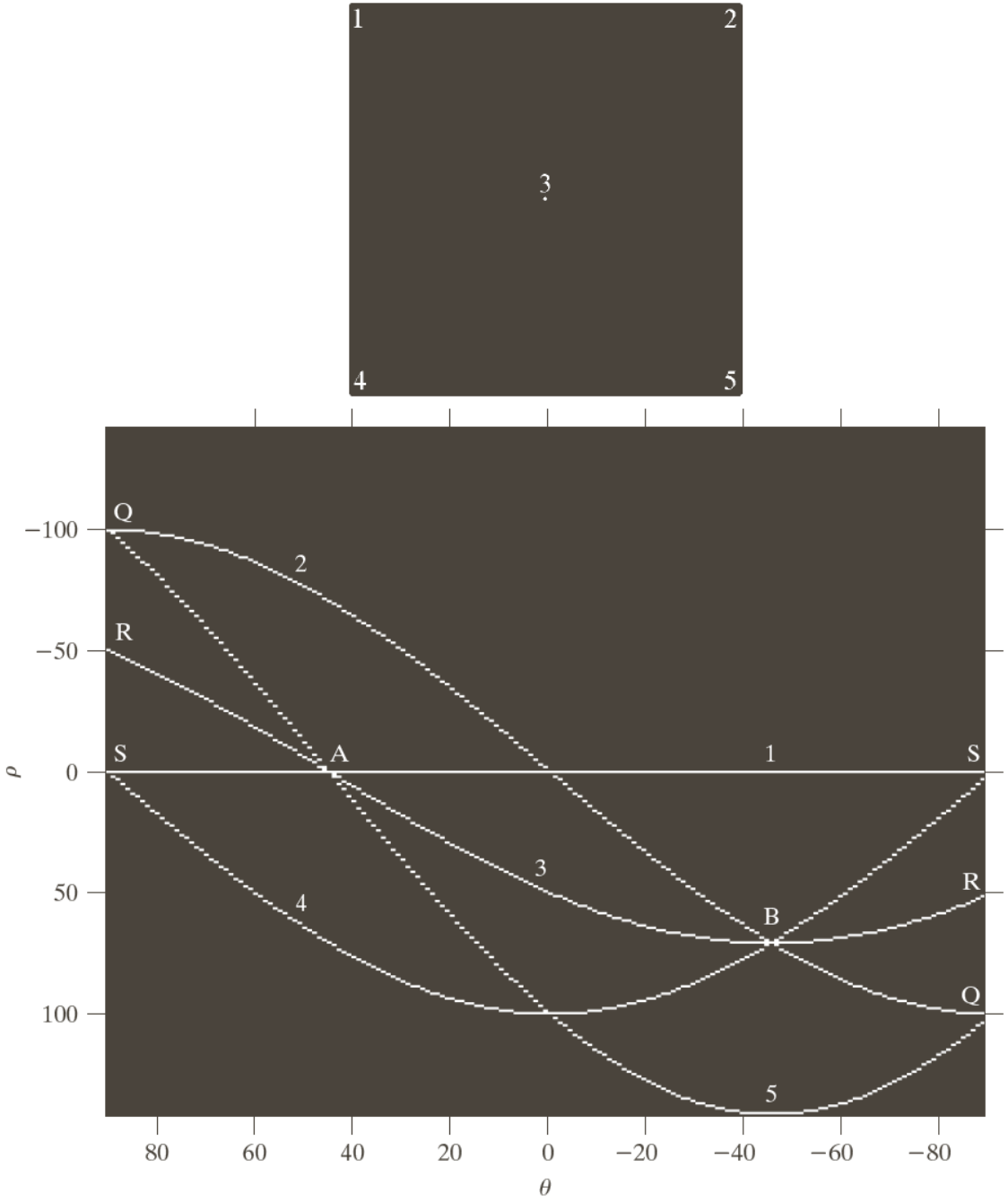


a



b

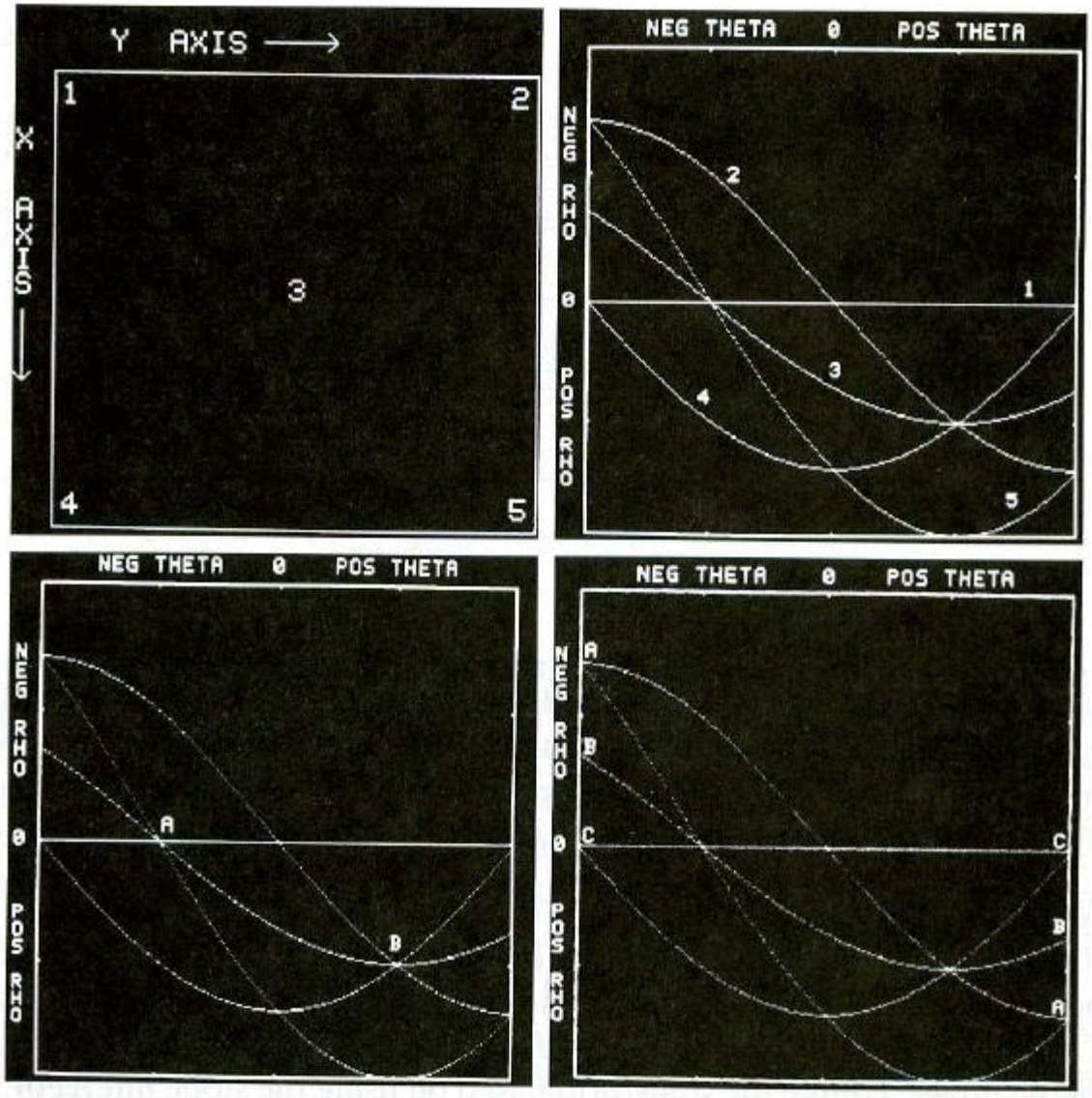

A
B
C
D
E
a
E
4
1
3
C
B
6
2
A
5
D
b
A
B
2
6
3
C
5
4
1
D
E
c



a
b

FIGURE 10.33
(a) Image of size 101×101 pixels, containing five points.
(b) Corresponding parameter space. (The points in (a) were enlarged to make them easier to see.)

Hough Transform



Implementation of the Hough transform

- Construct an array representing θ, ρ
- For each point, render the curve (θ, ρ) into this array, adding one at each cell
- Difficulties
  - how big should the cells be? (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)
- How many lines?
  - count the peaks in the Hough array
- This method can be extended to other type of curves also by using the equation for the desired curve, e.g. circle:

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2$$

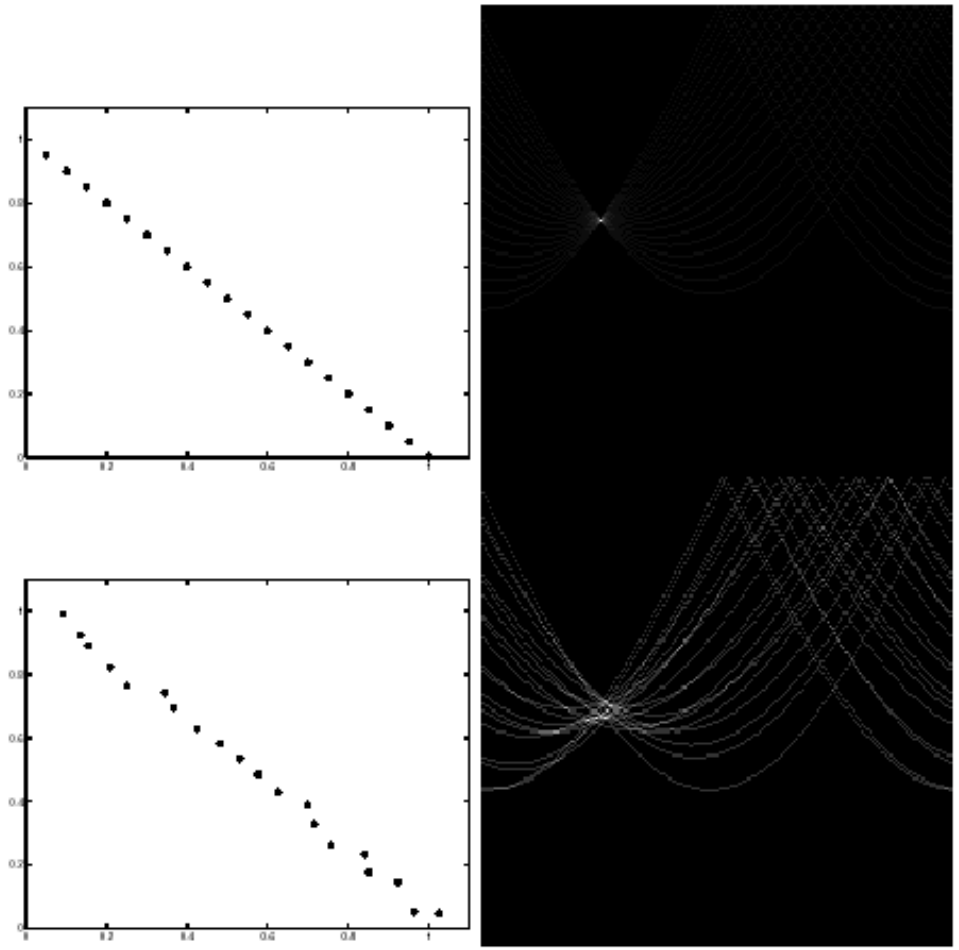


Figure 16.2. The Hough transform maps each point like token to a curve of possible lines (or other parametric curves) through that point. These figures illustrate the Hough transform for lines. The left hand column shows points, and the right hand column shows the corresponding accumulator arrays (the number of votes is indicated by the grey level, with a large number of votes being indicated by bright points). The top shows a set of 20 points drawn from a line next to the accumulator array for the Hough transform of these points. Corresponding to each point is a curve of votes in the accumulator array; the largest set of votes is 20. The horizontal variable in the accumulator array is θ and the vertical variable is r; there are 200 steps in each direction, and r lies in the range $[0, 1.55]$. In the center, these points have been offset by a random vector each element of which is uniform in the range $[0, 0.05]$; note that this offsets the curves in the accumulator array shown next to the points; the maximum vote is now 6.

