Mathematical Foundations for Image Processing

Mathematical Tools Used in Digital Image Processing

Mathematical modelling of an image

- We denote images by two-dimensional functions of the form $f(x, y)$.
- The value of f at coordinates (x, y) is a scalar quantity and it is nonnegative and finite.
- Function $f(x, y)$ is characterized by two components:
  1. the amount of source illumination incident on the object being viewed
  2. the amount of illumination reflected by the objects in the scene
- These are called the **illumination** and **reflectance** components.
- They are denoted by $i(x, y)$ and $r(x, y)$.

- The two functions combine as a product to form $f(x, y)$

$$\therefore \boxed{f(x, y) = i(x, y) \cdot r(x, y)}$$

where

$$0 \leq i(x, y) \leq \infty$$

and

$$0 \leq r(x, y) \leq 1$$

- Reflectance is bounded by 0 (total absorption) and 1 (total reflectance).
- $i(x, y)$ is determined by the illumination source.
- $r(x, y)$ is determined by the characteristics of the imaged objects.

- Following are typical values of illumination $i(x, y)$:
  - On a clear day, the sun may produce in excess of 90,000 lum/m^2 of illumination on the surface of the earth.
  - This value decreases to less than 10,000 lum/m^2 on a cloudy day.
  - On a clear evening, a full moon yields about 0.1 lum/m^2 of illumination.
  - The typical illumination level in a commercial office is about 1,000 lum/m^2.
- Similarly, the following are typical values of reflectance $r(x, y)$:
  - 0.01 for black velvet
  - 0.65 for stainless steel
  - 0.80 for flat-white wall paint
  - 0.90 for silver-plated metal
  - 0.93 for snow

- Let the intensity (gray level) of a monochrome image at any coordinates (x, y) be denoted by $\boxed{l = f(x, y)}$.
- It lies in the range $\boxed{L_{min} \leq l \leq L_{max}}$.
- $\boxed{L_{min} = i_{min} r_{min}}$ and it is nonnegative
- $\boxed{L_{max} = i_{max} r_{max}}$ and it is finite.
- In the absence of additional illumination, typical indoor values are $L_{min} \approx 10\ lum/m^2$ and $L_{max} \approx 1000\ lum/m^2$.
- The interval $[L_{min}, L_{max}]$ is called the intensity (or gray) scale.
- In practice, this interval is scaled to $[0, 1]$, or $[0, C]$.
- $l = 0$ is considered black and $l = 1$ (or C) is considered white on the scale.
- All intermediate values are shades of gray varying from black to white.

Image Sampling and Quantization

- To create a digital image, we need to convert the continuous sensed data into a digital format.
- This requires two processes: **sampling** and **quantization**.
- An image may be continuous with respect to the x and y coordinates, and also in amplitude.
- To digitize it, we have to sample the function in both coordinates and also in amplitude.
- Digitizing the coordinate values is called **sampling**.
- Digitizing the amplitude values is called **quantization**.

- Fig. 1 shows a continuous image f that we want to convert to digital form.
- The one-dimensional function in Fig. 2 is a plot of amplitude (intensity level) values of the continuous image along the line segment AB in Fig. 1.
- The random variations are due to image noise.

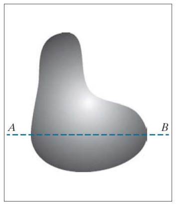


Fig. 1

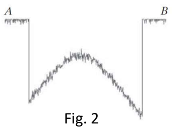


Fig. 2

- To sample this function, we take equally spaced samples along line AB, as shown in Fig. 3. (***Sampling***)
- The set of dark squares constitute the sampled function.
- The vertical gray bar in Fig. 3 depicts the intensity scale divided into eight discrete intervals, ranging from black to white. (***Quantization***)
- The digital samples resulting from both sampling and quantization are shown as white squares in Fig. 4.

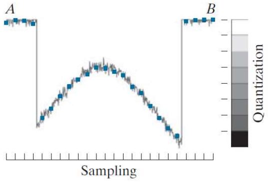


Fig. 3

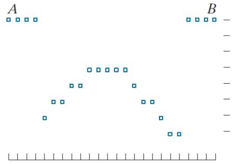
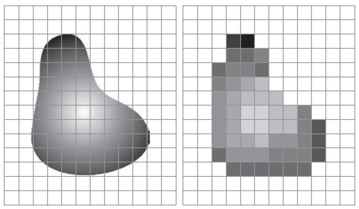
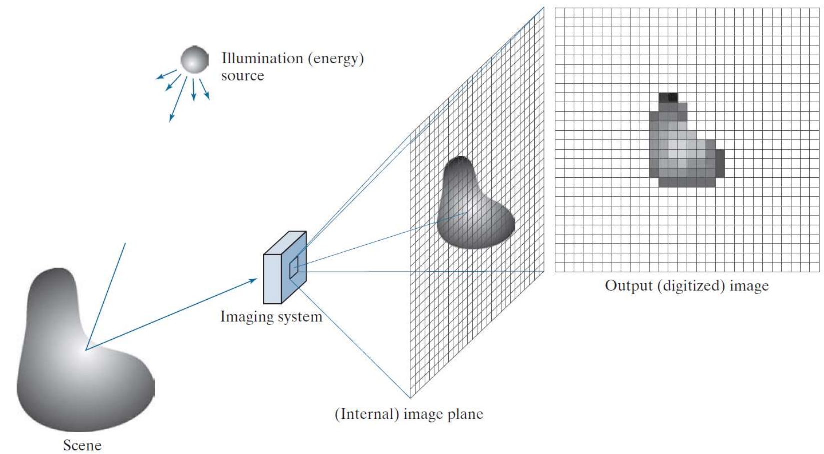


