

# *Computer Vision*

## Course 3

### 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 ;  $\eta(x,y)$  the noise, uncorrelated and has 0 average value.

For a random variable  $z$  with mean  $m$ ,  $E[(z-m)^2]$  is the variance ( $E(\cdot)$  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:

## Computer Vision

---

### Course 3

---

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

$E(\bar{g}(x, y))$  is the expected value of  $\bar{g}$ , and  $\sigma_{\bar{g}(x, y)}^2$  and  $\sigma_{\eta(x, y)}^2$  are the variances of  $\bar{g}$  and  $\eta$ , 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  $E(\bar{g}(x, y)) = f(x, y)$ , 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

## *Computer Vision*

---

### **Course 3**

---

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.

---

**Course 3**

---

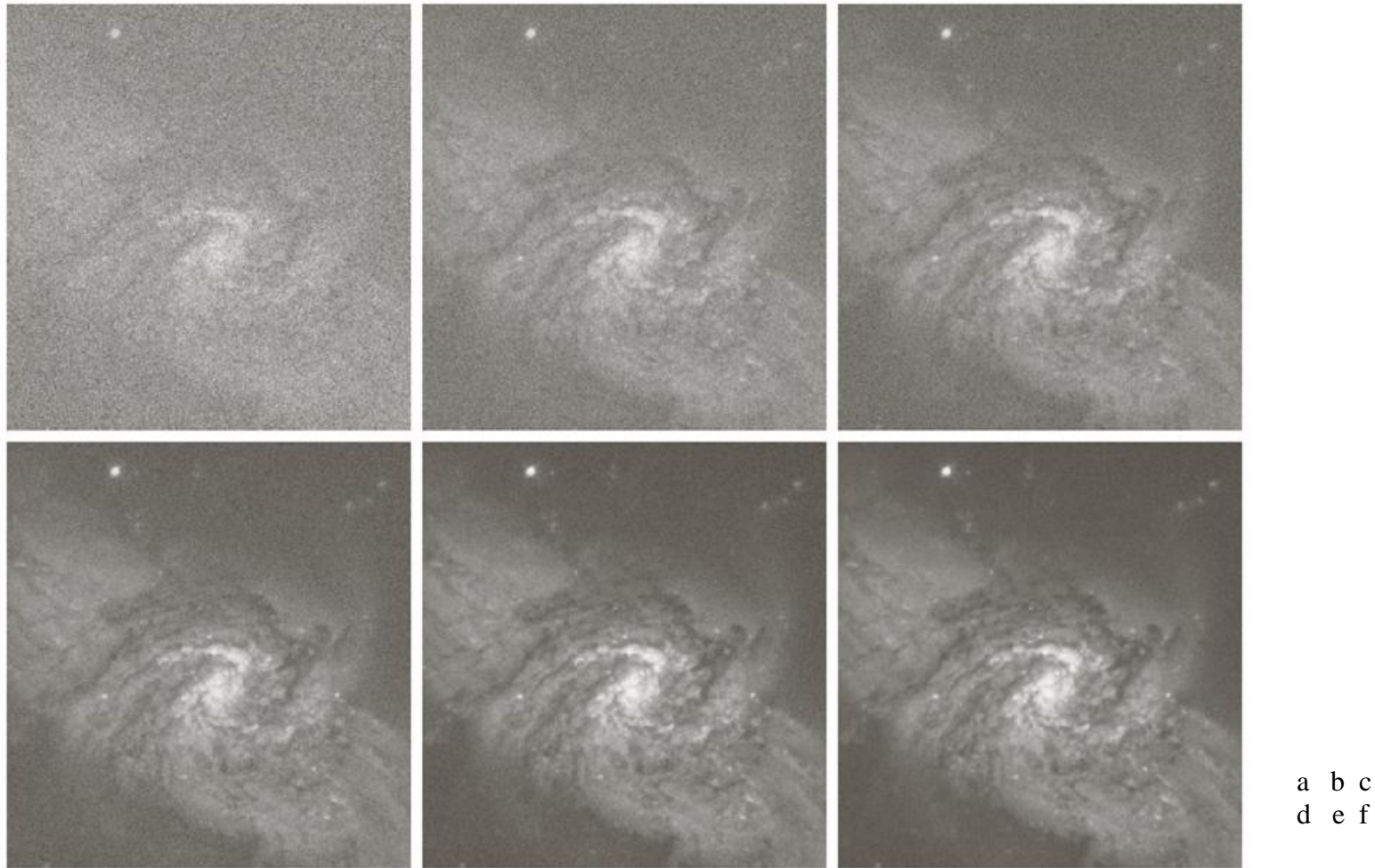


Fig. 3 Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise (left corner); Results of averaging 5, 10, 20, 50, 100 noisy images

## Computer Vision

---

### Course 3

---

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

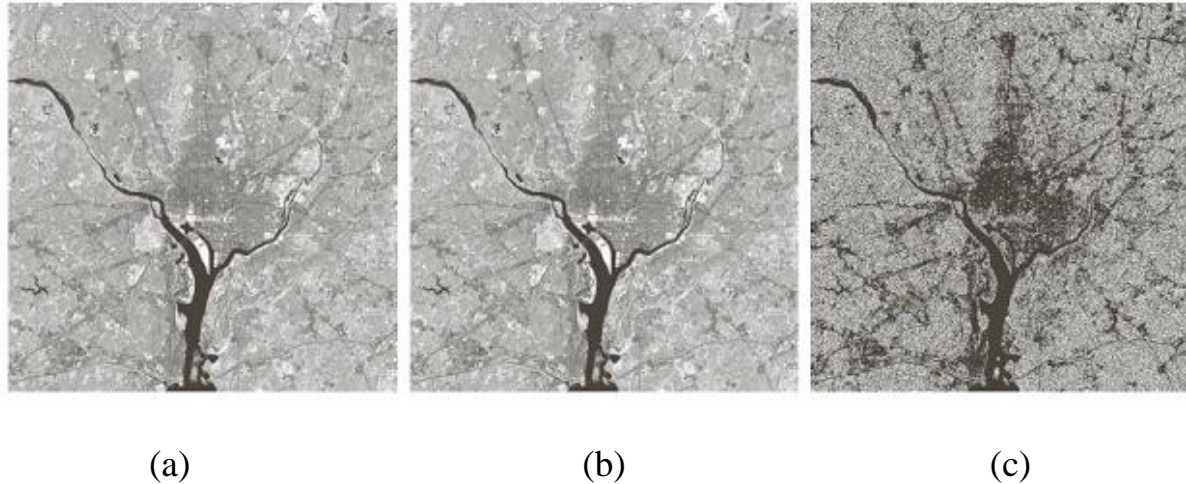


Fig. 4 (a) Infrared image of Washington DC area; (b) Image obtained from (a) by setting to zero the least significant bit of each pixel; (c) the difference between the two images

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).

