Representing digital images

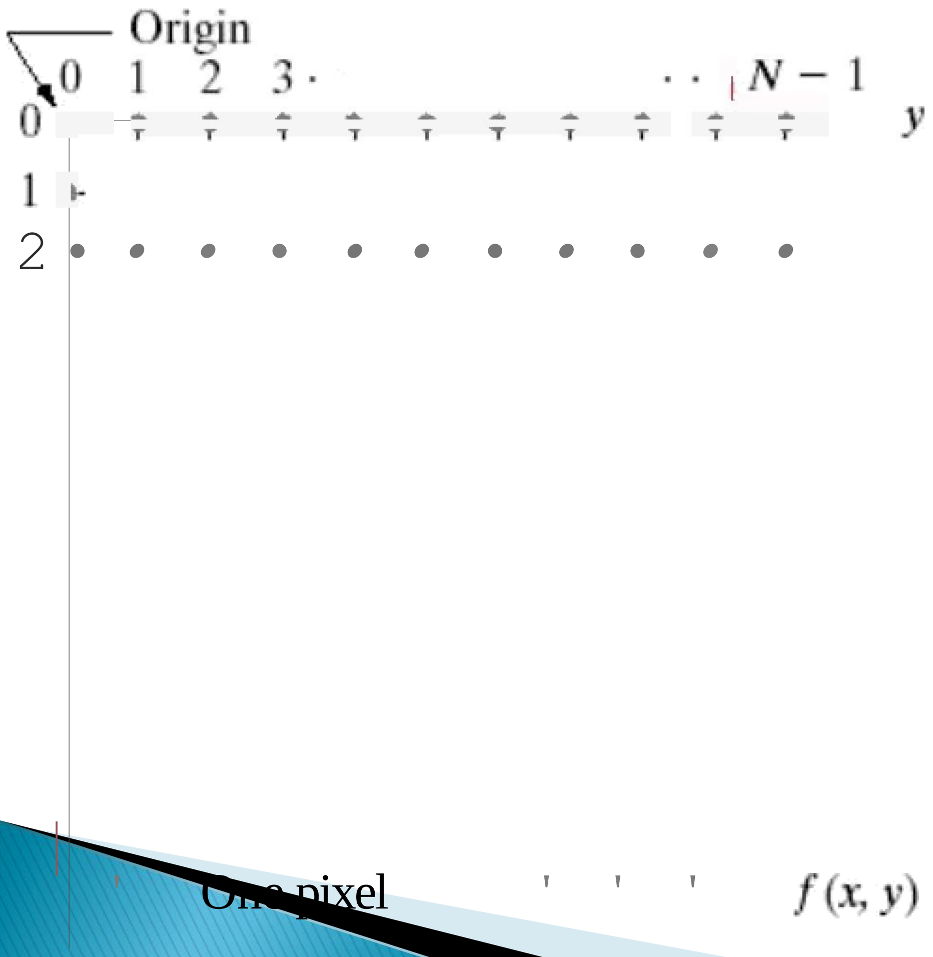


FIGURE 2.18
Coordinate convention used in this book to represent digital imagen

Representing digitai images (cont.)

- Matrix form

$$\left(\begin{array}{cccc} f(0,0) & t(0,1) & & f(0,N-1) \\ f(1,0) & i(0,1) & & f(1,N-1) \\ & & & \\ f(M—1,0) & f(M—1,1) & & f(M-1,N-1) \end{array}\right)_{MxN}$$

ibitsm to store thee g M xeN x k

gray level = 2*

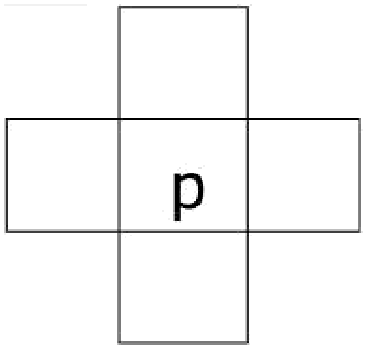
Representing digitai images (cont.)