Fig. 4



- Starting at the top of the continuous image and carrying out this procedure downward, line by line, produces a two-dimensional digital image.
- In addition to the number of discrete levels used, the accuracy achieved in quantization is highly dependent on the noise content of the sampled signal.



Representing Digital images

- Let $f(s,t)$ represent a continuous image function of two continuous variables, s and t.
- We convert this function into a digital image by sampling and quantization.
- Suppose that we sample the continuous image into a digital image, $f(x,y)$, containing M rows and N columns.
- Here (x,y) are discrete coordinates.
- Let $x = 0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$.

- We define the origin of an image at the top left corner.
- The value of the digital image at the origin is $f(0,0)$.
- Its value at the next coordinates along the first row is $f(0,1)$.
- Here, the notation $(0,1)$ is used to denote the second sample along the first row.
- In general, the value of a digital image at any coordinates (x,y) is denoted $f(x,y)$, where x and y are integers.
- Fig. 1, Fig. 2 and Fig. 3 in the next slide shows three ways of representing $f(x,y)$.

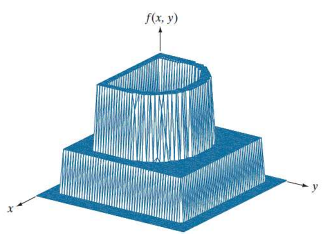


Fig. 1

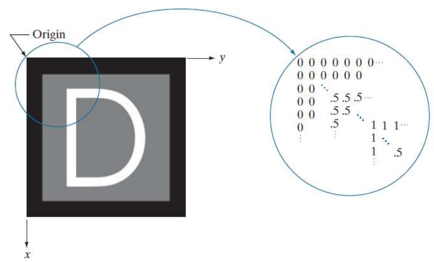


Fig. 2

Fig. 3

- Fig. 1 is a plot of the function. This representation is useful when working with grayscale sets whose elements are expressed in the form (x, y, z), where x and y are co-ordinates and z is gray level.
- Fig. 2 is more common, and it shows $f(x, y)$ as it would appear on a computer display or photograph. This type of representation includes color images, and allows us to view results at a glance.
- Fig. 3 shows, the third representation is a matrix composed of the numerical values of $f(x, y)$. This is the representation used for computer processing.

- We write the representation of an $M \times N$ numerical array as

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

- The right side of this equation is a digital image represented as an array of real numbers.
- Each element of this array is called an image element, picture element, **pixel**, or pel.

- Fig. 4 shows a graphical representation of an image array.
- Here the x and y axis are used to denote the rows and columns of the array.
- $f(i, j)$ is a pixel with coordinates (i, j).

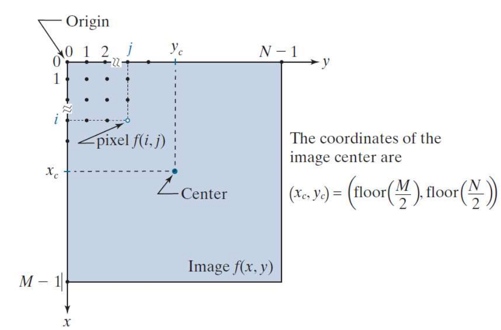


Fig. 4

- We can also represent a digital image in a traditional matrix form:

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}$$

here $a_{i,j} = f(i,j)$.

- There is a convention in image to start at the top left and move to the right, one row at a time.
- By convention, in mathematics also, the first element of a matrix is at the top left of the array.
- ***Mathematically, digital images in reality are matrices***.

- The center of an $M \times N$ digital image with origin at $(0,0)$ and ending at $(M-1, N-1)$ is obtained by dividing M and N by 2 and rounding down to the nearest integer.
- This operation sometimes is denoted using the floor operator
- For example, the center of an image of size 1023×1024 is at $(511, 512)$.
- The number, b, of bits required to store a digital image is