## Computer Vision

---

### Course 3

---

#### *Mask mode radiography*

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

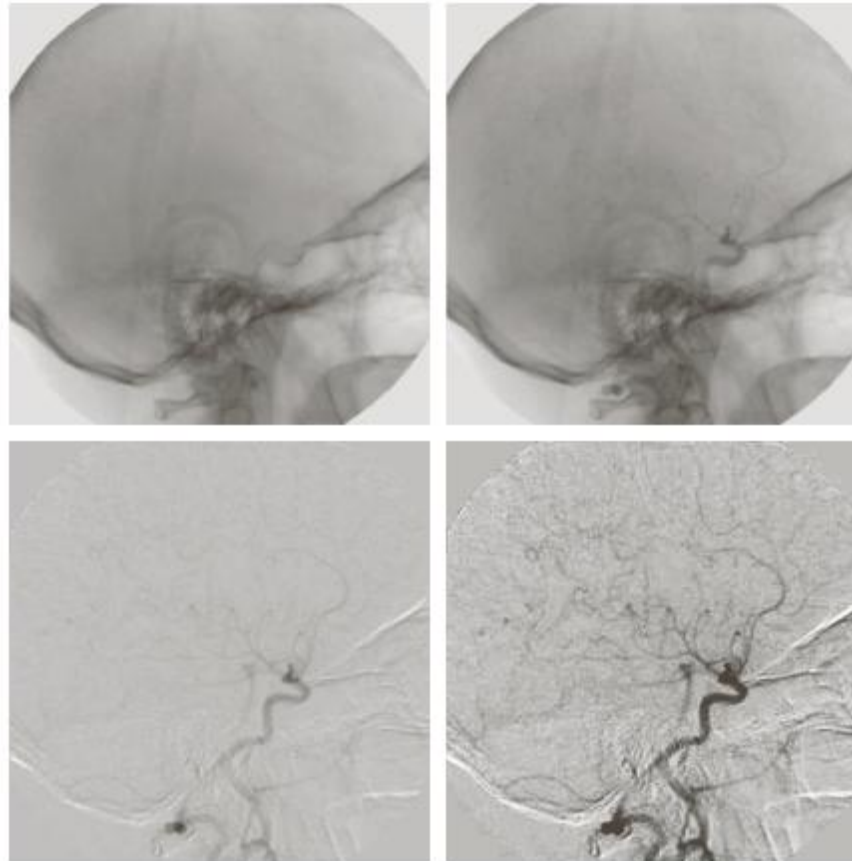
$h(x, y)$ , the *mask*, is an X-ray image of a region of a patient's body, captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source. The procedure consists of injecting an X-ray contrast medium into the patient's bloodstream, taking a series of images 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.

# Computer Vision

## Course 3



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

## Computer Vision

---

### Course 3

---

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

## Computer Vision

---

### Course 3

---

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)

## Computer Vision

---

### Course 3

---

In practice, most images are displayed using 8 bits  $\rightarrow$  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  $m \times n$  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:

1. a spatial transformation of coordinates
2. 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





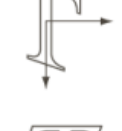
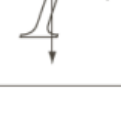
*Affine transform*

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

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  $3 \times 3$  matrix equal to the matrix product of the scaling, rotation, and translation matrices from Table 1.

# Computer Vision

## Course 3

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	$\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v \cos \theta - w \sin \theta$ $y = v \sin \theta + w \cos \theta$	
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

## Computer Vision

---

### Course 3

---

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:*

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

## Computer Vision

---

### Course 3

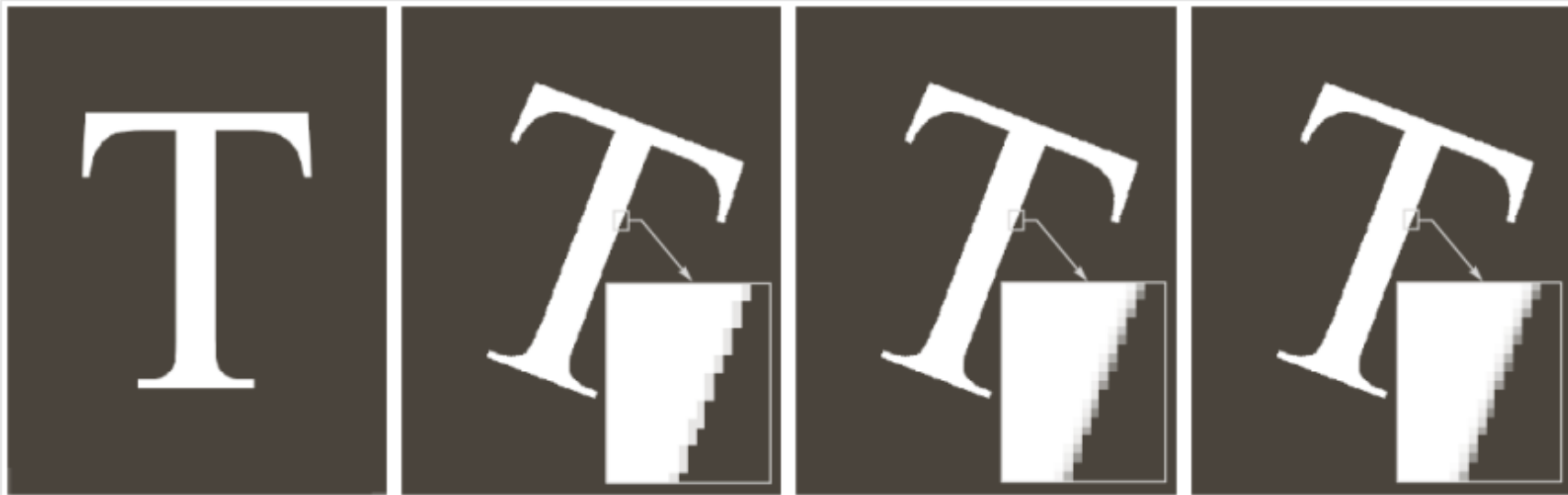
---

→ *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  $21^\circ$  clockwise using nearest neighbor interpolation to assign intensity values to the spatially transformed pixels. (c) Image rotated  $21^\circ$  using bilinear interpolation. (d) Image rotated  $21^\circ$  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

