Image and Video Processing

ELEX 7815

John Dian

Image and video processing – concept (Cont.)

- ✓ Processing example
  - ❖ Changes in color / size/ brightness
    - Old movies to color
    - Various display size
    - Quality Enhancement
  - ❖ Object detection and recognition
    - Face detection/ red eye
    - Construction example
  - ❖ Comparing
    - Finger print
    - Industrial PCB
    - Search based on Image
  - ❖ Finding motion
    - Position recovery
    - Speed and acceleration recovery
    - Car license plate
    - Security /alarm
  - ❖ Pattern recognition
    - Barcode
    - Shiny code
  - ❖ Compression
    - Storage
    - Transmission / network capabilities
  - ❖ Synthetic image and video
    - 3D movie
    - Virtual world
  - ✓ Image video and projector
    - digital ink
    - Digital Piano
  - ❖ Watermarking

Image and video processing - concept

- ✓ Digital image
  - ❖ Visual representation of an object, a person, or a scene produced by an optical device.
  - ❖ An image is represented in two-dimensional domain using a finite number of points, usually referred to as pixels
- ✓ *Digital* video
  - ❖ A sequence of images representing visual information over time
  - ❖ A digital video signal is represented in three-dimensional domain
- ✓ Digital Image & video processing
  - ❖ Modifying a digital image or a digital video sequence using computer algorithms for the purpose of
    - ▪ Enhancement : techniques to improve the image or video to be more suitable than the original one for a specific application
    - ▪ Restoration : techniques to improve the image or video that were subject to degradation &noise
    - ▪ Compression : techniques to reduce the size of an image or video
    - ▪ Segmentation: techniques to partition the objects inside the image or video
    - ▪ Recognition : the techniques to assign label to an object inside an image or video
- ✓ The result of processing

Input : image or video Output : attributes extracted from the image or video

Input : image or video Output : image or video

Image and video processing - concept (Cont.)

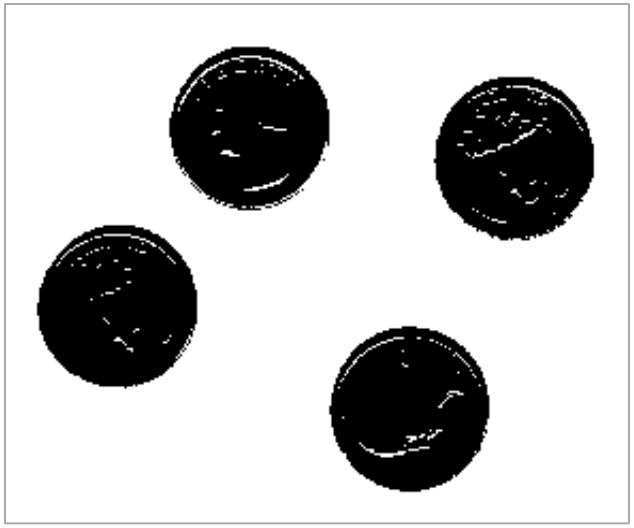
Edge detection

Blurring

Sharpening

Image and video processing - concept (Cont.)

Segmentation →

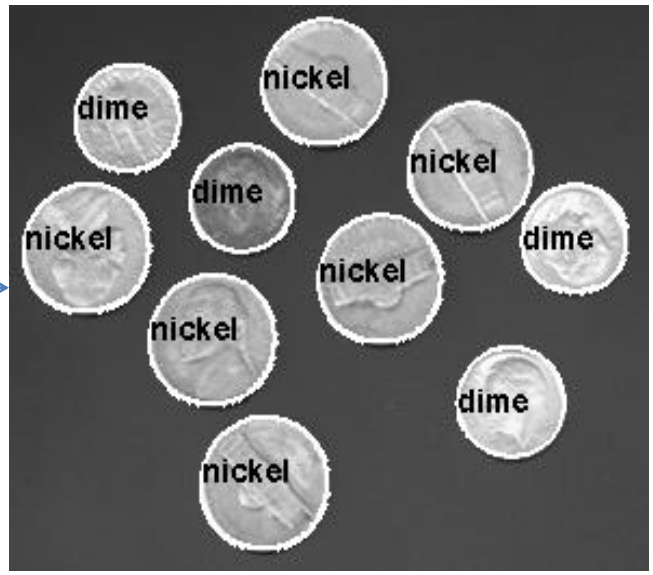


Noise removal →

Image and video processing - concept (Cont.)

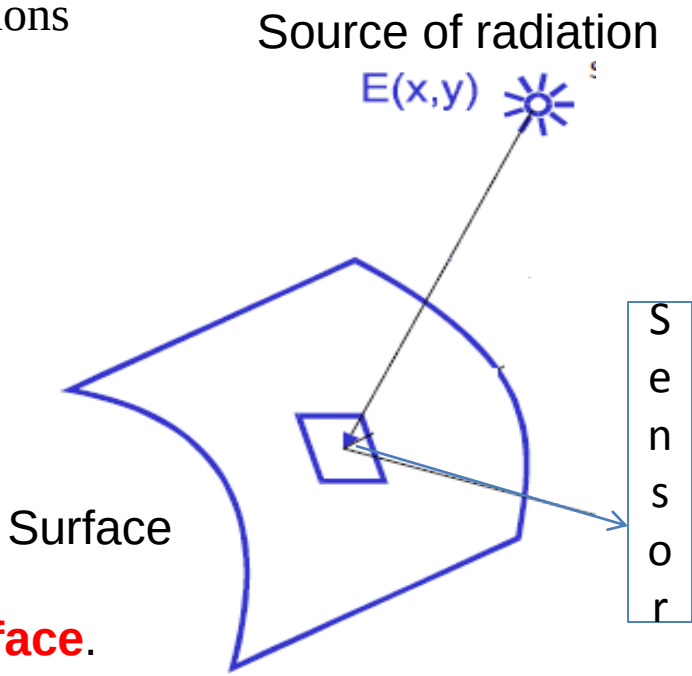
Lossy Image compression

Object recognition



Energy source for image

- ✓ Images can by created synthetically by computers or taken by a sensor.
- ✓ In capturing images by a sensor, usually we will assume that source of radiation is within visible light frequency domain but it could be other electromagnetic frequency bands (gamma rays, X rays, Ultraviolet ,microwave, radio waves, acoustic, or ultrasound)
  - ❖ Gamma rays : medicine and astronomy
  - ❖ X rays: Medical and industrial application
  - ❖ Ultraviolet: lithography, industrial application
  - ❖ Visible: most familiar, personal and industrial applications
  - ❖ Infrared : used in conjunction with the visible range
  - ❖ Microwaves : radar
  - ❖ Radio waves : medicine (MRI) and astronomy
  - ❖ Acoustic: medical and geological exploration
  - ❖ Ultrasonic: medical and manufacturing



Sensors receive signals reflected from the surface.

ELEX 7815

- ✓ Instructor: Dr. **John Dian** (office SW1-3071) local 8219
- ✓ Prerequisites:
  - ❖ ELEX 7710 :Signal Theory and Processing
- ✓ Text:
  - ❖ Class notes will be posted on the shared drive
  - ❖ Main Reference
    - ▪ Rafael C. Gonzalez, Richard E. Woods, ``Digital Image Processing", 3rd Edition, Prentice Hall; 2007
  - ❖ Second Reference
    - ▪ O. Marques, "Practical Image and Video Processing Using MATLAB", Wiley, 2011 (Recommended)
- ✓ Evaluation:

Laboratory:	20%	Midterm exam(s)	30%
Final Exam:	40%	Assignments	10%

Image acquisition & presentation

Visual perception

- ✓ In application whose goal is to enhance the quality of the image or video for human perception, It is important to understand the capabilities and limitations of human visual system (HVS)
- ✓ HVS is composed of eye as a visual sensor, brain as a processing and optical nerve as a transmission path between the eye and the brain.

Eye sections

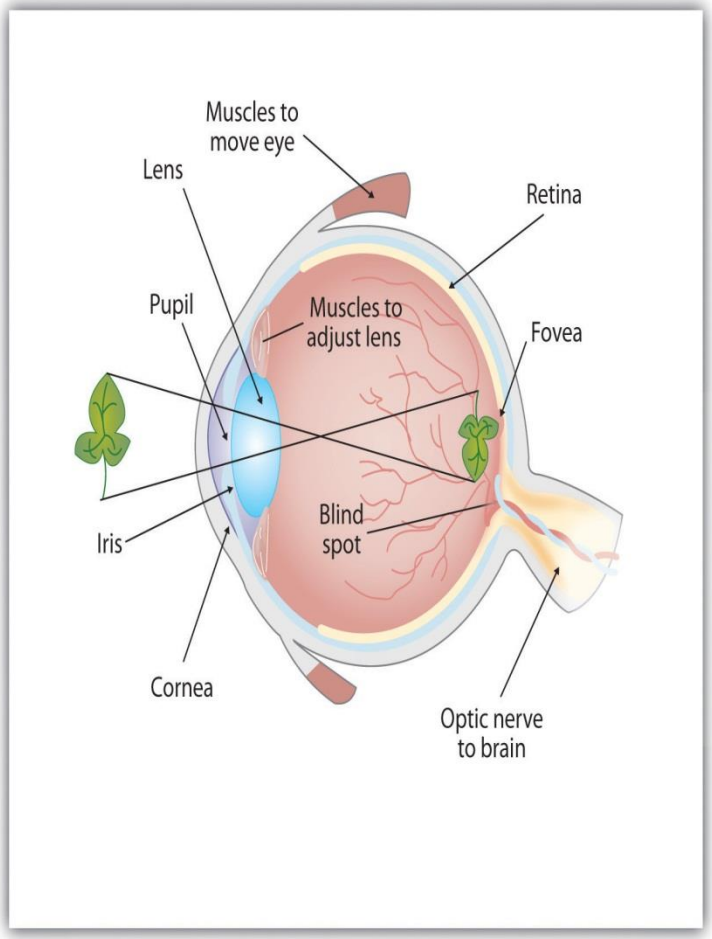
Lens : is used to focus an image. There are some muscles for controlling the movement of the lens

Iris diaphragm: is used to control the amount of light that enters the eye

Pupil: the central opening of the iris is called pupil and its diameter varies inversely proportional with to the amount of incoming light.

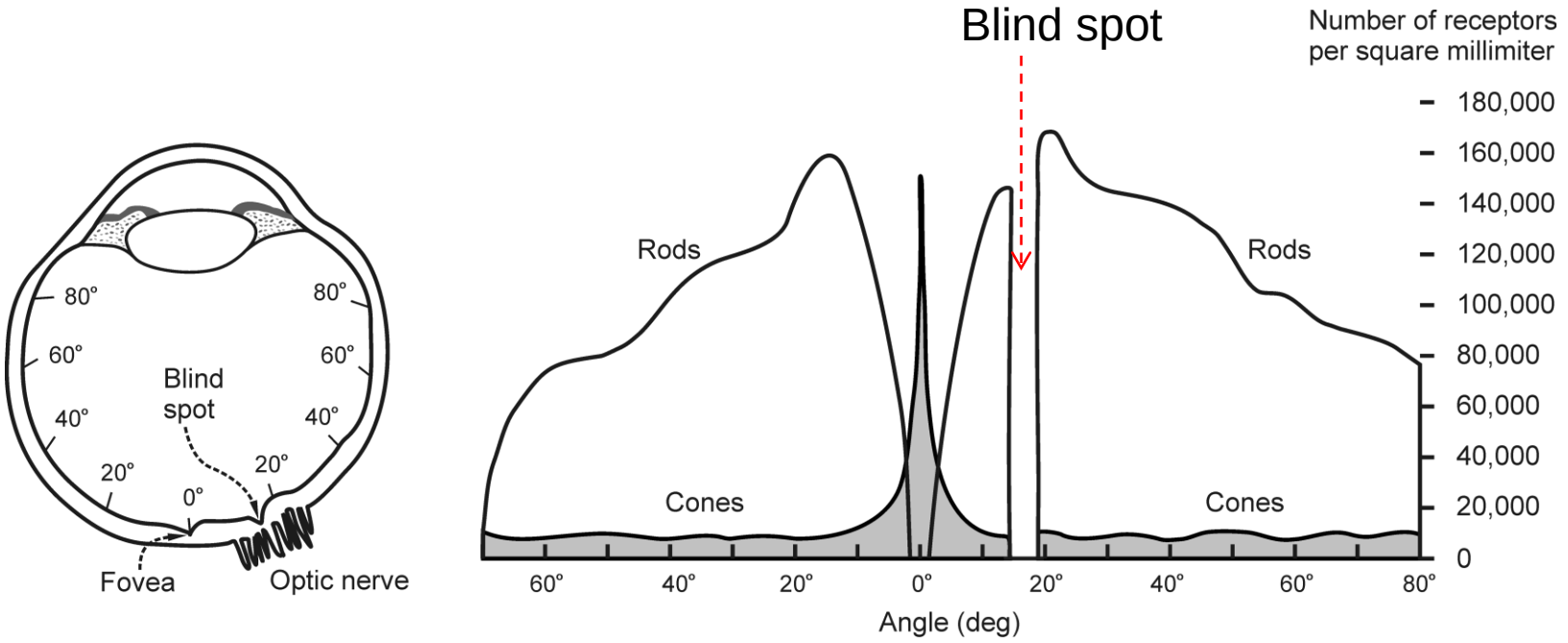
Retina: innermost membrane of the eye which is coated with photosensitive receptors called cones and rods.

Fovea: center of retina is called fovea and its covered by cones (6-7 millions) which are very sensitive to color. The rest of retina is covered by rods (75-150 millions) which gives a general picture of field of view.



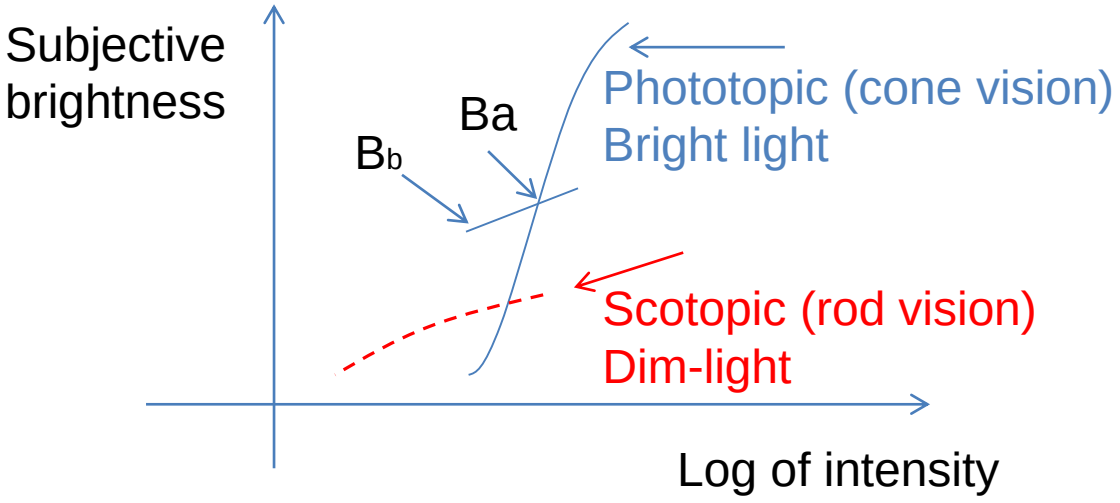
Vision sensors in HVS system

- ✓ Blind spot is an area with no vision sensors (cones and rods). If receptor density is measured in degrees from the center of fovea (0'), blind spot is 20' off the center.
  - ❖ Cone vision is called **photonic** or bright light vision.
  - ❖ Rod vision is called **scotopic** or dim-light vision



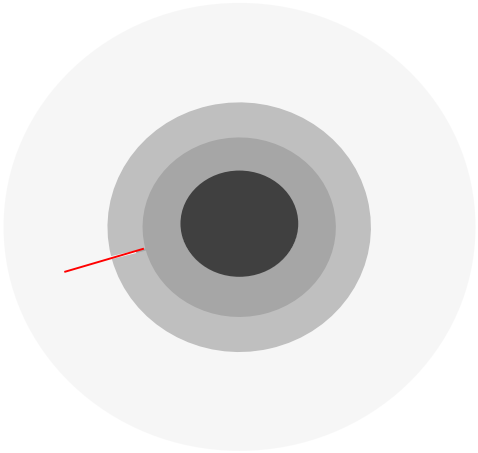
Brightness perception

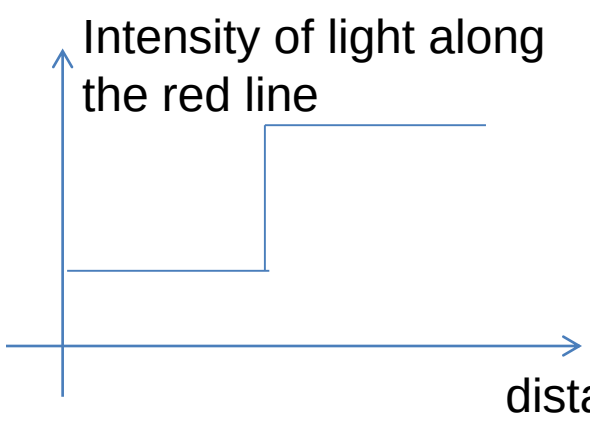
- ✓ Perceived brightness is a function of intensity and It is proportional to the logarithm of the light intensity incident on the eye.
- ✓ The range for brightness perception is large, but our visual system can't operate over this large range of illumination levels simultaneously.
- ✓ At any condition, the perceived brightness is called brightness adaptation which is small range as compared to range of illumination levels.
    - For instance, If eye is adopted to Ba, then the adaption level is the short intersecting curve having a level B_b at the end.

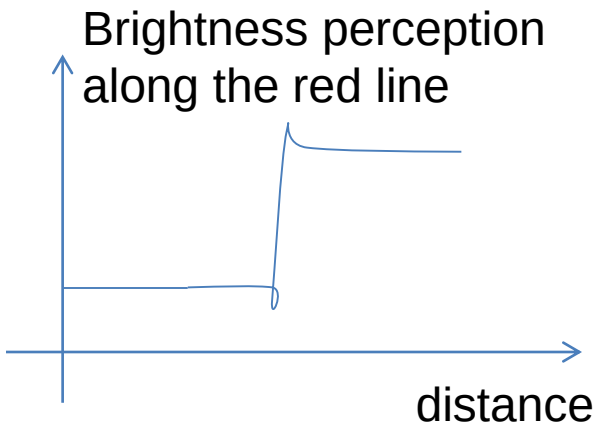


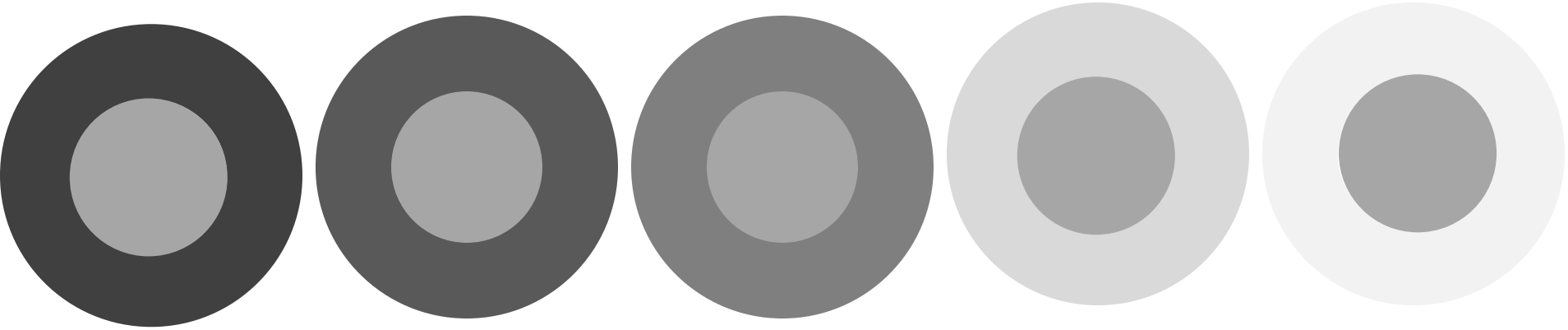
Brightness perception (Cont.)