Hough Transform

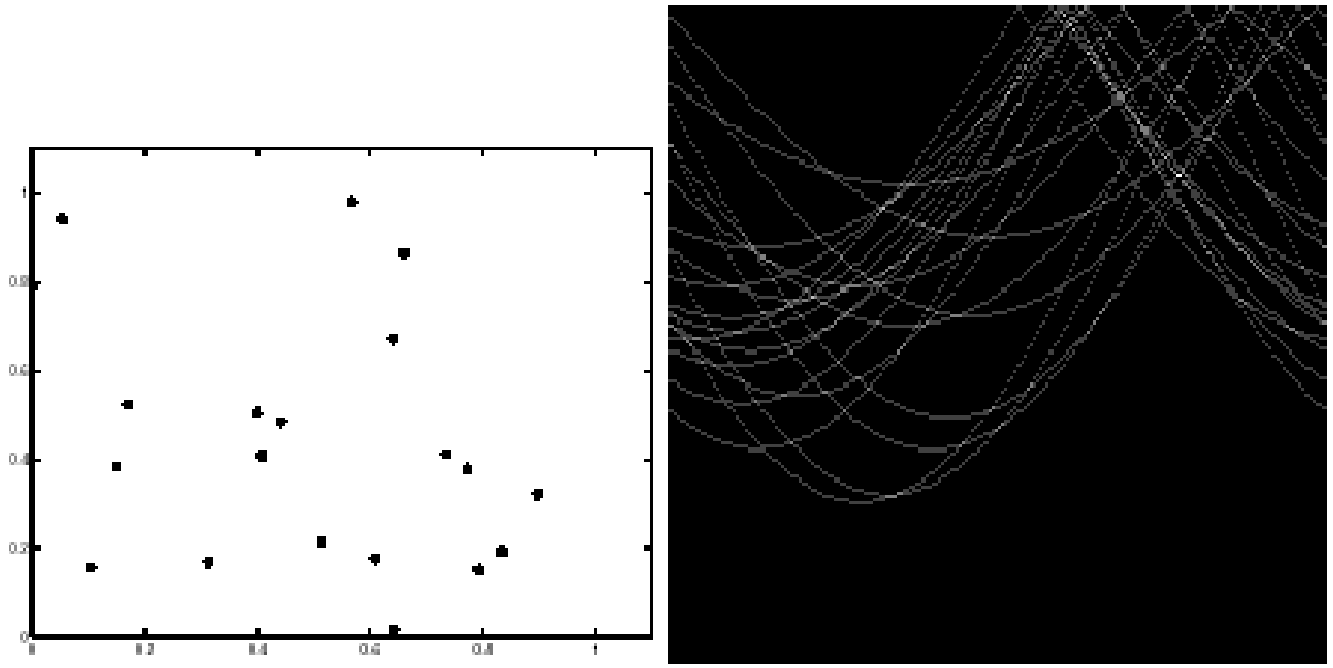
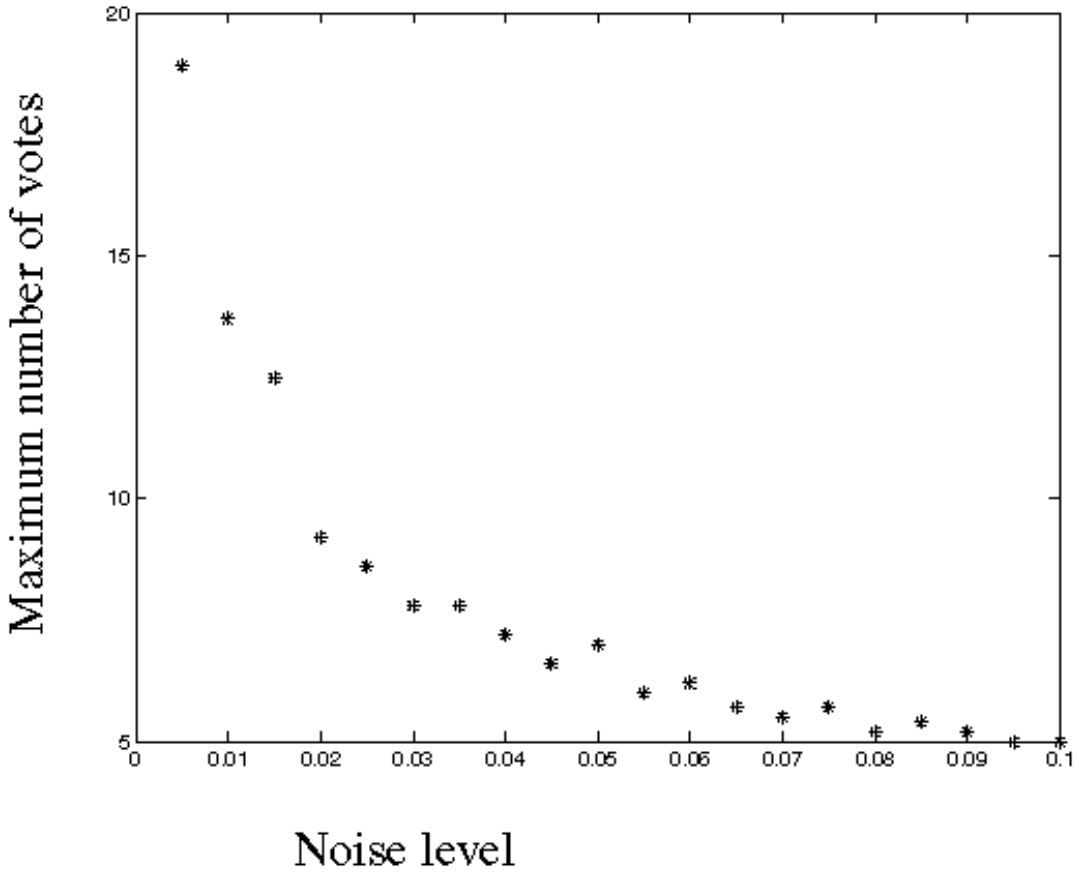


Figure 16.3. The Hough transform for a set of random points can lead to quite large sets of votes in the accumulator array. As in figure 16.2, the left hand column shows points, and the right hand column shows the corresponding accumulator arrays (the number of votes is indicated by the grey level, with a large number of votes being indicated by bright points). In this case, the data points are noise points (both coordinates are uniform random numbers in the range $[0,1]$); the accumulator array in this case contains many points of overlap, and the maximum vote is now 4. Figures 16.4 and explore noise issues somewhat further.

Noise Limitations of Hough Transform

- Two main limitations:noise and cell size



Results for a specimen of line with 20 points, with different amounts of noise

Hough Transform

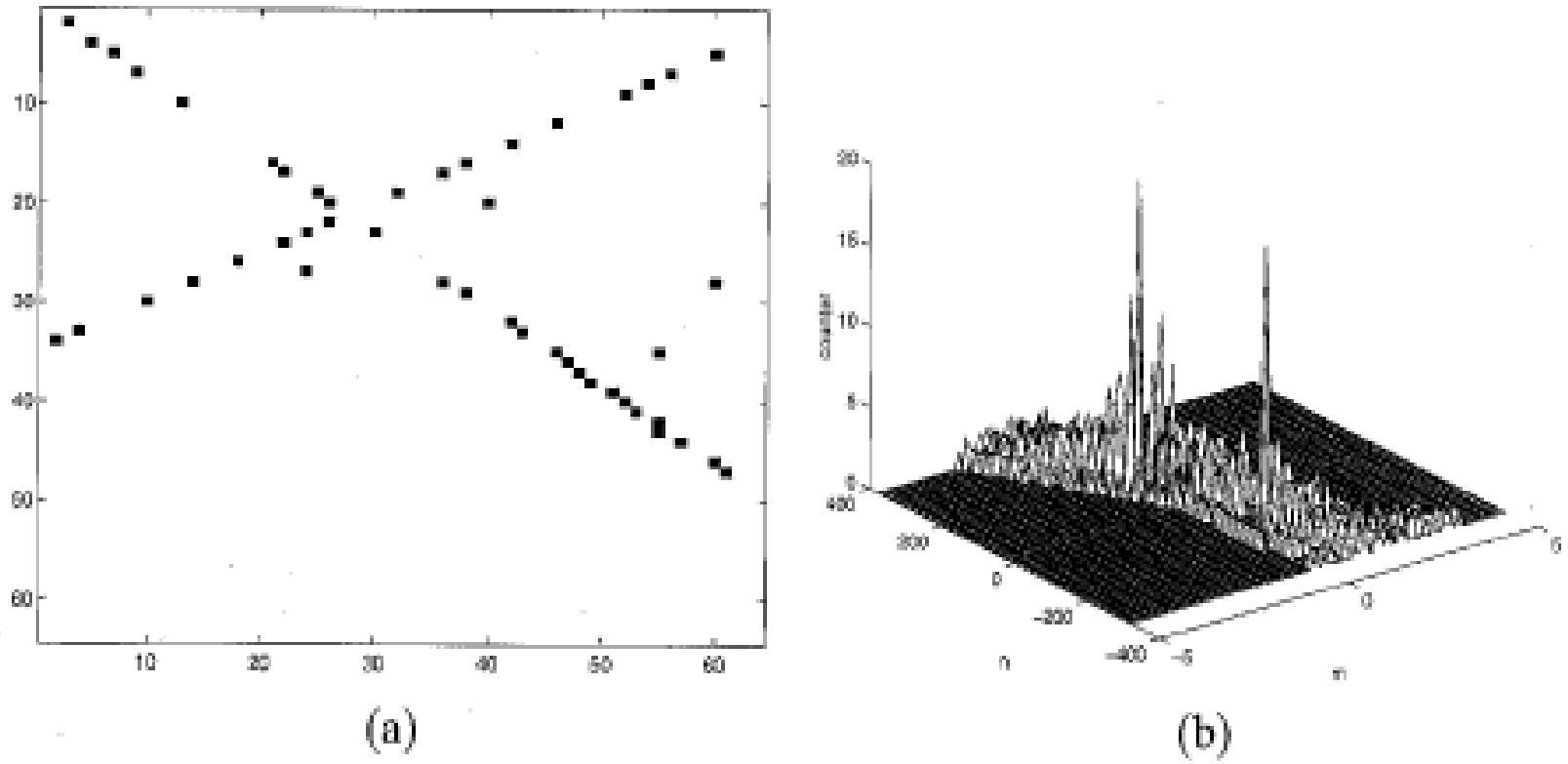


Figure 5.2 (a) An image containing two lines, sampled irregularly, and several random points. (b) Plot of the counters in the corresponding parameter space (how many points contribute to each cell (m, n)). Notice that the main peaks are obvious, but there are many secondary peaks.

Image Segmentation

Image Segmentation

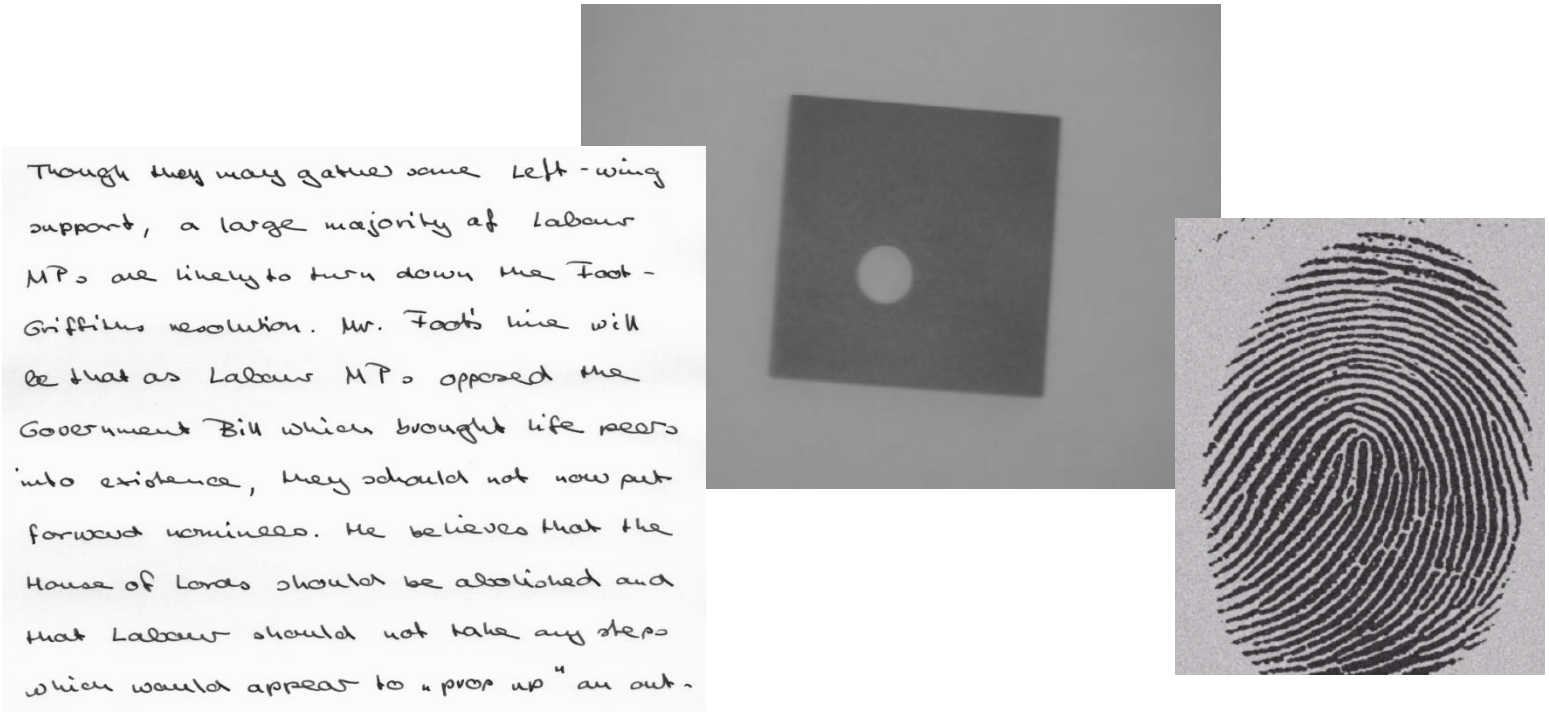
- Group similar components (such as, pixels in an image, image frames in a video)
- Applications: Finding tumors, veins, etc. in medical images, finding targets in satellite/aerial images, finding people in surveillance images, summarizing video, etc.

Image Segmentation

- Segmentation algorithms are based on one of two basic properties of gray-scale values:
  - Discontinuity
    - Partition an image based on abrupt changes in gray-scale levels.
    - Detection of isolated points, lines, and edges in an image.
  - Similarity
    - Thresholding, region growing, and region splitting/merging.