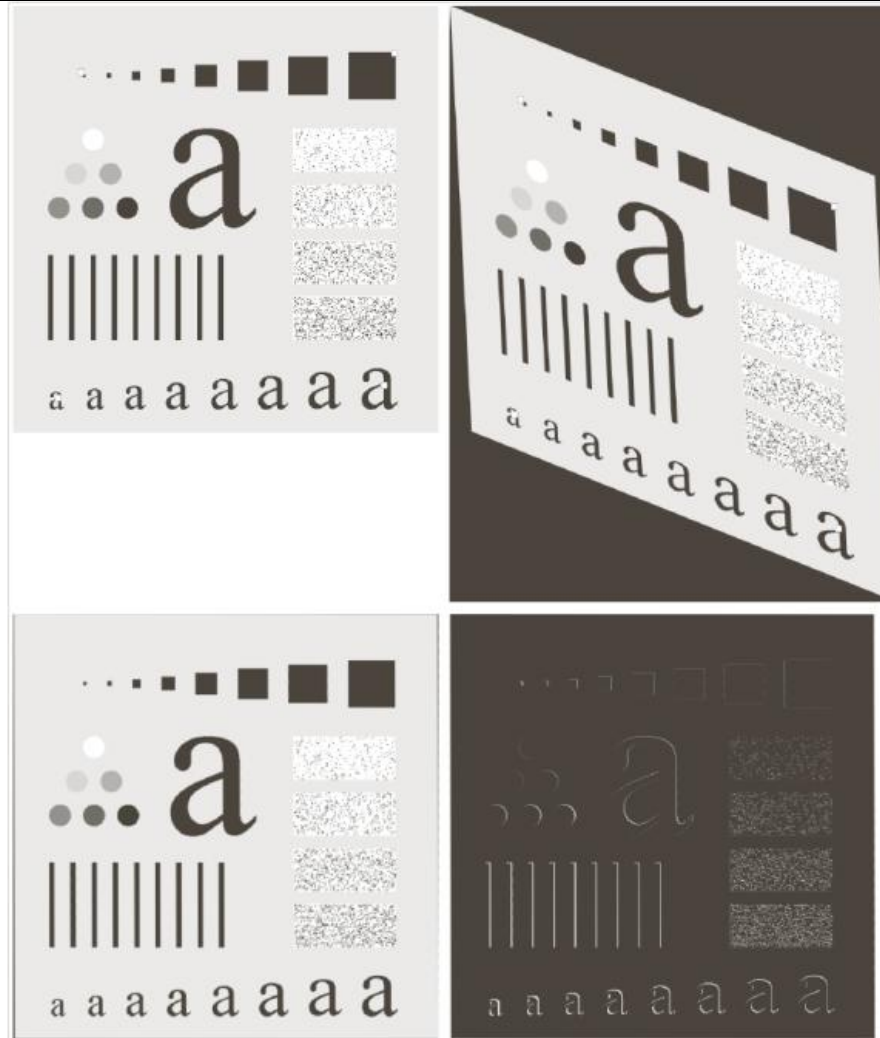
control points (CPs) in the literature. Methods that extract keypoints from an image are very useful.

- 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

# Computer Vision

## Course 3



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

# Computer Vision

---

## Course 3

---

### Probabilistic Methods

$z_i$  = the values of all possible intensities in an  $M \times N$  digital image,  $i=0, 1, \dots, L-1$

$p(z_k)$  = the probability that the intensity level  $z_k$  occurs in the given image

$$p(z_k) = \frac{n_k}{M \cdot N}$$

$n_k$  = the number of times that intensity  $z_k$  occurs in the image ( $MN$  is the total number of pixels in the image)

$$\sum_{k=0}^{L-1} p(z_k) = 1$$

The mean (average) intensity of an image is given by:

$$m = \sum_{k=0}^{L-1} z_k p(z_k)$$

# Computer Vision

---

## Course 3

---

The variance of the intensities is:

$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k)$$

The variance is a measure of the spread of the values of  $z$  about the mean, so it is a measure of image contrast. Usually, for measuring image contrast the standard deviation ( $\sigma$ ) is used.

The  $n$ -th moment of a random variable  $z$  about the mean is defined as:

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_k - m)^n p(z_k)$$

$$(\mu_0(z) = 1, \mu_1(z) = 0, \mu_2(z) = \sigma^2)$$

$\mu_3(z) > 0 \rightarrow$  the intensities are biased to values higher than the mean;

$\mu_3(z) < 0 \rightarrow$  the intensities are biased to values lower than the mean;

## Computer Vision

---

### Course 3

---

$\mu_3(z) = 0 \rightarrow$  the intensities are distributed approximately equally on both side of the mean