- ✓ There are phenomena that shows that perceived brightness are not a simple function of intensity.
  - The visual system tends to undershoot or overshoot around the boundary.
  - The perceived brightness of an area also depends on the contrast between the area and its surroundings





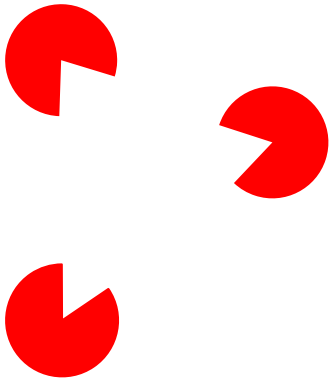




Optical illusion

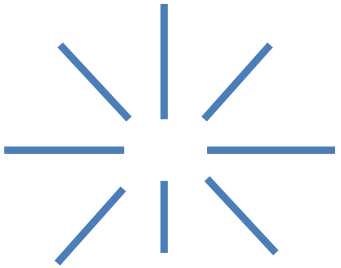
- ✓ Optical illusion: eye wrongly perceives geometrical information, size of an object or fills in non-existing information

Illusion of lines of Triangle.
Eye fills in the outline of the triangle.



Perceiving wrong size.
Both lines have the same size

eye wrongly perceives geometrical information. All horizontal lines are parallel.



Illusion of a rectangle.
Eye fills in the complete rectangle.

Optical illusion (Cont.)

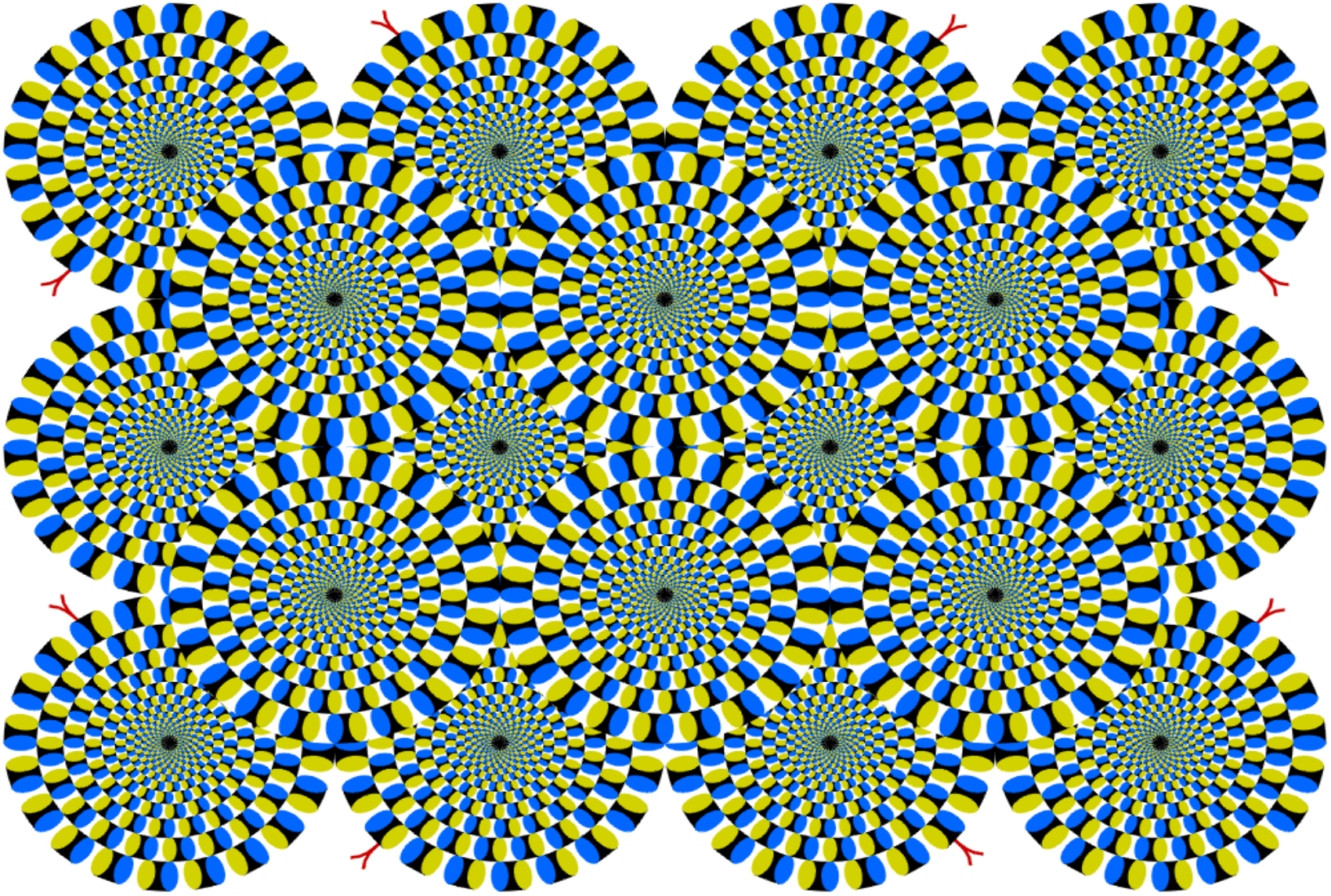


Image sensing and acquisition

- ✓ For natural images we need a light source . The energy of a light source having wavelength of λ at the location of (x', y', z') is $E(x', y', z', \lambda)$
- ✓ Each point such as (x', y', z') in the scene has a reflectivity function to a specific wavelength. We can define the reflectivity function as $r(x', y', z', \lambda)$
- ✓ Light reflects from a point and the reflected light is captured by an imaging sensor . The energy of light received by the sensor can be defined as

$$C(x', y', z', \lambda) = E(x', y', z', \lambda)\ r(x', y', z', \lambda)$$

- ✓ Each 3-D coordinates in real world is projected to a 2-D coordinate inside the sensor. The projection (P) can be considered as

$$C_p(x, y, \lambda) = P[C(x', y', z', \lambda)]$$

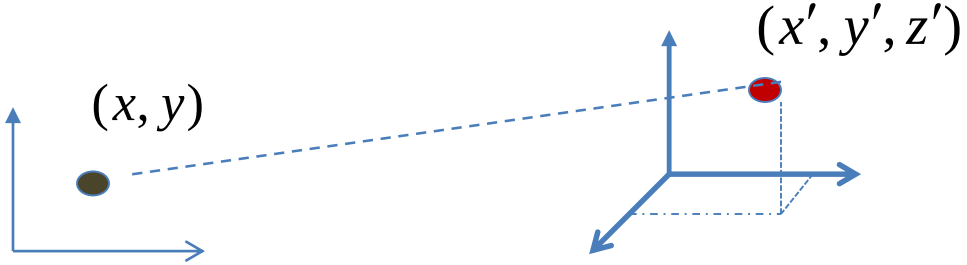


Image sensing and acquisition (cont.)

- ✓ Perspective Projection is a type of projection which has been widely used. HVS and camera sensor use this type of projection. In this type of projection the objects closer to the sensor appear larger in size.
- ✓ For each sensor, we can define a *sensitivity function* $V(\lambda)$. *This function* determines how sensitive it is in capturing the range of *wavelengths* present in $C_p(x, y, \lambda)$
- ✓ The amount of reflected light that is captured at the camera coordinates (x, y) *can be calculated as*

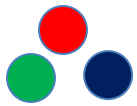
$$f(x, y) = \int C_p(x, y, \lambda)\, V(\lambda)\, d\lambda$$

- ✓ For a camera that captures color images, imagine that it has *three sensors* at each location (x, y) with sensitivity functions tuned to the wavelengths of the colors red, green and blue, outputting *three* image functions:

$$f_R(x, y) = \int C_p(x, y, \lambda) V_R(\lambda)\, d\lambda$$

$$f_G(x, y) = \int C_p(x, y, \lambda) V_G(\lambda)\, d\lambda$$

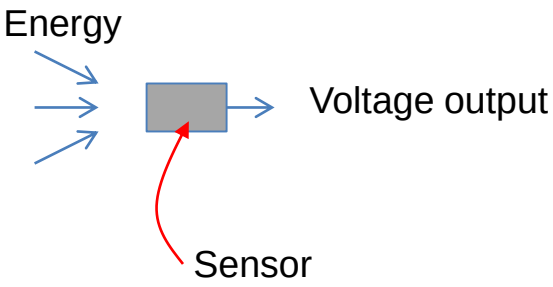
$$f_B(x, y) = \int C_p(x, y, \lambda) V_B(\lambda)\, d\lambda$$



$C_p(x, y, \lambda)$ ←

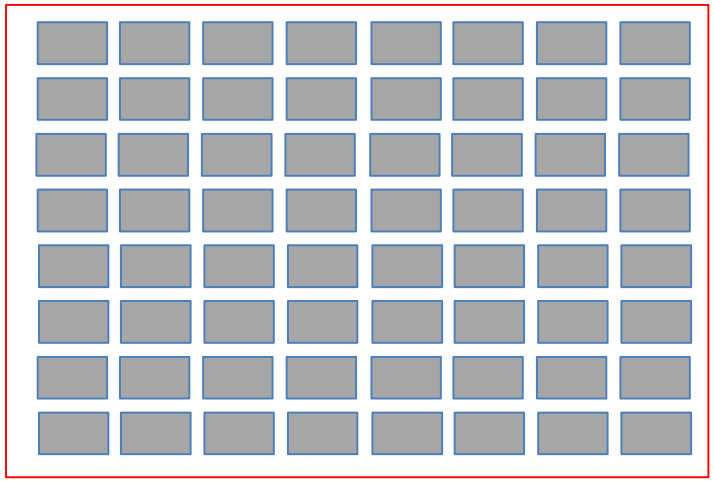
Image sensing and acquisition

- ✓ Incoming energy reflected from a surface is captured by a sensor element projected to a 2-D plane and depending on the sensor sensitivity function generates a voltage
  - ❖ The sensor that is wildly used as imaging sensor in industry is CCD (charge-coupled device).
  - ❖ An image is projected through a lens onto CCD which has capacitor accumulate an electric charge proportional to the light intensity. The charge is then is converted to voltage.



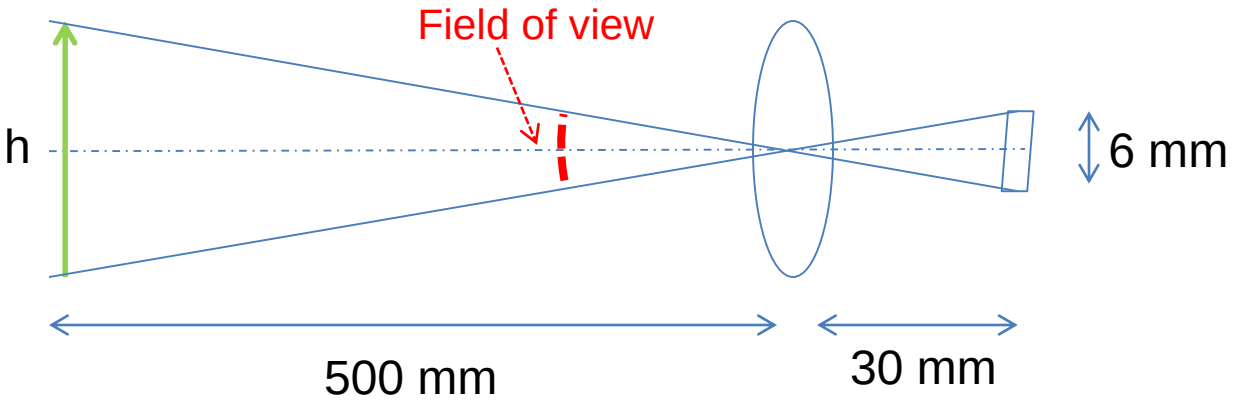
- ✓ To capture an image a collections of sensors are arranged in various form such as to capture images
  - ❖ Line of images sensor
    - ▪ 1-D CCD, used in fax machines

  - ❖ Array of images sensor
    - ▪ (2-D CCDs, used often in digital camera)



CCD - Example

- ✓ In a camera, an image is projected through a lens onto a CCD array. The lens has a fixed focal length of 30mm. The image is located 500m from the lens. The array has a size of 6x6 mm and has 4000x4000 sensor elements. How many sensor elements per mm will this camera is able to detect?



$$\frac{6}{30} = \frac{h}{500} \Rightarrow h = 100$$

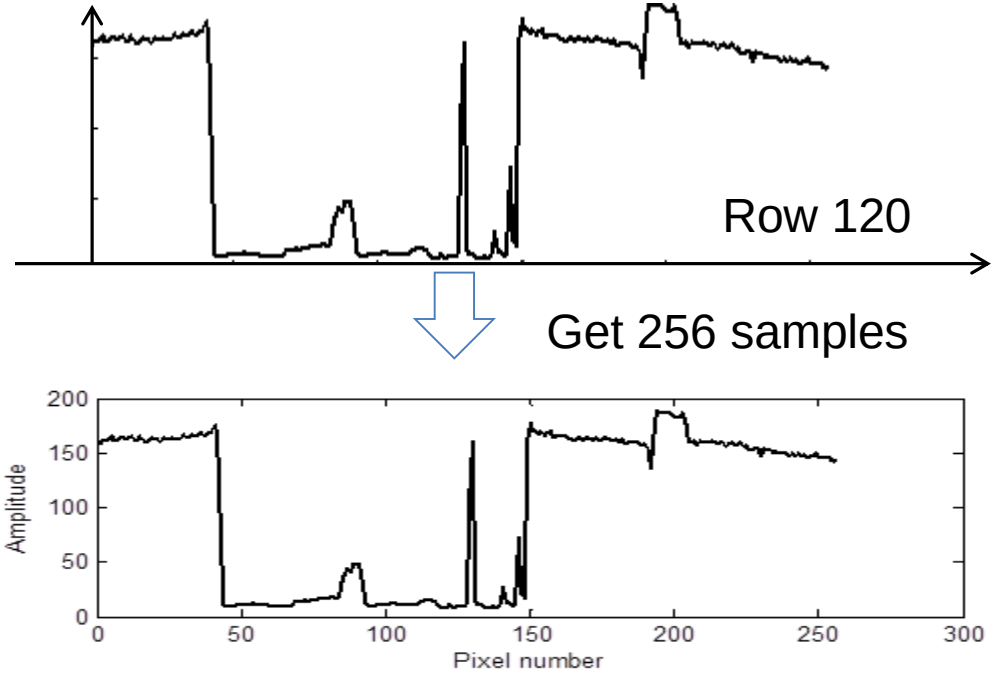
  - ❖ The resolution of one line is 4000/100= 40 elements per mm.

- ✓ Image resolution can be measured in various ways. Basically, resolution quantifies how close lines can be to each other and still be visibly resolved. Resolution units can be tied to physical sizes such as lines per mm. Line pairs are often used instead of lines; a line pair comprises a dark line and an adjacent light line. How many line pairs per mm will this camera is able to detect?
  - ❖ To find the line pairs per mm, the number of elements per mm must be divided by 2.
  - ❖ So 40/2= 20 LP per mm.

Image sensing and acquisition

- ✓ The image function $f(x, y)$ varies in a continuum given by the respective intervals. Digital computers cannot process parameters/functions that vary in a continuum. To discretize, we need the following tasks:
  - ❖ Sampling
  - ❖ Quantization.
- ✓ To sample, the intensity should be read in equally spaced intervals along a given row. The intensity values resulted from sampling process has a continuous ranges of intensity values. The quantization process changes the sampled values in discrete quantities.
- ✓

Input image



Digital image representation

$$f(x,y)=\begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & f(2,3) & \cdots & f(2,N) \\ \vdots & \vdots & \vdots & & \vdots \\ & & & f(x-1,y-1)\ \ f(x-1,y)\ \ f(x+1,y) & \\ & & & f(x,y-1)\ \ f(x,y)\ \ f(x,y+1) & \\ & & & f(x+1,y-1)\ \ f(x+1,y)\ \ f(x+1,y+1) & \\ \vdots & \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & f(M,3) & & f(M,N) \end{bmatrix}$$

- ✓ A pixel M located at (x,y) has
  - ❖ 4 horizontal and vertical neighbors called $N_4(T)$
  - ❖ 4 diagonal neighbors called $N_D(T)$
  - ❖ Together 4 horizontal and vertical as well as 4 diagonal neighbors are called $N_8(T)$

$$N_8(T) = N_4(T) + N_D(T)$$

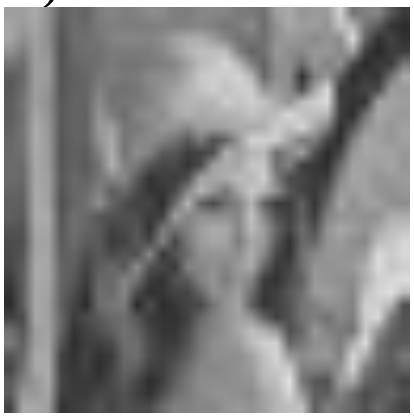
Spatial resolution (based on sampling)

- ✓ Spatial *resolution*
  - ❖is determined by how sampling was carried out and simply refers to the smallest discernible detail in an image
    - ❖ Vision specialists will often talk about pixel size
    - ❖ Graphic designers will talk about *dots per inch* (DPI)

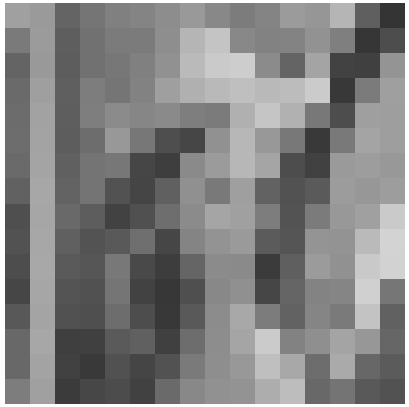
512X512

128X128

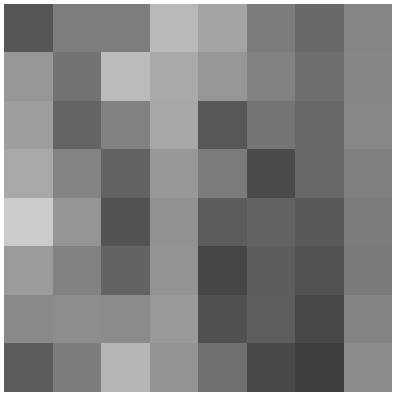
64X64



32X32



16X16



8X8

Intensity resolution (based on quantization)

- ✓ Is determined by how many intensity levels exist in an image or the number of bits used to quantize an image

Image quantized to 8 bits

Image quantized to 7 bits

Image quantized to 5 bits

Image quantized to 3 bits

Image quantized to 2 bits

Image quantized to 1 bits

Intensity Transformation
(point processing)

Point Processing

- ✓ The reason for point processing is
  - ❖ To enhance the image quality for human viewing
  - ❖ To modify the image in such a way that is more suitable for further processing and feature detection

- ✓ In point processing, the value of a pixel **r** in original image is changed to a new value **s** using a transformation which can be shown as:

$$s = T\,[r]$$

- ✓ Examples of the transformation that can be used in point processing are:
  - ❖ Linear point processing $s = c.\,r + b$
    - Negative (contrast reverse) c=-1, b=255
  - ❖ Power law (Gamma transformation) $s = c.\,r^{\gamma}$
  - ❖ Log transform $s = c.\,\log(1+r)$
  - ❖ Piecewise linear transformation

$$\begin{cases} s = c_1 r + b_1 & 0 < r < a_1 \\ \vdots & \\ s = c_n r + b_n & r > a_n \end{cases}$$