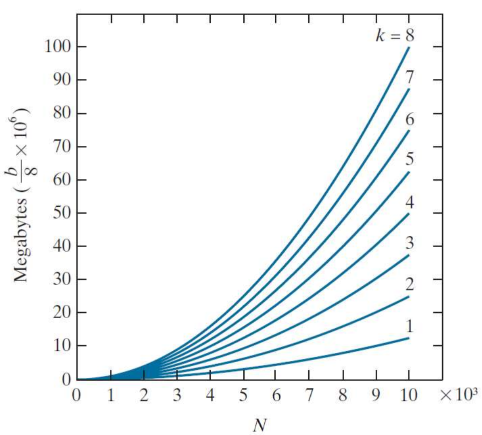
$$b = M \times N \times k$$

- When $M = N$, this equation becomes

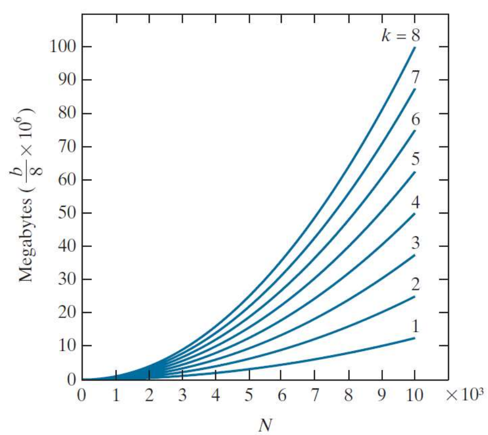
$$b = N^2 \times k$$

Here k is an integer.

- Figure shows the number of megabytes required to store square images for various values of N and k.
- An image can have 2^k possible intensity levels. Such image is called a k-bit image.



- For e.g. $2^8 = 256$ -level image is called an 8-bit image.
- Storage requirements for large 8-bit images is significant.
- An 8-bit image of size 10000×10000 requires $8 \times 10000 \times 10000$ bits $= 8 \times 10^8$ bits $= 100$ MB storage space.



Spatial and Intensity resolution

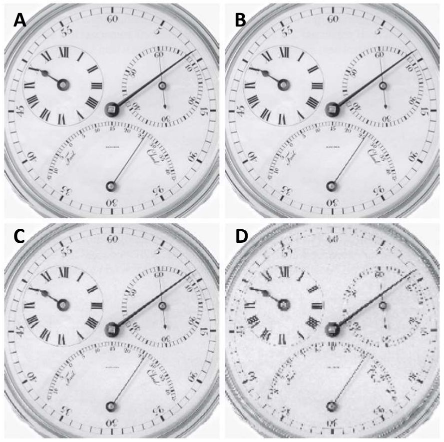
1. **Spatial Resolution:**
   - Spatial resolution is a measure of the smallest discernible (noticeable) detail in an image.
   - More commonly, it is measured dots per inch (dpi).
   - Newspapers are printed with a resolution of 75 dpi.
   - Magazines are printed with a resolution of 133 dpi.
   - Glossy brochures are printed with a resolution of 175 dpi.
   - Book are printed with a resolution of 2400 dpi.

21

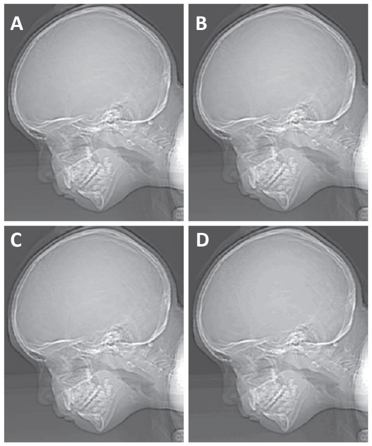
2. **Intensity Resolution:**
   - Intensity resolution refers to the smallest discernible (noticeable) change in intensity level.
   - The number of intensity levels usually is an integer power of two.
   - The most common number is 8 bits i.e. image whose intensity is quantized into 256 levels has 8 bits of intensity resolution.
   - 16 bits are used in some applications in which enhancement of specific intensity ranges is necessary.
   - Intensity quantization using 32 bits is rare.

22

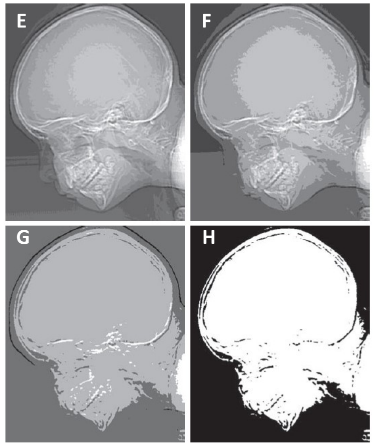
- Effects of reducing spatial resolution.
- The images shown are at:
  A. 930 dpi
  B. 300 dpi
  C. 150 dpi
  D. 72 dpi



- Effects of reducing intensity resolution.
- The images shown are at following intensity levels
  A. 256
  B. 128
  C. 64
  D. 32

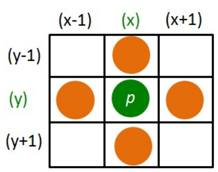


- The images shown are at following intensity levels
  - E. 16
  - F. 8
  - G. 4
  - H. 2



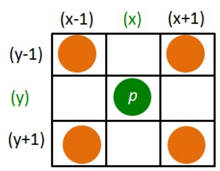
Neighbors of a pixel

1. A pixel p at coordinates (x, y) has two horizontal and two vertical neighbors with coordinates
$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$
This set of pixels, called the **4-neighbors** of p, is denoted $N_4(p)$.
2. The four **diagonal neighbors** of p have coordinates
$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$
and are denoted $N_D(p)$.
3. Diagonal neighbors, together with the 4-neighbors, are called the **8-neighbors** of p, denoted by $N_8(p)$.