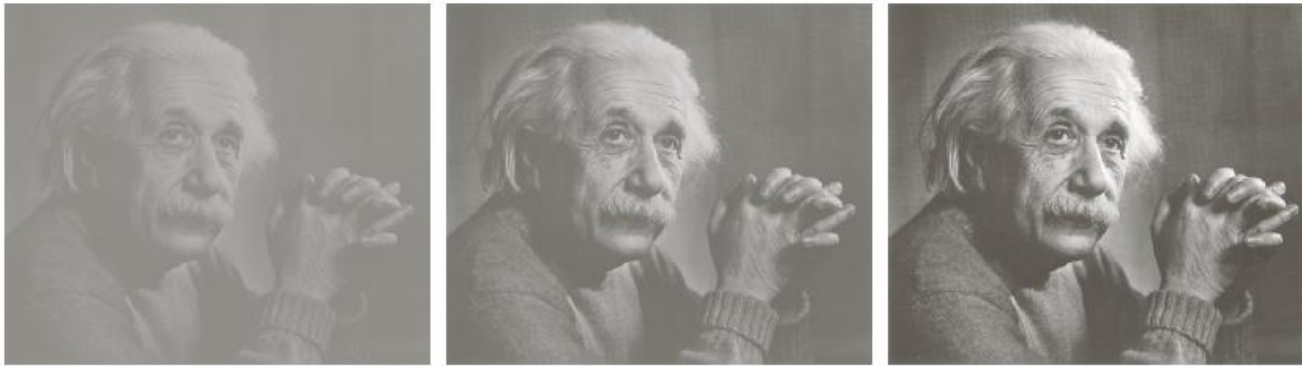


Fig.1

(a) Low contrast

(b) medium contrast

(c) high contrast

Figure 1(a) – standard deviation  $\sigma = 14.3$  (variance = **204.5**)

Figure 1(b) – standard deviation  $\sigma = 31.6$  (variance = **998.6**)

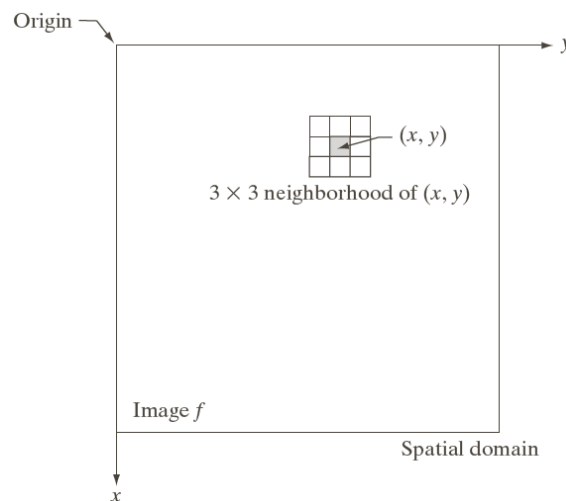
Figure 1(c) – standard deviation  $\sigma = 49.2$  (variance = **2420.6**)

### Intensity Transformations and Spatial Filtering

$$g(x, y) = T[f(x, y)]$$

$f(x, y)$  – input image ,  $g(x, y)$  – output image ,  $T$  – an operator on  $f$  defined over a neighborhood of  $(x, y)$ .

- the neighborhood of the point  $(x, y)$ ,  $S_{xy}$  usually is rectangular, centered on  $(x, y)$ , and much smaller in size than the image



# Computer Vision

## Course 3

- *spatial filtering*, the operator  $T$  (the neighborhood and the operation applied on it) is called *spatial filter* (*spatial mask*, *kernel*, *template* or *window*)

$S_{xy} = \{(x, y)\} \rightarrow T$  becomes an *intensity* (gray-level or *mapping*) *transformation function*

$$s = T(r)$$

$s$  and  $r$  are denoting, respectively, the intensity of  $g$  and  $f$  at  $(x, y)$ .

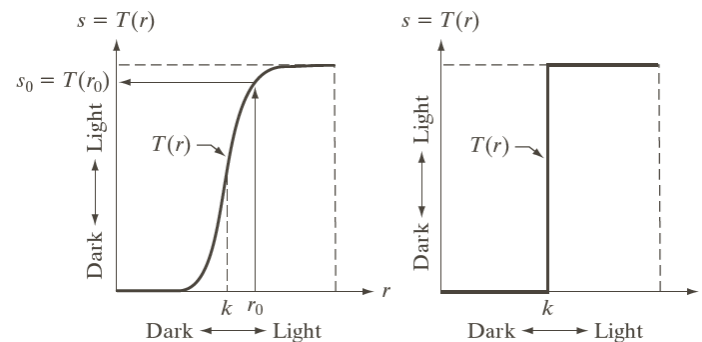


Fig. 2  
Intensity transformation functions.  
left - contrast stretching  
right - thresholding function

Figure 2 left -  $T$  produces an output image of higher contrast than the original, by darkening the intensity levels below  $k$  and brightening the levels above  $k$  – this technique is called *contrast stretching*.

Figure 2 right -  $T$  produces a binary output image. A mapping of this form is called *thresholding* function.

### Some Basic Intensity Transformation Functions

#### Image Negatives

The negative of an image with intensity levels in  $[0, L-1]$  is obtain using the function

$$s = T(r) = L - 1 - r$$

- equivalent of a photographic negative
- technique suited for enhancing white or gray detail embedded in dark regions of an image