- L = 2^k gray levels, gray scales [0,...,L-1]
- The dynamic range of an image
  - [min(image)   max(image)]
  - If the dynamic range of an image spans a significant portion of the gray scale -> high contrast
  - Otherwise, low dynamic range results in a dull, washed out gray look

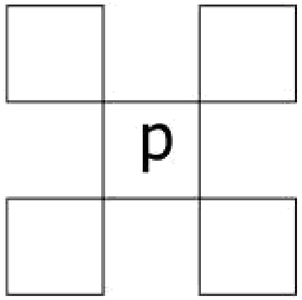
Relationships Between Pixels

Neighbors a pixel

- 4-neighbors of p: $N_4(p)$



- Diagonal neighbors: $N_D(p)$



- 8-neighbors = 4-neighbors-rdiagonal neighbors : $N_8(p)$

Adjacency, connectivity', regions, and boundaries

- Connectivity of pixels
  - They are neighbors
  - Their gray levels satisfy a specified criterion of similarity
  - Concept about regions and boundaries
- Adjacency
  - 4-adjacency:p and q with intensity from V and q is in N_{4}fp\
  - 8-adjacency: p and q with intensity from V and q is in N_{8}fp)

Connectivit:y and adjacency (cont.)

- m-adjacency(mixed adjacency): p and q having intensity from V and
  - q is in N,(p), or
  - q is in $N_{o\ e}$) and N,(p) nN_4(q) has no pixels whose values are from V

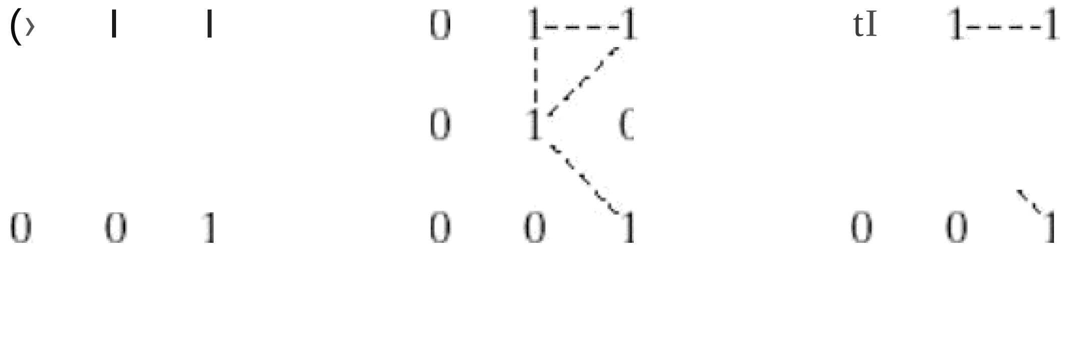


FIGURE 2.26 (a) Arrangement of pixels; (b) pixels that are 8-adjacent (shown dashed) to the center pixel; (c) *m*-adjacency.

Path

- A path from p: (x,y) to : (s,t) is a sequence of pixels:

  (x,y), (xc.ve).(xc.y). , , (xc .ve).‹s,t)