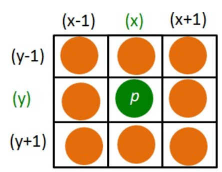
4-neighbors $N_4(p)$

$(x+1, y)$,
$(x-1, y)$,
$(x, y+1)$,
$(x, y-1)$



Diagonal neighbors $N_d(p)$

$(x+1, y+1)$,
$(x+1, y-1)$,
$(x-1, y+1)$,
$(x-1, y-1)$



8-neighbors $N_8(p)$

- The set of image locations of the neighbors of a point p is called the **neighborhood** of p.
- The neighborhood is said to be **closed** if it contains p.
- Otherwise, the neighborhood is said to be **open**.

Adjacency of pixels

- Let V be the set of intensity values used to define adjacency.
- In a binary image, $V = \{1\}$ if we are referring to adjacency of pixels with value 1.
- In a grayscale image, the idea is the same, but set V typically contains more elements.
- For example, if we are dealing with the adjacency of pixels whose values are in the range 0 to 255, set V could be any subset of these 256 values.

- We consider three types of adjacency:
  1. **4-adjacency**: Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.
  2. **8-adjacency**: Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.
  3. ***m*-adjacency (mixed adjacency)**: Two pixels p and q with values from V are m-adjacent if
     a) q is in $N_4(p)$, or
     b) q is in $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V.

- Mixed adjacency is a modification of 8-adjacency, and is introduced to eliminate the ambiguities that may result from using 8-adjacency.
- For example, consider the pixel arrangement in Fig and let $V = \{1\}$.

$$\begin{array}{ccc} 0 & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \end{array}$$

- The three pixels at the top show multiple (ambiguous) 8-adjacency, as indicated by the dashed lines.

$$\begin{array}{ccc} 0 & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \end{array}$$

- This ambiguity is removed by using m-adjacency:

$$\begin{array}{ccc} 0 & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \end{array}$$

- In other words, the center and upper-right diagonal pixels are not m-adjacent because they do not satisfy condition (b) of m-adjacency.

An arrangement of pixels	Pixels that are 8-adjacent	m-adjacency
$\begin{array}{ccc} 0 & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \end{array}$	$\begin{array}{ccc} 0 & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \end{array}$	$\begin{array}{ccc} 0 & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \end{array}$

Path between pixels

- A digital path (or curve) from pixel p with coordinates (x_0, y_0) to pixel q with coordinates (x_n, y_n) is a sequence of distinct pixels with coordinates

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

  where points (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$.
- In this case, n is the length of the path.
- If $(x_0, y_0) = (x_n, y_n)$, the path is a closed path.
- We can define 4-path, 8-path, or m-path, depending on the type of adjacency specified.

- For example, the paths in Fig. 1 between the top right and bottom right points are 8-paths, and the path in Fig. 2 is an m-path.

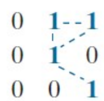


Fig. 1: Pixels that are 8-adjacent

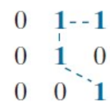


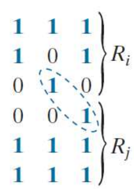
Fig. 2: m-adjacency

Connected pixels

- Let S represent a subset of pixels in an image.
- Two pixels p and q are said to be connected in S if there exists a path between them consisting entirely of pixels in S.
- For any pixel p in S, the set of pixels that are connected to it in S is called a connected component of S.
- If it only has one component, and that component is connected, then S is called a connected set.

Region of image

- Let R represent a subset of pixels in an image.
- We call R a region of the image if R is a connected set.
- Two regions, R_i and R_j are said to be adjacent if their union forms a connected set.
- Regions that are not adjacent are said to be disjoint.
- We consider 4-adjacency and 8-adjacency when referring to regions.
- In Fig., The two regions of 1’s are adjacent only if 8-adjacency is used. Because 4-path between the two regions does not exist, so their union is not a connected set



- Suppose an image contains K disjoint regions, $R_k, k = 1, 2, \dots, K$, none of which touches the image border.
- Let R_u denote the union of all the K regions, and let $(R_u)^c$ denote its complement
- Recall that the complement of a set A is the set of points that are not in A.
- We call all the points in R_u the **foreground**, and all the points in $(R_u)^c$ the **background** of the image.
- The **boundary** (also called the **border** or contour) of a region R is the set of pixels in R that are adjacent to pixels in the complement of R.

- Here also we must specify the connectivity being used to define adjacency.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

- The point circled is not a member of the border of the 1-valued region if 4-connectivity is used between the region and its background.
- Because the only possible connection between that point and the background is diagonal.
- Adjacency between points in a region and its background is defined using 8-connectivity to handle situations such as this.