# Computer Vision

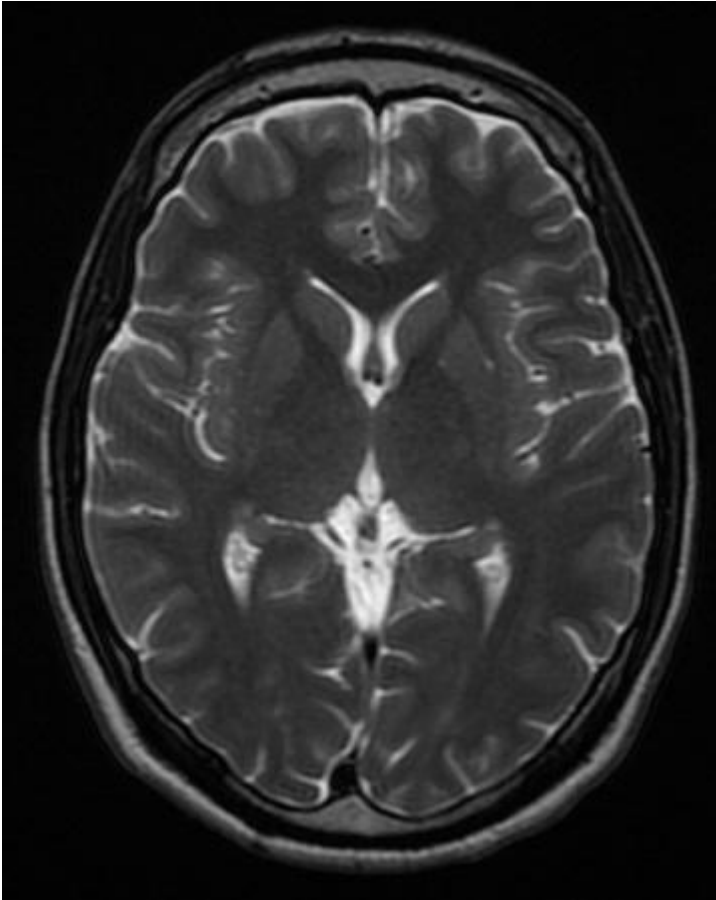
## Course 3



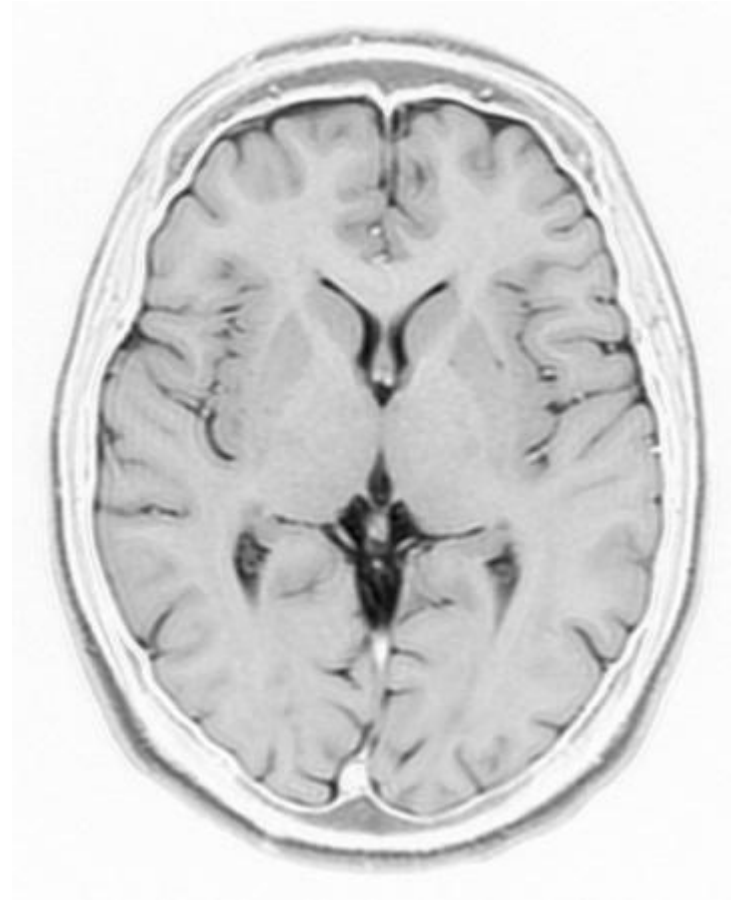
Original



Negative image



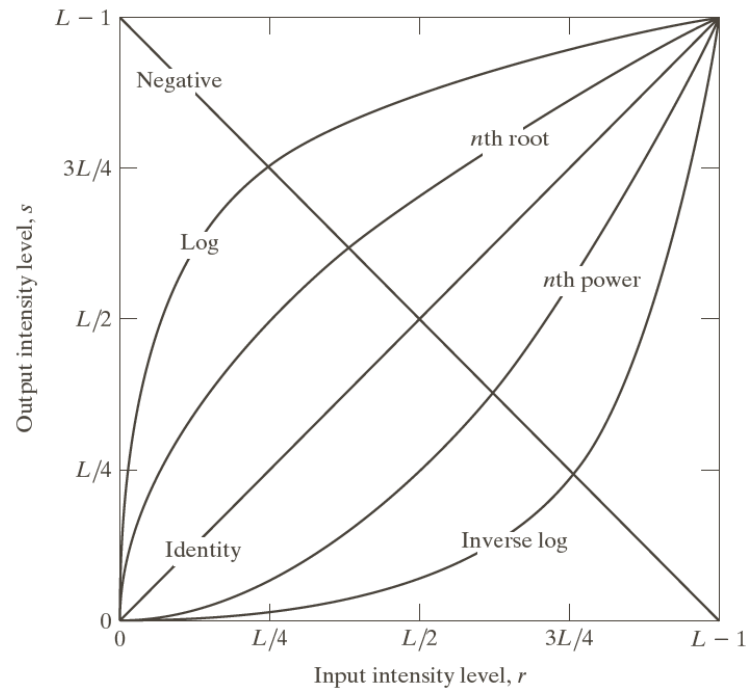
Original



Negative

### Log Transformations

$$s = T(r) = c \log(1+r) , \quad c - \text{constant} , \quad r \geq 0$$



Some basic intensity transformation functions

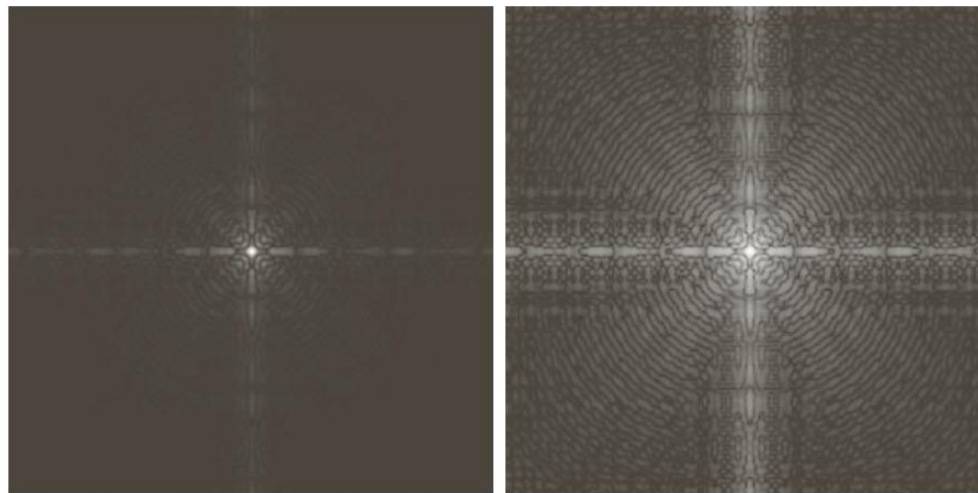
## Computer Vision

---

### Course 3

---

This transformation maps a narrow range of low intensity values in the input into a wider range. An operator of this type is used to expand the values of dark pixels in an image while compressing the higher-level values. The opposite is true for the inverse *log* transformation. The *log* functions compress the dynamic range of images with large variations in pixel values.



