COMPUTER VISION

Course 2

Anca Ignat
Faculty of Computer Science
Alexandru Ioan Cuza University of Iași

Representing Digital Images

(x,y) → ***x = 0,1,…,M-1 , y = 0,1,…,N-1 –*** *spatial variables* or *spatial coordinates*

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

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \in \mathbb{R}^{M \times N}, \quad a_{i,j} = f(x=i, y=j) = f(i,j)$$

$a_{i,j}$ – image element, pixel

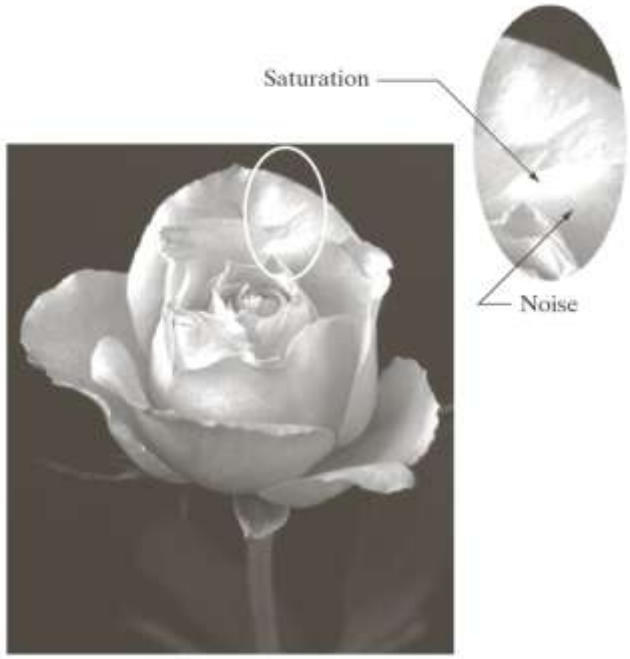
f(0,0) **–** the upper left corner of the image

$$M, N \geq 0, \quad L=2^k$$

$$a_{i,j} \in \mathbb{N} \ , \quad a_{i,j} \in [0, L-1]$$

Dynamic range of an image = the ratio of the maximum measurable intensity to the minimum detectable intensity level in the system

Upper limit – determined by *saturation,* lower limit - *noise*



http://xahlee.info/img/image_editing.html

HYATT

HYATT

HYATT

HYATT

Number of bits required to store a digitized image:

$$b = M \times N \times k \ , \quad \text{for } M = N \ , \ b = N^2 k$$

When an image can have 2^k intensity levels, the image is referred as a ***k-bit image***
256 discrete intensity values – 8-bit image

TABLE 2.1
Number of storage bits for various values of N and k.

N/k	1 ($L = 2$)	2 ($L = 4$)	3 ($L = 8$)	4 ($L = 16$)	5 ($L = 32$)	6 ($L = 64$)	7 ($L = 128$)	8 ($L = 256$)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

Spatial and Intensity Resolution

Spatial resolution – the smallest discernible detail in an image

Measures: *line pairs per unit distance, dots (pixels) per unit distance*

Image resolution = the largest number of discernible line pairs per unit distance

(e.g. 100 line pairs per mm)

Dots per unit distance are commonly used in printing and publishing

In U.S. the measure is expressed in *dots per inch* (***dpi***)

(newspapers are printed with 75 dpi, glossy brochures at 175 dpi)

Intensity resolution – the smallest discernible change in intensity level

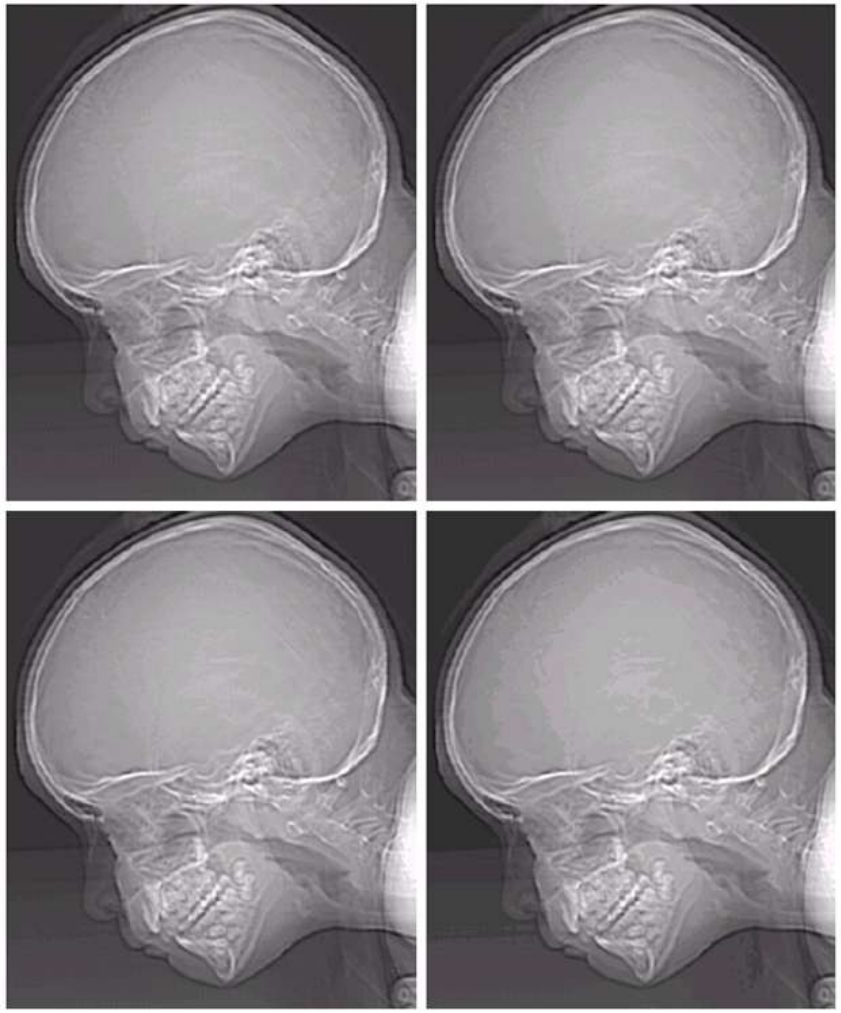
The number of intensity levels (***L***) is determined by hardware considerations