- The **inner border** of the 1-valued region is the region itself.
- This border does not satisfy the definition of a closed path.
- On the other hand, the **outer border** of the region does form a closed path around the region.

$$\begin{matrix} 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 \end{matrix}$$

Distance measures used in image

- For pixels p, q, and s, with coordinates (x, y), (u, v), and (w, z), respectively, D is a distance function or metric if

  a) $D(p, q) \geq 0$ and $D(p, q) = 0$ iff $p = q$,
  b) $D(p, q) = D(q, p)$, and
  c) $D(p, s) \leq D(p, q) + D(q, s)$.

- The Euclidean distance between p and q is defined as

$$\boxed{D_e(p, q) = [(x-u)^2 + (y-v)^2]^{\frac{1}{2}}}$$

- For this distance measure, the pixels having a distance less than or equal to some value r from (x, y) are the points contained in a disk of radius r centered at (x, y).
- The D_4 distance, called the city-block distance between p and q is defined as

$$\boxed{D_4(p, q) = |x-u| + |y-v|}$$

- In this case, pixels having a D_4 distance from (x, y) that is less than or equal to some value d form a diamond centered at (x, y).

- For example, the pixels with D_4 distance ≤ 2 from (x, y) (the center point) form the following contours of constant distance:

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

- The pixels with $D_4 = 1$ are the 4-neighbors of (x, y).

- The D_8 distance (called the chessboard distance) between p and q is defined as

$$\boxed{D_8(p, q) = \max(|x - u|, |y - v|)}$$

- In this case, the pixels with D_8 distance from (x, y) less than or equal to some value d form a square centered at (x, y).
- For example, the pixels with D_8 distance ≤ 2 form the following contours of constant distance:

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}$$

- The pixels with $D_8 = 1$ are the 8-neighbors of the pixel at (x, y).

- The D_m distance between two points is defined as the shortest m-path between the points.
- In this case, the distance between two pixels will depend on the values of the pixels along the path, as well as the values of their neighbors.
- For instance, consider the following arrangement of pixels and assume that p, p_2, and p_4 have a value of 1, and that p_1 and p_3 can be 0 or 1.

$$\begin{array}{ccc} & p_3 & p_4 \\ p_1 & p_2 & \\ p & & \end{array}$$

$$\begin{array}{ccc} & p_3 & p_4 \\ p_1 & p_2 & \\ p & & \end{array}$$

- Suppose that we consider adjacency of pixels valued 1, $V = \{1\}$.
- If p_1 and p_3 are 0, the D_m distance between p and p_4 is 2.
- If p_1 is 1, then p_2 and p will no longer be m-adjacent and the D_m distance becomes 3.
- If p_3 is 1, then also the D_m distance is 3.
- Finally, if both $p1$ and $p3$ are 1, the length of the D_m distance is 4.

Various operations on Images

- Images can be viewed equivalently as matrices.
- So there are many situations in which operations between images are carried out using matrix theory.
- There are two types of operations:
  1. Matrix operations
  2. Elementwise operations

- Consider the following two 2 × 2 images (matrices):

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

- The **matrix product** of the images is formed using the rules of matrix multiplication:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

- But, in image processing, elementwise operations are widely used.

- The elementwise product is obtained by multiplying pairs of corresponding pixels.
- The **elementwise product** (denoted by ⊕ or ⊙) of these two images is

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

- The elementwise product of two matrices is also called the **Hadamard product** of the matrices.

- In elementwise operations, when we refer to raising an image to a power, we mean that each individual pixel is raised to that power;
- Similarly, when we refer to dividing an image by another, we mean that the division is between corresponding pixel pairs, and so on.
- The symbol ⊖ is often used to denote elementwise division.

Linear operations and Nonlinear operations

- One of the most important classifications of an image processing method is whether it is linear or nonlinear.
- Consider a general operator, $\mathcal{H}$, that produces an output image, $g(x,y)$, from a given input image, $f(x,y)$:

$$\boxed{\mathcal{H}[f(x,y)] = g(x,y)}$$

- Given two arbitrary constants, a and b, and two arbitrary images $f_1(x,y)$ and $f_2(x,y)$, $\mathcal{H}$ is said to be **a linear operator** if

$$\mathcal{H}[af_1(x,y) + bf_2(x,y)] = a\mathcal{H}[f_1(x,y)] + b\mathcal{H}[f_2(x,y)]$$

i.e. $\boxed{\mathcal{H}[af_1(x,y) + bf_2(x,y)] = ag_1(x,y) + bg_2(x,y)}$

$$\boxed{\mathcal{H}[af_1(x,y) + bf_2(x,y)] = ag_1(x,y) + bg_2(x,y)}$$

- This equation indicates that the output of a linear operation applied to the sum of two inputs is the same as performing the operation individually on the inputs and then summing the results. This is called **property of additivity**.
- In addition, the output of a linear operation on a constant multiplied by an input is the same as the output of the operation due to the original input multiplied by that constant. This is called **property of homogeneity**.
- An operator that fails to satisfy above equation, is said to be **nonlinear**.