Thresholding

- Segmentation into two classes/groups
  - Foreground (Objects)
  - Background



Thresholding

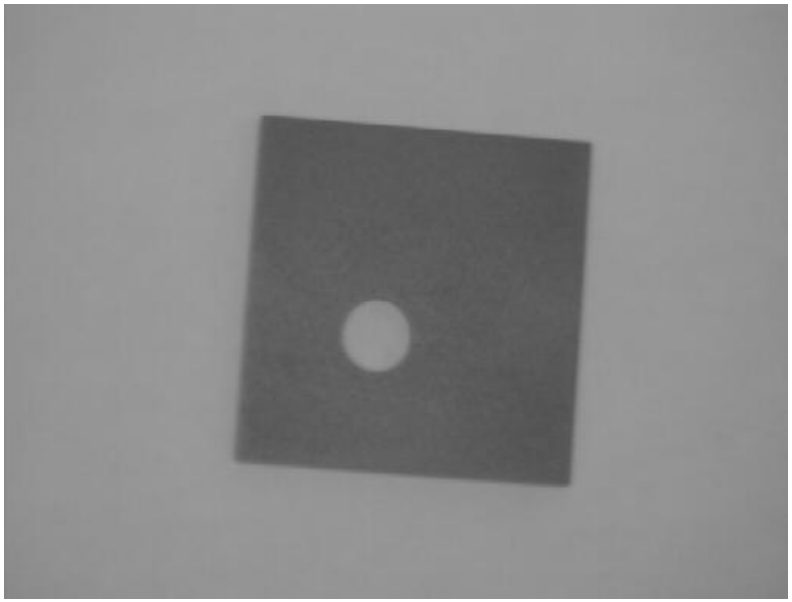
$$g(x,y)=\begin{cases}1 & if \;\; f(x,y)>T \\ 0 & if \;\; f(x,y)\le T\end{cases}$$

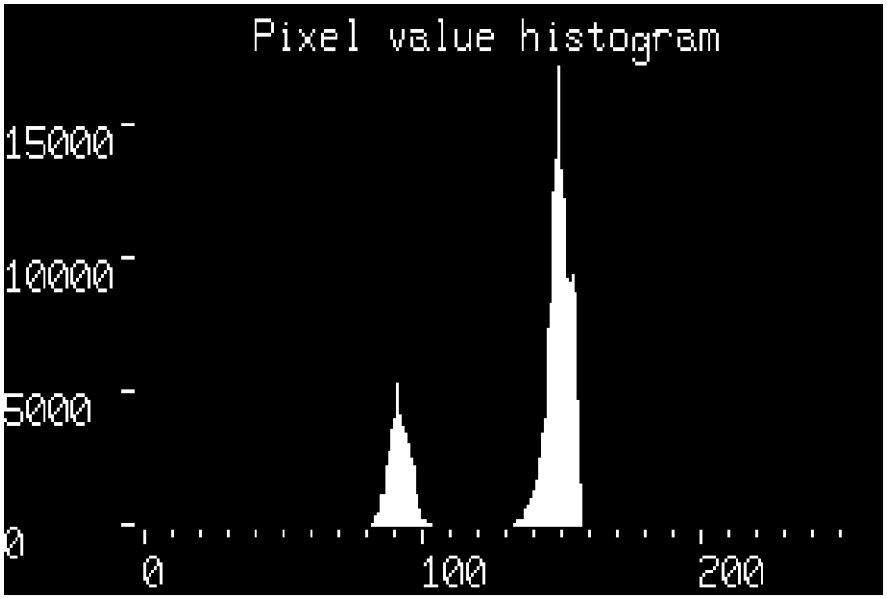
Objects & Background

- Global Thresholding
- Local/Adaptive Thresholding

Global Thresholding

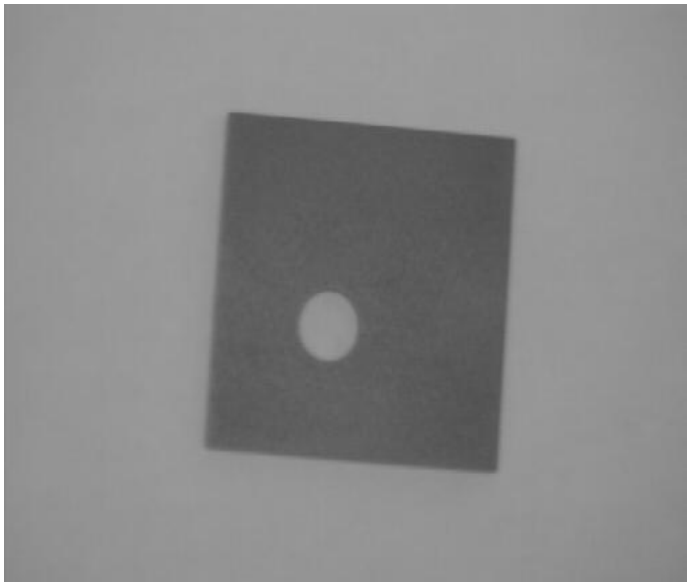
- Single threshold value for entire image
- Fixed ?
- Automatic
  - Intensity histogram

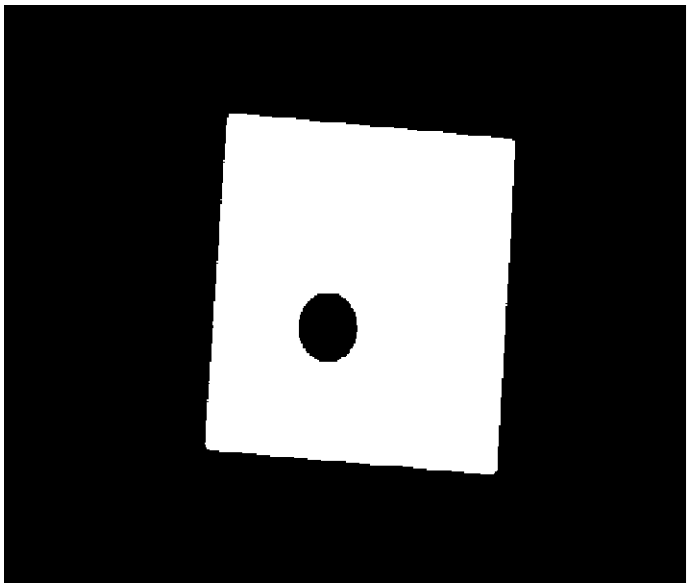




Global Thresholding

- Single threshold value for entire image
- Fixed ?
- Automatic
  - Intensity histogram

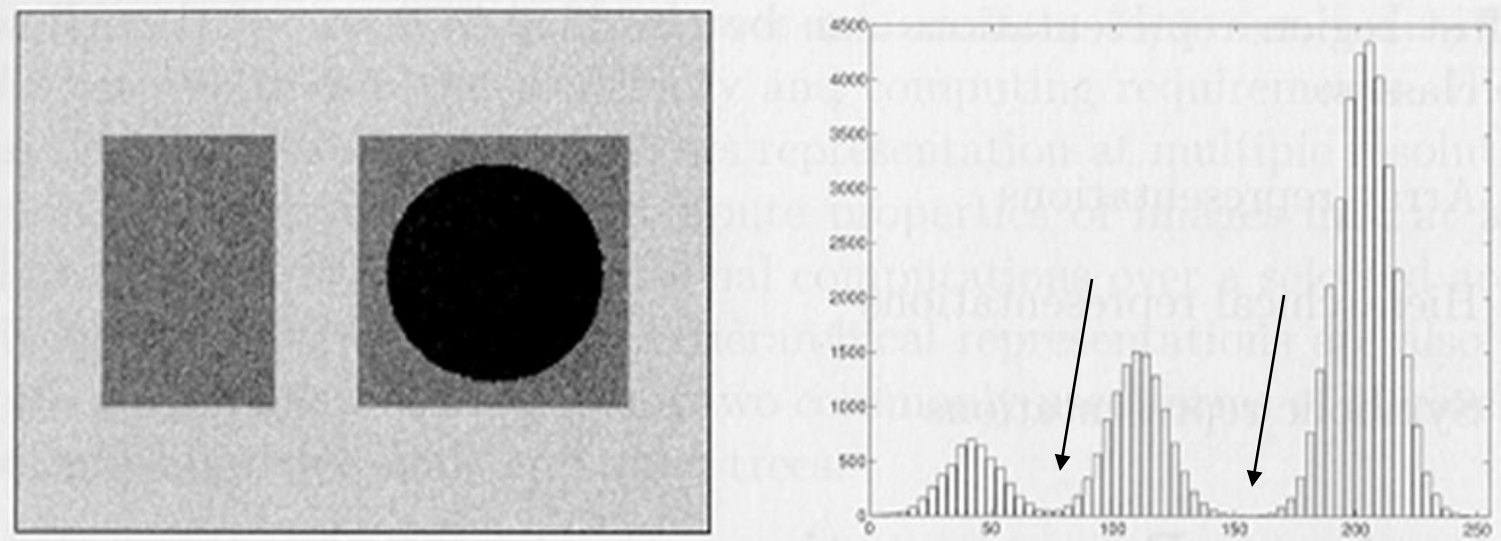




Global Thresholding

- Estimate an initial T
- Segment Image using T: Two groups of pixels G1 and G2
- Compute average gray values m1 and m2 of two groups
- Compute new threshold value T=1/2(m1+m2)
- Repeat steps 2 to 4 until: abs(T_i – T_{i-1})<epsilon

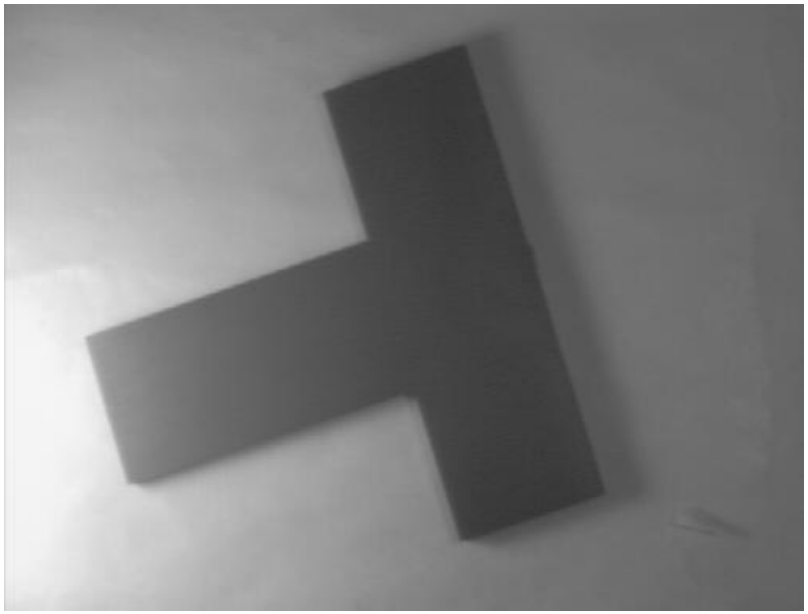
Global Thresholding

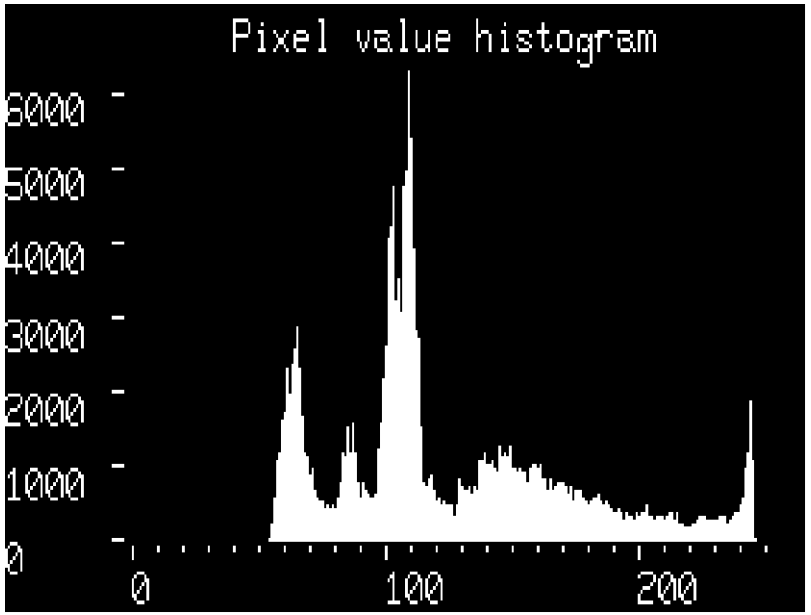


Multilevel thresholding

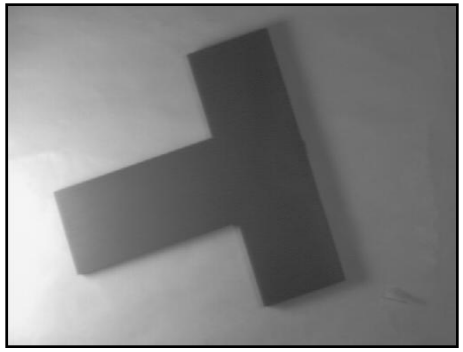
Thresholding

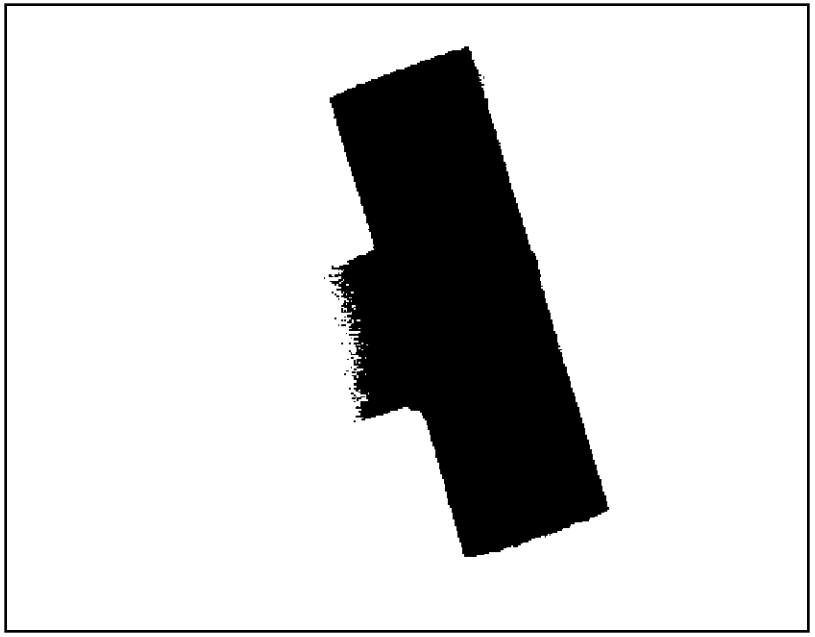
- Non-uniform illumination:

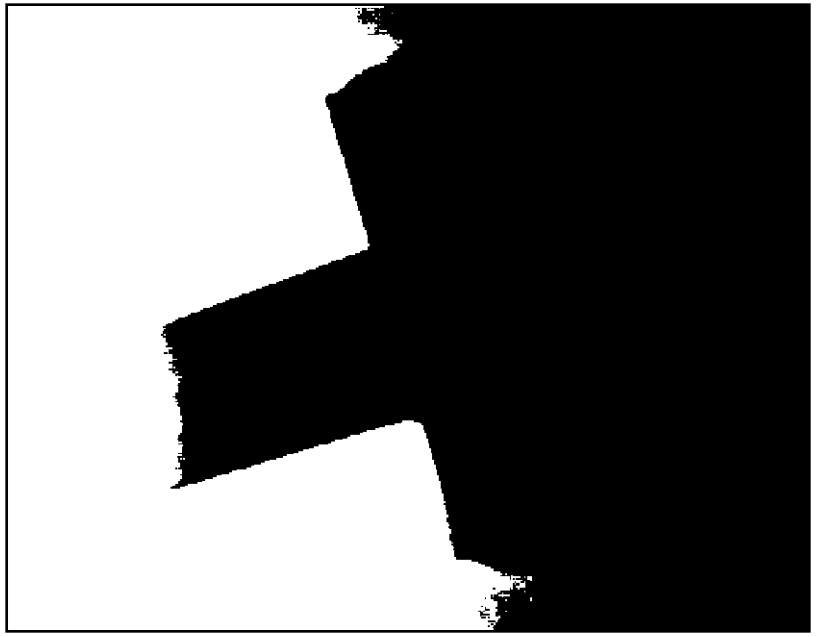




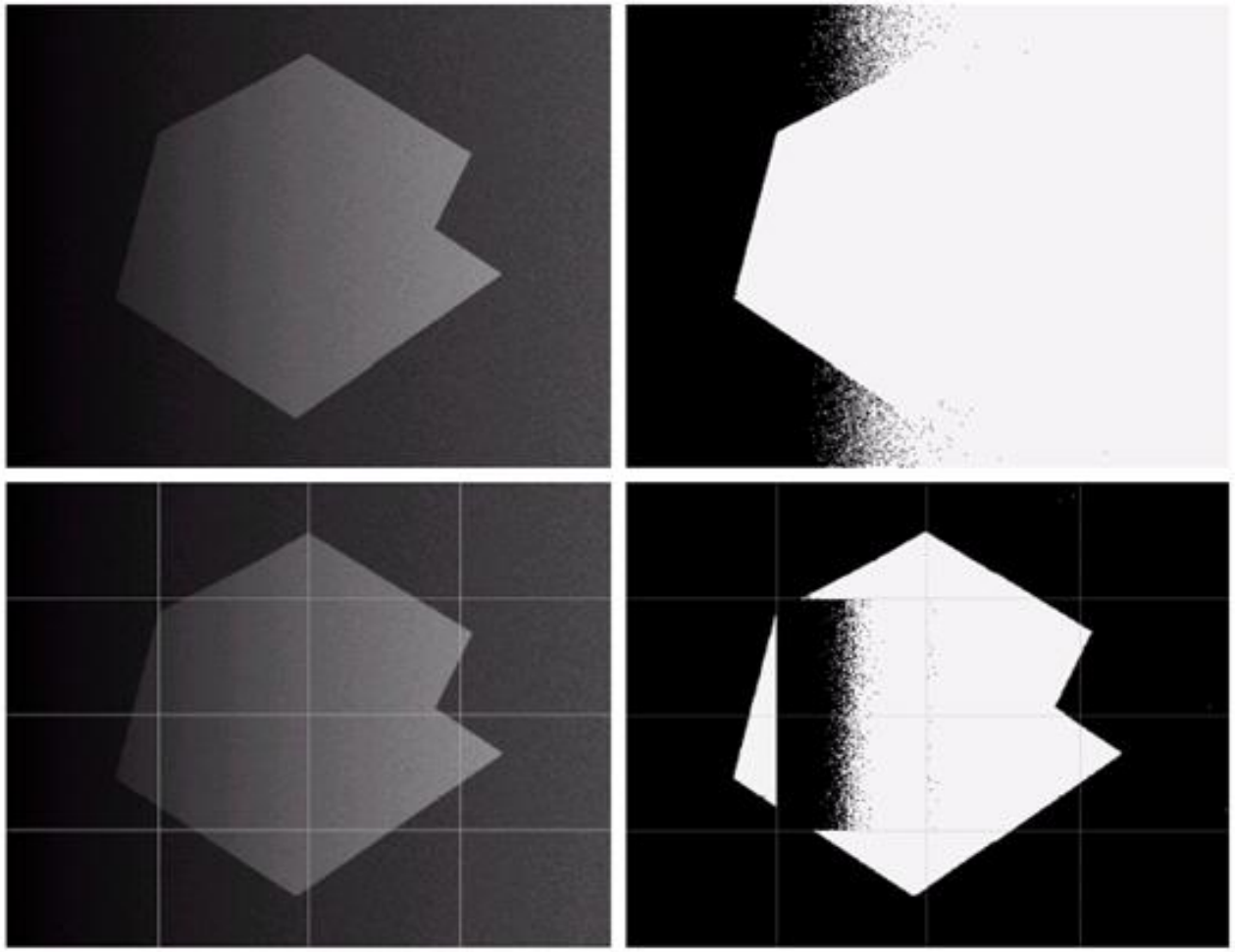
Global Thresholding







Adaptive Thresholding



Adaptive Thresholding

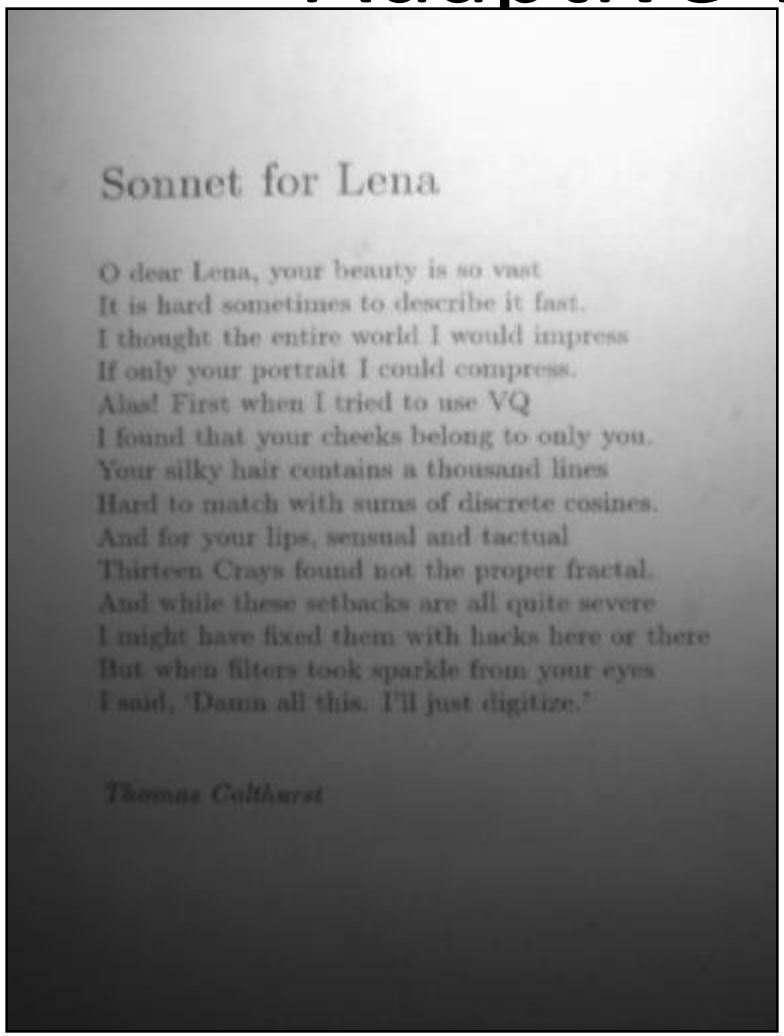
- Threshold: function of neighboring pixels

$$T = mean$$

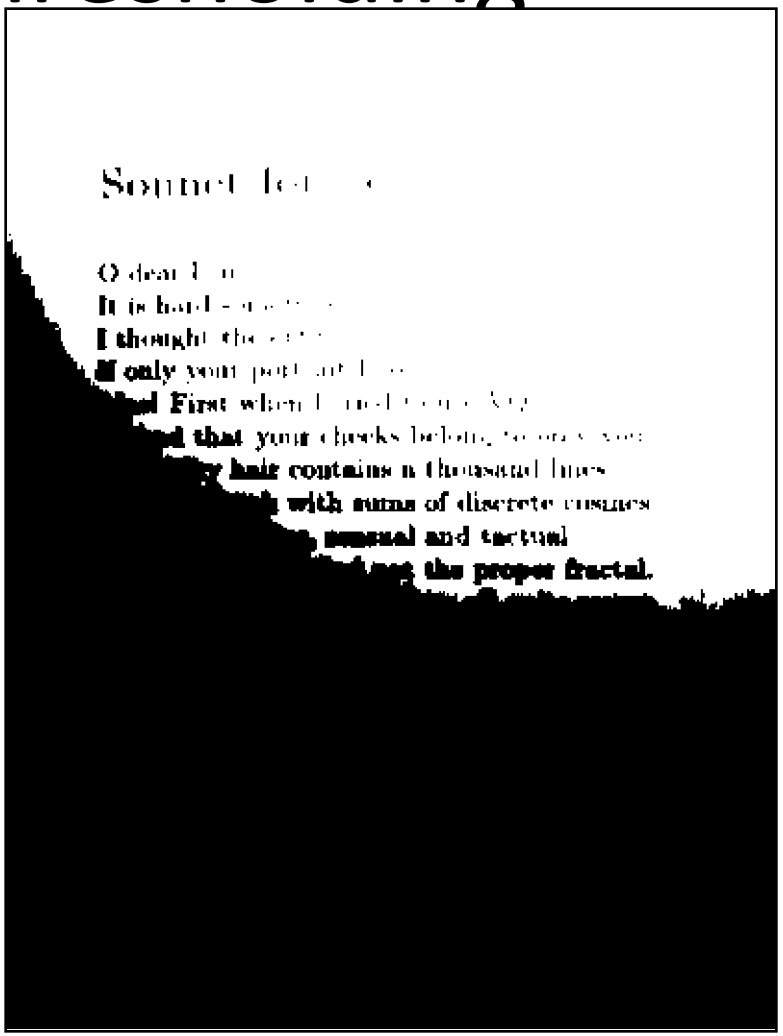
$$T = median$$

$$T = \frac{\max + \min}{2}$$

Adaptive Thresholding

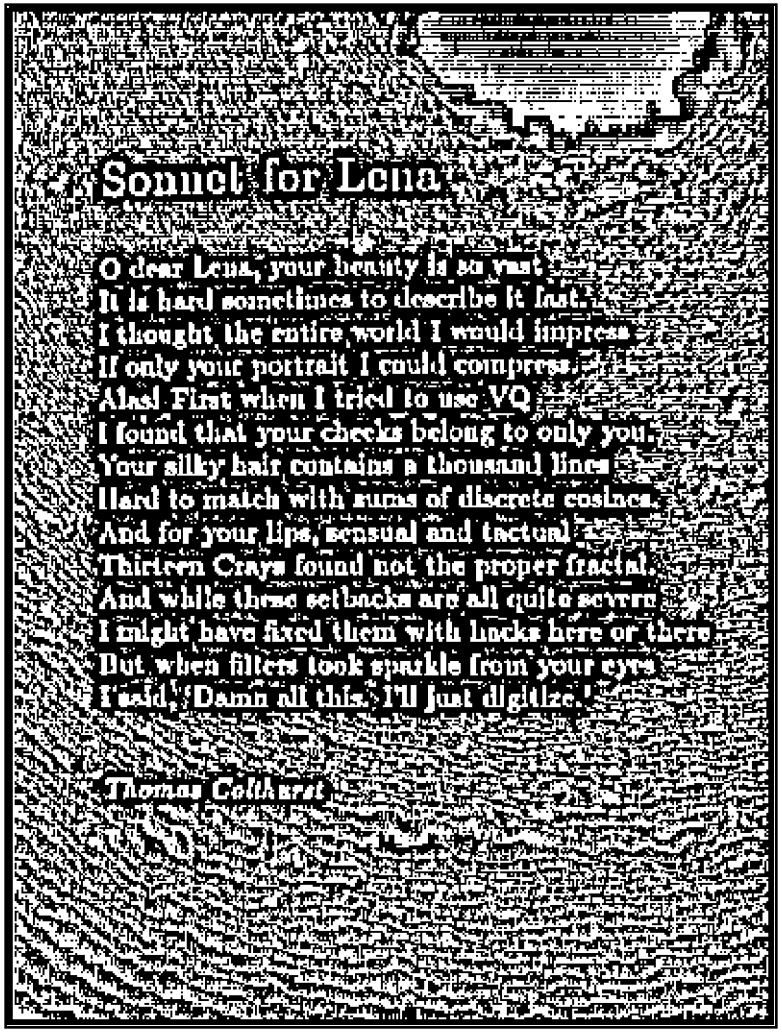


Original Image

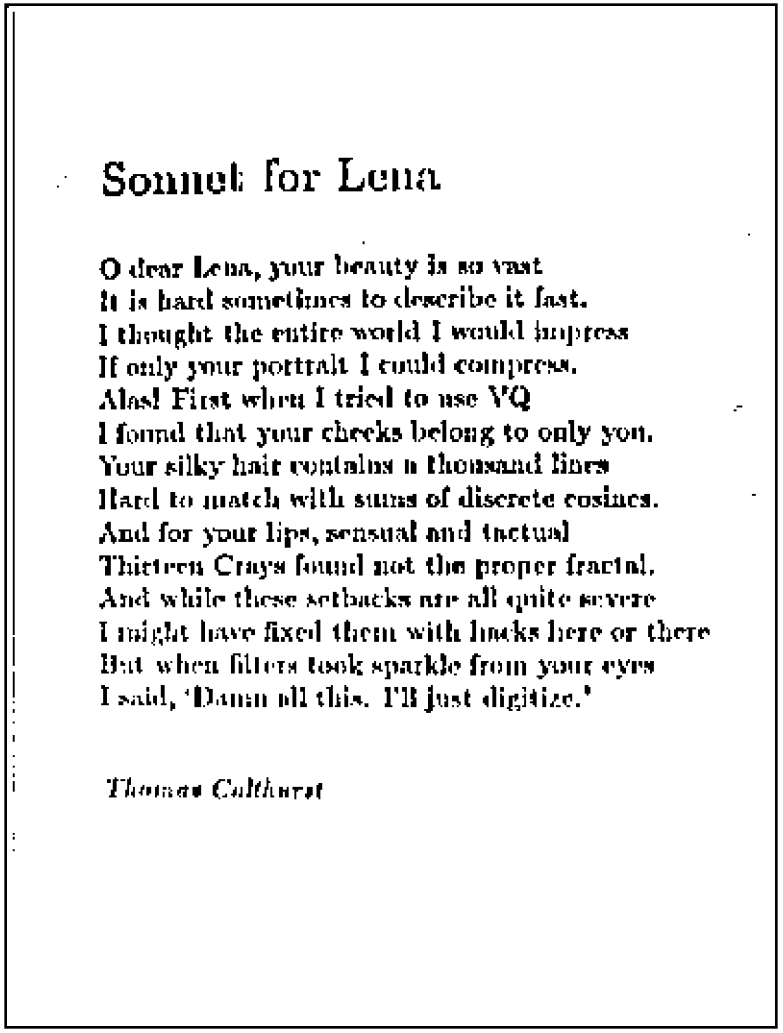


Global Thresholding

Adaptive Thresholding



T=mean, neighborhood=7x7



T=mean-Const., neighborhood=7x7

Adaptive Thresholding

- Niblack Algorithm

$$T = m + k \times s$$

$$m = \text{mean}$$

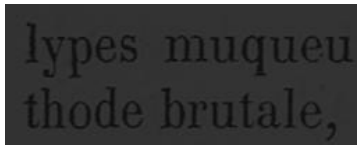
$$s = \text{standard deviations}$$

$$k = \text{ Niblack constant}$$

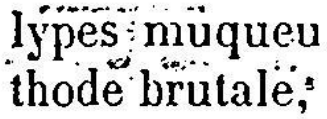
- Neighborhood size???