$L=2^k$ – most common $k = 8$

Intensity resolution, in practice, is given by ***k (***number of bits used to quantize intensity)

Fig.1 Reducing spatial resolution: 1250 dpi(upper left), 300 dpi (upper right)
150 dpi (lower left), 72 dpi (lower right)

Reducing the number of gray levels: 256, 128, 64, 32



Reducing the number of gray levels: 16, 8, 4, 2

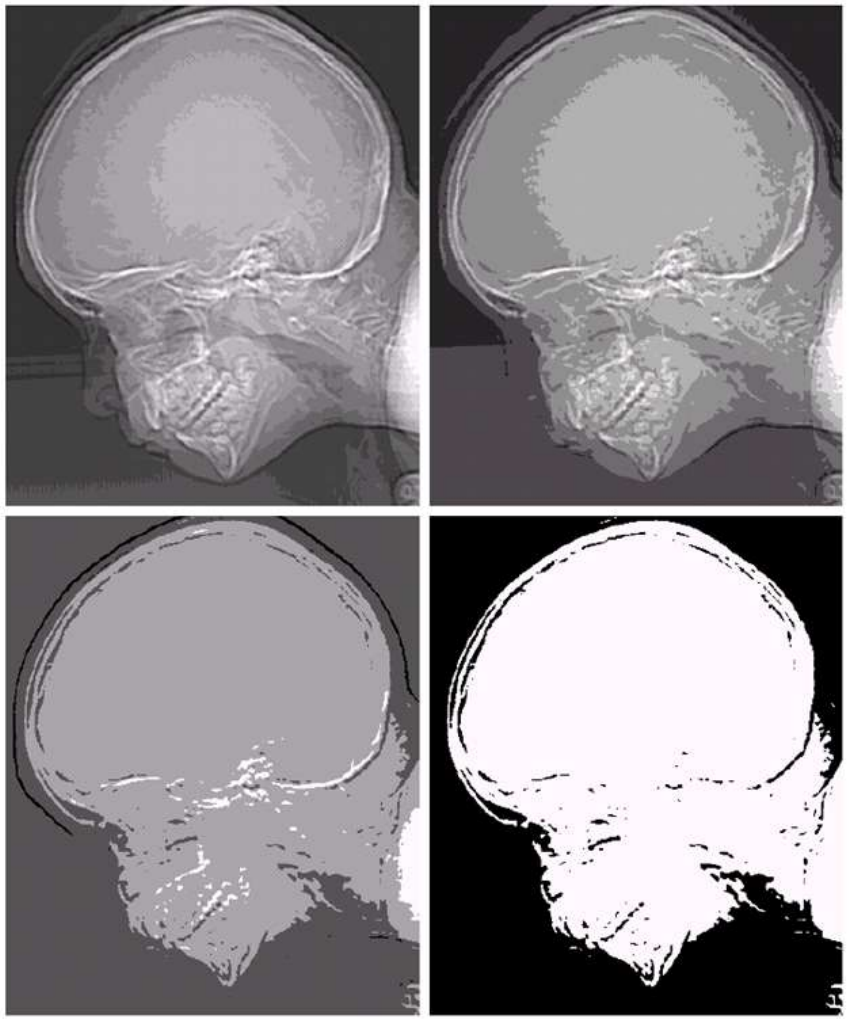


Image Interpolation

- used in zooming, shrinking, rotating, and geometric corrections

Shrinking, zooming – image resizing – image resampling methods

Interpolation is the process of using known data to estimate values at unknown locations

Suppose we have an image of size ***500 × 500*** pixels that has to be enlarged 1.5 times to ***750×750*** pixels. One way to do this is to create an imaginary ***750 × 750*** grid with the same spacing as the original, and then shrink it so that it fits exactly over the original image. The pixel spacing in the ***750 × 750*** grid will be less than in the original image.

Problem: assignment of intensity-level in the new ***750 × 750*** grid

255	250	200	220
245	200	180	200
138	145	150	160
124	130	125	129

→

255	?	250	?	200	?	220	?
?	?	?	?	?	?	?	?
245	?	200	?	180	?	200	?
?	?	?	?	?	?	?	?
138	?	145	?	150	?	160	?
?	?	?	?	?	?	?	?
124	?	130	?	125	?	129	?
?	?	?	?	?	?	?	?

Nearest neighbor interpolation: assign for every point in the new grid (***750 × 750***) the intensity of the closest pixel (nearest neighbor) from the old/original grid (***500 × 500***). This technique has the tendency to produce undesirable effects, like severe distortion of straight edges.

Bilinear interpolation – assign for the new ***(x, y)*** location the following intensity:

$$v(x, y) = a\,x + b\,y + c\,x\,y + d$$

where the four coefficients are determined from the 4 equations in 4 unknowns that can be written using the 4 nearest neighbors of point ***(x, y).***

Bilinear interpolation gives much better results than nearest neighbor interpolation, with a modest increase in computational effort.

Bicubic interpolation – assign for the new ***(x, y)*** location an intensity that involves the 16 nearest neighbors of the point:

$$v(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}c_{i,j}x^{i}y^{j}$$

The coefficients $c_{i,j}$ are obtained solving a ***16x16*** linear system:

$$\sum_{i=0}^{3}\sum_{j=0}^{3}c_{i,j}x^{i}y^{j}=\text{intensity levels of the 16 nearest neighbors of }(x,y)$$