- As an example, suppose that $\mathcal{H}$ is the sum operator, Σ.
- The function performed by this operator is simply to sum its inputs.
- To test for linearity, we start with the left side and attempt to prove that it is equal to the right side:

$$\begin{aligned}
\therefore LHS &= \mathcal{H}[af_1(x,y) + bf_2(x,y)] \\
&= \sum[af_1(x,y) + bf_2(x,y)] \\
&= \sum[af_1(x,y)] + \sum[bf_2(x,y)] \\
&= a\sum[f_1(x,y)] + b\sum[f_2(x,y)] \\
&= a\mathcal{H}[f_1(x,y)] + b\mathcal{H}[f_2(x,y)] \\
&= ag_1(x,y) + bg_2(x,y) = RHS
\end{aligned}$$

- We conclude that the **sum operator is linear**.

- On the other hand, suppose that we are working with the max operation, whose function is to find the maximum value of the pixels in an image.
- Consider the following two images

$$f_1 = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \quad \text{and} \quad f_2 = \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}$$

- Suppose that we let $a = 1$ and $b = -1$. To test for linearity, we again start with the left side.

$$\max\left\{(1)\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} + (-1)\begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}\right\} = \max\left\{\begin{bmatrix} -6 & -3 \\ -2 & -4 \end{bmatrix}\right\} = -2$$

- Working next with the right side, we obtain

$$(1)\max\left\{\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}\right\} + (-1)\max\left\{\begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}\right\} = 3 + (-1)7 = -4$$

- The left and right sides are not equal in this case, so we have proved that the **max operator is nonlinear**.

Arithmetic operations

- Arithmetic operations between two images $f(x, y)$ and $g(x, y)$ are denoted as

$$s(x,y) = f(x,y) + g(x,y)$$
$$d(x,y) = f(x,y) - g(x,y)$$
$$p(x,y) = f(x,y) \times g(x,y)$$
$$v(x,y) = f(x,y) \div g(x,y)$$

- These are elementwise operations, performed between corresponding pixel pairs in f and g $for\ x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$.

- Here $s, d, p,$ and v are images of size $M \times N$.
- Arithmetic operations are important in digital image processing.
- Some applications are as follows
  1. **Noise reduction** using image addition (averaging) [Page 86]
  2. **Comparing** images using subtraction [Page 87]
  3. **Shading correction and masking** using image multiplication and division [Page 90]

Set operations

- Let the elements of a grayscale image be represented by a set A whose elements are of the form (x, y, z).
- Here x and y are spatial coordinates, and z denotes intensity values.
- We define the complement of A as the set

$$A^c = \{(x, y, K - z) | (x, y, z) \in A\}$$

- A^c is the set of pixels of A whose intensities have been subtracted from a constant K.
- K is maximum intensity value in the image, $2^k - 1$, where k is the number of bits used to represent z.

- Let A denote the 8-bit grayscale image.
- Suppose that we want to form the negative of A using grayscale set operations.
- The negative is the set complement, and this is an 8-bit image.
- For that, we have to let $K = 255$ in the set defined above.

$$A^c = \{(x, y, 255 - z) | (x, y, z) \in A\}$$

- The union of two grayscale sets A and B with the same number of elements is defined as the set

$$A \cup B = \{\max(a, b) \mid a \in A, b \in B\}$$

- The max operation is applied to pairs of corresponding elements.
- The union is an array formed from the maximum intensity between pairs of spatially corresponding elements.

Set operations involving grayscale images.

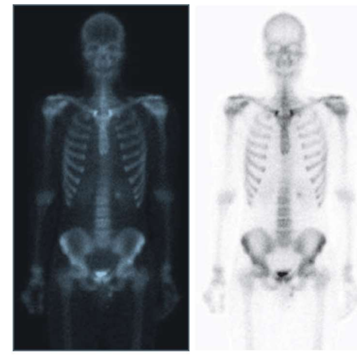


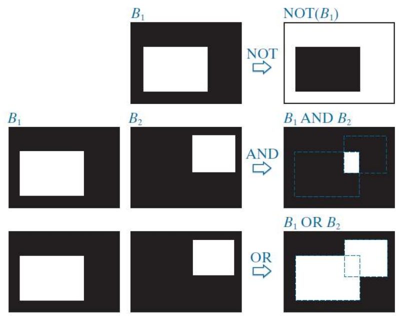
Fig. 1 Original image
Fig. 2 Image negative obtained using grayscale set complementation.

- Let X is a set of M equally spaced values on the x-axis and Y is a set of N equally spaced values on the y-axis.
- The Cartesian product of these two sets define the coordinates of an M −by−N rectangular array (i.e., the coordinates of an image).
- Let X and Y denote the specific $x-$ and $y-$coordinates of a group of 8-connected, 1-valued pixels in a binary image.
- Then set $X \times Y$ represents the region (object) comprised of those pixels.

Logical operations

- We have seen various logical operators like AND, OR etc.
- When applied to two binary images, AND & OR operate on pairs of corresponding pixels between the images.
- They are elementwise operators.
- The NOT of binary image is an array obtained by changing all 1-valued pixels to 0, and vice versa.
- The AND of two binary images contains a 1 at all spatial locations where the corresponding elements of both images are 1; the operation yields 0's elsewhere.