Document Binarization

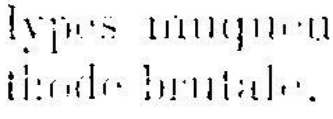
- Local Thresholding – Examples

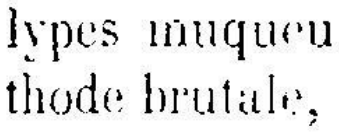


Original

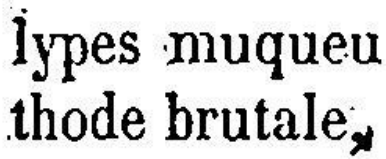


Niblack

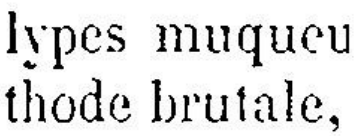

Sauvola



Wolf



Feng



NICK

Image Morphology

Introduction

- Morphology

  A branch of biology which deals with the form and structure of animals and plants

- Mathematical Morphology
  - A tool for extracting image components that are useful in the representation and description of region shapes
  - The language of mathematical morphology is Set Theory

Morphology: Quick Example

Image after segmentation

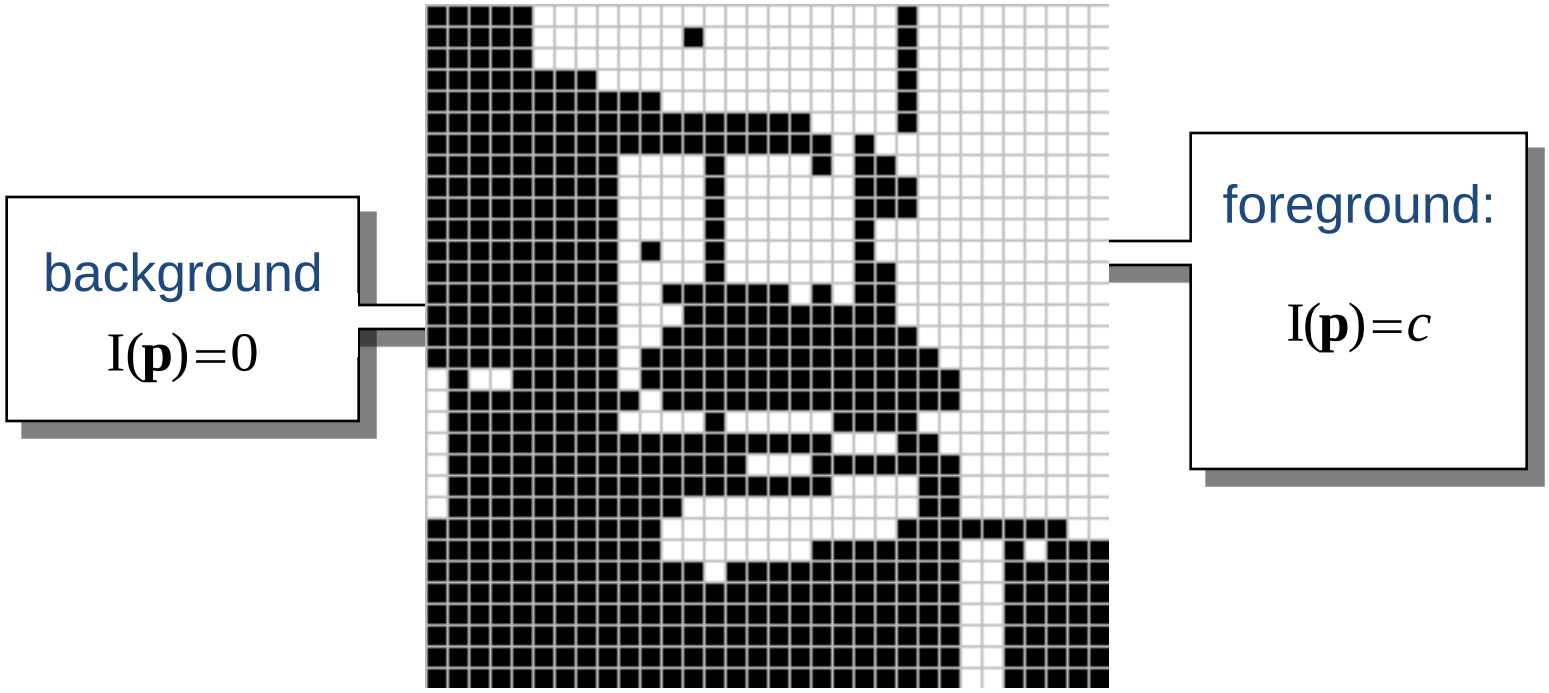
Image after segmentation and morphological processing

Introduction

Morphological image processing describes a range of image processing techniques that deal with the shape (or morphology) of objects in an image

Sets in mathematical morphology represents objects in an image. E.g. Set of all white pixels in a binary image.

Introduction



This represents a digital image. Each square is one pixel.

Set Theory

- The set space of binary image is Z^2
  - Each element of the set is a 2D vector whose coordinates are the *(x,y)* of a black (or white, depending on the convention) pixel in the image
- The set space of gray level image is Z^3
  - Each element of the set is a 3D vector: *(x,y)* and intensity level.

NOTE:

Set Theory and Logical operations are covered in:
Section 9.1, Chapter # 9, 2nd Edition DIP by Gonzalez
Section 2.6.4, Chapter # 2, 3rd Edition DIP by Gonzalez

Set Theory

- Let A be a set in Z^2. if $a = (a_1, a_2)$ is an element of A, then we write

$$a \in A$$

- If a is not an element of A, we write

$$a \notin A$$

- Set representation

$$A = \{(a_1, a_2), (a_3, a_4)\}$$

- Empty or Null set

$$A = \varnothing$$

Set Theory

- Subset: if every element of A is also an element of another set B, the A is said to be a subset of B

$$A \subseteq B$$

- Union: The set of all elements belonging either to A, B or both

$$C = A \bigcup B$$

- Intersection: The set of all elements belonging to both A and B

$$D = A \bigcap B$$

Set Theory

- Two sets A and B are said to be disjoint or mutually exclusive if they have no common element

$$A \bigcap B = \varnothing$$

- Complement: The set of elements not contained in A

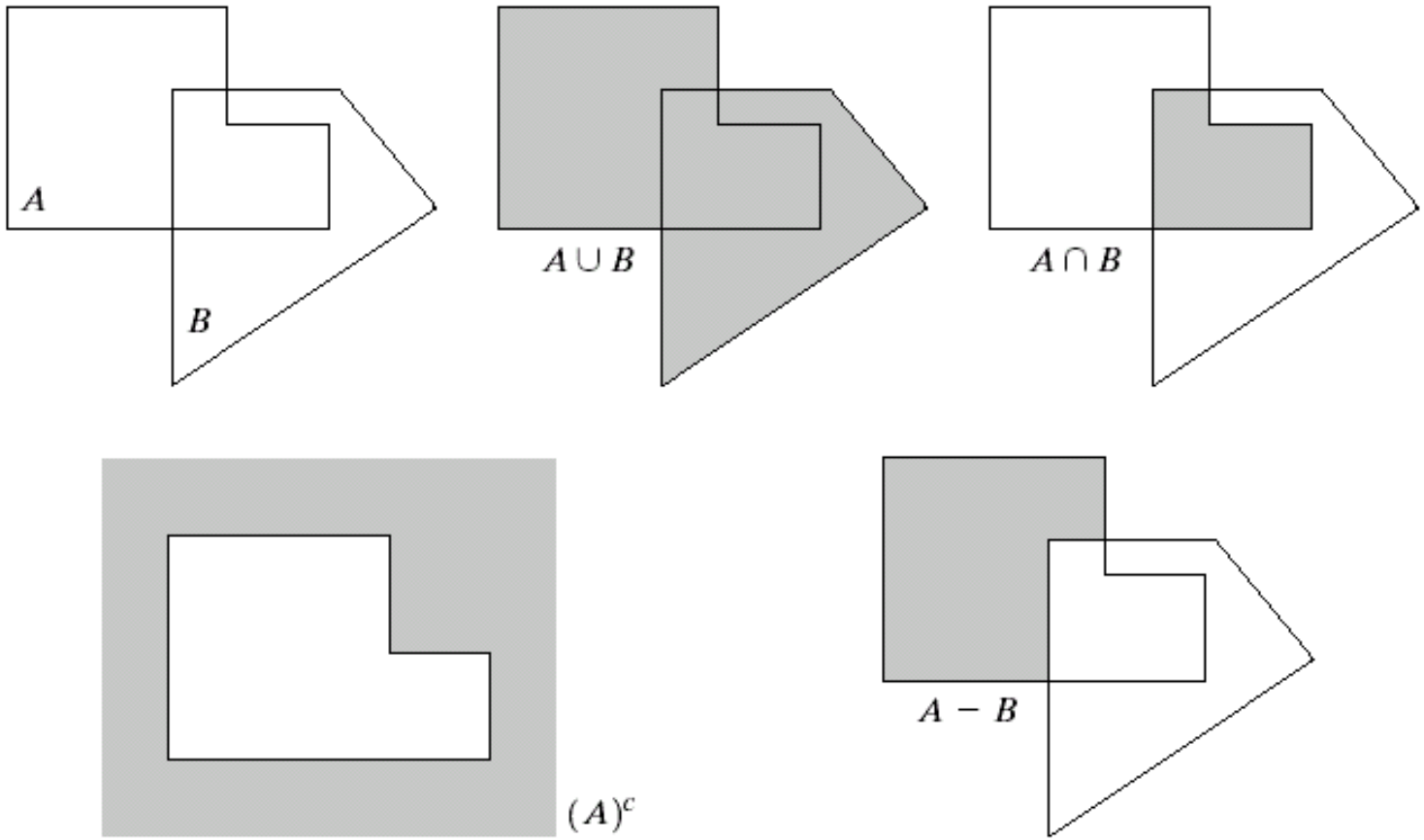
$$A^c = \{w \mid w \notin A\}$$

- Difference of two sets A and B, denoted by A – B, is defined as

$$A - B = \{w \mid w \in A, w \notin B\} = A \bigcap B^c$$

i.e. the set of elements that belong to A, but not to B

Set Theory



a b c
d e

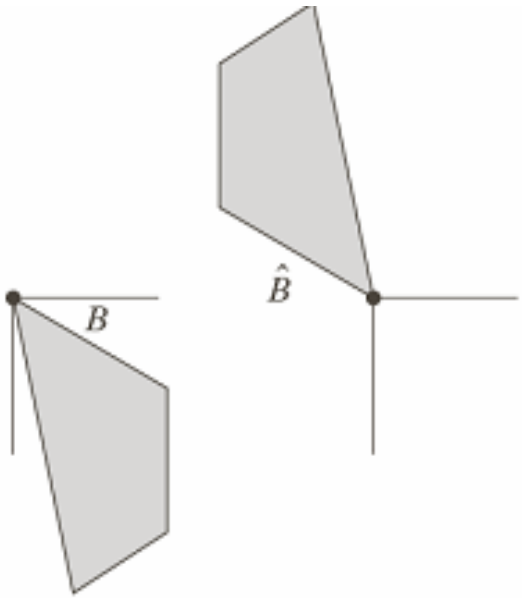
FIGURE 9.1
(a) Two sets A and B. (b) The union of A and B. (c) The intersection of A and B. (d) The complement of A. (e) The difference between A and B.