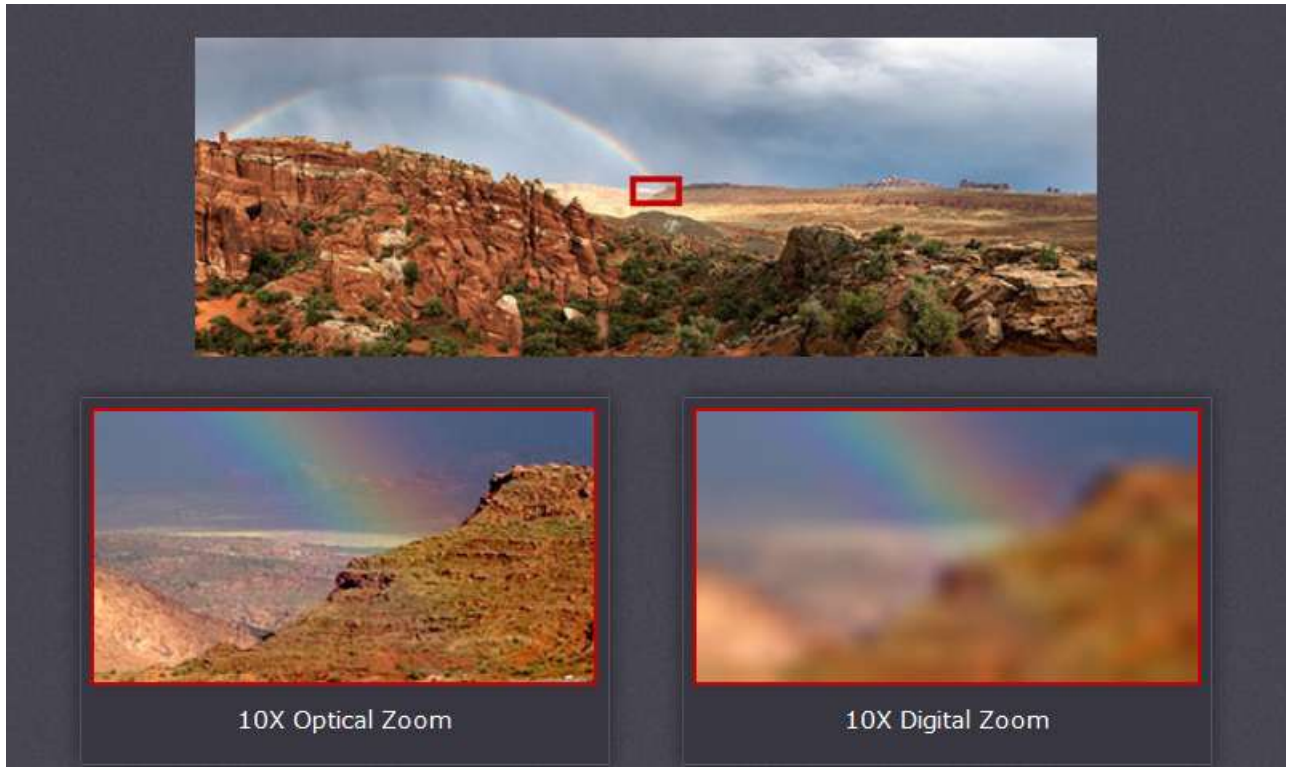
Generally, bicubic interpolation does a better job of preserving fine detail than the bilinear technique. Bicubic interpolation is the standard used in commercial image editing programs, such as Adobe Photoshop and Corel Photopaint.

Figure 2 (a) is the same as Fig. 1 (d), which was obtained by reducing the resolution of the 1250 dpi in Fig. 1(a) to 72 dpi (the size shrank from ***3692 × 2812*** to ***213 × 162***) and then zooming the reduced image

back to its original size. To generate Fig. 1(d) nearest neighbor interpolation was used (both for shrinking and zooming).

Figures 2(b) and (c) were generated using the same steps but using bilinear and bicubic interpolation, respectively. Figures 2(d)+(e)+(f) were obtained by reducing the resolution from 1250 dpi to 150 dpi (instead of 72 dpi)

Fig. 2 – Interpolation examples for zooming and shrinking (nearest neighbor, linear, bicubic)



https://www.cambridgeincolour.com/tutorials/image-interpolation.htm

Neighbors of a Pixel

A pixel ***p*** at coordinates ***(x, y)*** has 4 *horizontal* and *vertical* neighbors:

$$\text{horizontal: } (x+1, y)\,,\ (x-1, y) \quad ; \quad \text{vertical: } (x, y+1)\,,\ (x, y-1)$$

This set of pixels, called the *4-neighbors* of ***p***, denoted by $N_4(p)$.

The 4 *diagonal* neighbors of ***p*** have coordinates:

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

are denoted $N_D(p)$.

The horizontal, vertical and diagonal neighbors are called the ***8-neighbors*** of ***p***, denoted $N_8(p)$.

If ***(x,y)*** is on the border of the image some of the neighbor locations in $N_D(p)$ and $N_8(p)$ fall outside the image.

Adjacency, Connectivity, Regions, Boundaries

Denote by ***V*** the set of intensity levels used to define adjacency.

- in a binary image ***V*** $\subseteq$ ***{0,1} (V={0} , V={1})***
- in a gray-scale image with 256 possible gray-levels, ***V*** can be any subset of ***{0,255}***

We consider 3 types of adjacencies:

(a) *4-adjacency* : two pixels ***p*** and ***q*** with values from ***V*** are *4-adjacent* if $q \in N_4(p)$

(b) *8-adjacency* : two pixels ***p*** and ***q*** with values from ***V*** are *8-adjacent* if $q \in N_8(p)$

(c) *m-adjacency* (mixed adjacency) : two pixels ***p*** and ***q*** with values from ***V*** are *m-adjacent* if :

- $q \in N_4(p)$ or
- $q \in N_D(p)$ 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. It is introduced to eliminate the ambiguities that often arise when *8*-adjacency is used. Consider the example:

$$V = \{1\} \text{ – binary image } \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & \cdots & 1 \\ & \vdots & \iddots & \\ 0 & 1 & & 0 \\ & & \ddots & \\ 0 & 0 & & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & \cdots & 1 \\ & \vdots & & \\ 0 & 1 & & 0 \\ & & \ddots & \\ 0 & 0 & & 1 \end{pmatrix}$$

The three pixels at the top (first line) in the above example show multiple (ambiguous) *8*-adjacency, as indicated by the dashed lines. This ambiguity is removed by using *m*-adjacency.