  consecutive pixels are adjacency

- Length = n
- It's a k-path if it is 4-, 8-, and m-adjacency

Growth of definitions

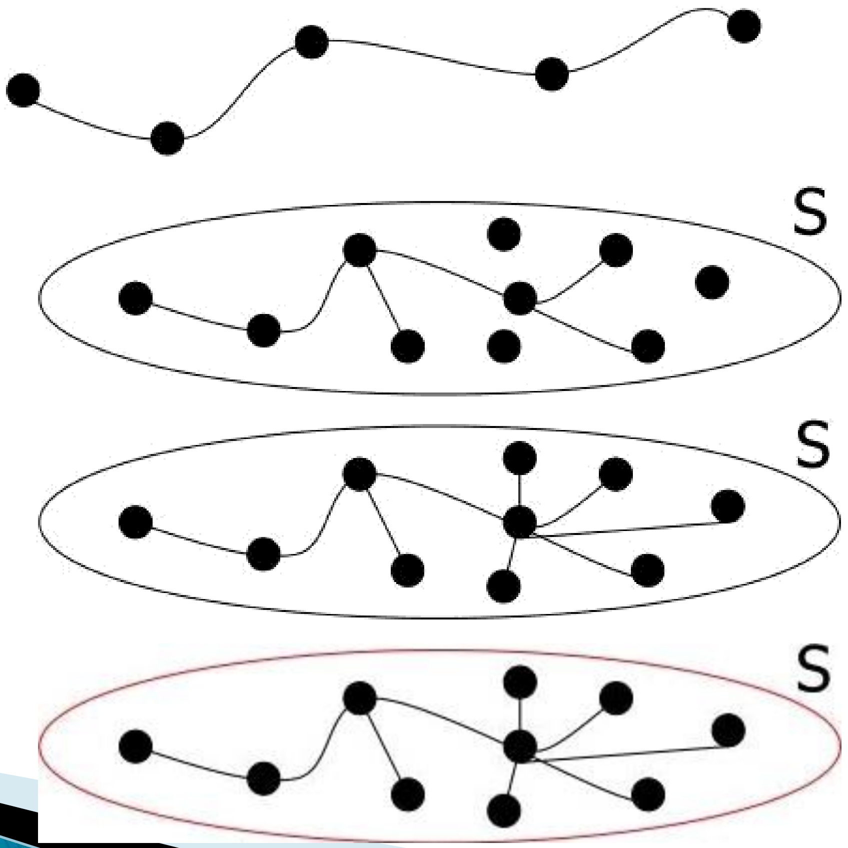
adjacency e e

path

connected component

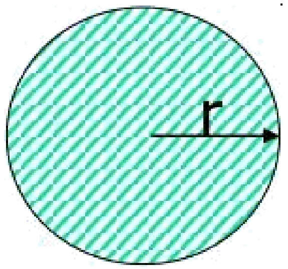
connected set (region)

boundary



Distance measure

- p: (x,y), : (s,t)
- Euclidean distance
  - $D_e(p,q)=[(x-s)^2+(y-t)^2]^{1'2}$
- D_4 distance
  - › D (p,q)'|x-s|+|y-t|
- Of distance
  - Dg(p,q)=max(|x-s|,|y-t|



```
    2
  2 1 2
2 1 0 1 2
  2 1 2
    2
```

```
2 1 0 1 2
```

Aspect ratio

- In image processing, aspect ratio refers to the proportional relationship between the width and height of an image.
- It is an important parameter that affects the visual appearance and suitability of an image for different applications, such as printing, display, and analysis.
- To calculate the aspect ratio of an image using OpenCV, we use method cv2.imread() function to read the image and shape attribute of an image array t ontain its dimension.

- import cv2 # Load the image
- img = cv2.imread('image.jpg')

  # Get the dimensions of the image
- height, width, channels = img.shape

  #Calculate the aspect ratio
- aspect_ratio = width / height

  # Display the aspect ratio
- print('Aspect ratio:', aspect_ratio)

Importance of image manipulation and enhancement

- **1. Improving image quality**: to improve the quality of an image by adjusting factors such as brightness, contrast, and color balance.
- **2. Removing unwanted elements:** It can help to improve the composition of an image and make it more visually appealing.
- **3. Highlighting specific features:** It help to highlight specific features in an image, such as the details in a landscape or the texture of a product.
- **4. Creating a consistent visual identity**: It help to create a consistent visual identity across a company's marketing materials or social media channels.
- **5. Meeting specific requirements**: resolution or color profile needed for printing or online use.

Definition and importance of removing noise

- Removing noise is the process of reducing or eliminating unwanted or irrelevant information from an image.
- Noise is any random variation in an image that is not part of the underlying structure or signal, and can be caused by factors such as low light conditions, camera sensor limitations, or interference in data transmission.
- Removing noise is important in image processing because it can improve the clarity and
- quality of the image, making it easier to analyze and interpret.

common techniques for removing noise include:

- 1. Gaussian filtering: Gaussian filtering is a type of low-pass filtering that can be used to remove high-frequency noise from an image. It works by convolving the image with a Gaussian kernel that reduces the contribution of high-frequency components.
- **cv2.GaussianBlur()** function from the OpenCV library to apply a Gaussian filter to an image.
- The function takes the input image and the standard deviation of the Gaussian kernel as parameters.

```
filtered = cv2.GaussianBlur(img, (5, 5), 0)
```

- 2. Median filtering: Median filtering is a non-linear filtering method that can be used to remove salt-and-pepper noise from an image. It works by replacing each pixel in the image with the median value of the neighboring pixels.

- **cv2.medianBlur()** function from the OpenCV library to apply a median filter to an image. The function takes the input image and the size
- of the median filter as parameters.

- **filtered = cv2.medianBlur(img, 5)**

- 3. Wavelet denoising: Wavelet denoising is a technique that uses a wavelet transform to decompose an image into multiple frequency bands, and then removes noise from each band separately. It can be used to remove both high-frequency noise and low-frequency noise from an image.

- **pywt** library to perform wavelet denoising. The **pywt.wavedec2()** function can be used to perform the wavelet decomposition, and the
- **pywt.waverec2()** function can be used to reconstruct the image after denoising.

- 4. Deep learning-based denoising: Deep learning-based denoising is a more recent technique that uses deep neural networks to learn how to remove noise from images. It involves training a neural network on a large dataset of noisy and clean images, and then using the network to remove noise from new images
- **TensorFlow or PyTorch** to implement deep learning-based denoising.
- model = tf.keras.models.load_model ('denoising_model.h5')
- denoised = model.predict(img)