  - ❖ Contrast stretching, amplitude scaling $s = 1/(1+m/r)^{e}$
    - Thresholding

$$\begin{cases} s = 0 & r \le r_m \\ s = 255 & r > r_m \end{cases}$$

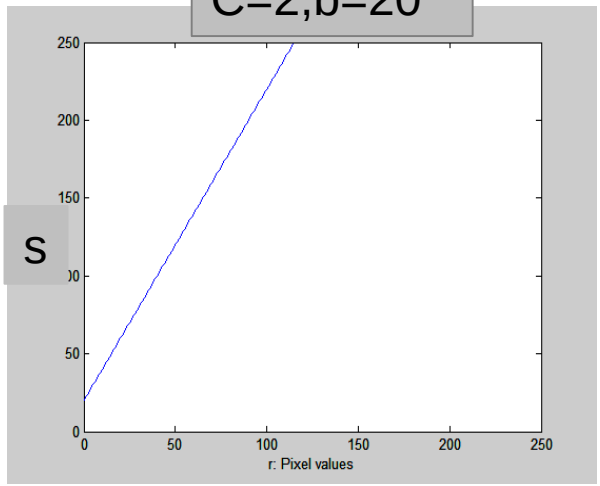
    - Auto contrast adjusting $s = \dfrac{L-1}{r_{\max} - r_{\min}}(r - r_{\min})$, L is the max. of intensity level

Linear Point Processing

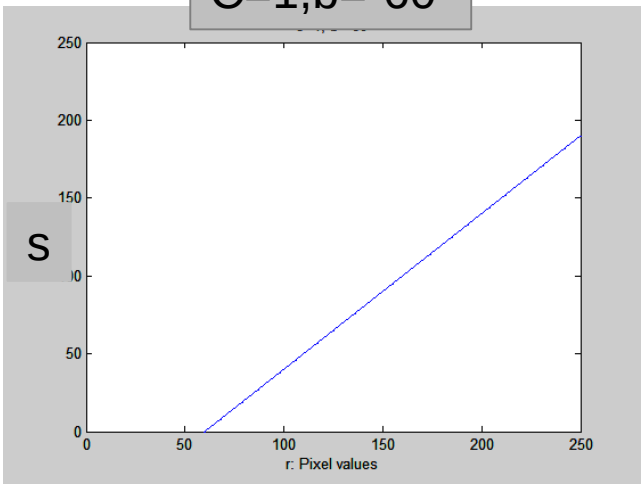
- ✓ Linear point processing

$$s = T[r] \Rightarrow s = c.r + b$$

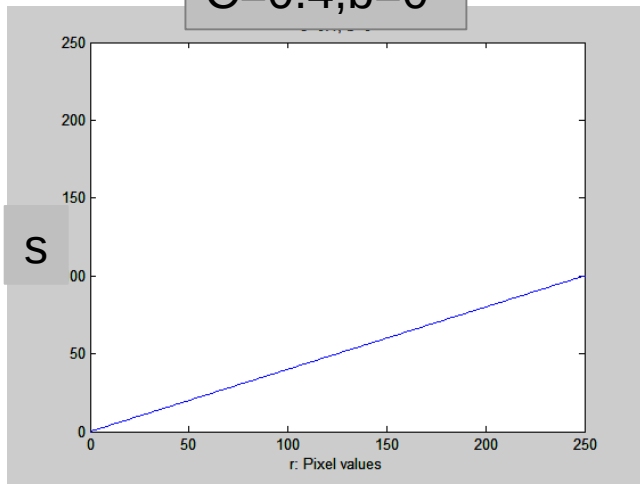
C=2,b=20



C=1,b=-60



C=0.4,b=0



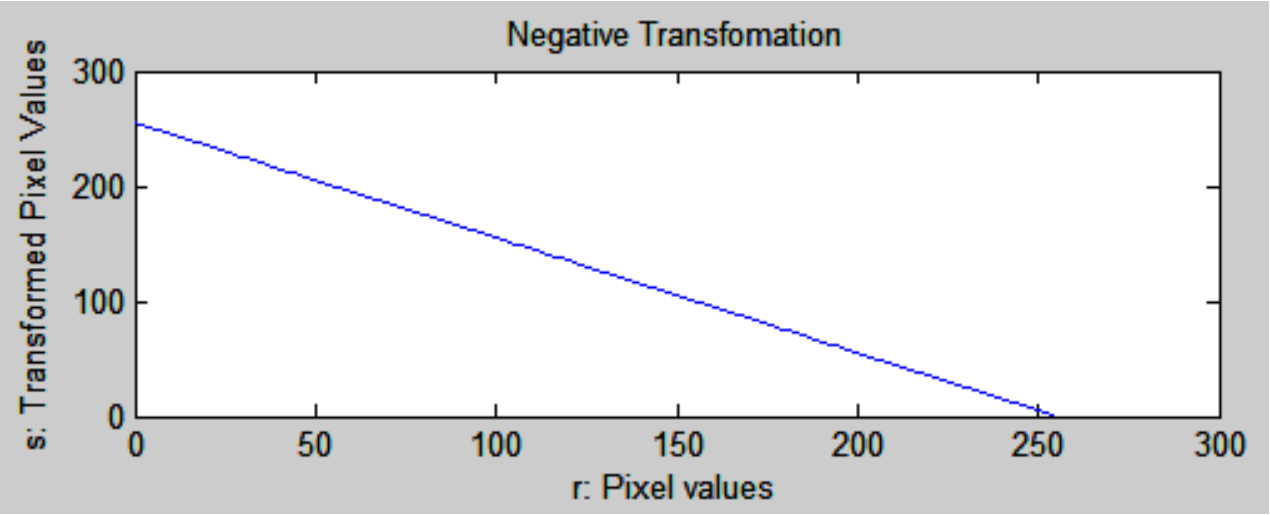
brightening

darkening

Contrast reduction

Negative (contrast reverse)

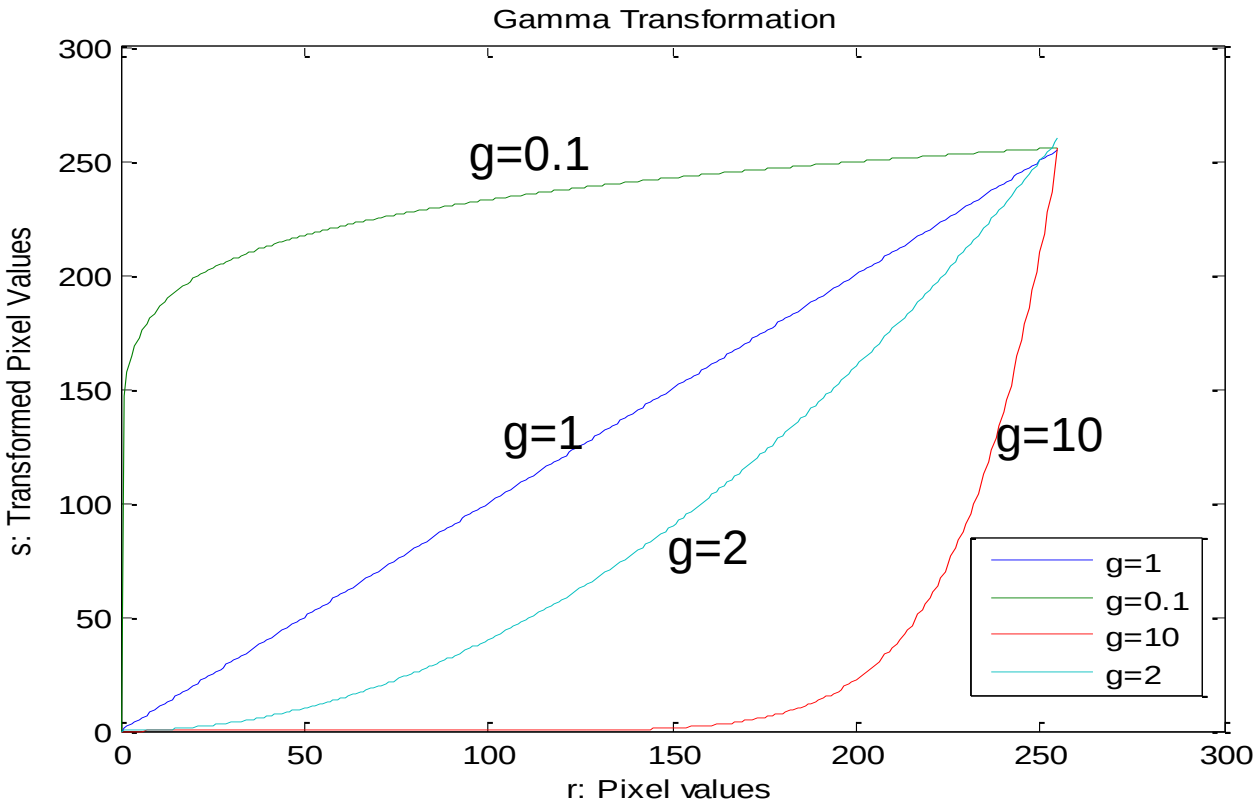
✓ Special case of linear point processing c= -1, b=255 -> $s = -r + 255$



Gamma Transformation

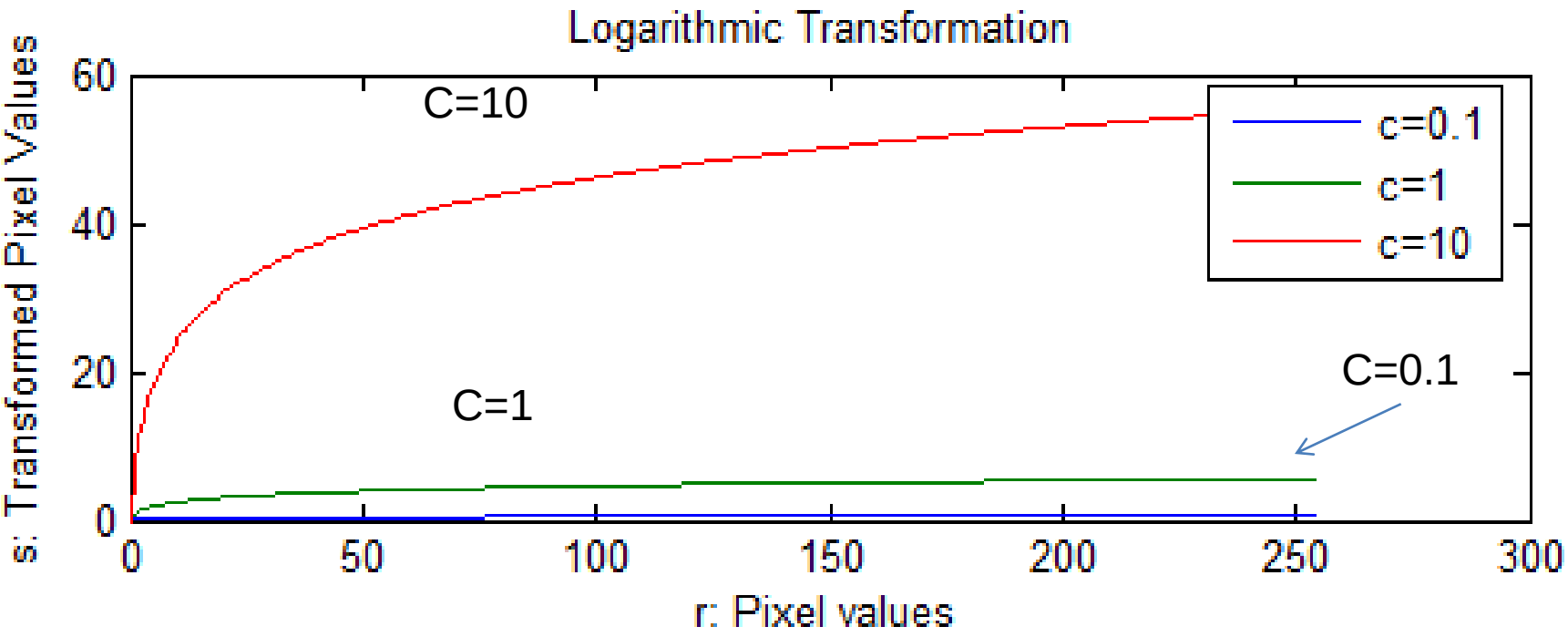
- ✓ Applying gamma transformation to an input image with gamma less than 1 produces a brighter image; while with gamma value greater than 1 produces darker image.

$$s = c.\, r^{\gamma}$$



Log Transformation

- ✓ It is a non-linear transformation $s = c.\ \log(1+r)$
- ✓ It is useful when we like to compress, or expand the dynamic range of pixel values in an image
  - ❖ It maps a narrow range of low intensity input levels into a wider range of output levels
  - ❖ Inverse log also can be used in which $s = \exp(r/c) - 1$
  - ❖ Consider that the values of an 2-D matrix are in range of [0, 20000], but most of them are close to zero. if you scale linearly the values to 8-bit [0 256] range, It would be hard to see anything. However, using a log transform , the dynamic range can be compressed.



Piecewise linear Transformation

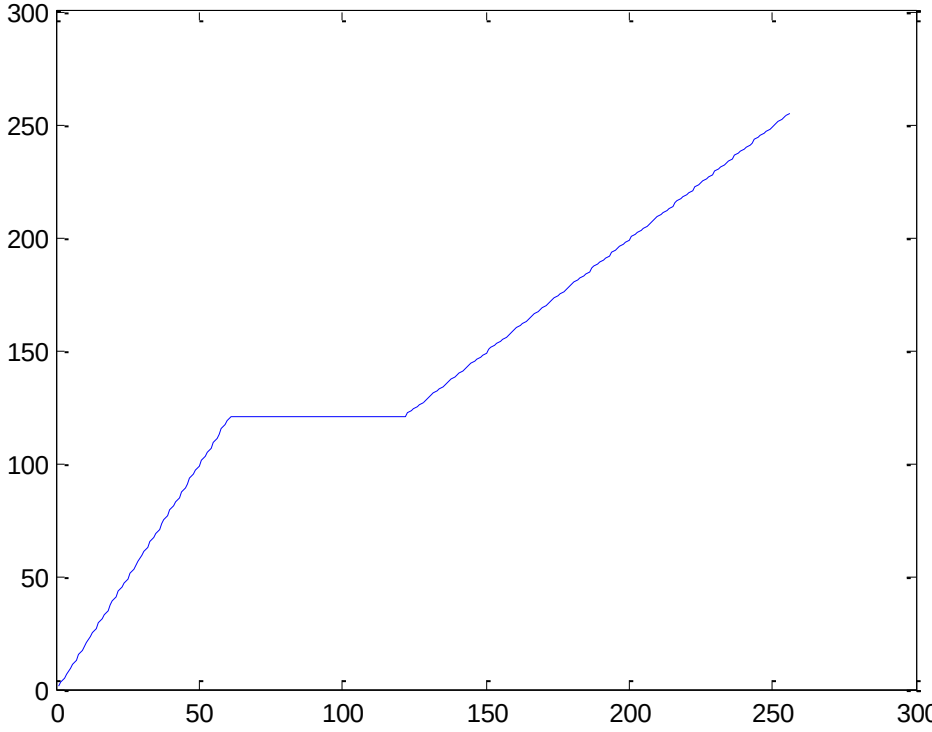
- ✓ It consists of several intervals of intensity values. For each interval, the transformation is defined as a linear equation.

$$\begin{cases} s = c_1 r + b_1 & 0 < r < a_1 \\ \vdots & \\ s = c_n r + b_n & r > a_n \end{cases}$$

- ✓ This shows that the different range of pixel intensity in the input image needs to be treated differently.
- ✓ **Intlut** function in MATLAB can be used to build the piecewise linear transformation
- ✓ Example

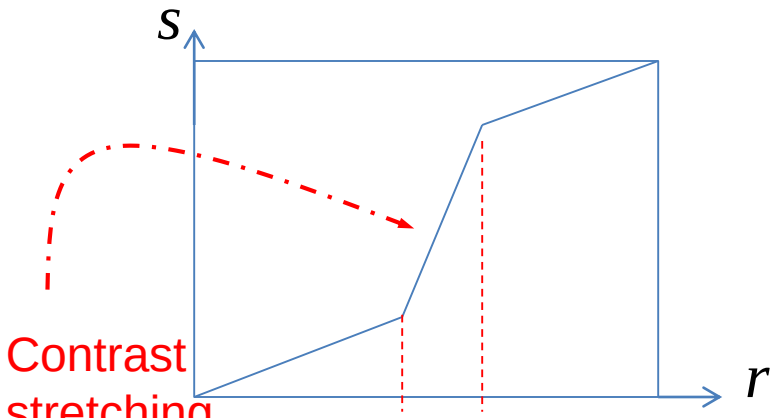
$$\begin{cases} s = 2r + 1 & 0 < r <= 60 \\ s = 121 & 60 < r <= 121 \\ s = r - 1 & r > 121 \end{cases}$$

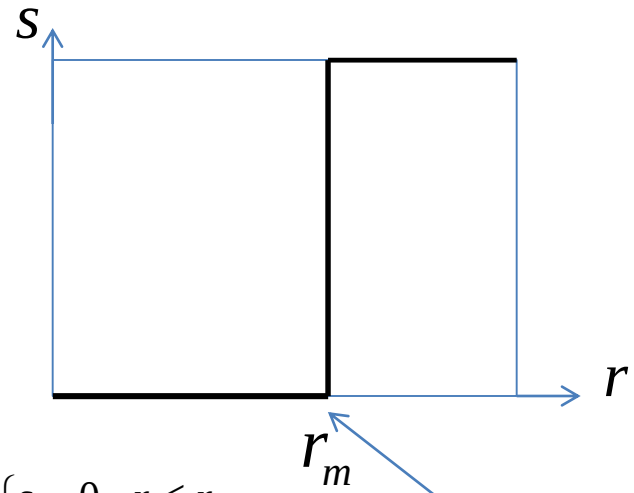
```
L=uint8(zeros([1 256]));
L(1:61)=2*(0:60)+1;
L(62:121)=121;
L(122:256)=(122:256)-1;
Transformed_image=intlut(input_image,L)
```



Contrast stretching, Thresholding & intensity slicing

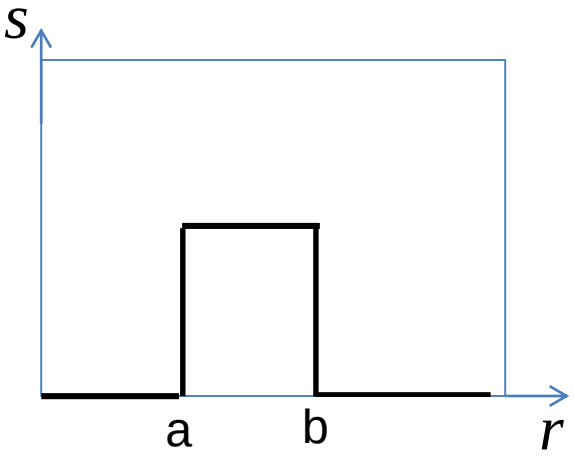
- ✓ Contrast stretching is a process that expands the range of intensity values in low contrast image