- The OR of these two images is an array that contains a 1 in locations where the elements of one of the image or both images are 1. The array contains 0's elsewhere.



Spatial operations

- Spatial operations are performed directly on the pixels of an image.
- We classify spatial operations into three broad categories:
  1. Single-pixel operations
  2. Neighbourhood operations
  3. Geometric spatial transformations

1. Single-Pixel Operations

- The simplest operation we perform on a digital image is to alter the intensity of its pixels individually using a transformation function, T, of the form

$$\boxed{s = T(z)}$$

- Here z is the intensity of a pixel in the original image
- And s is the intensity of the corresponding pixel in the processed image.

- For example, Fig. shows the transformation used to obtain the negative of an 8-bit image.
- This transformation could be used, for example, to obtain the negative image instead of using sets.

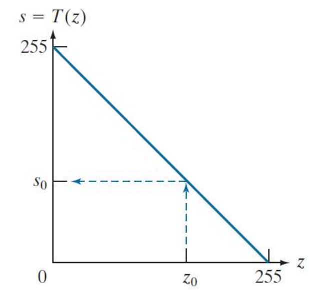


Figure: Intensity transformation function used to obtain the digital equivalent of photographic negative of an 8-bit image.

2. Neighbourhood Operations

- Let S_{xy} denote the set of coordinates of a neighborhood centered on an arbitrary point (x, y) in an image f.
- Neighborhood processing generates a corresponding pixel at the same coordinates in an output (processed) image g.
- The value of that central pixel is determined by some operation on the neighborhood of pixels in the input image with coordinates in the set S_{xy}.
- For example, we can compute the average value of the pixels in a rectangular neighborhood of size $m \times n$ centered on (x, y).

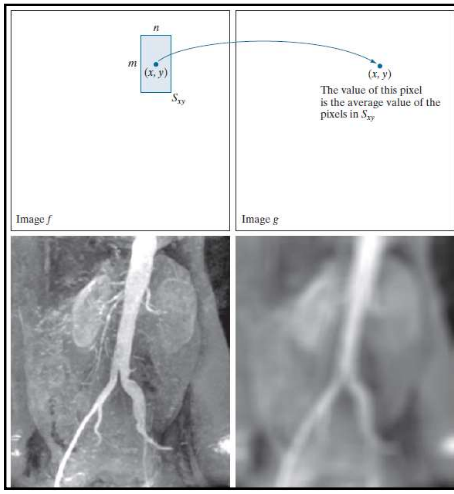
65

- We can express this averaging operation as

$$g(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} f(r, c)$$

- Here r and c are the row and column coordinates of the pixels whose coordinates are in the set S_{xy}.
- Image g is created by varying the coordinates (x, y) so that the center of the neighborhood moves from pixel to pixel in image f.
- Then the neighborhood operation is repeated at each new location.
- Figures illustrate the process.

66



- Figure shows Local averaging using neighborhood processing.
- The procedure is illustrated in (a) and (b) for a rectangular neighborhood.
- (c) An aortic angiogram.
- (d) The result of using averaging with $m = n = 41$.
- The images are of size 790 × 686 pixels.

67

3. Geometric Transformations

- We use geometric transformations modify the spatial arrangement of pixels in an image.
- Geometric transformations of digital images consist of two basic operations.
  1. Spatial transformation of coordinates.
  2. Intensity interpolation that assigns intensity values to the spatially transformed pixels.

68

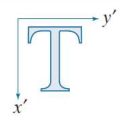
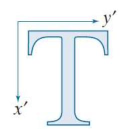
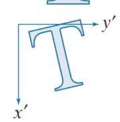
- The transformation of coordinates may be expressed as

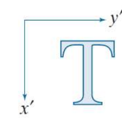
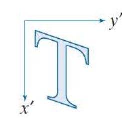
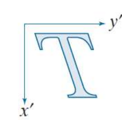
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Here (x, y) are pixel coordinates in the original image and (x', y') are the corresponding pixel coordinates of the transformed image.
- For example, the transformation $(x', y') = (x/2, y/2)$ shrinks the original image to half its size in both spatial directions.
- Other geometric Transformations are scaling, translation, rotation, and shearing.
- So $(x', y') = (x/2, y/2)$ represents **scaling transformation**.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Here A is called **Affine matrix** or **Affine transformation**.
- This transformation can scale, rotate, translate, or sheer an image, depending on the values chosen for the elements of matrix A.
- Following table shows the matrix values used to implement these transformations.

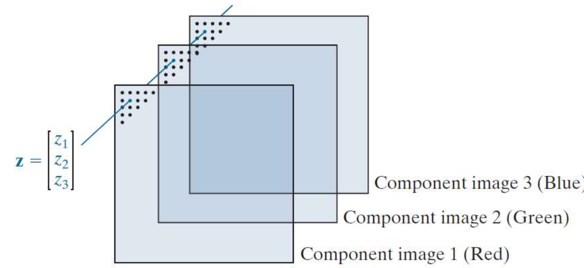
Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to −1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x\cos\theta - y\sin\theta$ $y' = x\sin\theta + y\cos\theta$	

Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

Vector and matrix operations

- Color images are formed in RGB color space by using red, green, and blue component images.
- Each pixel of an RGB image has three components, which can be organized in the form of a column vector $\vec{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$.
- $\boldsymbol{z_1}$ is the intensity of the pixel in the **red** image.
- $\boldsymbol{z_2}$ is the intensity of the pixel in the **green** image.
- $\boldsymbol{z_3}$ is the intensity of the pixel in the **blue** image.