A *(digital) path* (or *curve*) from pixel ***p*** with coordinates ***(x,y)*** to ***q*** with coordinates ***(s,t)*** is a sequence of distinct pixels with coordinates:

$$(x_0, y_0) = (x, y)\ ,\ (x_1, y_1), \ldots , (x_n, y_n) = (s, t)$$

$$(x_{i-1}, y_{i-1}) \text{ and } (x_i, y_i) \text{ are adjacent, } i = 1, 2, \ldots, n$$

The *length* of the path is ***n***. If $(x_0, y_0) = (x_n, y_n)$ the path is *closed*.

Depending on the type of adjacency considered the paths are: *4*-, *8*-, or *m*-paths.

Let ***S*** denote 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 only of pixels from ***S***.

S is a *connected set* if there is a path in ***S*** between any 2 pixels in ***S.***

Let ***R*** be a subset of pixels in an image. ***R*** is a *region* of the image if ***R*** is a connected set.

Two regions R_1 and R_2 are said to be *adjacent* if $R_1 \cup R_2$ form a connected set. Regions that are not adjacent are said to be *disjoint*. When referring to regions only *4*- and *8*-adjacency are considered.

Suppose that an image contains ***K*** disjoint regions, R_k, $k = 1,...,K$, none of which touches the image border.

$$R_u = \bigcup_{k=1}^{K} R_k \quad , \quad (R_u)^c - \text{the complement of } R_u$$

We call all the points in R_u the *foreground* of the image and the points in $(R_u)^c$ the *background* of the image.

The *boundary (border* or *contour*) of a region ***R*** is the set of points that are adjacent to points in the complement of ***R, (R)^c***. The border of an image is the set of pixels in the region that have at least one background neighbor. This definition is referred to as the *inner border* to distinguish it from the notion of *outer border* which is the corresponding border in the background.

Distance measures

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

(a) $D(p, q) \geq 0$, $D(p, q) = 0$ iff $p=q$

(b) $D(p, q) = D(q, p)$

(c) $D(p, z) \leq D(p, q) + D(q, z)$

The *Euclidean distance* between ***p*** and ***q*** is defined as:

$$D_e(p,q) = \left[(x-s)^2 + (y-t)^2\right]^{\frac{1}{2}} = \sqrt{(x-s)^2 + (y-t)^2}$$

The pixels ***q*** for which $D_e(p,q) \leq r$ are the points contained in a disk of radius ***r*** centered at ***(x, y)***.

The ***D_4 distance*** *(*also called *city-block distance)* between ***p*** and ***q*** is defined as:

$$D_4(p,q) = |x-s| + |y-t|$$

The pixels ***q*** for which $D_4(p,q) \leq r$ form a diamond centered at ***(x,y)***.

$D_4 \leq 2 \rightarrow$

		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

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:

$$D_8(p,q) = \max\{|x-s|, |y-t|\}$$

The pixels ***q*** for which $D_8(p,q) \leq r$ form a square centered at ***(x, y)***.

$D_8 \leq 2 \rightarrow$

$$\begin{matrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{matrix}$$

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

D_4 and ***D_8*** distances are independent of any paths that might exist between ***p*** and ***q*** because these distances involve only the coordinates of the point.

Array versus Matrix Operations

An array operation involving one or more images is carried out on a *pixel-by-pixel* basis.

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

Array product:

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

Matrix product:

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

We assume array operations unless stated otherwise!

Linear versus Nonlinear Operations

One of the most important classifications of image-processing methods is whether it is linear or nonlinear.

$$H\left[f(x,y)\right] = g(x,y)$$

H is said to be a *linear operator* if:

$$H\left[a\,f_1(x,y) + b\,f_2(x,y)\right] = a\,H\left[f_1(x,y)\right] + b\,H\left[f_2(x,y)\right]$$

$$\forall a,b \in \mathbb{R}\ ,\ \forall f_1, f_2 - \text{images}$$

Example of nonlinear operator:

$$H\left[f\right] = \max\{f(x,y)\} = \text{the maximum value of the pixels of image } f$$

$$f_1 = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix},\ f_2 = \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix},\ a = 1,\ b = -1$$

$$\max\{a\, f_1 + b\, f_2\} = \max\left\{1\begin{bmatrix}0 & 2\\ 2 & 3\end{bmatrix} + (-1)\begin{bmatrix}6 & 5\\ 4 & 7\end{bmatrix}\right\} = \max\left\{\begin{bmatrix}-6 & -3\\ -2 & -4\end{bmatrix}\right\} = -2$$

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

Arithmetic Operations in Image Processing

Let ***g(x,y)*** denote a corrupted image formed by the *addition* of noise:

$$g(x,y) = f(x,y) + \eta(x,y)$$

f(x,y) – the noiseless image ; ***η(x,y)*** the noise, uncorrelated and has 0 average value.

For a random variable ***z*** with mean ***m***, ***E[(z-m)^2]*** is the variance (***E(·)*** is the expected value). The covariance of two random variables ***z_1*** and ***z_2*** is defined as ***E[(z_1-m_1) (z_2-m_2)].*** The two random variables are *uncorrelated* when their covariance is 0.

Objective: reduce noise by adding a set of noisy images $\{g_i(x,y)\}$ (technique frequently used in image enhancement)

$$\bar{g}(x,y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x,y)$$

If the noise satisfies the properties stated above, we have:

$$E\left(\bar{g}(x,y)\right) = f(x,y) \quad , \quad \sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)}$$

$E(\bar{g}(x,y))$ is the expected value of $\bar{g}$, and $\sigma^2_{\bar{g}(x,y)}$ and $\sigma^2_{\eta(x,y)}$ are the variances of $\bar{g}$ and η, respectively.