Set Theory

- Reflection of set B

$$\hat{B} = \{w \mid w = -b, \quad \text{for } b \in B\}$$

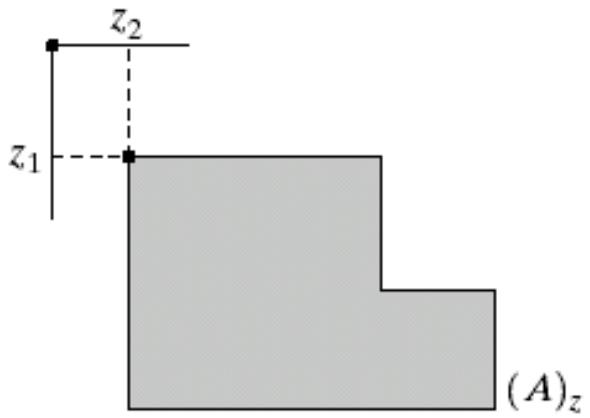
i.e. the set of element *w*, such that *w* is formed by multiplying each of two coordinates of all the elements of set *B* by -1



Set Theory

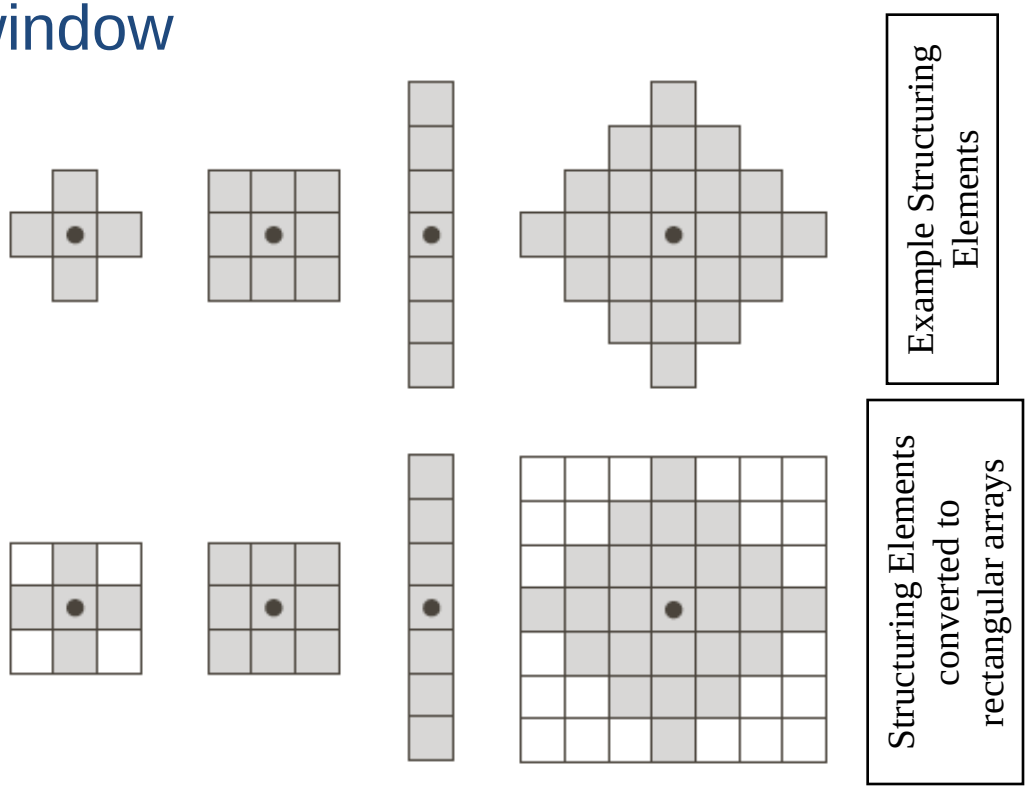
- Translation of set A by point z = (z_1,z_2), denoted $(A)_z$, is defined as

$$(A)_z = \{w \mid w = a + z, \quad \text{for } a \in A\}$$



Structuring Element

A structuring element is a small image – used as a moving window



Structuring Element

For simplicity we will use rectangular structuring elements with their origin at the middle pixel

1	1	1
1	**1**	1
1	1	1

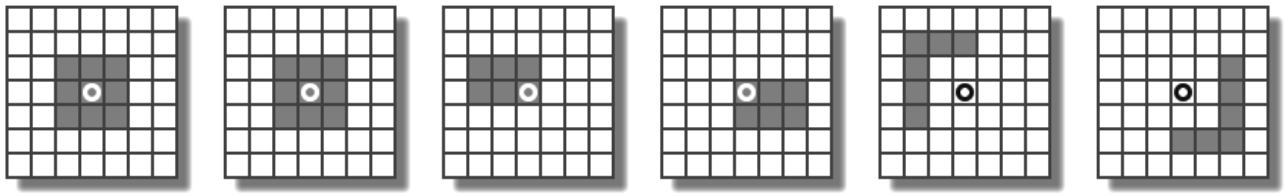
0	1	0
1	**1**	1
0	1	0

0	0	1	0	0
0	1	1	1	0
1	1	**1**	1	1
0	1	1	1	0
0	0	1	0	0

Structuring Element: Reflection

$$\hat{B}(x,y) = B(-x,-y)$$

$\hat{B}$ is B rotated by 180° around its origin.



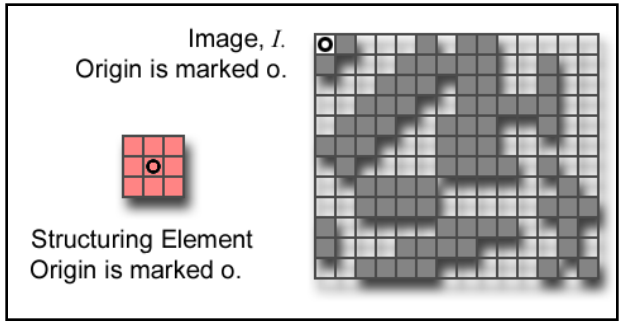
B_1 $\hat{B}_1$ B_2 $\hat{B}_2$ B_3 $\hat{B}_3$

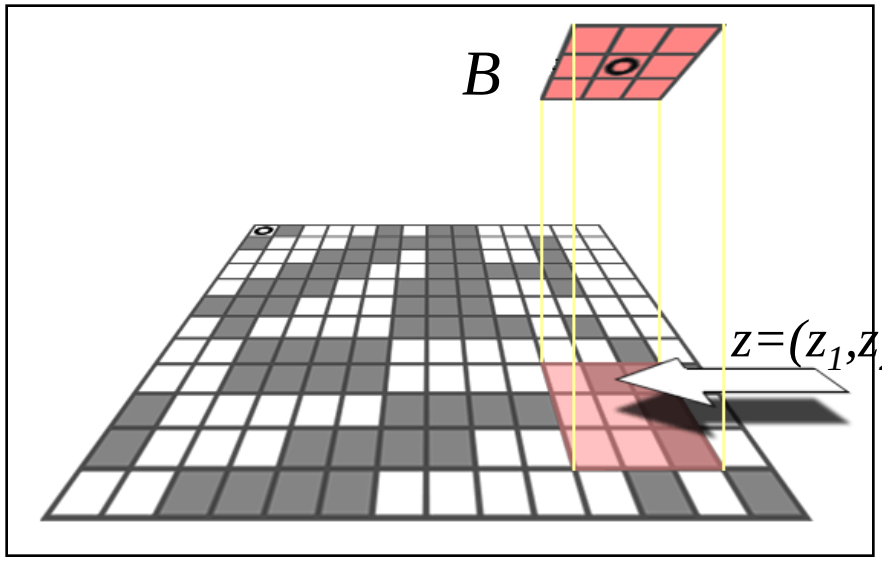
Structuring Element: Translation

Let I be an image and *B* a SE.

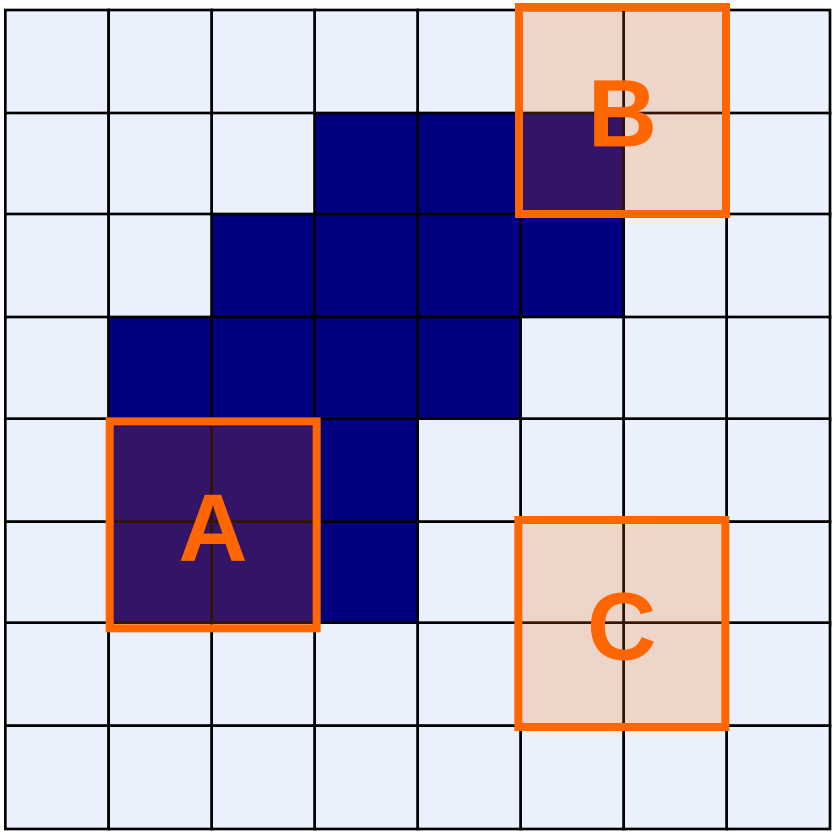
$(B)_z$ means that *B* is moved so that its origin coincides with location z in image *I*.

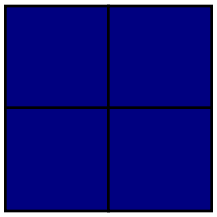
$(B)_z$ is the *translate* of *B* to location *z* in *I*.





Structuring Elements: Hits & Fits





Structuring Element

Fit: All *on pixels* in the structuring element cover *on pixels* in the image

Hit: Any *on pixel* in the structuring element covers an *on pixel* in the image

All morphological processing operations are based on these simple ideas

Fitting & Hitting

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	B	1	1	1	0	C	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	A	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0

1	1	1
1	1	1
1	1	1

Structuring Element 1

0	1	0
1	1	1
0	1	0

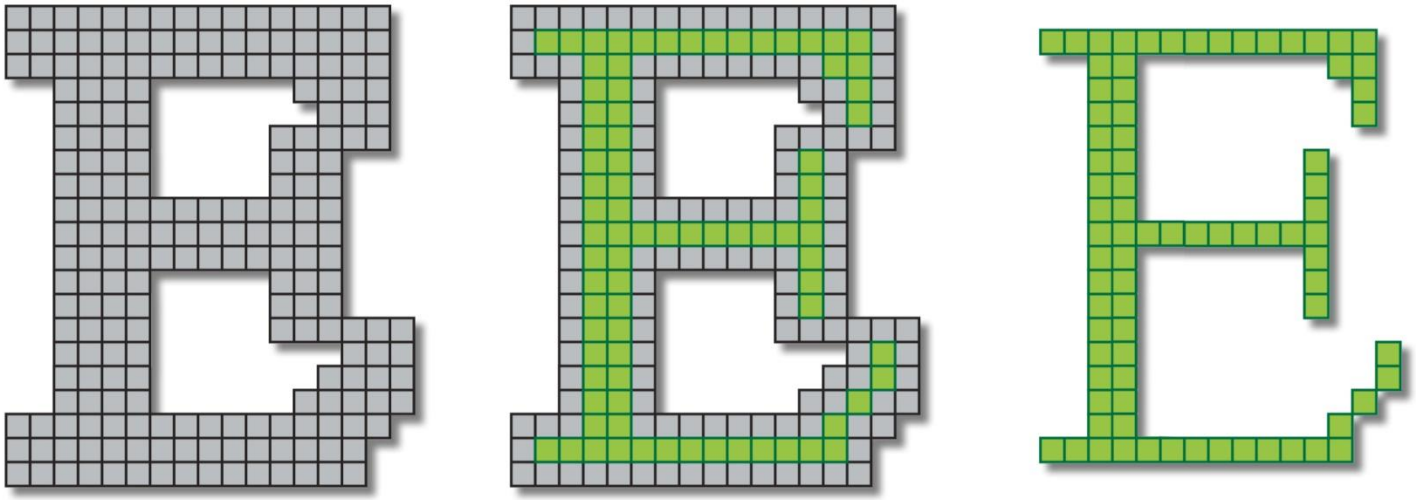
Structuring Element 2

Fundamental Operations

- Fundamentally morphological image processing is very like spatial filtering
- The structuring element is moved across every pixel in the original image to give a pixel in a new processed image
- The value of this new pixel depends on the operation performed

There are two basic morphological operations: **erosion** and **dilation**

Erosion



Erosion

Definition 1:

The erosion of two sets A and B is defined as:

$$A ! \ B = \{ z \mid (B)_z \subseteq A \}$$

i.e. The Erosion of *A* by *B* is the set of all points *z*, such that B, translated by *z*, is contained in *A*

Erosion

Definition 2:

Erosion of image *f* by structuring element *s* is given by $f \ominus s$

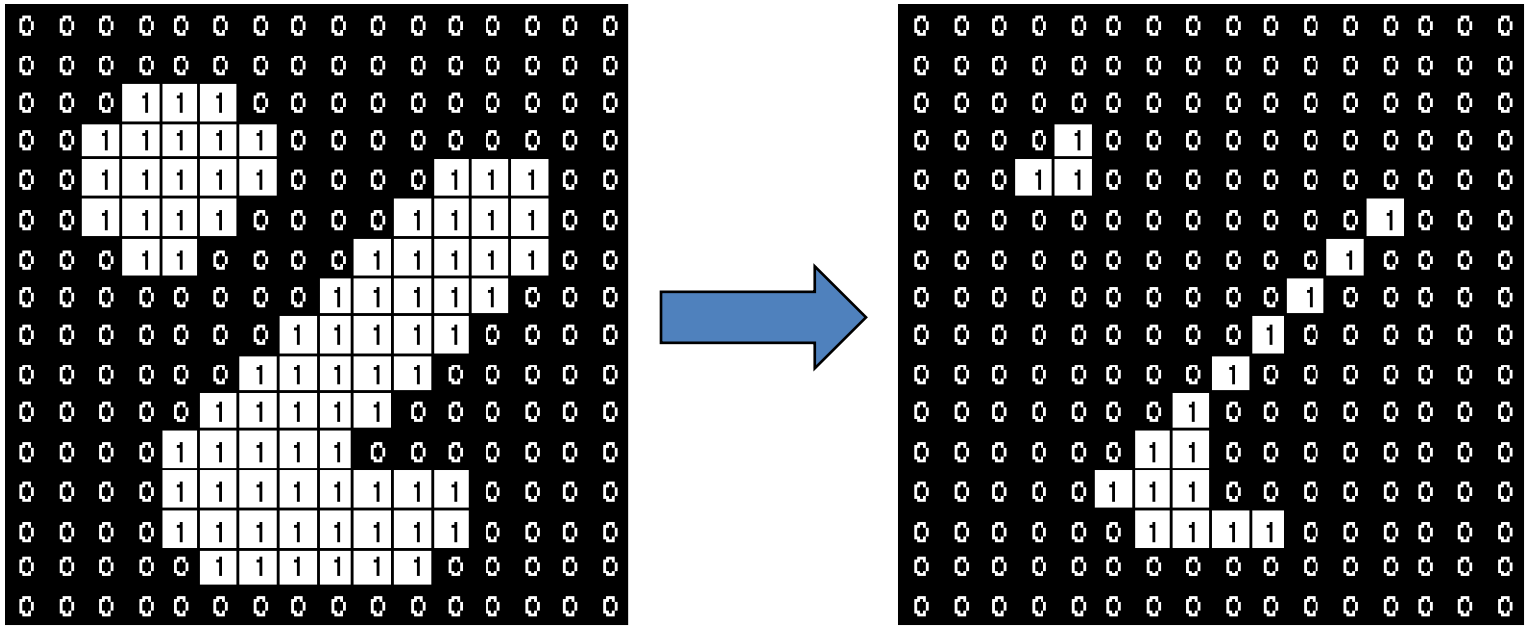
The structuring element s is positioned with its origin at *(x, y)* and the new pixel value is determined using the rule:

$$g(x,y) = \begin{cases} 1 \text{ if } s \text{ fits } f \\ 0 \text{ otherwise} \end{cases}$$