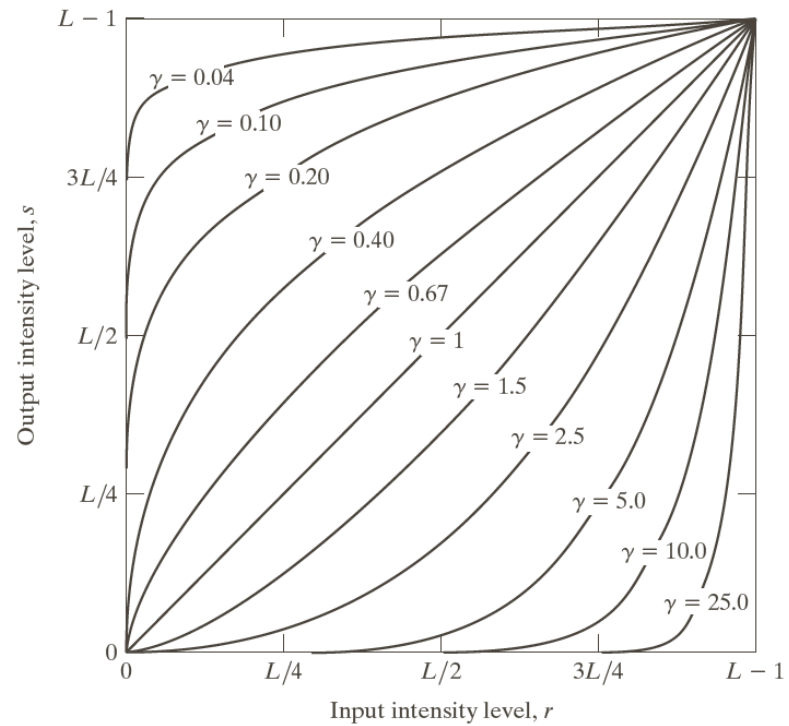
a b  
(a) – Fourier spectrum  
(b) – log transformation  
applied to (a),  $c=1$   
Fig. 4

Figure 4(a) – intensity values in the range  $0$  to  $1.5 \times 10^6$

Figure 4(b) = log transformation of Figure 4(a) with  $c=1$  – range  $0$  to  $6.2$

### Power-Law (Gamma) Transformations

$$s = T(r) = c r^\gamma, \quad c, \gamma - \text{positive constants} \quad (s = c(r + \epsilon)^\gamma)$$



Plots of gamma transformation for different values of  $\gamma$  ( $c=1$ )

## Computer Vision

---

### Course 3

---

Power-law curves with  $\gamma < 1$  map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input values. The curves with  $\gamma > 1$  have the opposite effect of those generated with values of  $\gamma < 1$ .

$c = \gamma = 1$  - identity transformation.

A variety of devices used for image capture, printing, and display respond according to a power law. The process used to correct these power-law response phenomena is called *gamma correction*.

# Computer Vision

## Course 3



a b

c d

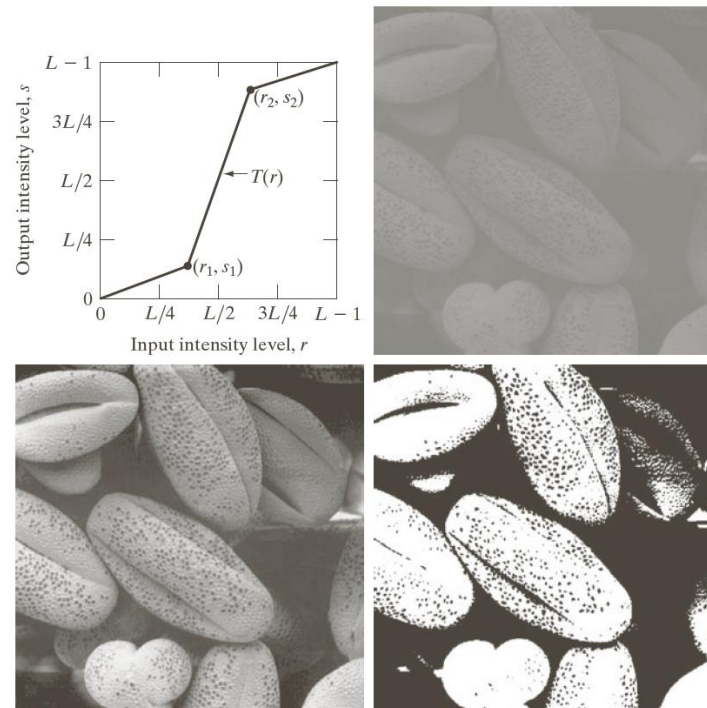
(a) – aerial image

(b) – (d) – results of applying gamma transformation with  $c=1$  and  $\gamma=3.0, 4.0$  and  $5.0$  respectively