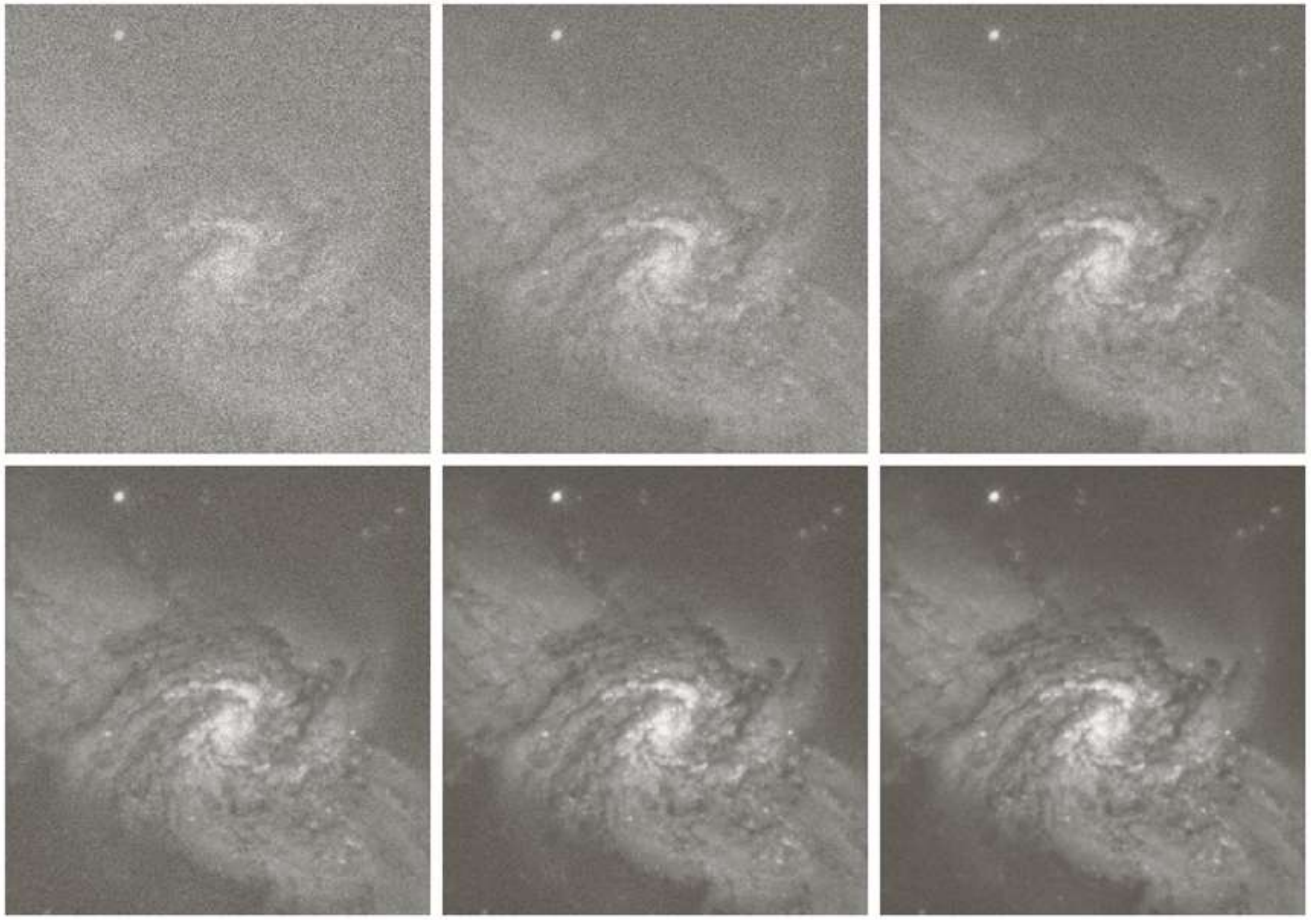
The standard deviation (square root of the variance) at any point in the average image is:

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta(x,y)}$$

As ***K*** increases, the variability (as measured by the variance or the standard deviation) of the pixel values at each location ***(x, y)*** decreases. Because $E\left(\bar{g}(x,y)\right) = f(x,y)$, this means that $\bar{g}(x,y)$ approaches ***f(x, y)*** as the number of noisy images used in the averaging process increases.

An important application of image averaging is in the field of astronomy, where imaging under very low light levels frequently causes sensor noise to render single images virtually useless for analysis. Figure 3(a) shows an 8-bit image in which corruption was simulated by adding to it Gaussian noise with zero

mean and a standard deviation of 64 intensity levels. Figures 3(b)-(f) show the result of averaging 5, 10, 20, 50 and 100 images, respectively.



a b c
d e f

Fig. 3 Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise (left corner, a); Results of averaging 5 (b), 10 (c), 20 (d), 50 (e), 100 (f) noisy images

A frequent application of image *subtraction* is in the enhancement of *differences* between images.

(a) (b) (c)

Fig. 4 (a) Infrared image of Washington DC area; (b) Image obtained from (a) by setting to zero the least significant bit of each pixel; (c) the difference between the two images

Figure 4(b) was obtained by setting to zero the least-significant bit of every pixel in Figure 4(a). The two images seem almost the same. Figure 4(c) is the difference between images (a) and (b). Black (0) values in Figure (c) indicate locations where there is no difference between images (a) and (b).