Erosion – How to compute

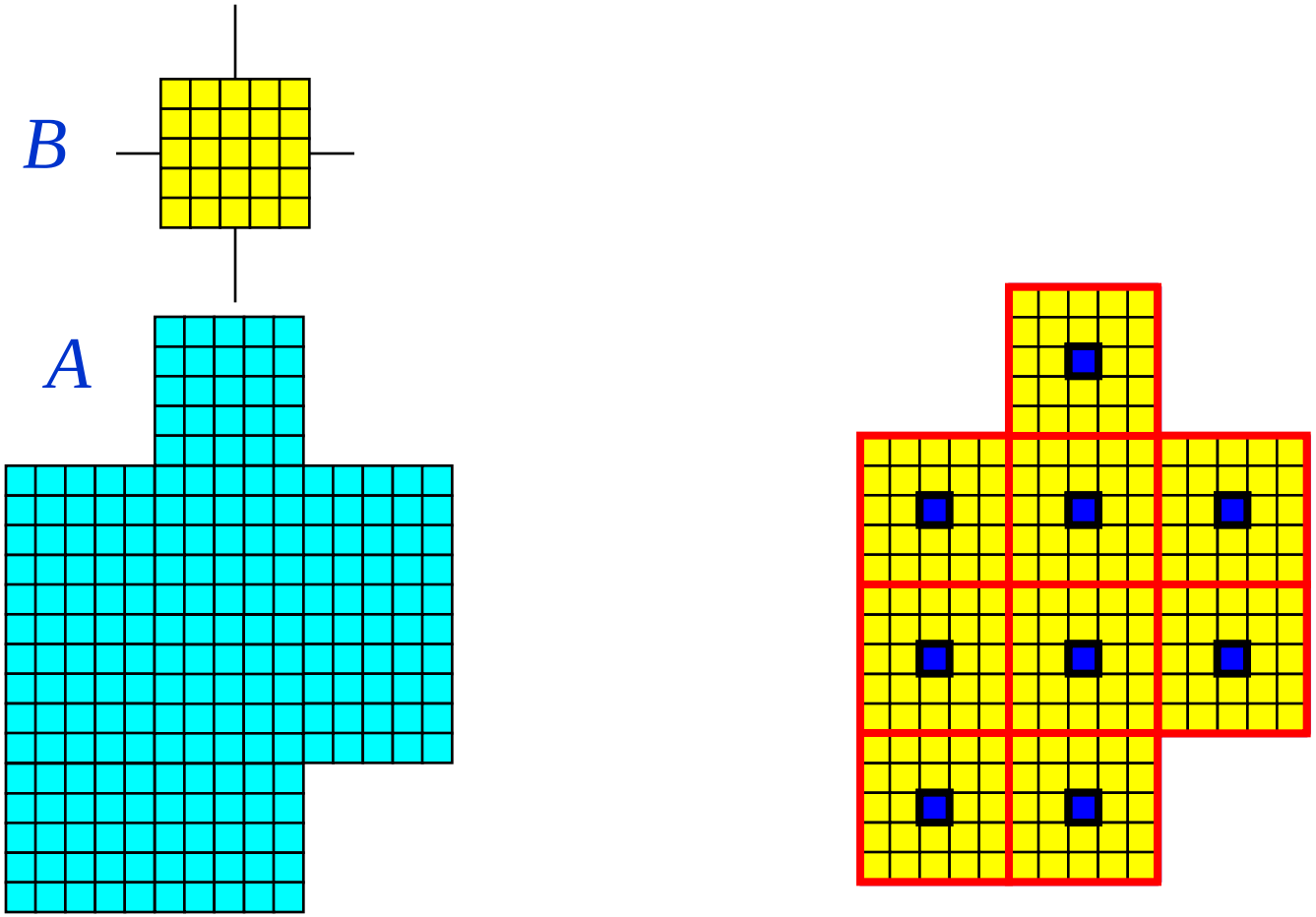
- For each foreground pixel (which we will call the *input pixel*)
  - Superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position.
  - If *for every* pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is.
  - If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value.