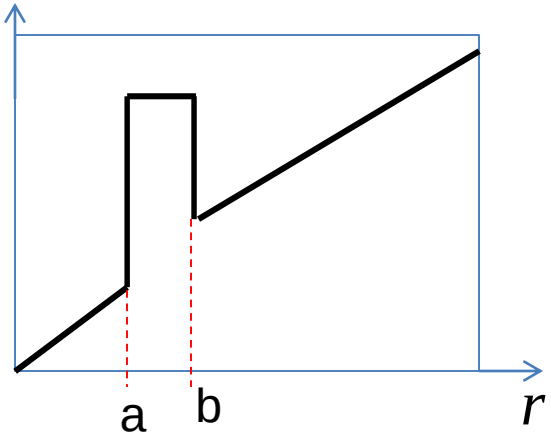




$$\begin{cases} s = 0 & r \le r_m \\ s = 255 & r > r_m \end{cases}$$

Thresholding



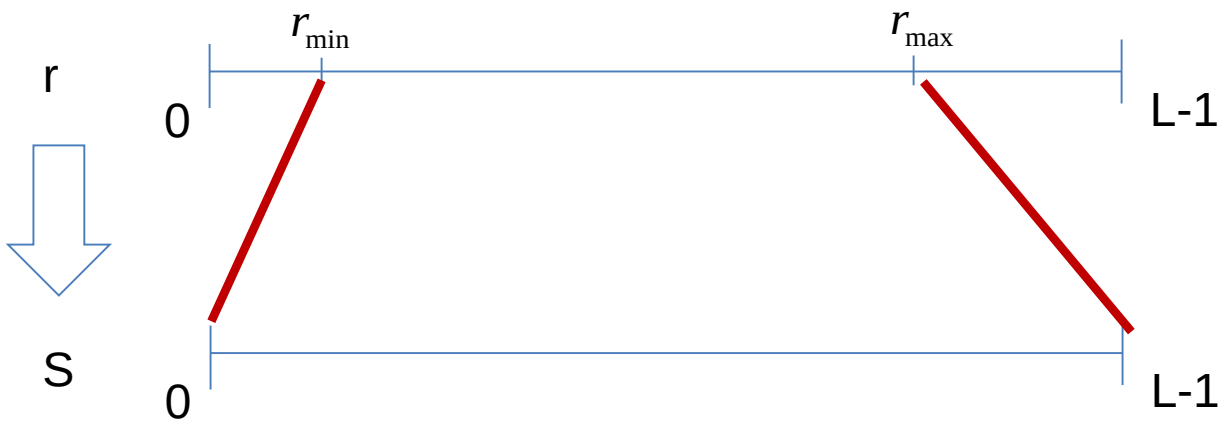


intensity slicing

Auto contrast adjusting

- ✓ This transformation maps the darkest pixel value in the original input image r_{min} to 0 and the brightest pixel value in the original image r_{max} to 255 and linearly change the values of the other pixels in the input image.
- ✓ If L-1 be the highest gray value in the input image (L is 256 for uint8 class), then the Auto contrast adjusting function can be defined as

$$s = \frac{L-1}{r_{max} - r_{min}}(r - r_{min})$$



- ✓ If **r** is the intensity of the original image **f(x,y)** , first subtract all the intensities such as **r** from the min value of **f(x,y)** , let say r_{min} , this produce image **g(x,y)** which intensity **r** in **g(x,y)** is changed to s_1 , where $s_1 = r - r_{min}$
- ✓ Then divide **g(x,y)** by its maximum intensity value, $r_{max} - r_{min}$, to find an the intensities in range [0 1]. Then multiply the result to L-1 to find a function with values in the range of [0, L-1]

Geometric Transformation and Interpolation

Geometric transformation

- ✓ It is a spatial transformation of the coordinates of pixels

$$(x', y') = T(x, y)$$

Pixel coordinates in the transformed image

Pixel coordinates in the original image

- ✓ So, the pixel in location (x, y) must be transformed to location (x', y')
- ✓ Affine transformation is a type of geometric transformation that is widely used. This transformation can be used to scale(resize), rotate, translate or shear an image by using correct values for transformation operator **T**
- ✓ The affine transform can be defined as

$$[x' \;\; y' \;\; 1] = [x \;\; y \;\; 1] \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix} \quad T$$

Affine Transformation Example

$$T = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling

$$T = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{bmatrix}$$

Translation

$$T = \begin{bmatrix} 1 & S_h & 0 \\ S_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear

✓ In IPT, we can use two functions to implement affine transformation

- 1- **maketform** function to define an affine transformation

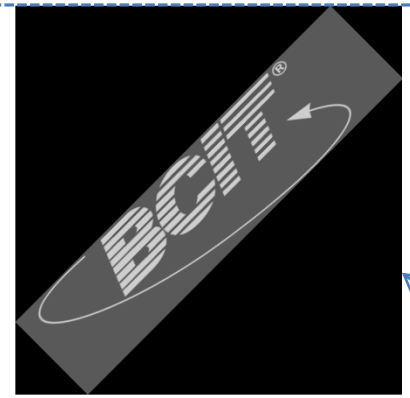
  T=**maketform**('affine', [sx 0 0 ;0 sy 0;0 0 1]');

- 2- **imtransform** function to apply the transformation to the original image

  Transformed_image=**imtransform(**original_image,T);

Original

Translated



Rotated

Sheared

Geometric transformation process

- ✓ Using the transformation, the coordinates of a pixel goes to new coordinates, but what pixel intensity should be used when
  - ❖ Several coordinates to the same coordinate in the transformed image
  - ❖ No pixel gets transformed to some coordinates of the transformed image

- ✓ There are two ways for geometric transformation processing
  - ❖ Forward mapping : this is based on what we explained so far. The question raised above somehow needs to be answered.
  - ❖ Inverse mapping : In this case for any pixel in the transformed image, we calculate the corresponding location in the original image using

$$[x, y, 1] = [x', y', 1]\, T^{-1}$$

x,y might not be an integer number. We need then to interpolate among the nearest pixels in the original image to find the value of the pixel in the transformed image.

Inverse mapping is more efficient . MATLAB uses inverse mapping

Image registration

- ✓ Image registration is the process of aligning an input image against a reference image.
  - ❖ The input and reference image are usually taken from the same scene, in different time or with different devices having different resolutions. The images might be taken from different distances and with different viewing angles
  - ❖ In this process usually the two images (input and reference) are known and we are trying to estimate a transformation to align the input image against the reference frame. Applying the transformation to input image, will produce a new image that should be close to reference image.
- ✓ One way for image registration is done by using some control points in the scene. If theses control points in two images cab be detected, then using a using bilinear approximation, a transformation for these two images can be found.

(x, y) *control points of reference image*

(x', y') *control points of input image*

$$x = a_1 x' + b_1 y' + c_1 x' y' + d_1$$

$$y = a_2 x' + b_2 y' + c_2 x' y' + d_2$$

  - ❖ In this case, we need 4 control points to find all the coefficients.
  - ❖ When the coefficients of the transform are found, use the transform to all the pixels in the input image to find a new image.
  - ❖ The new image is the aligned or register image

Image registration-Example

✓ Application:

detecting missing components on a PCB board in a assembly line

- ❖ Reference image →the correct PCB with all the correct components $R(x, y)$
- ❖ Input image→ image taken from the same product during the operation $I(x, y)$

✓ Register the image $I(x, y)$ against the reference image to align the image to reference image. Lets call the new image as $I'(x, y)$

✓ Subtract $I'(x, y)$ from $R(x, y)$ to find the difference image

$$d(x, y) = I'(x, y) - R(x, y)$$

✓ If all the pixel in $d(x, y)$ are zero or smaller than a threshold value , the product is OK, otherwise needs to be checked out.

$$d(x, y) \leq |T|$$

T can be positive or negative

Image Interpolation

✓**interpolation** is a method of constructing new data points within the range of a set of known data points. One application in image processing is resizing.

- ❖Resizing: image of size 512x512 pixels needs to be enlarged to size 765X765 pixels.

✓Image interpolation methods: Find where each pixel in the original image will be transformed in enlarged image.in the enlarged image, for the locations that no pixel from the original pixel exist, assign the intensity value of those pixels, based on

- ❖1- **nearest neighbor interpolation method**
  - ▪assign the values based on the intensity values of the nearest neighbors
- ❖2- **bilinear interpolation**
  - ▪assign the intensity values in location (x,y) based on the following formula. The value for coefficients a, b, c, d can be calculated by solving four equations using 4 nearest neighbors.

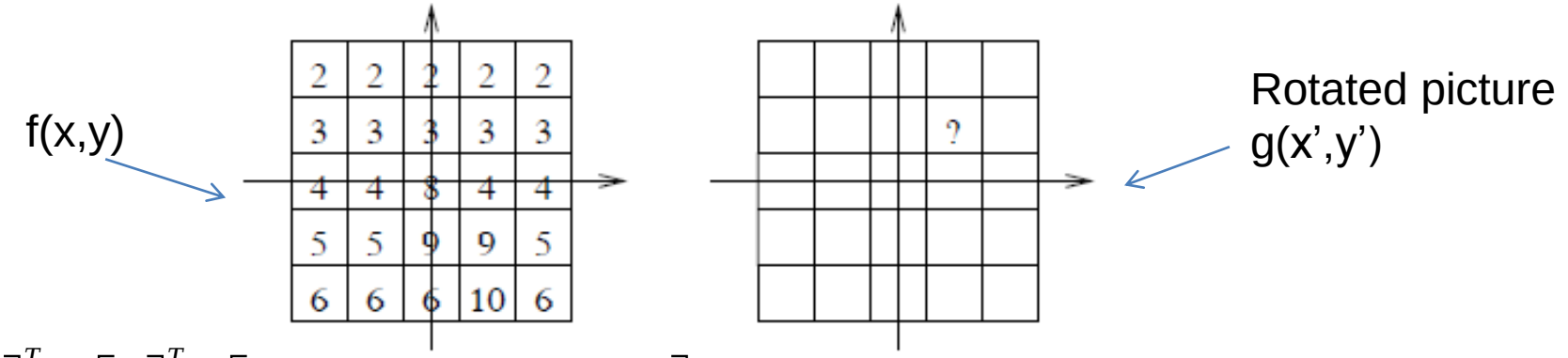
$$f(x, y) = ax + by + cxy + d$$

- ❖3- **Bicubic interpolation**
  - ▪ assign the intensity values in location (x,y) based on the following formula . The value for 16 coefficients of a_{ij} can be calculated by solving 16 equations using 16 nearest neighbors.

$$f(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i x^j$$

Interpolation – Example I

- ✓ The image $f(x, y)$ is going to be rotated 60◦ counterclockwise. Compute the pixel value of the pixel marked with a question mark (?). Use nearest neighbor interpolation.



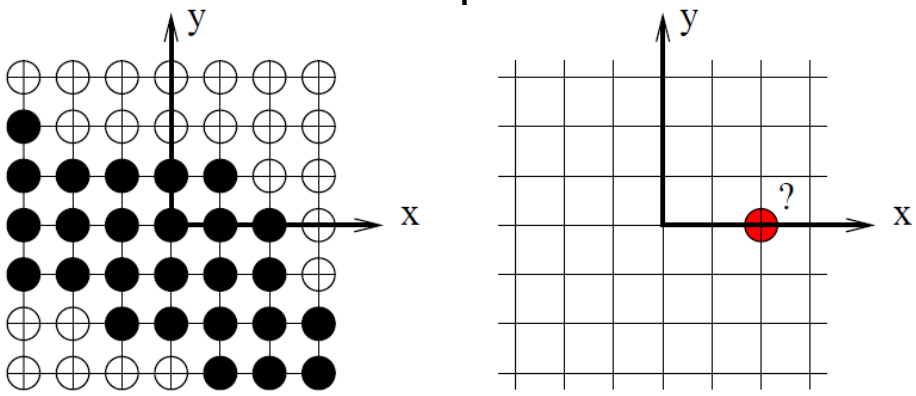
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^T \times \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^T = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T \times \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \theta = 60, \quad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1.366 \\ -0.366 \\ 1 \end{bmatrix} \Rightarrow g(1,1) = f(1,0) = 4$$

the pixel which is in marked with a question mark will get the value 4.

Interpolation – Example II

- ✓ The image below will be rotated 20◦ clockwise. Calculate the value at the point marked with a question mark. Use bi-linear interpolation. Assume that the white points have value 1 and that the black points have value 0.

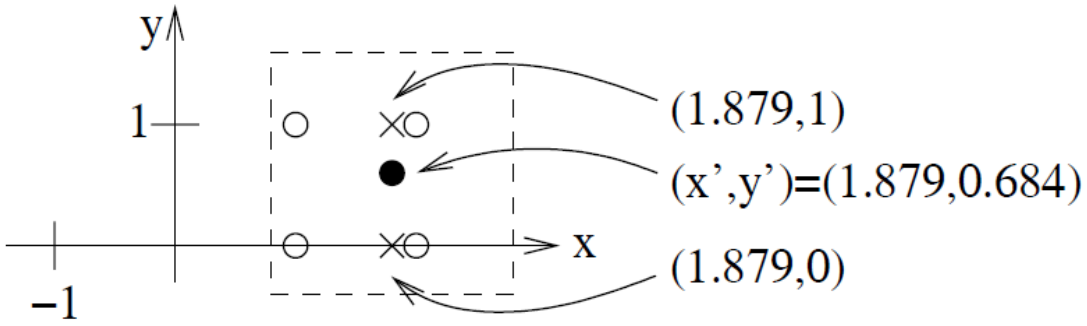


$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^T \times \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^T = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T \times \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- ✓ The point $(x', y') = (2, 0)$ and the angle $\theta = -20^\circ$ inserted in the formula above gives the coordinate $(x, y) \approx (1.879, 0.684)$. There is three ways to interpolate in 2D,
- ✓ 1) 1D interpolation in the x-direction followed by 1D interpolation in the y-direction
  2) 2D interpolation directly
- ✓ 3) solving $f(x, y) = ax + by + cxy + d$ equation using 4 neighbors to find the value of a,b,c,d and using f(x,y) to find the new value

Interpolation – Example II (cont.)

$$h(x)=\begin{cases}1-|x| & -1\le x\le 1\\ 0 & \text{else where}\end{cases}$$



✓ **Method 1)** 1D interpolation in the x-direction followed by 1D interpolation in the y-direction

$f(1.879, 1) = f(1, 1)\, h(-0.879) + f(2, 1)\, h(0.121) = (0)\ (0.121) + (1)\ (0.879) = 0.879,$

$f(1.879, 0) = f(1, 0)\, h(-0.879) + f(2, 0)\, h(0.121) = (0)\ (0.121) + (0)\ (0.879) = 0,$

$f(1.879, 0.684) = f(1.879, 0)\, h(-0.684) + f(1.879, 1)\, h(0.316) = (0)\ (0.316) + (0.879)\ (0.684) = 0.6012$

✓ **Method 2)** 2D interpolation directly ($h2D(x, y) = h(x)\, h(y)$)

$f(1.879, 0.684) = f(1, 1)\, h2D(-0.879, 0.316) + f(2, 1)\, h2D(0.121, 0.316) +$
$f(1, 0) h2D(-0.879, -0.684) + f(2, 0)\ h2D(0.121, -0.684)$

$= (0)\ (0.121)\ (0.684) + (1\ )\ (0.879)\ (0.684) + (0)\ (0.121)\ (0.316) + (0)\ (0.879)\ (0.316) = 0.6012$

✓ **Method 3)**

$$f(x,y)=ax+by+cxy+d$$
$$f(1,0)=0 \Rightarrow a+d=0$$
$$f(2,0)=0 \Rightarrow 2a+d=0 \qquad \Rightarrow a=0, b=-1, c=1, d=0$$
$$f(1,1)=0 \Rightarrow a+b+c+d=0 \qquad f(x,y)=-y+xy$$
$$f(2,1)=1 \Rightarrow 2a+b+2c+d=1 \qquad f(0.684,1.879)=0.6012$$

Assignment #1

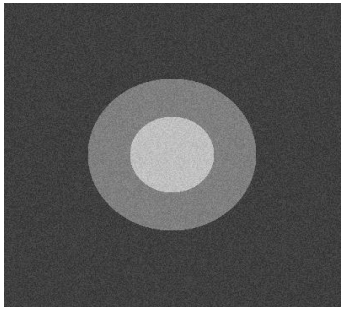
1) A video is composed of images that are 512x384 pixels, where each pixel is 3 bytes or 24 bits (consisting of red/green/blue color information). The video is 50 minutes and 40 seconds long. Images play at a rate of 23 frames per second. With no compression, how big would this file be? (Compressed file is 350 MB.)

2) we can treat the human fovea as a square sensor array of size 1.5 mm x 1.5 mm, containing about 337,000 cones. Also, the space between the cones is equal to width of the cones.

a) What is the field of view (in degrees) of the human fovea? Assuming the focal length of the eye is 17 mm.

b) A person observes a fishing boat out at sea that is approximately 1 mile away. The person claims that they can see a seagull following the boat. Is that possible? Justify your answer with quantitative estimates. Assume for simplicity that the size of the image of the seagull must cover at least two receptors (cones). Seagulls can range in size from 29 cm to 76 cm.

3) It is often useful to generate a synthetic image with known properties that can be used to test algorithms. Generate an image composed of two concentric circles as shown below. The inner circle should have a radius of 50 pixels and a mean value of 192. The outer circle should have a radius of 100 pixels and a mean value of 128. The background should have a mean value of 64. Add uniform random noise to each pixel in the range -16 .. +16. Save the image in "tif" format, and make sure the saved image looks correct.



4) You are given one image f1. You are asked to synthesize an image f2 that is f1 shifted 5pixels to the right, 2 pixels up, and then rotated by 30 degree in clockwise direction. Assuming f1 uses image coordinate (x,y), and f2 uses image coordinate (u,v).
a)Find the forward mapping function from f1 to f2. b) Find the inverse mapping function from f2 to f1.
C. Explain how you can generate f2 using step (a) or (b)?

Histogram Processing

Definition of histogram

- ✓ Histogram of a $M \times N$ image with intensity levels in the range of [0, L-1] is defined as

$$h(r_k) = n_K$$

Kth intensity level in the range of [0, L-1]

Number of pixels in the image with the intensity value equal to r_k

- ✓ The normalized histogram is defined as

$$p_r(r_k) = \frac{h(r_k)}{M \times N} = \frac{n_k}{M \times N} \qquad k = 0,1,2,\ldots,L-1$$

- ✓ Normalized histogram function
  - ❖ It is the probability of occurrence of intensity level r_k in an image
  - ❖ The sum of all components of a normalized histogram function is equal to 1

$$\sum_{k=0}^{L-1} p_r(r_k) = 1$$

Histogram processing

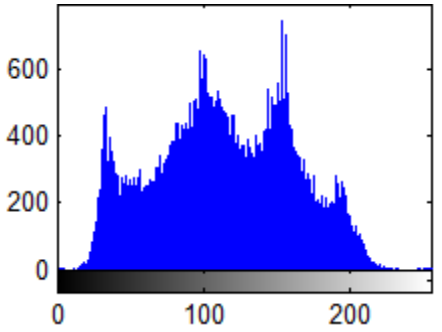
- ✓ Histograms provides useful image statistics that can be used in many image processing techniques such as image enhancement, image segmentation, image compression
- ✓ Histogram is a very popular tool in image processing. It has low computational complexity and can be performed in software or hardware (for real-time image processing)

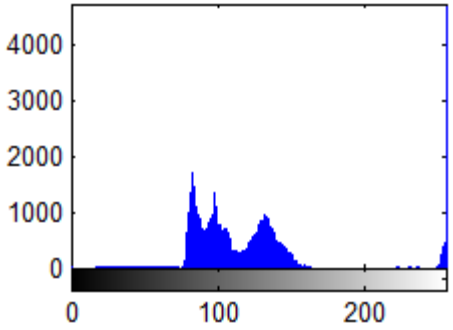
High contrast image

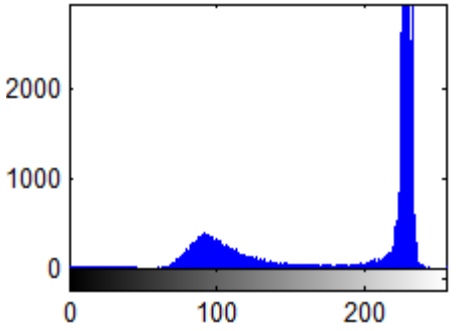
Low contrast image

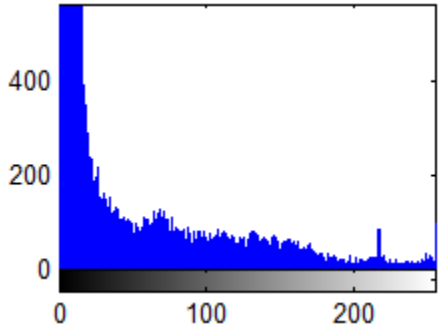
Bright image

Dark image









A note on Random variables

- ✓ If a random variable r with the probability density function (PDF) of $p_r(r)$ goes under transformation T(r), then the output is a random variable s=T(r) which its probability density function can be calculated as