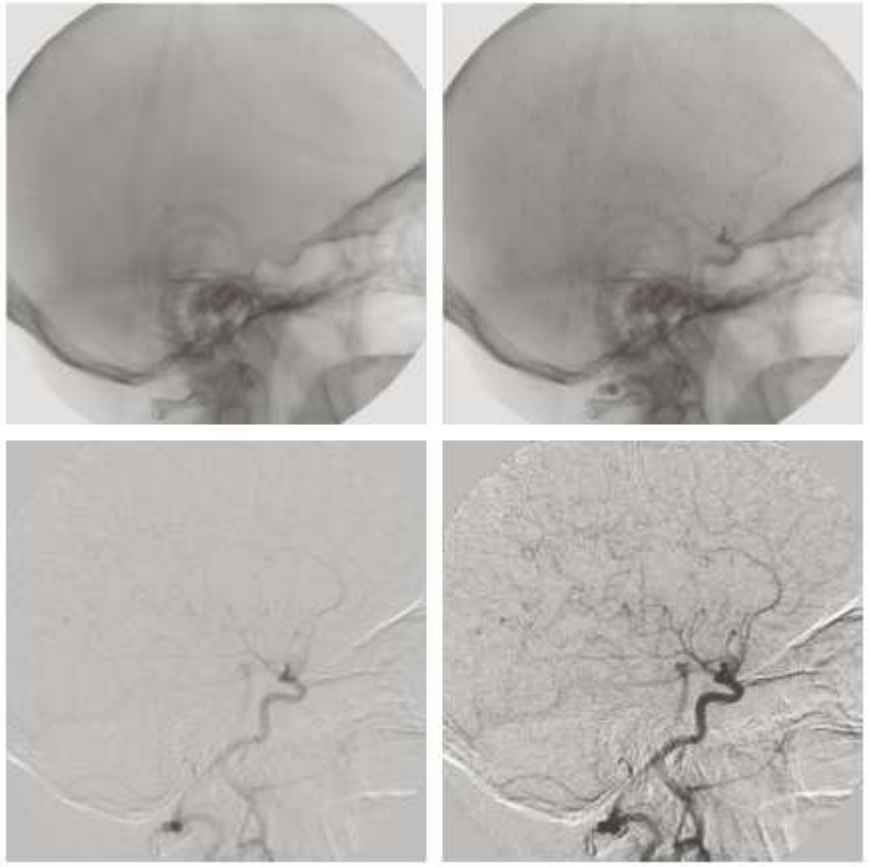
Mask mode radiography

$$g(x, y) = f(x, y) - h(x, y)$$

h(x, y), the *mask*, is an *X*-ray image of a region of a patient's body, captured by an intensified TV camera (instead of traditional *X*-ray film) located opposite an *X*-ray source. The procedure consists of injecting an *X*-ray contrast medium into the patient's bloodstream, taking a series of images called *live images* (denoted ***f(x, y)***) of the same anatomical region as ***h(x, y)***, and subtracting the mask from the series of incoming live images after injection of the contrast medium.

In ***g(x, y)*** we can find the differences between ***h*** and ***f***, as enhanced detail.

Images being captured at TV rates, we obtain a movie showing how the contrast medium propagates through the various arteries in the area being observed.



a b
c d
Fig. 5 – Angiography – subtraction example (a) – mask image; (b) – live image ; (c) – difference between (a) and (b); (d) - image (c) enhanced

An important application of image *multiplication* (and *division*) is *shading correction*.

Suppose that an imaging sensor produces images in the form:

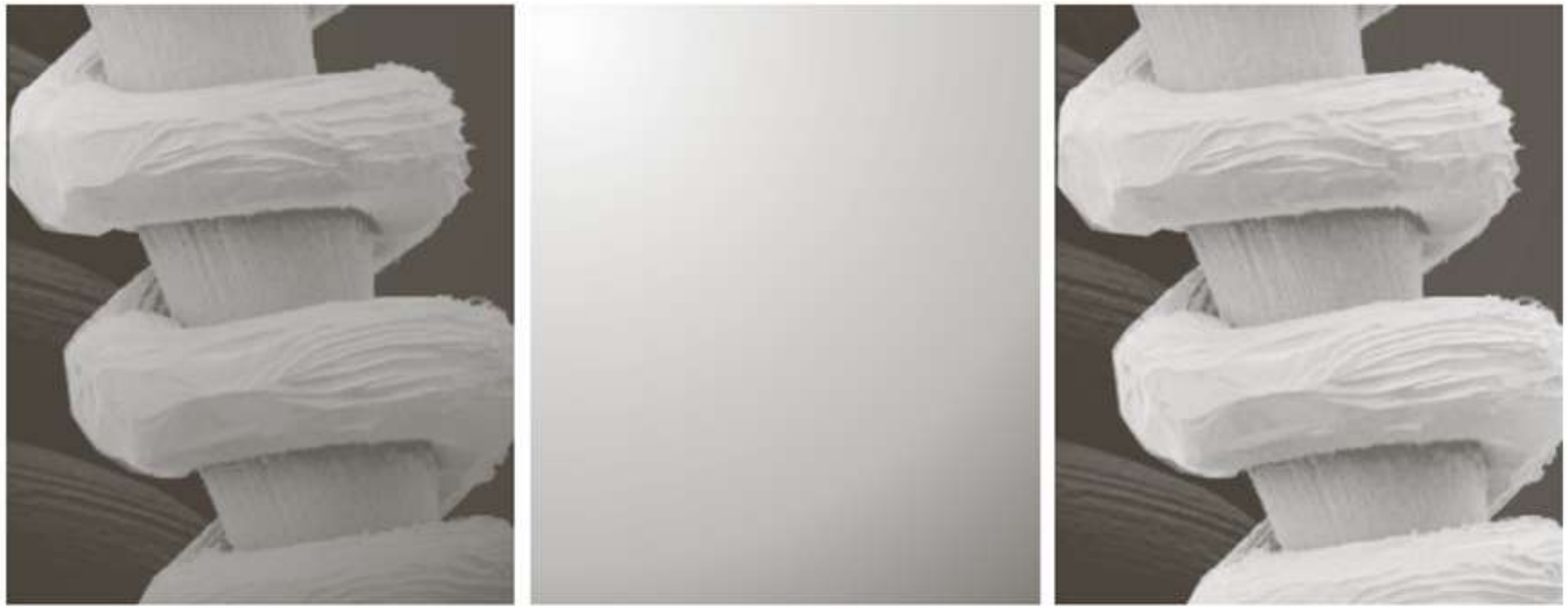
$$g(x,y) = f(x,y) \cdot h(x,y)$$

f(x, y) – the “perfect image” , ***h(x, y)*** – the shading function

When the shading function is known:

$$f(x,y) = \frac{g(x,y)}{h(x,y)}$$

h(x, y) is unknown but if we have access to the imaging system, we can obtain an approximation to the shading function by imaging a target of constant intensity. When the sensor is not available, often the shading pattern can be estimated from the image.



(a) (b) (c)

Fig. 6 Shading correction (a) – Shaded image of a tungsten filament, magnified 130 ×; (b) - shading pattern ; (c) corrected image

Another use of image *multiplication* is in *masking*, also called *region of interest (ROI*), operations. The process consists of multiplying a given image by a mask image that has 1's (white) in the ROI and 0's elsewhere. There can be more than one ROI in the mask image and the shape of the ROI can be arbitrary, but usually is a rectangular shape.