Piecewise-Linear Transformations Functions

Contrast stretching

- a process that expands the range of intensity levels in an image so it spans the full intensity range of the recording tool or display device



a b  
c d

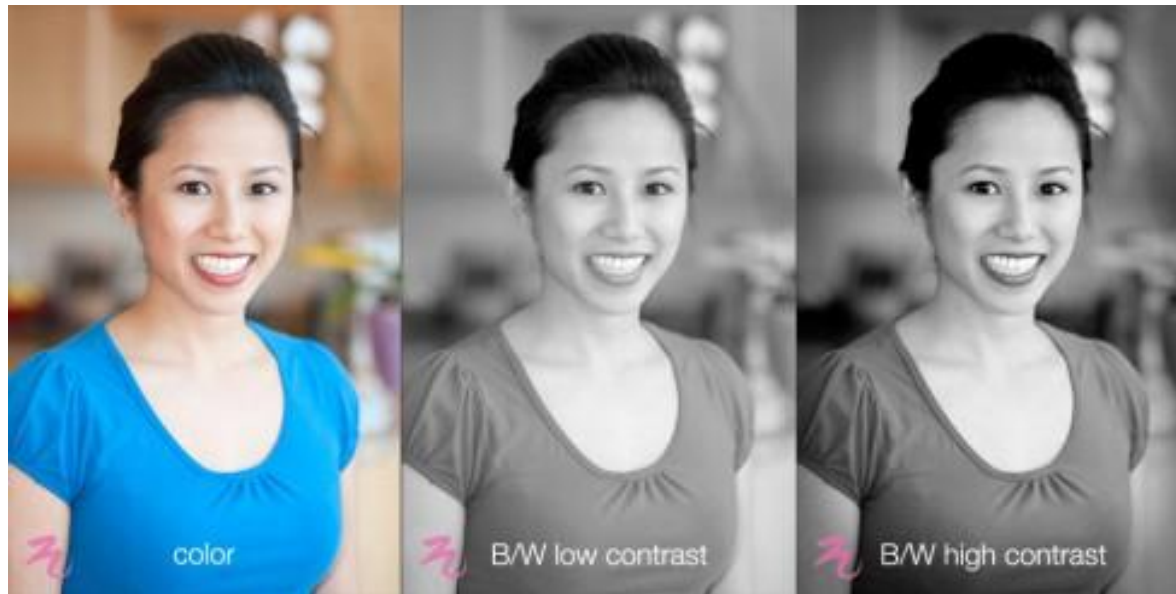
Fig.5

# Computer Vision

---

## Course 3

---



[\(http://www.tripletwist.com/makeuphairblog/bridal/nikond3/\)](http://www.tripletwist.com/makeuphairblog/bridal/nikond3/)

$$T(r) = \begin{cases} \frac{s_1}{r_1} r & r \in [0, r_1] \\ \frac{s_2(r - r_1)}{(r_2 - r_1)} + \frac{s_1(r_2 - r)}{(r_2 - r_1)} & r \in [r_1, r_2] \\ \frac{s_2(L - 1 - r)}{(L - 1 - r_2)} & r \in [r_2, L - 1] \end{cases}$$

## Computer Vision

---

### Course 3

---

$r_1 = s_1$  ,  $r_2 = s_2$   $\rightarrow$  identity transformation (no change)

$r_1 = r_2$  ,  $s_1 = 0$  ,  $s_2 = L-1$   $\rightarrow$  *thresholding function*

Figure 5(b) shows an 8-bit image with low contrast.

Figure 5(c) - contrast stretching, obtained by setting the parameters  $(r_1, s_1) = (r_{\min}, 0)$ ,  $(r_2, s_2) = (r_{\max}, L-1)$  where  $r_{\min}$  and  $r_{\max}$  denote the minimum and maximum gray levels in the image, respectively. Thus, the transformation function stretched the levels linearly from their original range to the full range  $[0, L-1]$ .

Figure 5(d) - the thresholding function was used with  $(r_1, s_1) = (m, 0)$ ,  $(r_2, s_2) = (m, L-1)$  where  $m$  is the mean gray level in the image.

The original image on which these results are based is a scanning electron microscope image of pollen, magnified approximately 700 times.

### **Intensity-level slicing**

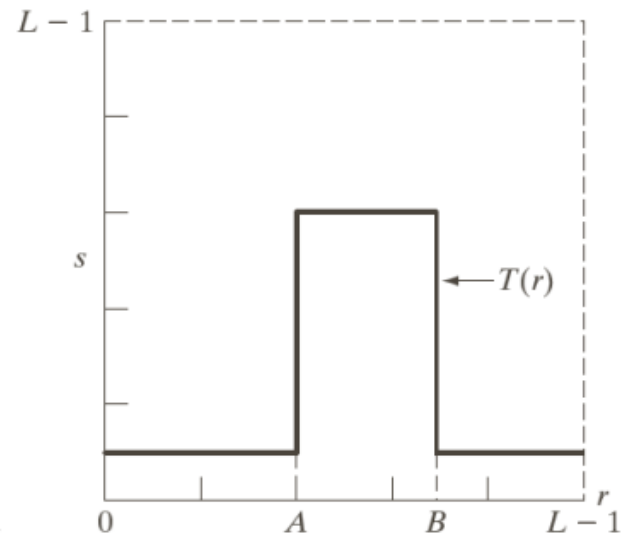
- highlighting a specific range of intensities in an image

There are two approaches for *intensity-level slicing*:

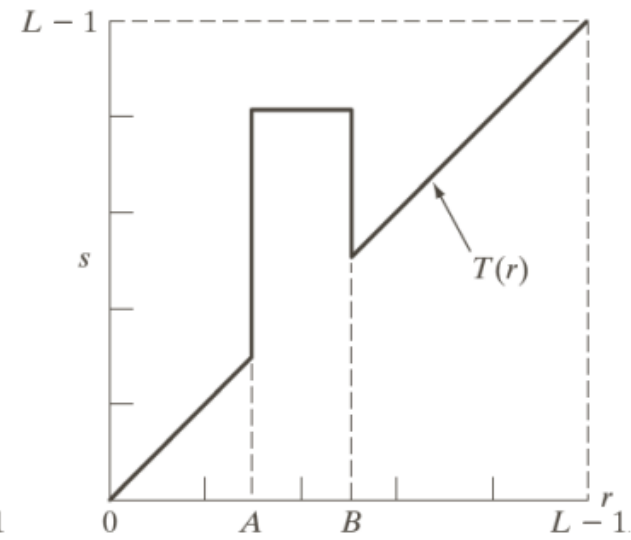
1. display in one value (white, for example) all the values in the range of interest and in another (say, black) all other intensities (Figure 3.11 (a))
2. brighten (or darken) the desired range of intensities but leaves unchanged all other intensities in the image (Figure 3.11 (b)).