73



- Thus, an RGB color image of size $M \times N$ can be represented by three component images of this size.
- It consists total of MN vectors of size 3×1.

74

- In general, a multispectral image involving n component images will result in $n-$ dimensional vectors like $\vec{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$.
- So vector and matrix operations are necessary.
- The **inner product** (also called the **dot product**) of two $n-$dimensional column vectors $\vec{a}$ and $\vec{b}$ is defined as

$$\vec{a}^T \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

- Here T indicates the transpose.

75

- The **Euclidean vector norm**, denoted by $\|\vec{z}\|$, is defined as the square root of the inner product:

$$\|\vec{z}\| = (\vec{z}^T \vec{z})^{\frac{1}{2}}$$

- It is nothing but length of vector $\vec{z}$.
- **Euclidean distance**, $D(\vec{z}, \vec{a})$, between vectors $\vec{z}$ and $\vec{a}$ in $n-$dimensional space is defined as the Euclidean vector norm

$$D(\vec{z}, \vec{a}) = \|\vec{z} - \vec{a}\| = [(\vec{z} - \vec{a})^T (\vec{z} - \vec{a})]^{\frac{1}{2}}$$

$$i.e.\, D(\vec{z}, \vec{a}) = [(z_1 - a_1)^2 + (z_2 - a_2)^2 + \cdots + (z_n - a_n)^2]^{\frac{1}{2}}$$

76

- Another advantage of pixel vectors is in **linear transformations**, represented as

$$\vec{w} = A(\vec{z} - \vec{a})$$

- Here A is a matrix of size $m \times n$, and $\vec{z}$ and $\vec{a}$ are column vectors of size $n \times 1$.
- We can express an image of size $M \times N$ as a column vector of dimension $MN \times 1$.
- The first M elements of the vector equal the first column of the image.
- The next M elements equal the second column and so on.

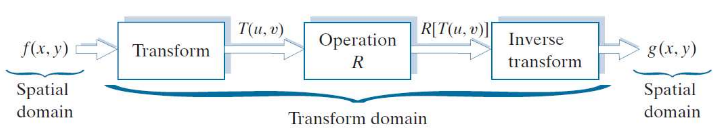
- Let $\vec{f}$ is an $MN \times 1$ vector representing an input image.
- Let $\vec{g}$ is an $MN \times 1$ vector representing a processed image.
- Then

$$\vec{g} = H\vec{f} + \vec{n}$$

- Here H is $MN \times MN$ matrix representing a linear process applied and $\vec{n}$ is $MN \times 1$ vector representing noise.

Image transforms

- Some image operations are not done in spatial domain, i.e. they are not done directly on pixels.
- Input images are transformed to new domain, then specific task is done in transformed domain and finally applying the inverse transform we return to the spatial domain.



- Linear transforms, denoted $T(u, v)$, can be expressed in the general form

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, r(x, y, u, v)$$

- Here $f(x, y)$ is an input image and $r(x, y, u, v)$ is called a **forward transformation kernel**.
- x and y are spatial variables, while M and N are the row and column dimensions of f.
- Variables u and v are called the **transform variables**.
- $T(u, v)$ is called the **forward transform** of $f(x, y)$.

- Given $T(u,v)$, we can recover $f(x,y)$ using the inverse transform of $T(u,v)$.

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)\, s(x,y,u,v)$$

- $s(x,y,u,v)$ is called an **inverse transformation kernel**.
- The nature of a transform is determined by its **kernel**.
- For example, **Fourier transform** is of particular importance in digital image processing.
- The **forward kernel** for Fourier transform is

$$r(x,y,u,v) = e^{-2\ \left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

81

- The **inverse kernel** for Fourier transform is

$$s(x,y,u,v) = \frac{1}{MN} e^{2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

- Here $i = \sqrt{-1}$, so these kernels are complex functions.
- Using these kernels is $T(u,v)$ and $f(x,y)$ we get,

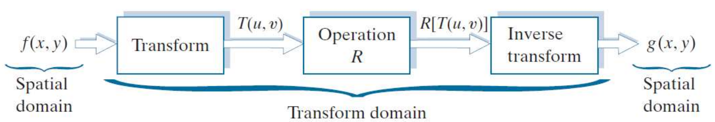
$$T(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) e^{-2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

and

$$f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} T(u,v) e^{2\pi\ \left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

82

Summary: **Image transforms**



- First, the input image is transformed.
- The transform is then modified by a predefined operation .
- Finally, the output image is obtained by computing the inverse of the modified transform.
- Thus, the process goes from the spatial domain to the transform domain, and then back to the spatial domain.

83