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| \qquad (1)$$

- ✓ Assume the following T(r) function

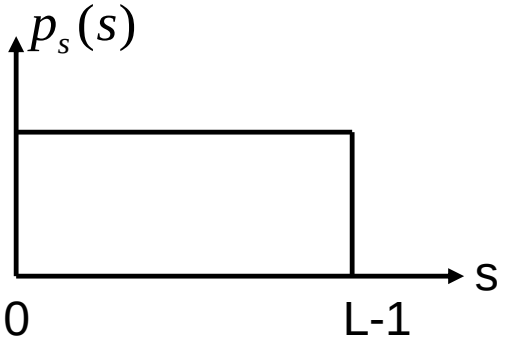
$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw \qquad (2)$$

- ✓ If we calculate the PDF of the transformed random variable s, $p_s(s)$, using (1) and (2)

$$\frac{ds}{dr} = \frac{d\ T(r)}{dr} = (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)\,dw\right] = (L-1)\ p_r(r)$$

$$\left|\frac{dr}{ds}\right| = \left|\frac{1}{(L-1)\ p_r(r)}\right|$$

$$p_s(s) = p_r(r)\left|\frac{1}{(L-1)\ p_r(r)}\right| = \frac{1}{L-1}$$



The relationship between random variable and an image

- ✓ The values of pixels within an image ,**r,** can be viewed as random variables. If all the pixel in the image go under a transformation such as T(r), the pixel values of the transformed image ,**s**, are also random variables.
- ✓ Therefore, we can use the transformation function, $T(r) = (L-1)\int_0^r p_r(w)\,dw$ to produce an image with flat histogram. For discrete values of an image, we deal with histograms instead of probability density functions. So, T(r) can be defined as

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{M \times N}\sum_{j=0}^{k} n_j$$

- ✓ Histogram equalization is the process of transforming an image in such a way that transformed image has a flat histogram
- ✓ Since a histogram is an approximation to a PDF, the result histogram usually is not a uniform histogram (flat histogram)

$$s_0 = T(r_0) = \frac{(L-1)}{M \times N} n_0$$

$$s_1 = T(r_1) = \frac{(L-1)}{M \times N}(n_0 + n_1)$$

$$s_2 = T(r_2) = \frac{(L-1)}{M \times N}(n_0 + n_1 + n_2)$$

$$\vdots$$

Histogram equalization

- ✓ An image of size 64×64 with intensity levels in the range of [0,…7], has the following histogram values, find the Transformation function and the equalized histogram.

r_k		$p_r(r_k)$	s_k	round(Sk)
0	750	0.183105	1.281738	1
1	1000	0.244141	2.990723	3
2	900	0.219727	4.528809	5
3	600	0.146484	5.554199	6
4	300	0.073242	6.066895	6
5	300	0.073242	6.57959	7
6	150	0.036621	6.835938	7
7	96	0.023438	7	7
	4096			

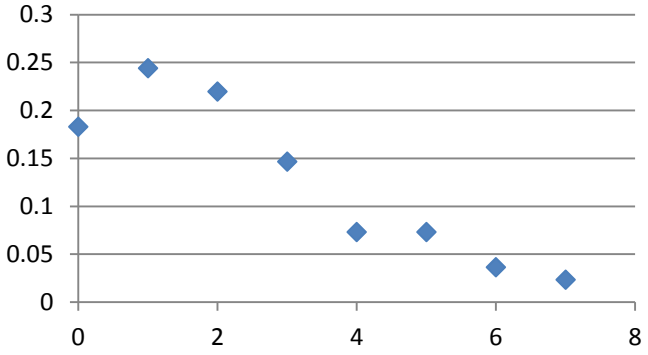
$$p_r(r_k) = \frac{n_k}{M \times N} \qquad s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7\,p_r(r_0)$$

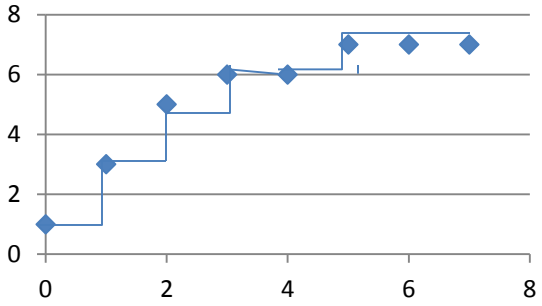
$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7\,p_r(r_0) + 7\,p_r(r_1)$$

1	750	0.183105
3	1000	0.244141
5	900	0.219727
6	900	0.219727
7	546	0.133301

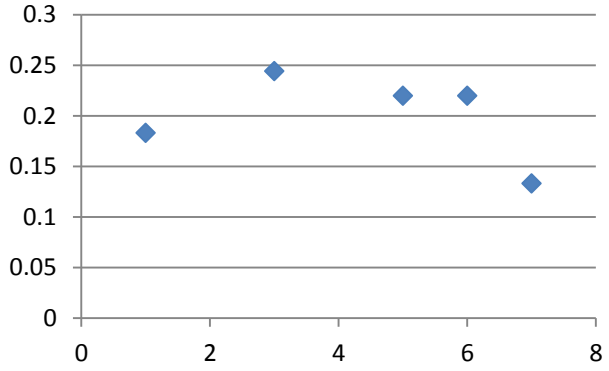
$p_r(r_k)$



$T(r_k)$



$p_s(s_k)$



Histogram Matching

- ✓ In some applications, it is needed to change the histogram of an image to a specific histogram. In this case, we can specify the shape of desired histogram for the processed image.
- ✓ The process used to generate a processed image that has a specified histogram is called histogram matching or histogram specification.
- ✓ **Histogram Matching Process**
  - ❖ 1. Find the histogram $p_r(r)$ of the input image and determine its equalization transformation:
$$s = T(r) = (L-1)\int_0^r p_r(w)\, dw$$
  - ❖ 2. Use the specified pdf , $p_z(r)$ of the output image to obtain the transformation function:
$$G(z) = (L-1)\int_0^z p_z(t)\, dt$$
  - ❖ 3. match s=G(z) and find the inverse transformation $z = G^{-1}(s)$ the mapping from s to z:
$$z = G^{-1}(T(r))$$
  - ❖ 4. Obtain the output image by equalizing the input image first; then for each pixel in the equalized image, perform the inverse mapping to obtain the corresponding pixel of the output image.

Histogram Matching- Example I

- ✓ Given a 3-bit gray-level input image A with the following histogram:

r_k	s_k
0	1028
1	3544
2	5023
3	3201
4	1867
5	734
6	604
7	383

- ✓ Design point operation to match it to the following desired histogram
  (the desired histogram can be the histogram of image B)

r_k	s_q
0	0
1	0
2	1638
3	4096
4	4916
5	4096
6	1638
7	0

Histogram Matching - Example I (cont.)

- ✓ 1- Equalize the input image A and find an image B which has a flat histogram

r_k	n_k	$p_r(r_k)$	s_k	round(s_k)
0	1028	0.062744	0.439209	0
1	3544	0.216309	1.953369	2
2	5023	0.30658	4.099426	4
3	3201	0.195374	5.467041	5
4	1867	0.113953	6.264709	6
5	734	0.0448	6.578308	7
6	604	0.036865	6.836365	7
7	383	0.023376	7	7
	16384			

- ✓ 2- Equalize the desired histogram

z_q	n_q	$p_z(z_q)$	$G(z) = s_q$	Round(s_q)
0	0	0	0	0
1	0	0	0	0
2	1638	0.099976	0.699829	1
3	4096	0.25	2.449829	2
4	4916	0.300049	4.550171	5
5	4096	0.25	6.300171	6
6	1638	0.099976	7	7
7	0	0	7	7
	16384			

Histogram Matching- Example I (cont.)

- ✓ 3- Find the inverse function $z = G^{-1}(s)$
  - ❖ For each value of S_k ,find the smallest value of Z_q , so that $G(z_q)$ is closest to the value of S_k . When more than one value of Z_q satisfies the given S_k , choose the smallest value by convention.

	Z_q
0	0
2	3
4	4
5	4
6	5
7	6

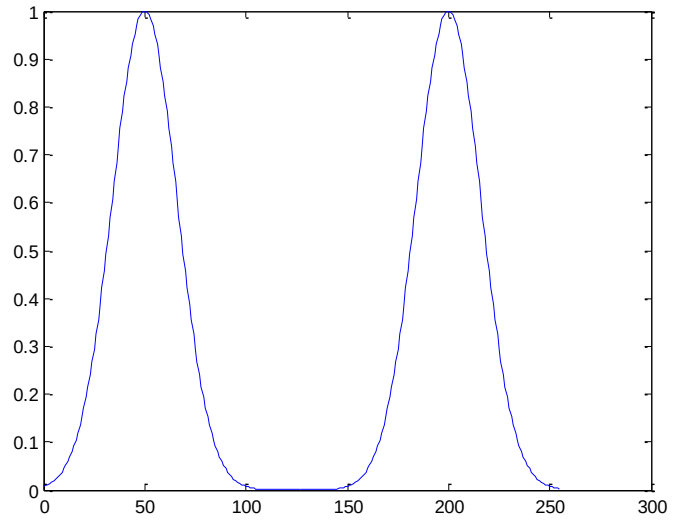
- ✓ 4- Take image B (found in step 1), and replace pixel s by z using the table II

Histogram Matching- Example II

Write a MATLAB program to read an image and match its histogram to histogram defined as h, where h is

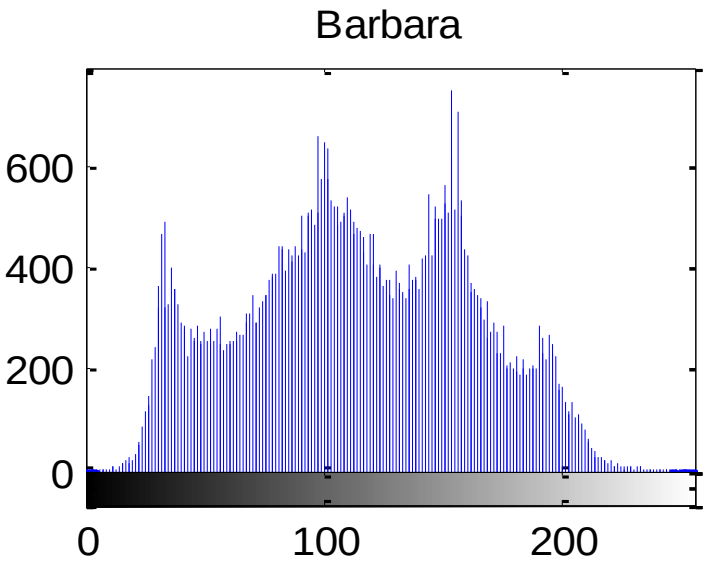
$$h = e^{\frac{-(d-50)^2}{2\times 16^2}} + e^{\frac{-(d-200)^2}{2\times 16^2}}$$

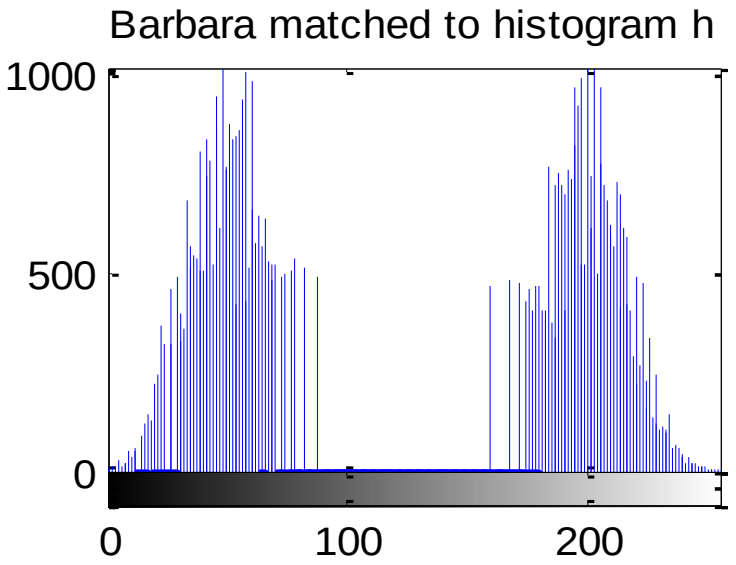
```
D = 0:255;
h = exp(-(D-50).^2/(2*16^2)) + exp(-(D-200).^2/(2*16^2));
Plot (D,h)
```




```
I=imread('barbara.jpg');
figure
subplot(2,2,1);imhist(I); title('Barbara');
subplot(2,2,3);imshow(I); title('Barbara');
im2 = histeq (I, h);
subplot(2,2,2);imhist(im2); title('Barbara matched to h histogram');
subplot(2,2,4);imshow(im2); title('Barbara matched to h histogram');
```

Histogram Matching – Example II (cont.)





Barbara

Barbara matched to histogram h

Local Histogram

- ✓ Local histogram processing
  - ❖ Simplest way for local histogram processing is to divide the image to sub-images and performing histogram processing(equalization , matching) on these sub-images instead of over entire image
    - If the image is divided into non-overlapping sub-images, the process will produce undesirable artifacts.
  - ❖ A better way for local histogram processing explained as follows:
    - Define an area
    - Compute the histogram processing function
    - Use the computed function to map the intensity of pixel in the center
    - The center of the area needs then to be moved to adjacent location and repeat the above steps

✓ Q:

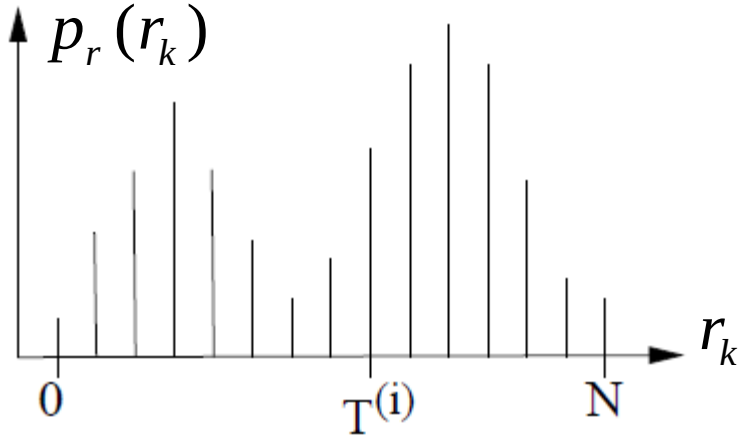
Histogram of intensity k in an area of n pixel is $P_r(r_k) = \frac{n_k}{n}$. If the area is shifted by one pixel to the right, calculate the new histogram of intensity k is terms of the $P_r(r_k)$

✓ A:

$$P_r(r_k)|_{new} = \frac{1}{n}[n_k + n_{k-Right} - n_{k-Left}] \qquad P_r(r_k)|_{new} = P_r(r_k) + \frac{1}{n}[n_{k-Right} - n_{k-Left}]$$

Thresholding using histogram

- ✓ Thresholding is used widely in image processing to produce binary images.
- ✓ An image with bright objects and a dark background can often be thresholded to a binary image. Such an image has a histogram with two peaks. Suppose that the histogram $p_r(r_k)$ looks as in the figure below.



- ✓ The main question is the thresholding point.
- ✓ A good threshold can be found with the midway method. The midway method is iterative and calculates mean values.

Thresholding- midway method

- ✓ midway method:
  - ❖ 1) Set $i = 0$ and choose an initial threshold $T^{(i)} = T^{(0)}$.
  - ❖ 2) Calculate the mean to the left $\mu_0 T^{(i)}$ and to the right $\mu_1 T^{(i)}$ of $T(i)$.
  - ❖ A new threshold $T^{(i+1)}$ is calculated as the mean of the two means,

$$T^{(i+1)} = (\mu_0 T^{(i)} + \mu_1 T^{(i)}) / 2$$

  - ❖ 4) Set $i = i + 1$. Repeat from 2.
  - ❖ The stop criteria is when $T(i+1) \approx T(i)$.

- ✓ The mean of the function to the left / right of the threshold $T^{(i)}$ can be calculated as:

$$\mu_0 T^{(i)} = \frac{\sum_{r_k=0}^{T^{(i)}} p_r(r_k).\, r_k}{\sum_{r_k=0}^{T^{(i)}} p_r(r_k)} \qquad \mu_1 T^{(i)} = \frac{\sum_{r_k=T^{(i)}+1}^{N} p_r(r_k).\, r_k}{\sum_{r_k=T^{(i)}+1}^{N} p_r(r_k)}$$

- ✓ A good initial threshold $T^{(0)}$ can also be found by calculating another mean.

$$T^{(0)} = \frac{\sum_{r_k=0}^{N} p_r(r_k).\, r_k}{\sum_{r_k=0}^{N} p_r(r_k)}$$

Example of Thresholding using midway method

r_k	$p_r(r_k)$	$r_k p_r(r_k)$
0	11	0
1	10	10
2	300	600
3	150	450
4	39	156
5	60	300
6	700	4200
7	30	210
	1300	5926

r_k	$p_r(r_k)$	$r_k p_r(r_k)$
0	11	0
1	10	10
2	300	600
3	150	450
4	39	156
5	60	300
	570	1516
6	700	4200
7	30	210
	730	4410

r_k	$p_r(r_k)$	$r_k p_r(r_k)$
0	11	0
1	10	10
2	300	600
3	150	450
4	39	156
	510	1216
5	60	300
6	700	4200
7	30	210
	790	4710

$$T^{(0)} = \frac{\sum_{r_k=0}^{N=8} p_r(r_k).\, r_k}{\sum_{r_k=0}^{N=8} p_r(r_k)} = 4.556, \quad T^{(0)} = 5$$

$$T^{(1)} = (\mu_0 T^{(0)} + \mu_1 T^{(0)})/2 = \frac{\sum_{r_k=0}^{T^{(0)}=5} p_r(r_k).\, r_k}{2\sum_{r_k=0}^{T^{(0)}=5} p_r(r_k)} + \frac{\sum_{r_k=6}^{8} p_r(r_k).\, r_k}{2\sum_{r_k=6}^{8} p_r(r_k)} = 4.35 \;\; T^{(1)} = 4$$

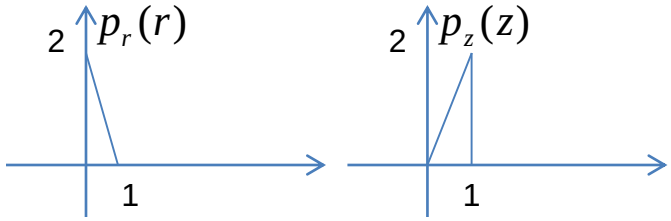
$$T^{(2)} = (\mu_0 T^{(1)} + \mu_1 T^{(1)})/2 = \frac{\sum_{r_k=0}^{T^{(0)}=4} p_r(r_k).\, r_k}{2\sum_{r_k=0}^{T^{(0)}=4} p_r(r_k)} + \frac{\sum_{r_k=5}^{8} p_r(r_k).\, r_k}{2\sum_{r_k=5}^{8} p_r(r_k)} = 4.17 \;\; T^{(2)} = 4$$

Assignment #2

1) An image has the PDF $p_r(r) = Ae^{-r}$. It is desired to transform the intensity values of this image so that its PDF changes to $p_z(z)$. Assume continuous quantities and find the transformation to perform this task.

$$p_r(r) = Ae^{-r} \qquad p_z(z) = Bze^{-z^2}$$

2) An image has the PDF shown below. It is desired to transform the intensity values of this image so that its PDF becomes $p_z(z)$. $p_z(z)$ is also shown below. Assume continuous quantities and find the transformation to perform this task.



3) Why do histogram equalization of a digital image usually not produce images with flat histograms?

4) A 50x70 image has 3-bit pixels. Its histogram, H(r), looks like a ramp as shown below. The counts in the histogram follow the formula $H(r) = kr$, where k is a constant.