(a) (b) (c)

Fig. 7 (a) – digital dental X-ray image; (b) - ROI mask for teeth with fillings; (c) product of (a) and (b)

In practice, most images are displayed using *8* bits → the image values are in the range ***[0,255]***.
TIFF, JPEG images – conversion to this range is automatic. The conversion depends on the system used.
Difference of two images can produce image with values in the range ***[-255,255]***
Addition of two images – range ***[0,510]***
Many software packages simply set the negative values to 0 and set to 255 all values greater than 255.
A more appropriate procedure: compute

$$f_m = f - \min(f)$$

$$f_s = K \frac{f_m}{\max(f_m)} \in [0\ ,\ K\ ] \quad (K = 255)$$

Spatial Operations

- are performed directly on the pixels of a given image.

There are three categories of spatial operations:

- single-pixel operations
- neighborhood operations
- geometric spatial transformations

Single-pixel operations

Change the values of intensity for the individual pixels

$$s = T(z)$$

where ***z*** is the intensity of a pixel in the original image and ***s*** is the intensity of the corresponding pixel in the processed image.

Neighborhood operations

Let S_{xy} denote a set of coordinates of a neighborhood centered on an arbitrary point ***(x, y)*** in an image ***f***. Neighborhood processing generates new intensity level at point ***(x, y)*** based on the values of the intensities of the points in S_{xy}. For example, if S_{xy} is a rectangular neighborhood of size ***m x n*** centered in ***(x, y)***, we can assign the new value of intensity by computing the average value of the pixels in S_{xy}.

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

The net effect is to perform local blurring in the original image. This type of process is used, for example, to eliminate small details and thus render “blobs” corresponding to the largest region of an image.

Geometric spatial transformations and image registration

- modify the spatial relationship between pixels in an image
- these transformations are often called *rubber-sheet* transformations (analogous to printing an image on a sheet of rubber and then stretching the sheet according to a predefined set of rules.

A geometric transformation consists of 2 basic operations:

- a spatial transformation of coordinates
- intensity interpolation that assign intensity values to the spatial transformed pixels

The coordinate system transformation:

$$(x, y) = T[(v, w)]$$

(v, w) – pixel coordinates in the original image

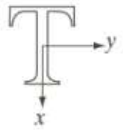
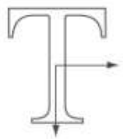
(x, y) – pixel coordinates in the transformed image

$T[(v,w)] = (\frac{v}{2}, \frac{w}{2})$ shrinks the original image half its size in both spatial directions

Affine transform

$$[x,y,1]=[v,w,1]T=[v,w,1]\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix} \Leftrightarrow \begin{aligned} x &= t_{11}v + t_{21}w + t_{31} \\ y &= t_{12}v + t_{22}w + t_{33} \end{aligned} \qquad \textbf{(AT)}$$

This transform can scale, rotate, translate, or shear a set of coordinate points, depending on the elements of the matrix ***T***. If we want to resize an image, rotate it, and move the result to some location, we simply form a ***3x3*** matrix equal to the matrix product of the scaling, rotation, and translation matrices from Table 1.

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

Affine transformations

The preceding transformations relocate pixels on an image to new locations. To complete the process, we have to assign intensity values to those locations. This task is done by using intensity interpolation (like nearest neighbor, bilinear, bi-cubic interpolation).

In practice, we can use equation ***(AT)*** in two basic ways:

→ *forward mapping*: scan the pixels of the input image ***(v, w)***, compute the new spatial location ***(x, y)*** of the corresponding pixel in the new image using ***(AT)*** directly;

Problems:

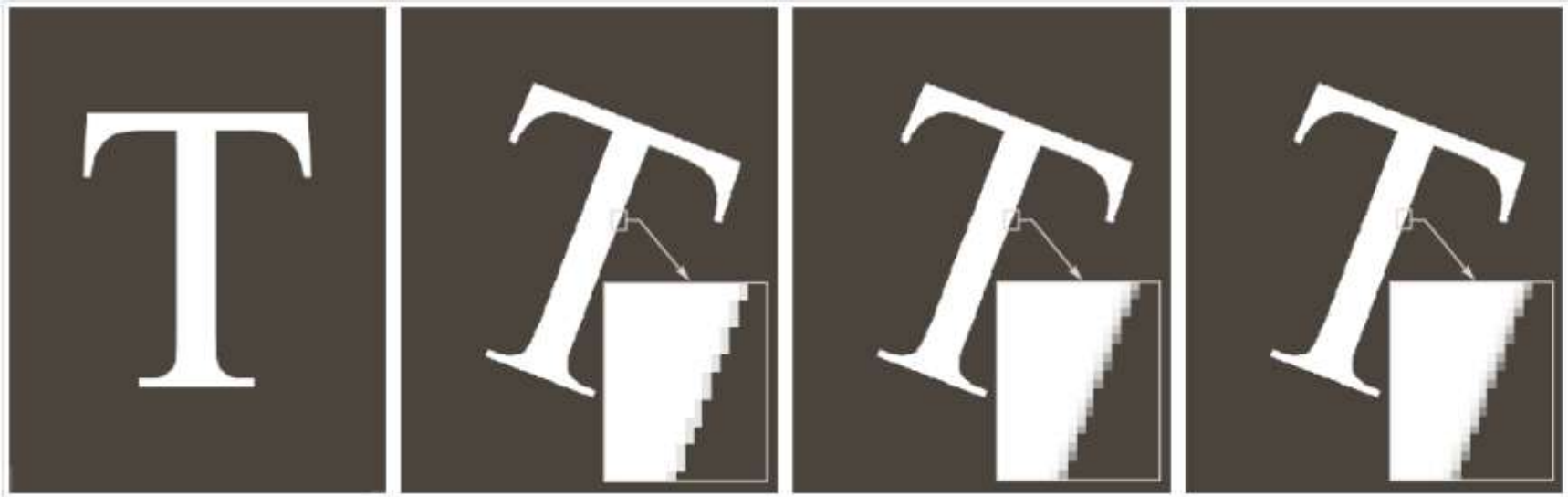
1. intensity assignment when 2 or more pixels in the original image are transformed to the same location in the output image,
2. some output locations have no correspondent in the original image (no intensity assignment)

→ *inverse mapping*: scans the output pixel locations, and at each location, ***(x, y)***, computes the corresponding location in the input image ***(v, w)***

$$(v, w) = T^{-1}(x, y)$$

It then interpolates among the nearest input pixels to determine the intensity of the output pixel value.