Erosion – How to compute

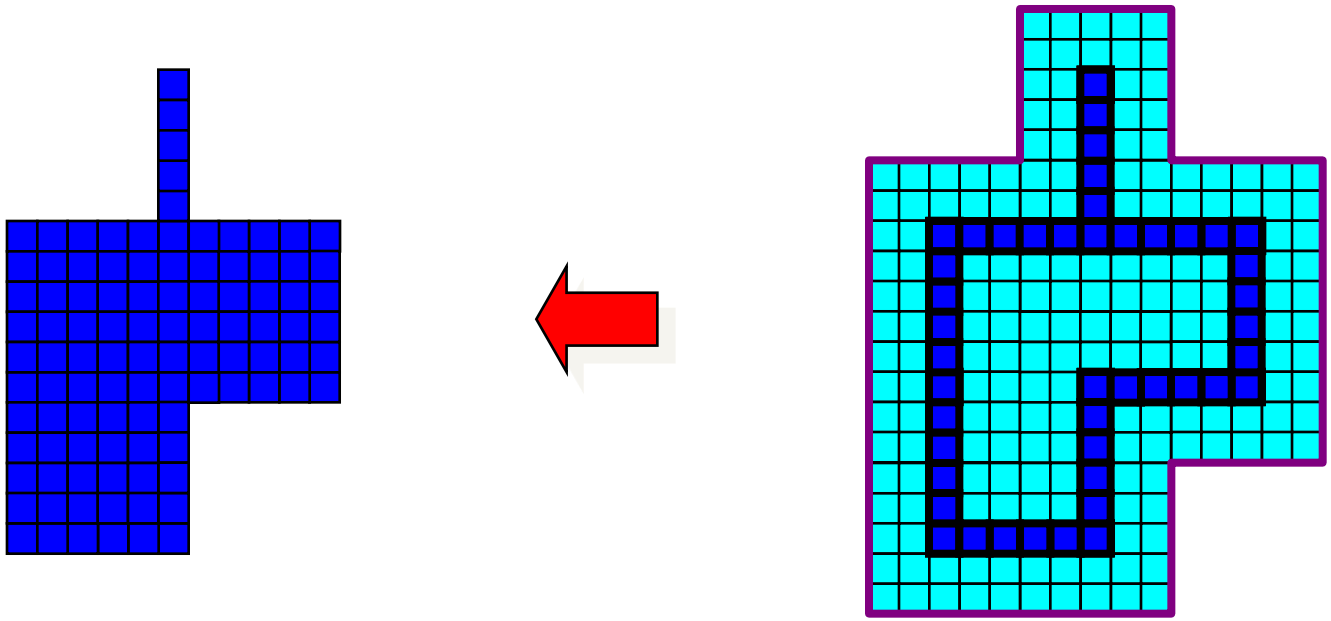


Erosion with a structuring element of size 3x3

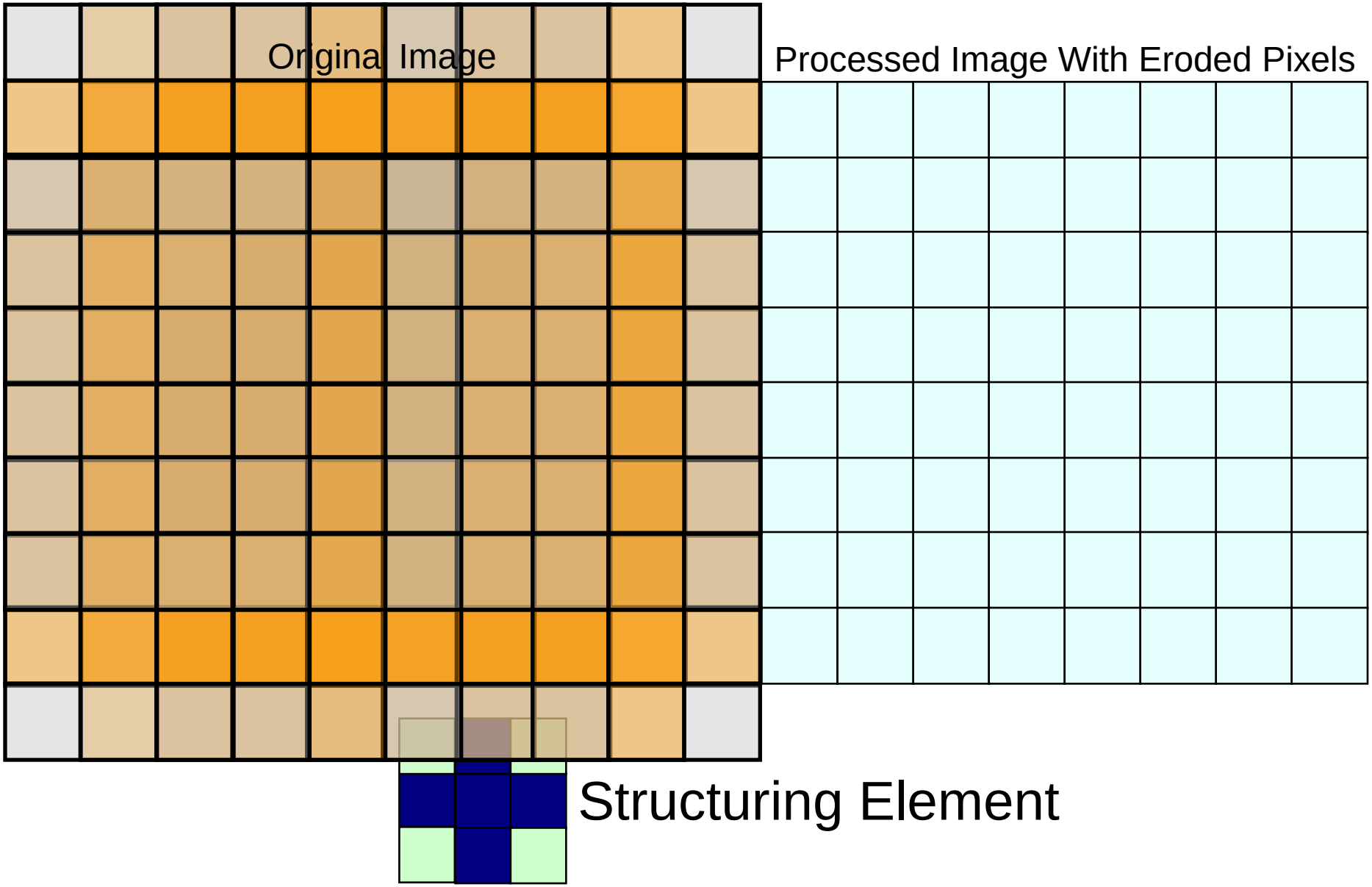
Erosion



Erosion

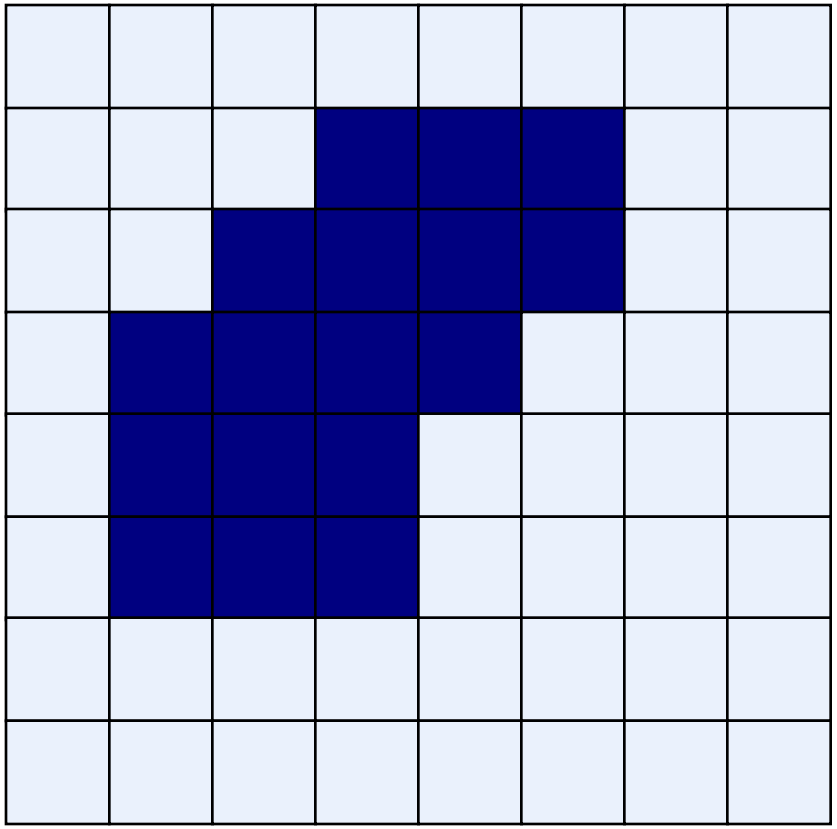


Erosion: Example

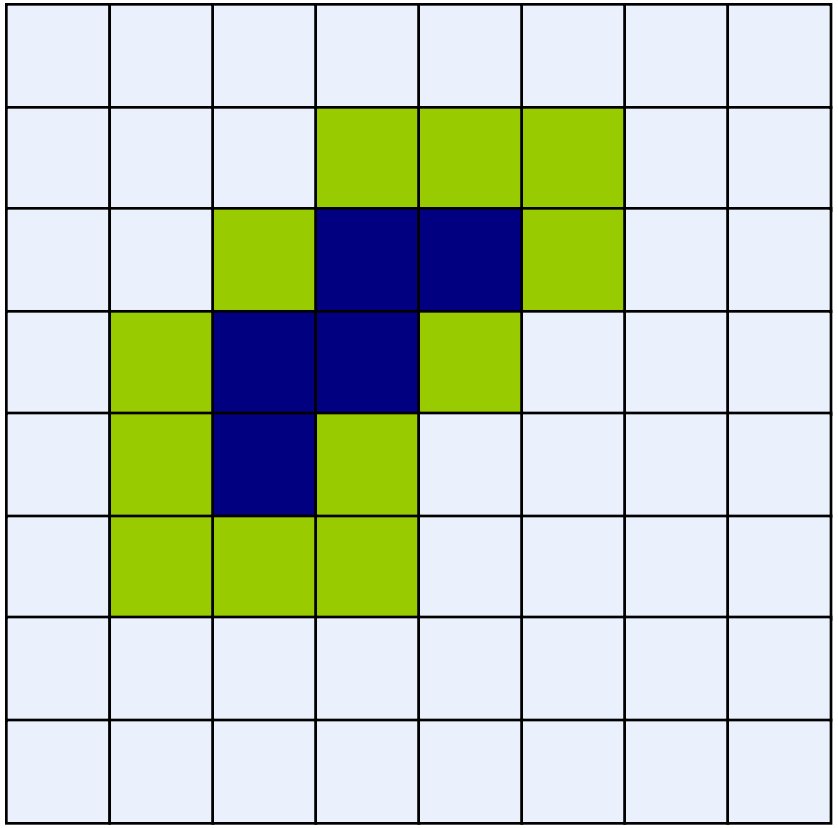


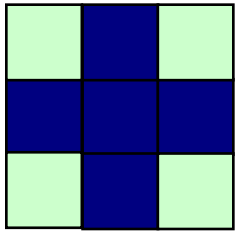
Erosion: Example

Original Image



Processed Image





Structuring Element

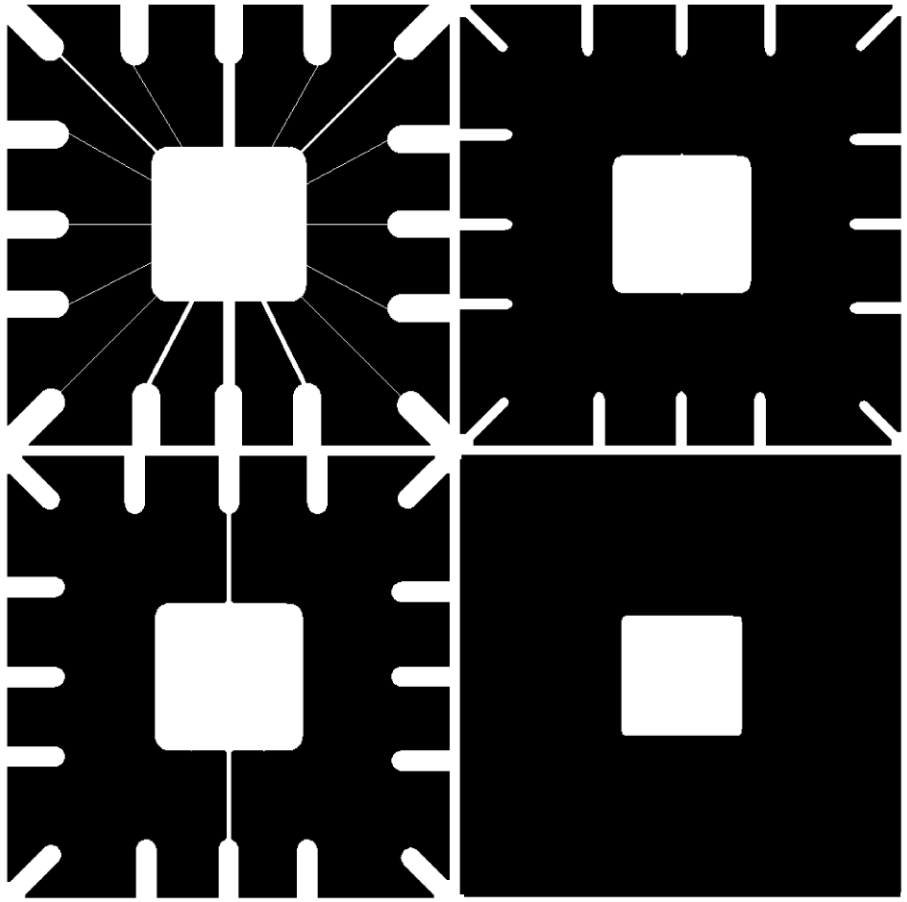
Erosion

- Effects
  - Shrinks the size of foreground (1-valued) objects
  - Smoothes object boundaries
  - Removes small objects

- Rule for Erosion

  In a binary image, if any of the pixel (in the neighborhood defined by structuring element) is 0, then output is 0

Erosion: Example 1



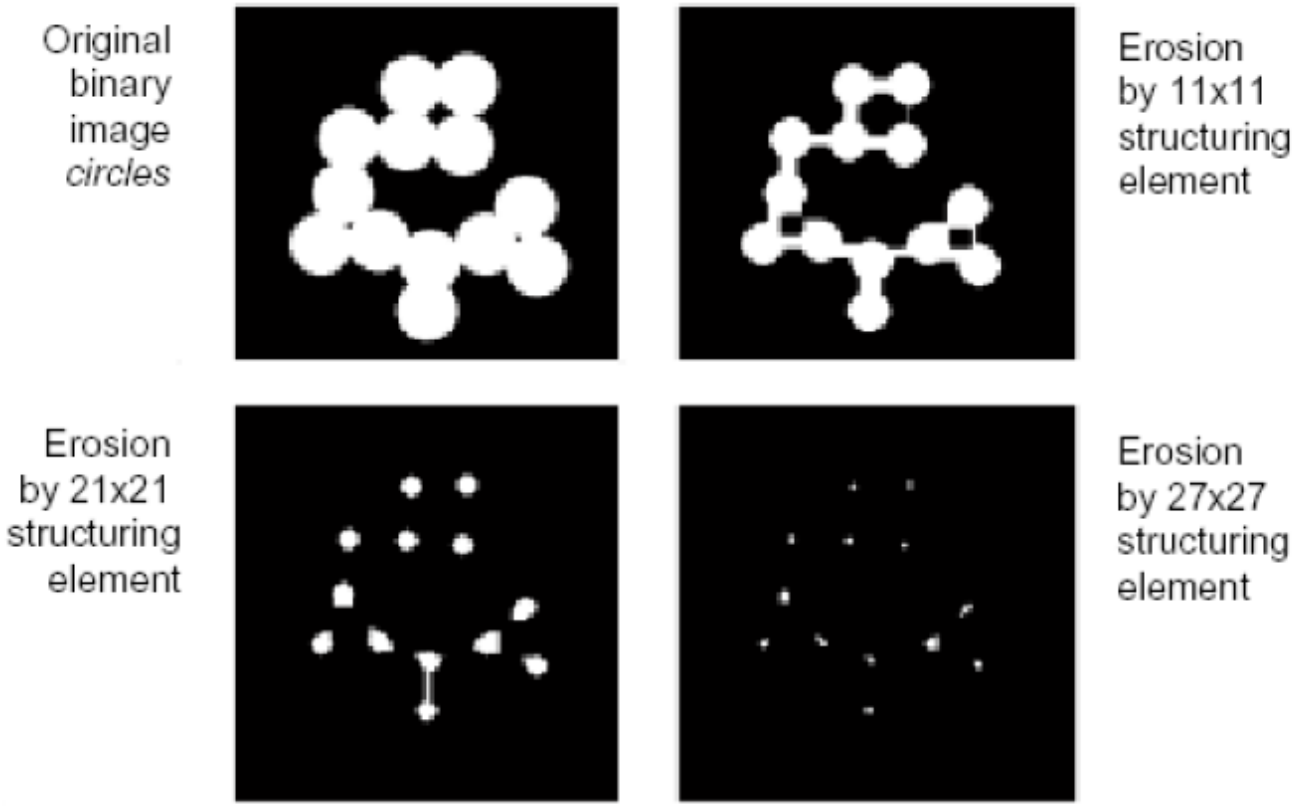
Original image

After erosion with a disc of radius 10

After erosion with a disc of radius 5

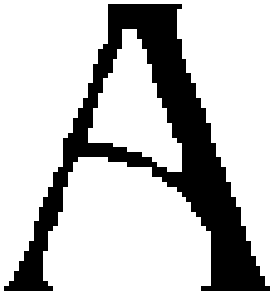
After erosion with a disc of radius 20

Erosion: Example 2

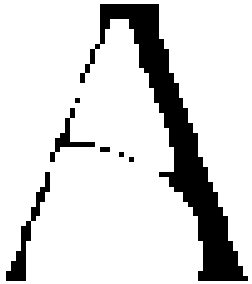


Erosion: Example 3

Original image



Erosion by 3*3 square structuring element



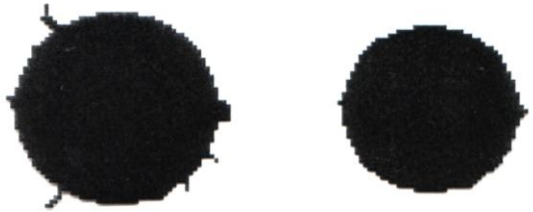
Erosion by 5*5 square structuring element

Note: In these examples a 1 refers to a black pixel!

Erosion

Erosion can split apart joined objects

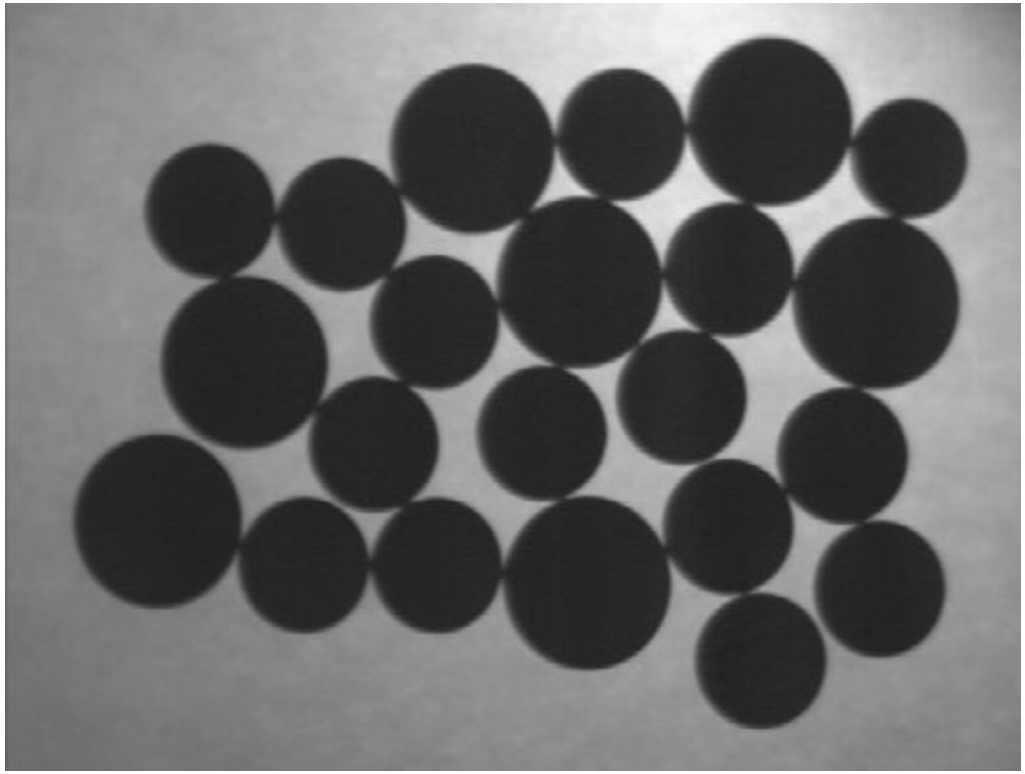
Erosion can strip away extrusions



Watch out: Erosion shrinks objects

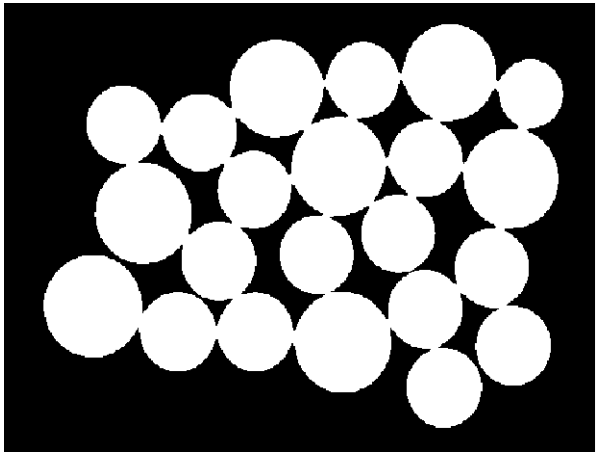
Exercise

Count the number of coins in the given image

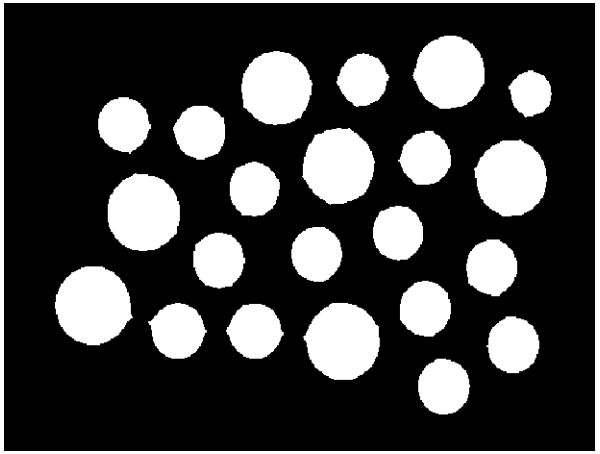


Exercise: Solution

Binarize the image



Perform Erosion



Use connected component labeling to count the number of coins

Readings from Book (3rd Edn.)

- Morphological Operations (Chapter – 9)
- Hough Transform (Chapter – 10)
- Segmentation (Chapter – 10)
- OTSU Algorithm (Chapter - 10)

Acknowledgements

- Digital Image Processing", Rafael C. Gonzalez & Richard E. Woods, Addison-Wesley, 2002
- Computer Vision for Computer Graphics, Mark Borg

Material in these slides has been taken from, the following resources