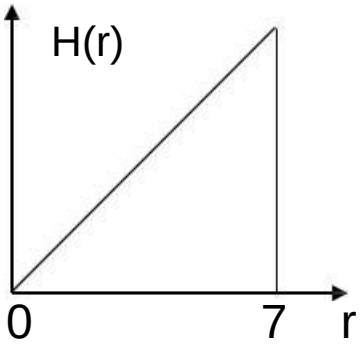
a) Determine the value of k.

b) Compute the mean and standard deviation of this image from the histogram.

c) Compute the transformation function $s = T(r)$ that will equalize the histogram.

d) Compute the histogram $H(s)$ of the resulting image, if it were transformed by T.

e) Compute the mean and standard deviation of the resulting image.



Spatial Filtering

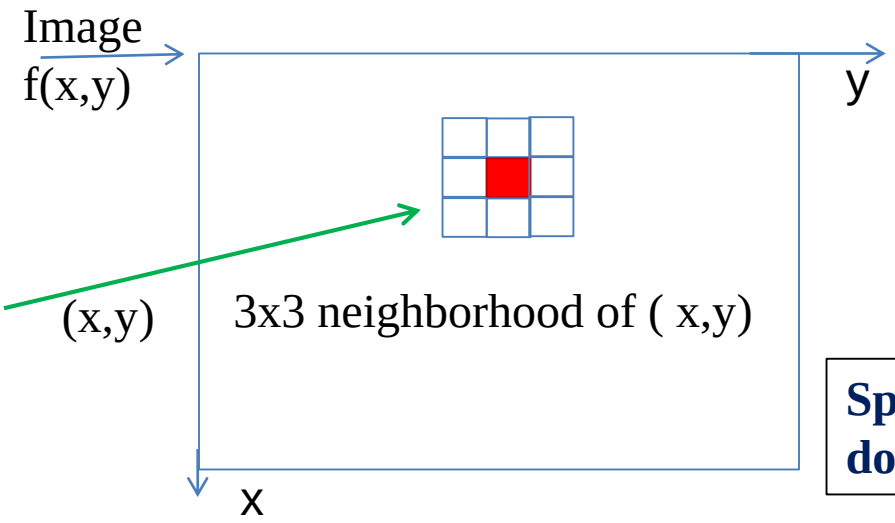
Spatial Filtering

- ✓ Spatial domain techniques operate directly on the pixels of an image as opposed to the frequency domain in which operation is performed on the Fourier transform of an image.
- ✓ Depending on the processing, we might choose to perform spatial or frequency domain techniques. Some processing tasks are easier or better suited to be done in spatial domain, while many others need to be done in frequency domain.
- ✓ A spatial domain process can be defined as

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

output image ← $g(x,y)$ = T [$f(x,y)$] → Input image

Operator defined over a neighborhood of point (x,y)



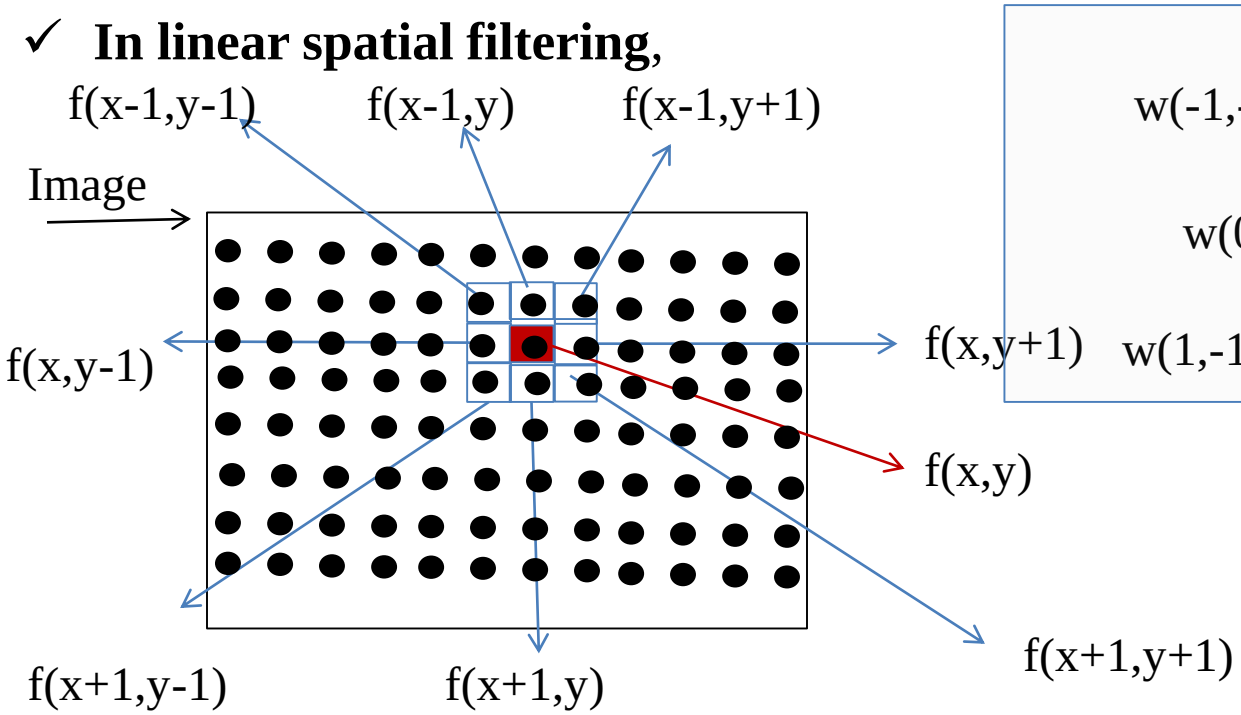
T

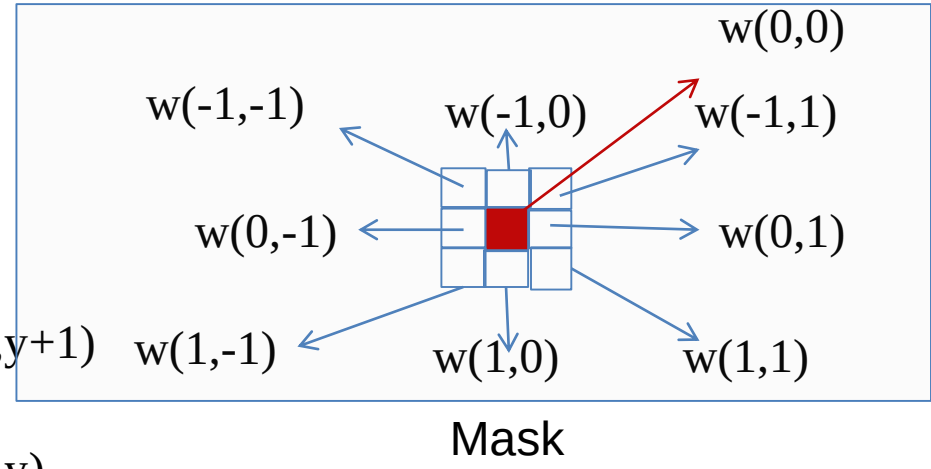
To find the value of g(x,y), The operator T must be applied to all the pixels in the neighborhood

Special case: if the neighborhood is 1x1, spatial domain is the same as point processing

Spatial Filtering- Fundamental

- ✓ To perform special filtering on an image, we need
  - ❖ To identify a neighborhood which typically represented by a rectangle and is called mask, window, kernel or template
  - ❖ To consider an operation which is performed on the image pixels in that neighborhood. If the operation is linear, the process is called linear spatial filtering. If the operation is non-linear, then the process is non-linear spatial filtering.
- ✓ The result of spatial filtering is a new pixel value for the given pixel.
- ✓ **In linear spatial filtering,**





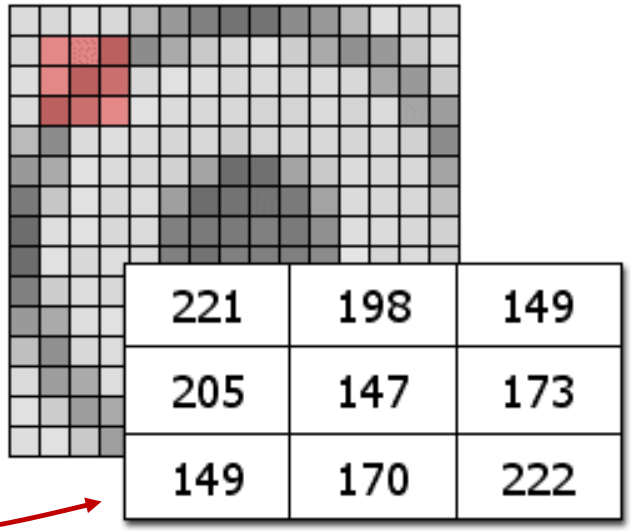
$$g(x,y) = w(-1,-1)\, f(x-1,y-1) + w(-1,0)\, f(x-1,y) + w(-1,1)\, f(x-1,y+1) + w(0,-1)\, f(x,y-1) + w(0,0)\, f(x,y) + w(0,1)\, f(x,y+1) + w(1,1)\, f(x+1,y-1) + w(1,0)\, f(x+1,y) + w(1,1)\, f(x+1,y+1)$$

New pixel value

Spatial filtering by Correlation or Convolution

- ✓ Correlation:
  - ❖ moving a filter mask over an image and computing the sum of products at each location (like what is shown in previous slide)
- ✓ Convolution
  - ❖ Convolution is the same as correlation, but the mask is first rotated by 180 degree and then the rotated mask should be moved over the image.

portion of an image, with a fragment of it highlighted and the greyscale values shown



-1	0	1
-2	0	2
-1	0	1

mask

Rotated mask by 180

1	0	-1
2	0	-2
1	0	-1

- ✓ New value for the middle pixel with value 147 using

Correlation: (221*-1)+ (198*0)+(149*1)+(205*-2)+(147*0)+(173*2)+(149*-1)+(170*0)+(22*1)= -63

Convolution: (221*1)+ (198*0)+(149*-1)+(205*2)+(147*0)+(173*-2)+(149*1)+(170*0)+(22*-1)= 63

Spatial filtering Examples

- ✓ To do spatial filtering using a mask of size $m \times n$, we need to specify $m \times n$ coefficients. These $m \times n$ coefficients should be chosen in such away that applying the spatial filter result in desired processing.
  - ❖ Processing examples which can be done using spatial filtering
  - ❖ Example I – smoothing filter (linear filter)
    - ▪ Smoothing filters can be used for blurring of an image (reducing sharp transition)
    - ▪ Noise reduction
    - ▪ Removal of small detail before object extraction
  - ❖ Example II- Median filter (non-linear filter)
    - ▪ Excellent noise reduction especially for impulse noise(salt and pepper noise)
  - ❖ Example III-Sharpening filter (linear filter)
    - ▪ To highlight transition in intensity
    - ▪ To shows edges or other types of discontinuities
    - ▪ Deemphasize areas of image that intensities won't change sharply

Spatial filtering- Example I (Smoothing filter)

- ✓ Smoothing filtering can be achieved by replacing all pixels in the image by the average of intensity of a 3x3 neighborhood created on those pixels .
- ✓ Smoothing filters also called averaging filter or low-pass filter
- ✓ To calculate the average, the mask can be selected as

1/9 X

1	1	1
1	1	1
1	1	1

→ Also called box-filter since all coefficients are equal

1/16 X

1	2	1
2	4	2
1	2	1

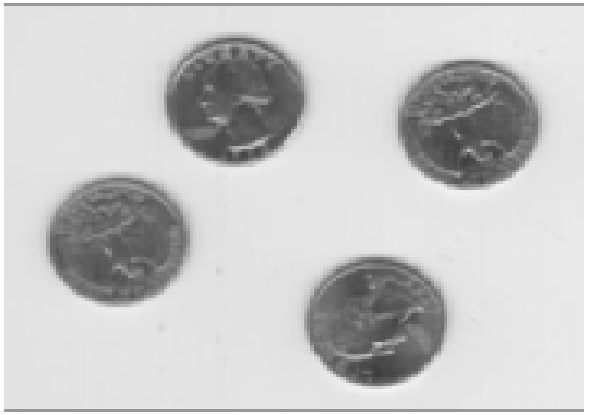
→ Also called weighted average, since all coefficients are not equal. It gives more weight to pixels at the center of the mask

Spatial filtering- Example I(Smoothing Filter)

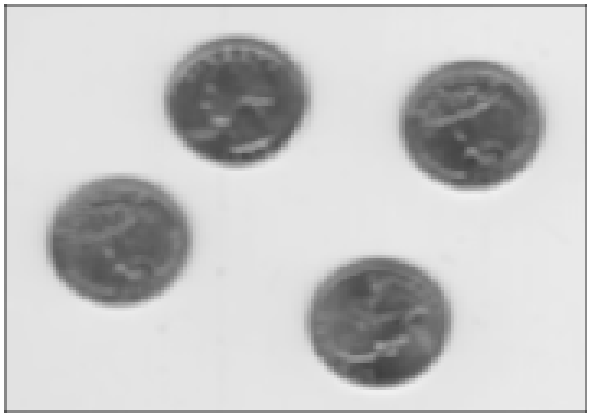
✓ Effect of window size in smoothing

Original

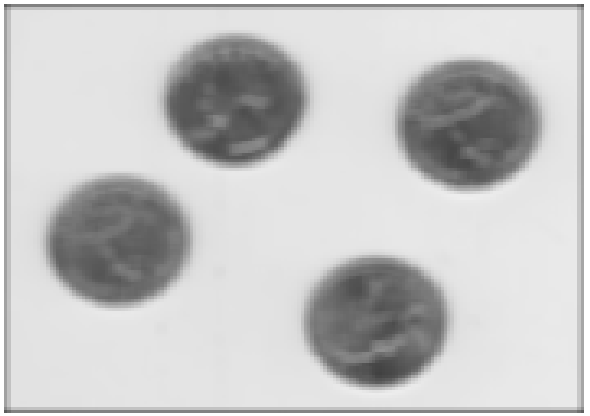
window 3x3



window 5x5



window 7x7



Spatial filtering- Example II(Median Filter)

- ✓ Median filter is a kind of order-statistic filters. These filters are not linear. In this filtering method, the value of a pixel is replaced by the median of the intensity value of the neighborhood of that pixel.
- ✓ To do median filtering
  - ❖ 1- sort the values of the pixels in the neighborhood from low to high
  - ❖ 2-find the median value
  - ❖ 3- assign this value to the corresponding pixel in the filtered image

10	**20**	**10**
15	**20**	**17**
20	**11**	**12**

10 10 11 12 15 17 20 20 20

3x3 neighborhood: the median is the 5th largest value

- ✓ Median is not the only order–statistic filters, the other example of these type of filters are
  - ❖ Max filter, Min filter, various percentage filters

Spatial filtering- Example III(Sharpening Filter)

- ✓ Smoothing filters use averaging, sharpening filters use differentiation.
- ✓ Differentiation can be achieved by derivation such as

❖ First order derivation $$\frac{\partial f(x)}{\partial x} = f(x+1) - f(x)$$

- on areas of constant intensities → results in zero
- along ramp-like transition in intensity → results non-zero
- start of a ramp or step intensity transition → results non-zero

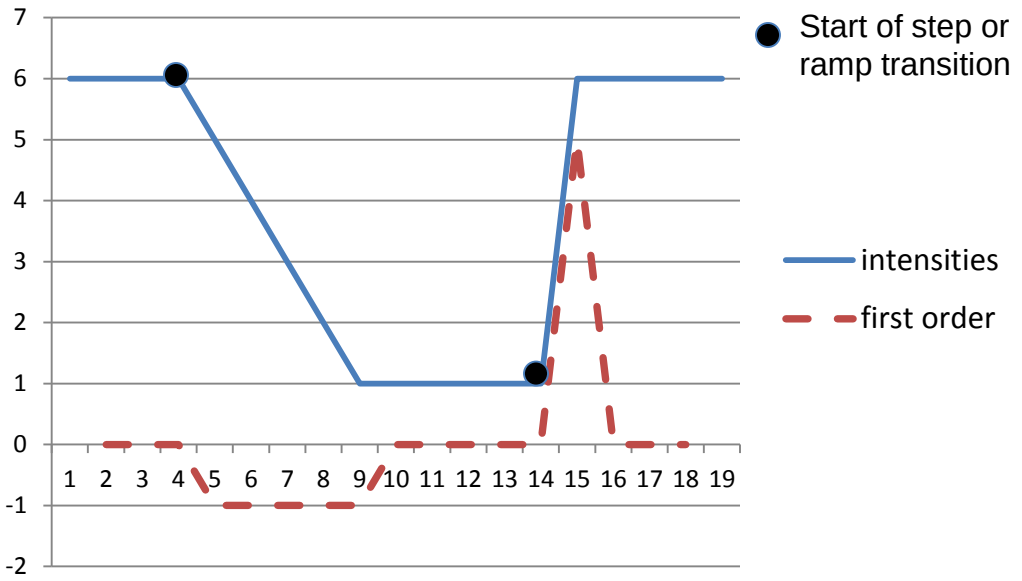
❖ Second-order derivation $$\frac{\partial^2 f(x)}{\partial x^2} = f(x+1) - 2f(x) + f(x-1)$$

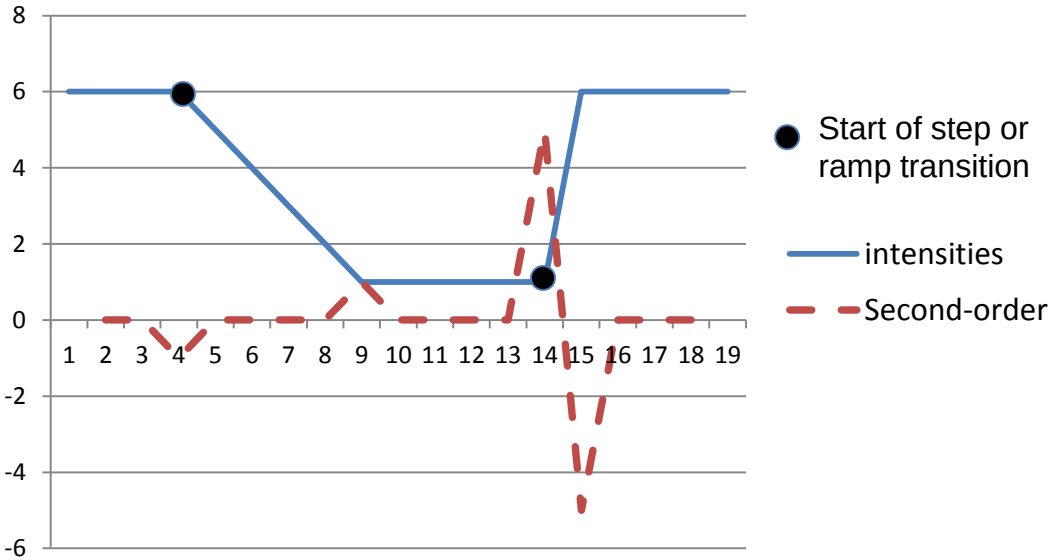
- on areas of constant intensities → results in zero
- along ramp-like transition in intensity → results zero
- start of a ramp or step intensity transition → results non-zero

Spatial filtering- Example III(Sharpening Filter) - (Cont.)

$$\frac{\partial f(x)}{\partial x} = f(x+1) - f(x) \qquad \frac{\partial^2 f(x)}{\partial x^2} = f(x+1) - 2f(x) + f(x-1)$$

intensities	first order	Second-order
6		
6	0	0
6	0	0
6	-1	-1
5	-1	0
4	-1	0
3	-1	0
2	-1	0
1	0	1
1	0	0
1	0	0
1	0	0
1	0	0
1	5	5
6	0	-5
6	0	0
6	0	0
6	0	0





Spatial filtering- Example III(Sharpening Filter) –(cont.)

- ✓ Edges in the image often are ramp-like transitions in intensity
  - ❖ Using First order derivation for sharpening
    - Produce thick edge, because is non-zero along the ramp
  - ❖ Using Second order derivation for sharpening
    - Produce double edge with one pixel thick, because is zero along the ramp

- ✓ Second order gives better edge representation and it is easier to implement. So usually second-order functions are used.
- ✓ For a 2-D image, the simplest derivative operator is Laplacian

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Spatial filtering- Example III(Sharpening Filter) –(cont.)