# Computer Vision

## Course 3



Highlights intensity range  $[A, B]$   
and reduces all other intensities to a  
lower level



Highlights range  $[A, B]$  and  
preserves all other intensities

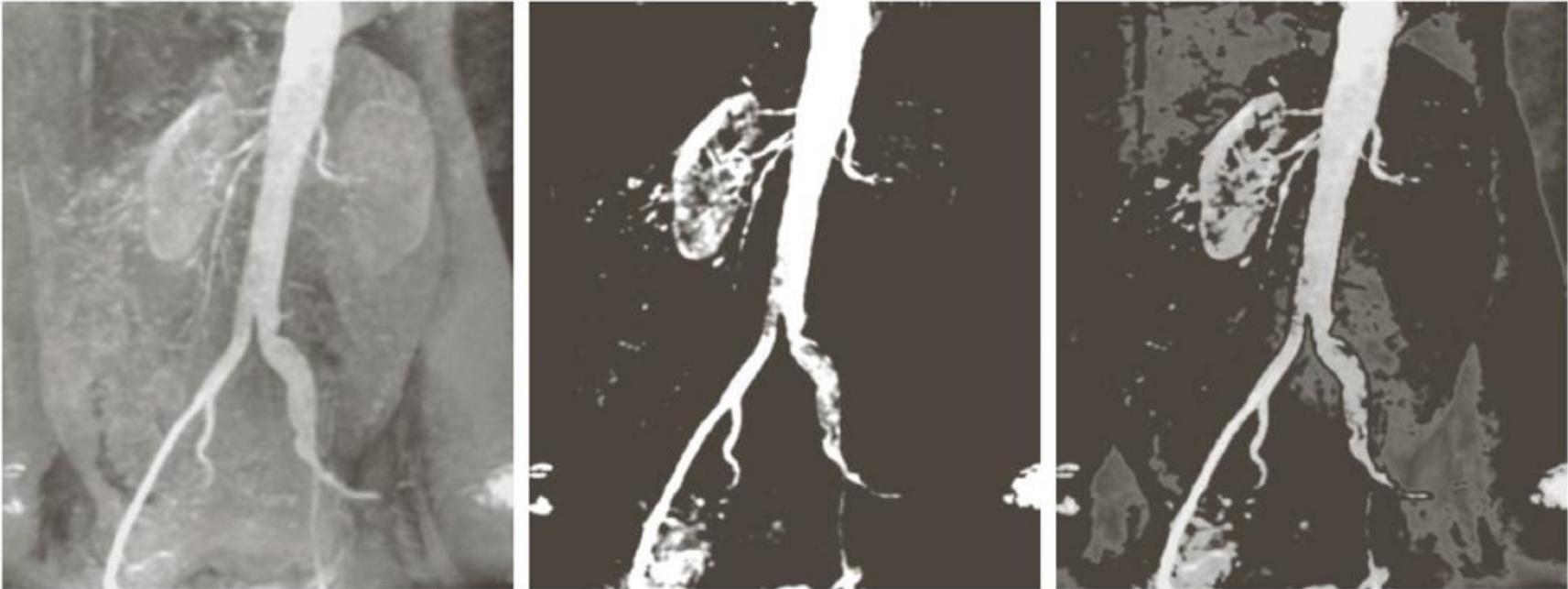


Fig. 6 - Aortic angiogram and intensity sliced versions

## *Computer Vision*

---

### **Course 3**

---

Figure 6 (left) – aortic angiogram near the kidney. The purpose of intensity slicing is to highlight the major blood vessels that appear brighter as a result of injecting a contrast medium. Figure 6(middle) shows the result of applying the first technique for a band near the top of the scale of intensities. This type of enhancement produces a binary image which is useful for studying the shape of the flow of the contrast substance (to detect blockages...)

In Figure 3.12(right) the second technique was used: a band of intensities in the mid-gray image around the mean intensity was set to black, the other intensities remain unchanged.

# Computer Vision

---

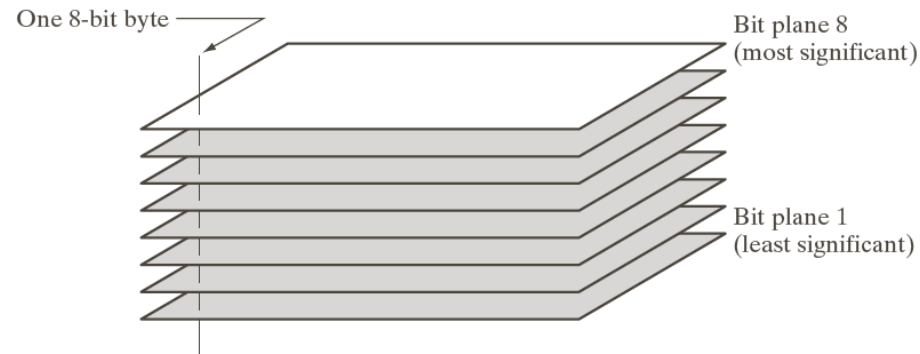
## Course 3

---

### Bit-plane slicing

For an 8-bit image,  $f(x, y)$  is a number in  $[0, 255]$ , with 8-bit representation in base 2

This technique highlights the contribution made to the whole image appearances by each of the bits. An 8-bit image may be considered as being composed of eight 1-bit planes (plane 1 – the lowest order bit, plane 8 – the highest order bit)



# Computer Vision

## Course 3



a	b	c
d	e	f
g	h	i

(a) An 8-bit gray-scale image of size  $500 \times 1192$  pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

# Computer Vision

---

## Course 3

---

The binary image for the 8-th bit plane of an 8-bit image can be obtained by processing the input image with a threshold intensity transformation function that maps all the intensities between 0 and 127 to 0 and maps all levels between 128 and 255 to 1.

The bit-slicing technique is useful for analyzing the relative importance of each bit in the image – helps in determining the proper number of bits to use when quantizing the image. The technique is also useful for image compression.



a b c

Images reconstructed using (a) bit planes 8 and 7; (b) bit planes 8, 7, and 6; and (c) bit planes 8, 7, 6, and 5.

---