Inverse mappings are more efficient to implement than forward mappings and are used in numerous commercial implementations of spatial transformations (MATLAB for ex.).



a b c d

(a) A 300 dpi image of the letter T. (b) Image rotated 21° clockwise using nearest neighbor interpolation to assign intensity values to the spatially transformed pixels. (c) Image rotated 21° using bilinear interpolation. (d) Image rotated 21° using bicubic interpolation. The enlarged sections show edge detail for the three interpolation approaches.

Image registration – align two or more images of the same scene

In image registration, we have available the input and output images, but the specific transformation that produced the output image from the input is generally unknown.

The problem is to estimate the transformation function and then use it to register the two images.

1. **Multi-view Analysis**: Images of the similar object or scene are captured from multiple viewpoints to gain a better representation of the scanned object or scene. Examples include mosaicing of images and shape recovery from the stereo.
2. **Multi-temporal Analysis**: Images of the same object/scene are captured at various times usually under dissimilar conditions to notice changes in the object/scene which emerged between the successive images' acquisitions. Examples include motion tracking, tracking the growth of tumors.

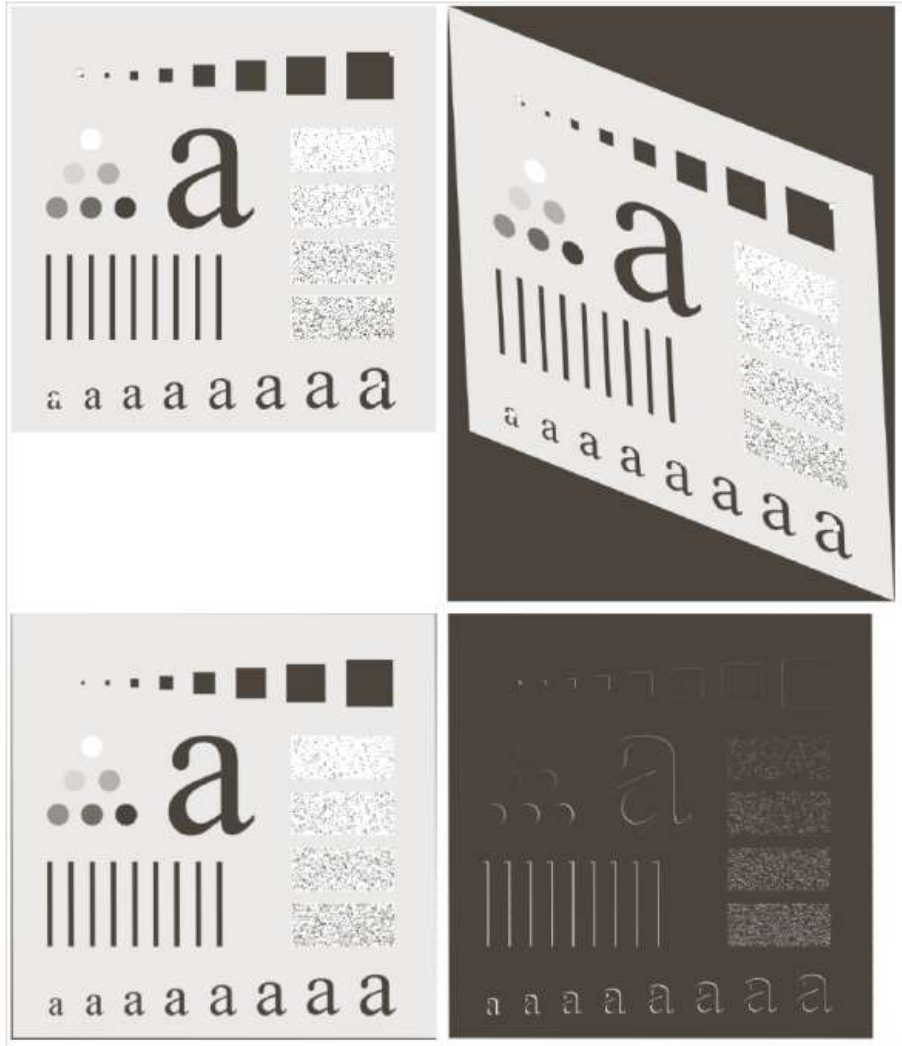
3. **Multi-modal Analysis**: Different sensors are used to acquire the images of the same object/scene to merge the information obtained from various sources to obtain details of the object/scene. Examples include integration of information from multiple sensors, such as MRI, ultrasound or CT, PET, single photon emission computed tomography (SPECT) or magnetic resonance spectroscopy (MRS).

The majority of the registration methods consists of the following four stages:

1) **Feature detection**. Salient and distinctive objects (closed-boundary regions, edges, contours, line intersections, corners, etc.) are manually or, preferably, automatically detected. For further processing, these features can be represented by their point representatives (centers of gravity, line endings, distinctive points), which are called control points (CPs) in the literature. Methods that extract keypoints from an image are very useful.

2) **Feature matching**. In this step, the correspondence between the features detected in the one of the two images and the other image is established. Various feature descriptors and similarity measures along with spatial relationships among the features are used for that purpose.
3) **Transform model estimation**. The type and parameters of the mapping functions, aligning the two images, are estimated. The feature correspondence provide the parameters of the mapping function.
4) **Image resampling and transformation**. Using the mapping function one image is transformed and then the result is compared with the other image.

Medical domain – deep learning approach



a b
c d
(a) – reference image
(b) – geometrically distorted image
(c) - registered image
(d) – difference between (a) and (c)