- ✓ The mask to implement the Laplacian formula is shown below

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

0	1	0
1	-4	1
0	1	0

- ✓ The other practical implementation of sharpening filters are

1	1	1
1	-8	1
1	1	1

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

- ✓ In general, the ones with 8, -8 at the center produce sharper-looking result as compared to the ones with 4 and -4 at the center. The reason is the fact that the ones with 8/-8 at the center detects changes along diagonals as well as horizontal and vertical directions.
- ✓ The Laplacian filter highlights intensity discontinuity in image and deemphasizes regions with slowly varying intensity..
- ✓ To sharpen an image the result needs to be added to the image

Spatial filtering- Example IV (High-boost filtering)

- ✓ A sharpening method used by printing industry is called high-boost filtering or unsharp-masking which has the following steps:
  - ❖ Smooth (blur) the original image $f(x,y)$ and call the blurred image as $\bar{f}(x,y)$
  - ❖ Subtract the smoothed image from the original image and call the result image as a mask $g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$
  - ❖ Add the mask image to the original image $g(x,y) = f(x,y) + g_{mask}(x,y)$
- ✓ For the original signal showing below, explain how High-boost filtering sharpen the image

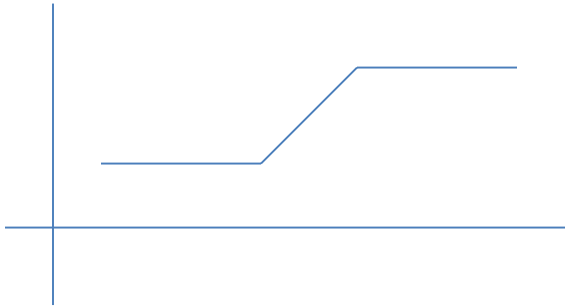


Image padding

- ✓ The Spatial filter should be applied to the entire pixels within an image by sliding the filter window over the image. However, there is an inherent problem when you are working with the image boundaries. The problem is that some of the "neighbors" are missing. There are several solution for this problem:
  - ❖ **Zero padding** where the missing value is replaced by zero. The disadvantage of zero padding is that it leaves dark artifacts around the edges of the filtered image (with white background).
  - ❖ **Replicate** where the missing value is replaced by the image boundary values.
  - ❖ **Symmetric** where the size of the image is extended by mirror-reflecting it across its border

Image

1	2	3
4	5	6
7	8	9

mask

1/4	1/4
1/4	1/4

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

Zero Padding

Replicate is the same as symmetric in this case

1	1	2	3	3
1	1	2	3	3
4	4	5	6	6
7	7	8	9	9
7	7	8	9	9

Padding example

$$f(x,y)=\begin{bmatrix}0 & -2 & 1\\ 0 & 0 & 0\\ 0 & 2 & 0\end{bmatrix} \qquad \textit{rotated}\quad f(x,y)=\begin{bmatrix}0 & 2 & 0\\ 0 & 0 & 0\\ 1 & -2 & 0\end{bmatrix}$$

- ✓ Calculate the correlation $f(x,y) \square f(x,y)$ and convolution $f(x,y) * f(x.y)$ using $f(x,y)$ as defined above. Use zero padding to calculate boundary values..

Zerro Padded $f(x,y)$

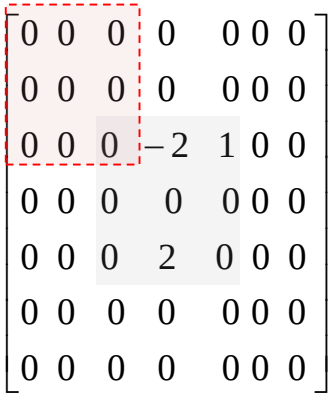
$$\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&-2&1&0&0\\0&0&0&0&0&0&0\\0&0&0&2&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}$$

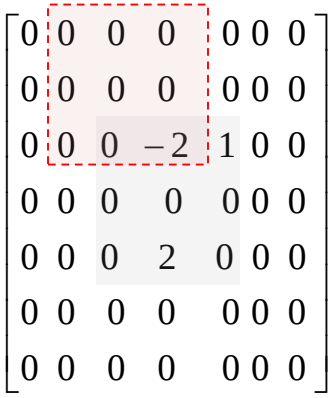
$f(x,y) \square f(x,y)$

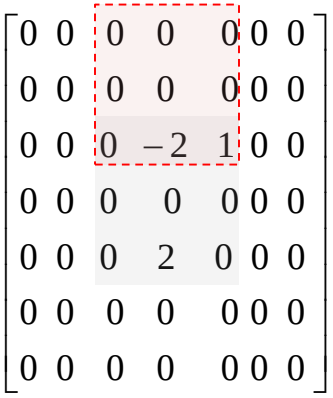
$$\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&-4&2&0&0\\0&0&0&0&0&0&0\\0&0&-2&9&-2&0&0\\0&0&0&0&0&0&0\\0&0&2&-4&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}$$

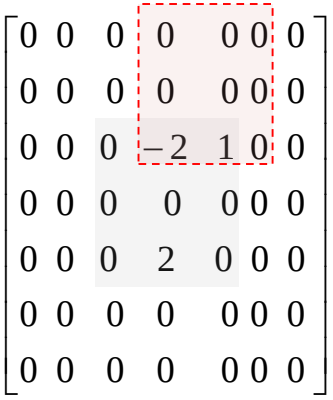
$f(x,y) * f(x.y)$

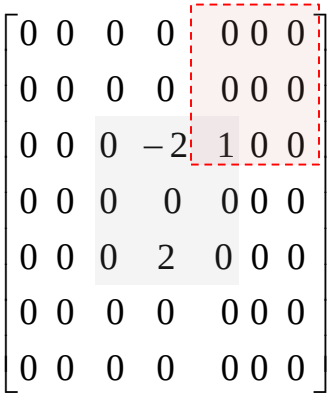
$$\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&4&-4&1&0\\0&0&0&0&0&0&0\\0&0&0&-8&4&0&0\\0&0&0&0&0&0&0\\0&0&0&4&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}$$







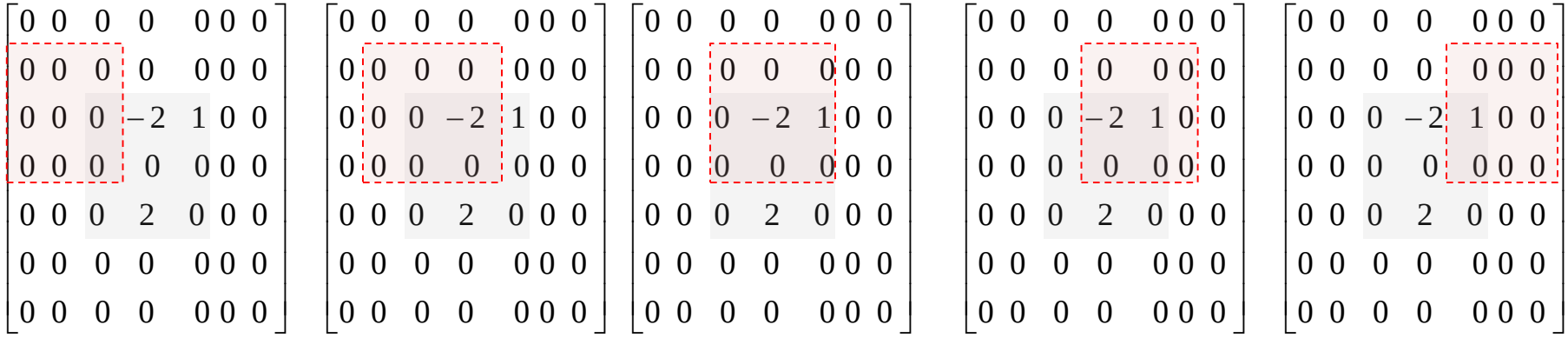




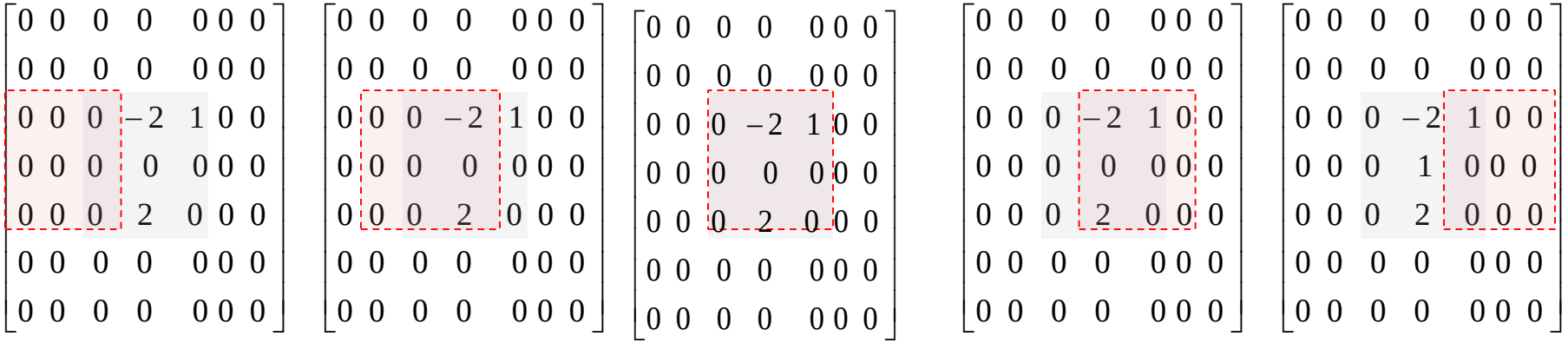
Correlation	0	0	-4	2	0
Convolution	0	0	4	-4	1

Padding example

$$f(x,y)=\begin{bmatrix}0 & -2 & 1\\ 0 & 0 & 0\\ 0 & 2 & 0\end{bmatrix} \qquad \textit{rotated} \quad f(x,y)=\begin{bmatrix}0 & 2 & 0\\ 0 & 0 & 0\\ 1 & -2 & 0\end{bmatrix}$$



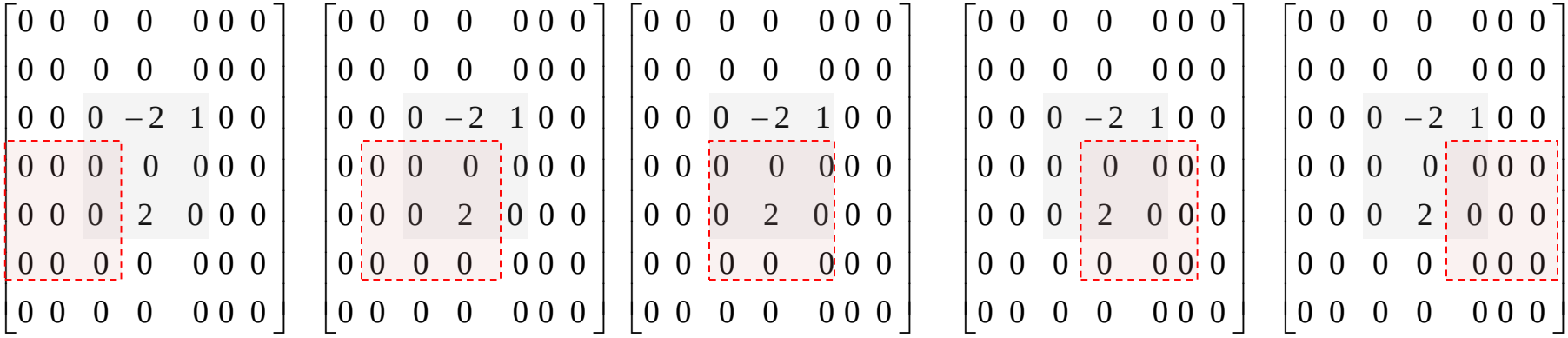
Correlation	0	0	0	0	0
Convolution	0	0	0	0	0



Correlation	0	-2	9	-2	0
Convolution	0	0	-8	4	0

Padding example

$$f(x,y)=\begin{bmatrix}0 & -2 & 1\\ 0 & 0 & 0\\ 0 & 2 & 0\end{bmatrix} \qquad \textit{rotated} \quad f(x,y)=\begin{bmatrix}0 & 2 & 0\\ 0 & 0 & 0\\ 1 & -2 & 0\end{bmatrix}$$



Correlation	0	0	0	0	0
Convolution	0	0	0	0	0

Correlation	0	2	-4	0	0
Convolution	0	0	4	0	0

Assignment #3

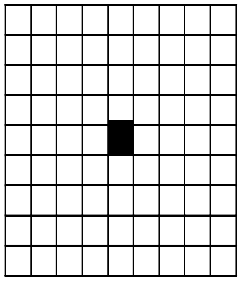
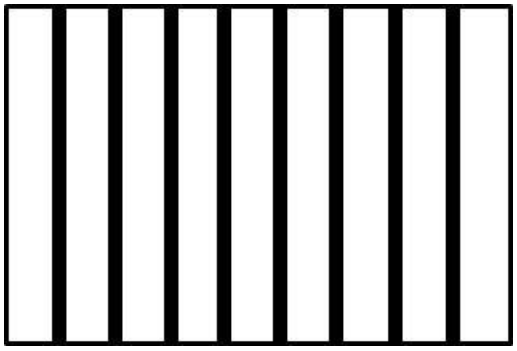


Image A

■ Pixels with intensity value of 1

□ Pixels with intensity value of 0

1) For the image below (let's call it image A), apply a 3x3 averaging low pass filter and call it image B. Apply the same averaging filter to Image B to produce image C. Draw image B and C. Explain what happens when you repeatedly apply an averaging filter to an image?



2) In Image shown here, each bar has a width of 6 pixels. The gaps between bars are 19 pixels. Explain the result of filtering this image using

a) An averaging mask of 25x25 b) An averaging mask of 20x20

3) In spatial filtering, a mask is applied to top-left corner of an image and then the center of the mask will be moved through the image. At each location, the sum of product of the mask coefficients with the corresponding pixel values at that location is calculated. Then the pixel value of the image corresponding with the center of the mask will be updated.

Considering an averaging mask of size $n \times n$ with coefficients of $1/n^2$. Calculate the minimum computational complexity of applying the averaging filter to an image. Remember that for an averaging mask all the coefficients are one considering a scale factor of $1/n^2$.

4) A digital image f(x,y) is changed to a black and white image b(x,y). b(x,y) is passed through a 3x3 blurring filter as shown here to produce output image o(x,y). What are possible intensity values in the output image.

$$\begin{bmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{bmatrix}$$

2-D Fourier Transformation

Fourier series and Transform

- ✓ Fourier series:
  - ❖ A periodic function (with period T) can be expressed as the sum of sines and /or cosine each multiplied by a different coefficient.

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{T}t} \qquad c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t)\ e^{-j\frac{2\pi n}{T}t}\ dt$$

- ✓ Fourier transform
  - ❖ A non-periodic function (with finite area under curve) can be expressed as the integral of sines and/or cosines multiplied by a weighing function

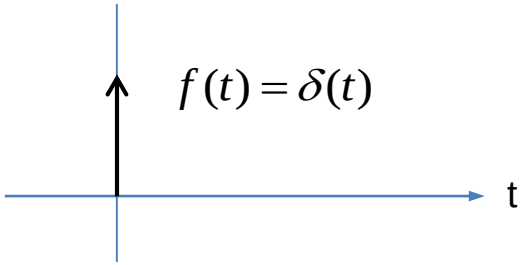
1-D: $$f(t) = \int_{-\infty}^{\infty} F(\mu)\ e^{j2\pi\mu t} d\mu \qquad F(\mu) = \Im\{f(t)\} = \int_{-\infty}^{\infty} f(t)\ e^{-j2\pi\mu t} dt$$

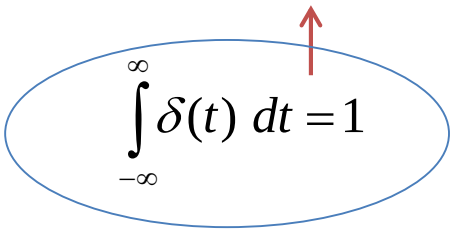
2-D: $$f(t,z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu,\nu)\ e^{j2\pi(\mu t+\nu z)} d\mu\, d\nu \qquad F(\mu,\nu) = \Im\{f(t,z)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t,z)\ e^{-j2\pi(\mu t+\nu z)} dt\ dz$$

1-D impulse function $\delta(t)$ and impulse train $s_{\Delta T}(t)$

✓ Impulse function is a spike of infinity amplitude and zero duration, having the unit area

$$\delta(t) = \begin{cases} \infty & if \ \ t = 0 \\ 0 & if \ \ t \neq 0 \end{cases}$$

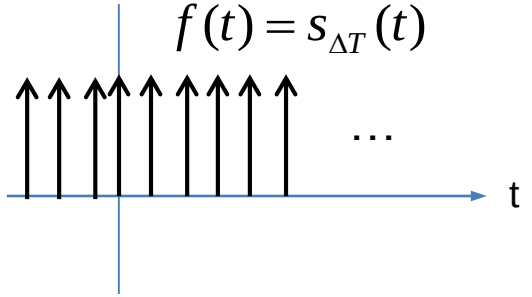




✓ One of the properties of impulse function is its Sifting property

$$\int_{-\infty}^{\infty} f(t)\ \delta(t)\ dt = f(0), \quad \int_{-\infty}^{\infty} f(t)\ \delta(t-t_0)\ dt = f(t_0)$$

✓ A series of impulse function can make a Impulse train which can be defined as $s_{\Delta T}(t)$ $\quad s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$



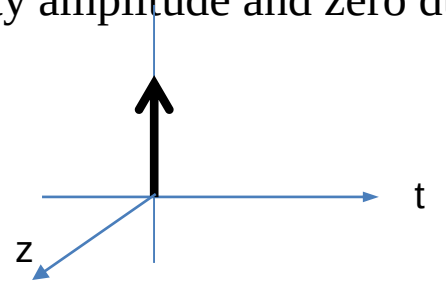
✓ The Fourier transform of $\delta(t)$, $\delta(t-t_0)$ and $s_{\Delta T}(t)$ are shown here.

$f(t)$	$F(\mu)$
$\delta(t)$	1
$\delta(t-t_0)$	$e^{-j2\pi\mu t_0}$
$s_{\Delta T}(t)$	$\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} \delta(\mu - \frac{n}{\Delta T})$

2-D impulse function $\delta(t,z)$ and impulse train $s_{\Delta T\,\Delta Z}(t,z)$

✓ Impulse function is a spike of infinity amplitude and zero duration, having the unit area

$$\delta(t,z)=\begin{cases}\infty & if\ \ t=z=0\\ 0 & otherwise\end{cases}$$



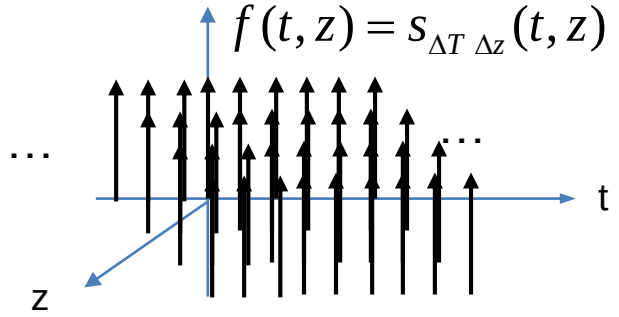
$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\delta(t,z)\,dt\,dz=1$$

✓ One of the properties of impulse function is its Sifting property

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(t,z)\,\delta(t,z)\,dt\,dz=f(0,0)\,,\quad \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(t,z)\,\delta(t-t_0,z-z_0)\,dt\,dz=f(t_0,z_0)$$

✓ 2-D Impulse train $s_{\Delta T\,\Delta z}(t,z)$ can be defined as

$$s_{\Delta T\,\Delta z}(t,z)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\delta(t-m\Delta T,z-n\Delta z)$$



$f(t,z)$	$F(\mu,\nu)$
$\delta(t,z)$	1
$\delta(t-t_0,z-z_0)$	$e^{-j2\pi(\mu t_0+\nu z_0)}$
$s_{\Delta T\,\Delta z}(t,z)$	$\frac{1}{\Delta T\,\Delta z}\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\delta(\mu-\frac{n}{\Delta T},\nu-\frac{m}{\Delta z})$

1-D sampled function

- ✓ Continuous function needs to be transformed into discrete values before getting processed by a computer which is done using sampling and quantization processes

$$\tilde{f}(t) = f(t)\, s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\ \ \delta(t - n\Delta T)$$

- ✓ The kth sampled value is

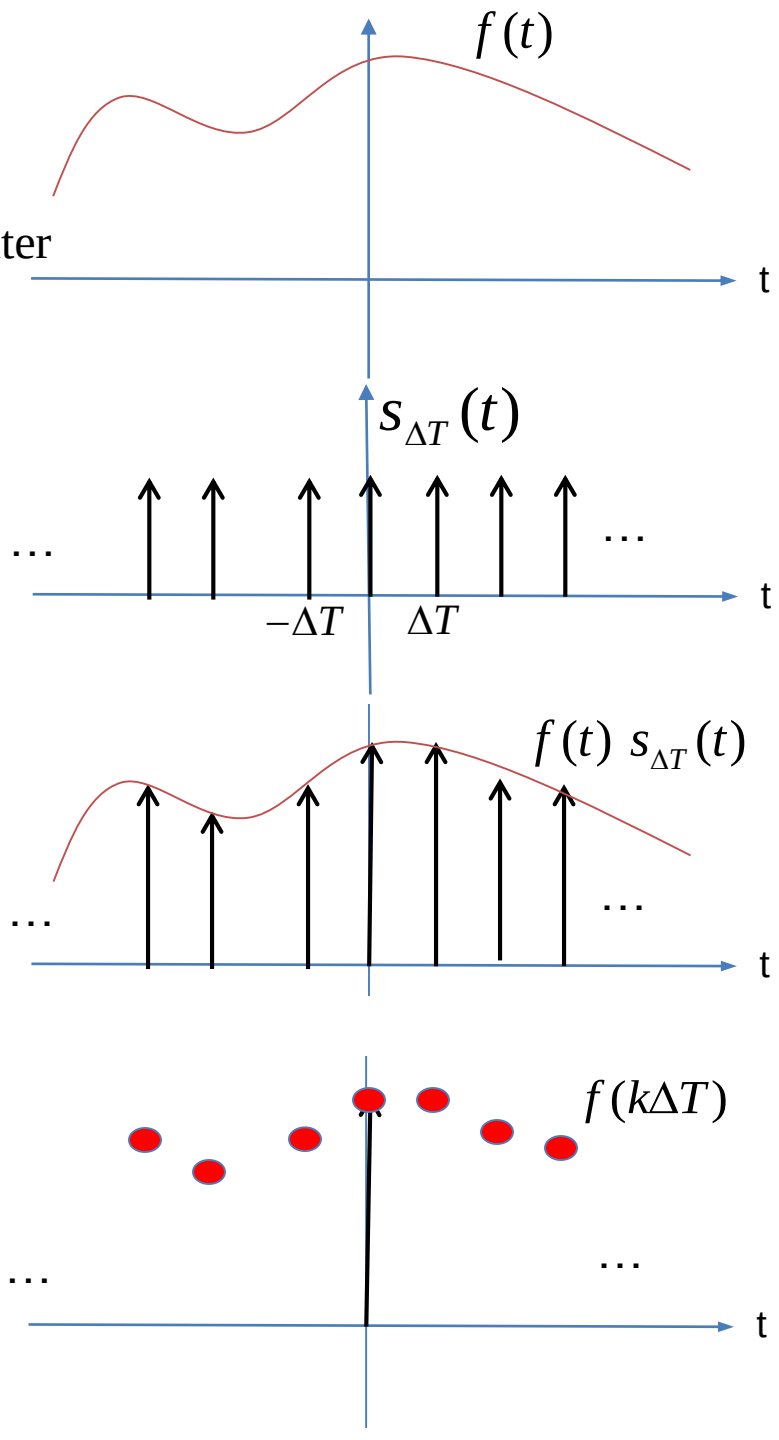
$$f_k = \int_{-\infty}^{\infty} f(t)\ \ \delta(t - k\Delta T)\ dt = f(k\Delta T)$$

- ✓ The Fourier transform of the sampled function is

$$\tilde{F}(\mu) = \Im\{\tilde{f}(t)\} = \Im\{f(t)\, s_{\Delta T}(t)\} = F(\mu) * S(\mu)$$

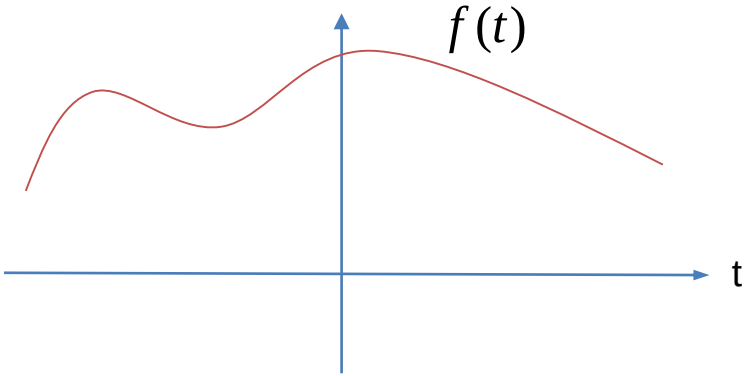
  - ❖ By replacing $S(\mu)$ by $\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} \delta(\mu - \frac{n}{\Delta T})$ and calculating the convolution

$$\tilde{F}(\mu) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} F(\mu - \frac{n}{\Delta T})$$

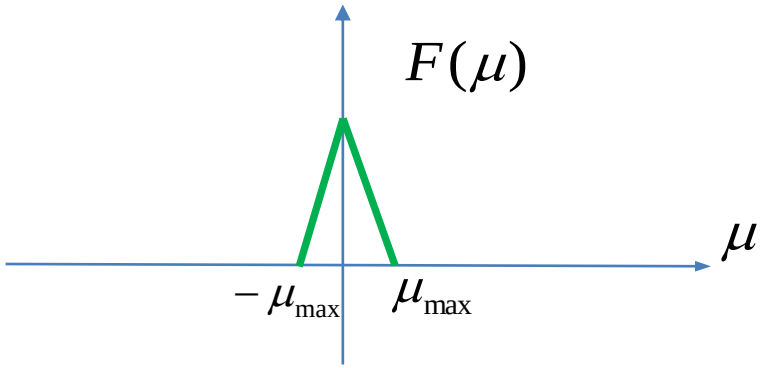


1-D sampled function (cont.)

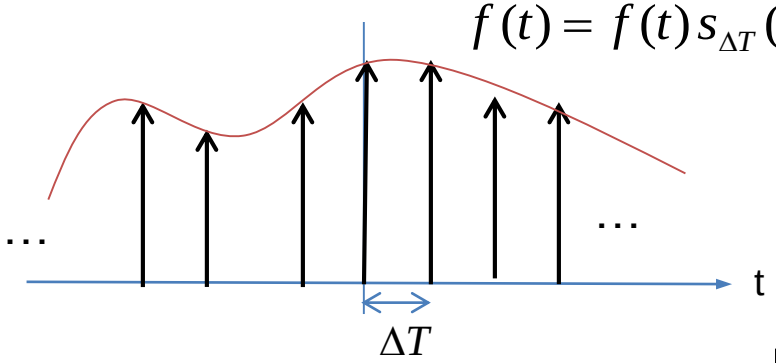
$$\tilde{F}(\mu)=\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}F(\mu-\frac{n}{\Delta T})$$



$\Im$



$$\tilde{f}(t)=f(t)s_{\Delta T}(t)$$

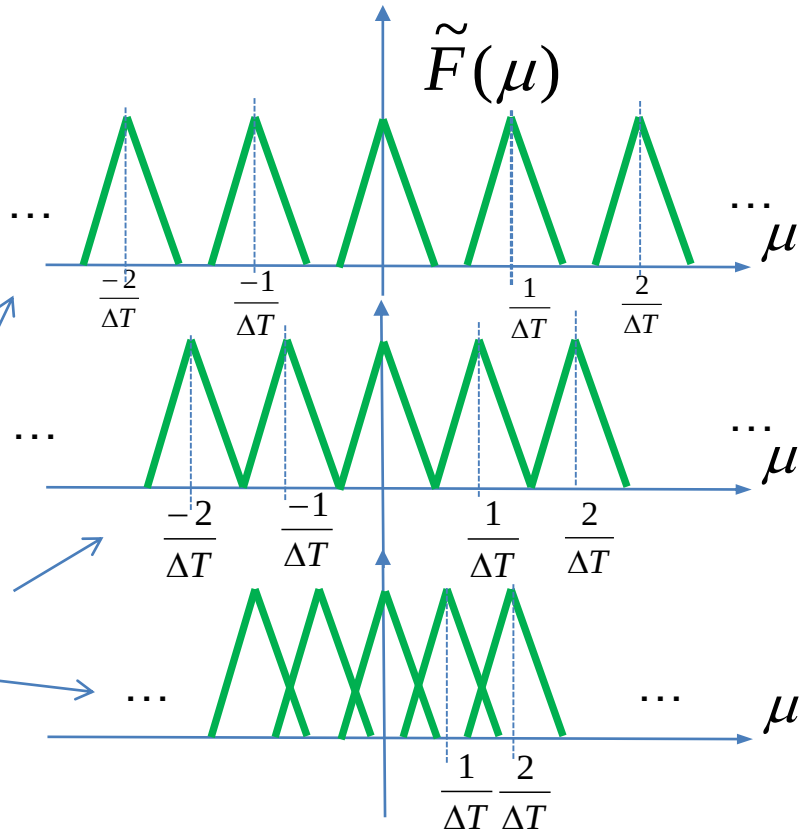


$\Im$

Over sampling

Critically sampling $\frac{1}{\Delta T}=2\mu_{\max}$

Under sampling $\frac{1}{\Delta T}<2\mu_{\max}$



1-D Sampling Theorem

- ✓ The condition [redacted] shows that a continuous , band limited function can be converted without loss of information from a set of its samples if the samples are obtained at a rate exceeding twice the highest frequency content of the function

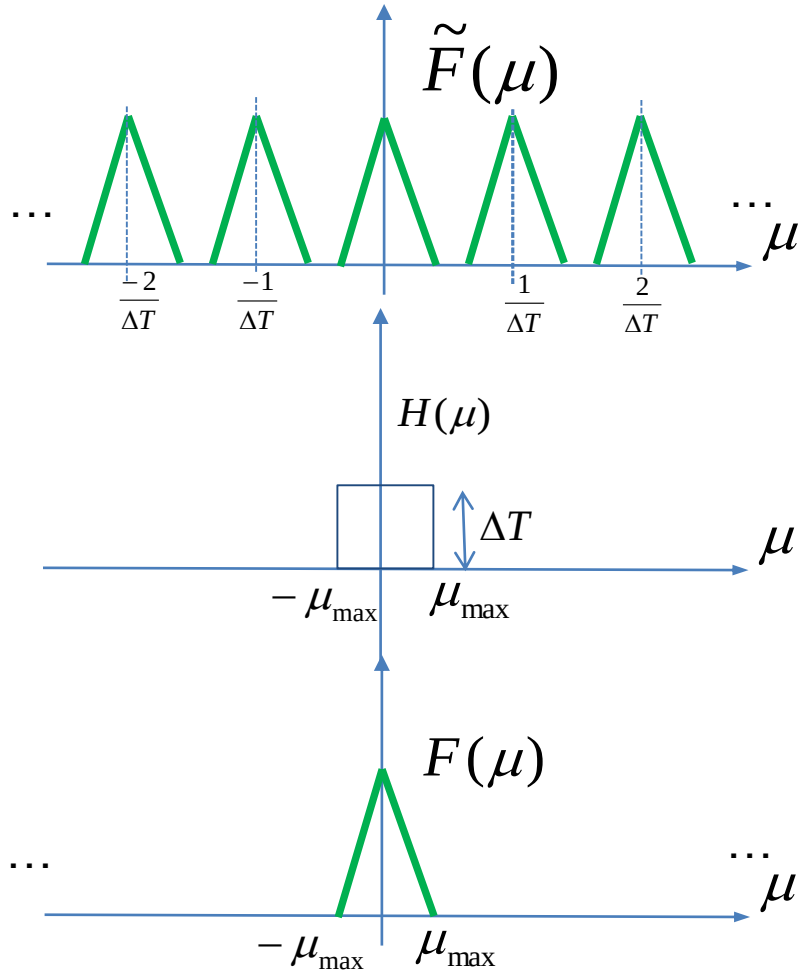
- ✓ To recover the main function from the sampled function, we need a lowpass filter. Lets consider $H(\mu)$ as the low pass filter that helps us to convert $\tilde{F}(\mu)$ to $F(\mu)$

$$H(\mu) = \begin{cases} \Delta T & -\mu_{\max} \le \mu \le \mu_{\max} \\ 0 & otherwise \end{cases}$$

$$F(\mu) = H(\mu)\,\tilde{F}(\mu)$$

- ✓ By using inverse Fourier transform, the function $f(t)$ can be recovered

$$\Im^{-1}[F(\mu)] = f(t)$$



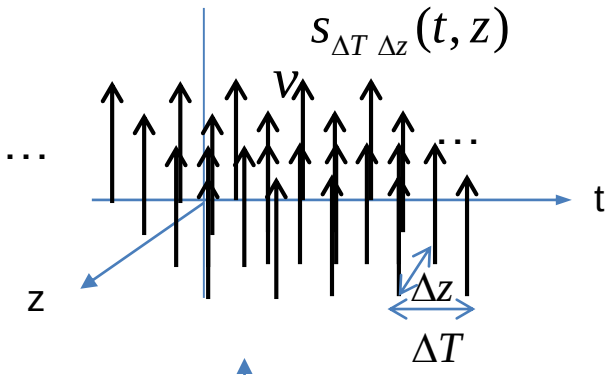
1-D Aliasing

- ✓ The Fourier transform of a sampled band-limited function is periodic. If the function has been sampled at a rate less than twice its highest frequency (under-sampling), then
  - ❖ The periods of the transform overlap with each other
  - ❖ As a result of this overlap, High frequency components of a function appears in low frequency components of the sampled function
  - ❖ No filter can filter a single period
  - ❖ Applying inverse Fourier transform will result in a corrupted function
- ✓ The effect caused by under-sampling is called aliasing
- ✓ Also, if a function is not band limited, we should see the aliasing effect. Since , in practice, we always limit the duration of a function to an interval such as [0, T], the sampled function is not band-limited anymore.
- ✓ The amount of aliasing can be reduced by smoothing the input function before sampling to attenuate the function's higher frequency. This process is called anti-aliasing. No action can be done after aliasing happened.

2-D sampling and aliasing

- ✓ In a manner similar to 1-D sampling, the sampling process in 2-D case can be modeled using a 2-D impulse train. If ΔT and Δz are the separations between samples along t and z axis, then the impulse train is

$$s_{\Delta T\,\Delta z}(t,z) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\delta(t-m\Delta T, z-n\Delta z)$$

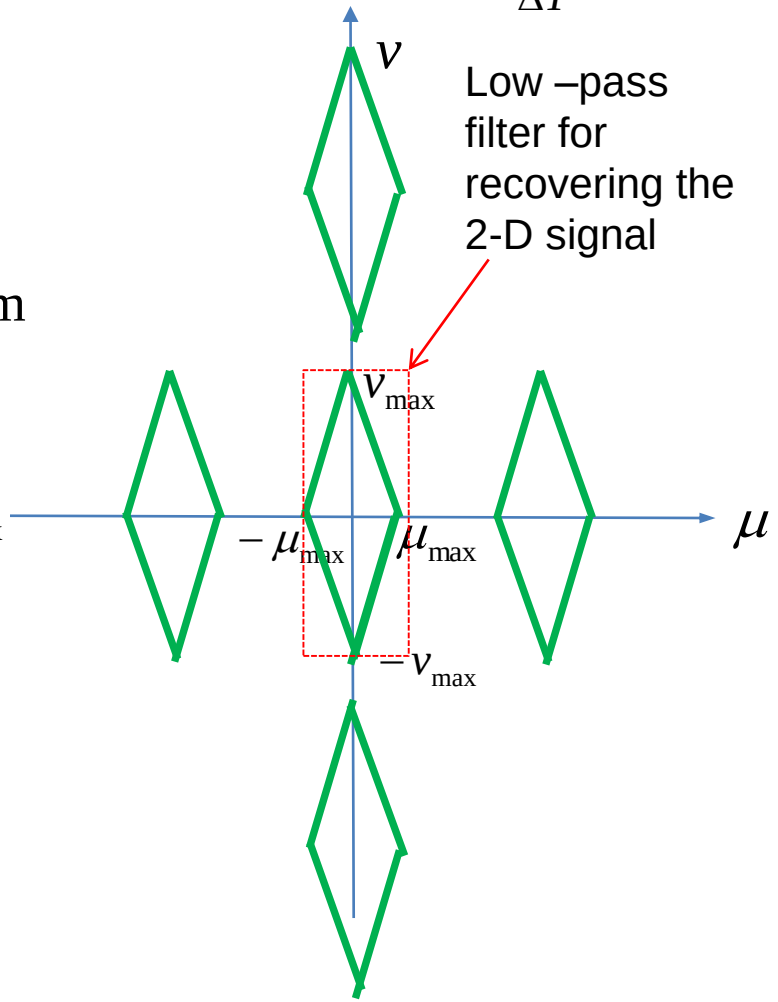


- ✓ for a 2-D band-limited function such as
  - ❖ Band-limited means that the Fourier transform of the function is zero outside of a rectangle-shape region defined by

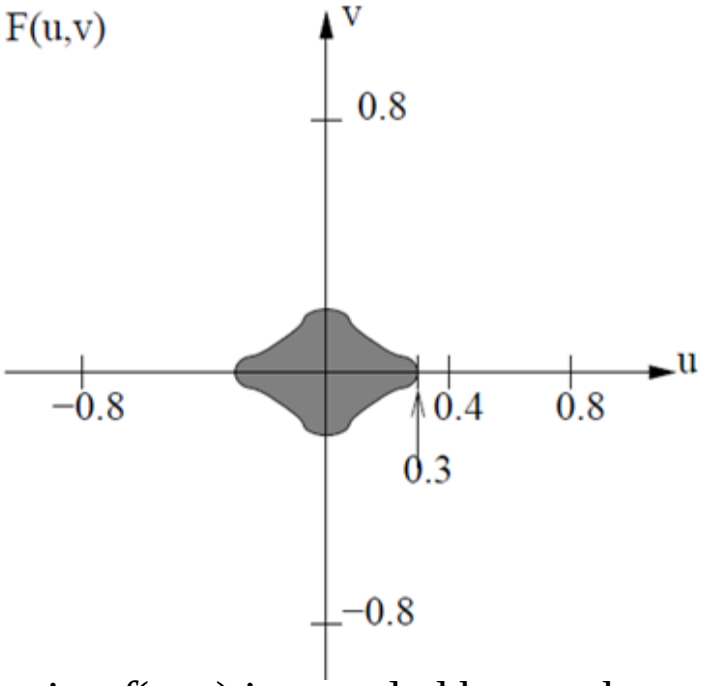
$$F(\mu,v)=0 \quad when \quad |\mu| \geq \mu_{max} \;\&\; |v| \geq v_{max}$$

  - ❖ Sampling theorem says that the function can be recovered from its samples if the sampling intervals are

$$\frac{1}{\Delta T} > 2\mu_{max} \qquad \frac{1}{\Delta z} > 2v_{max}$$



2-D sampling / aliasing example



✓ The function $f(x, y)$ has a 2D Fourier transform $F(u, v)$ as shown here. The function $f(x, y)$ is sampled by a 2D train to

$$g(x,y) = f(x,y)\sum_n \delta(x - n/0.8)\sum_n \delta(y - m/0.8)$$

a) Sketch $G(u, v)$ in the (u, v)-plane and grade the axes

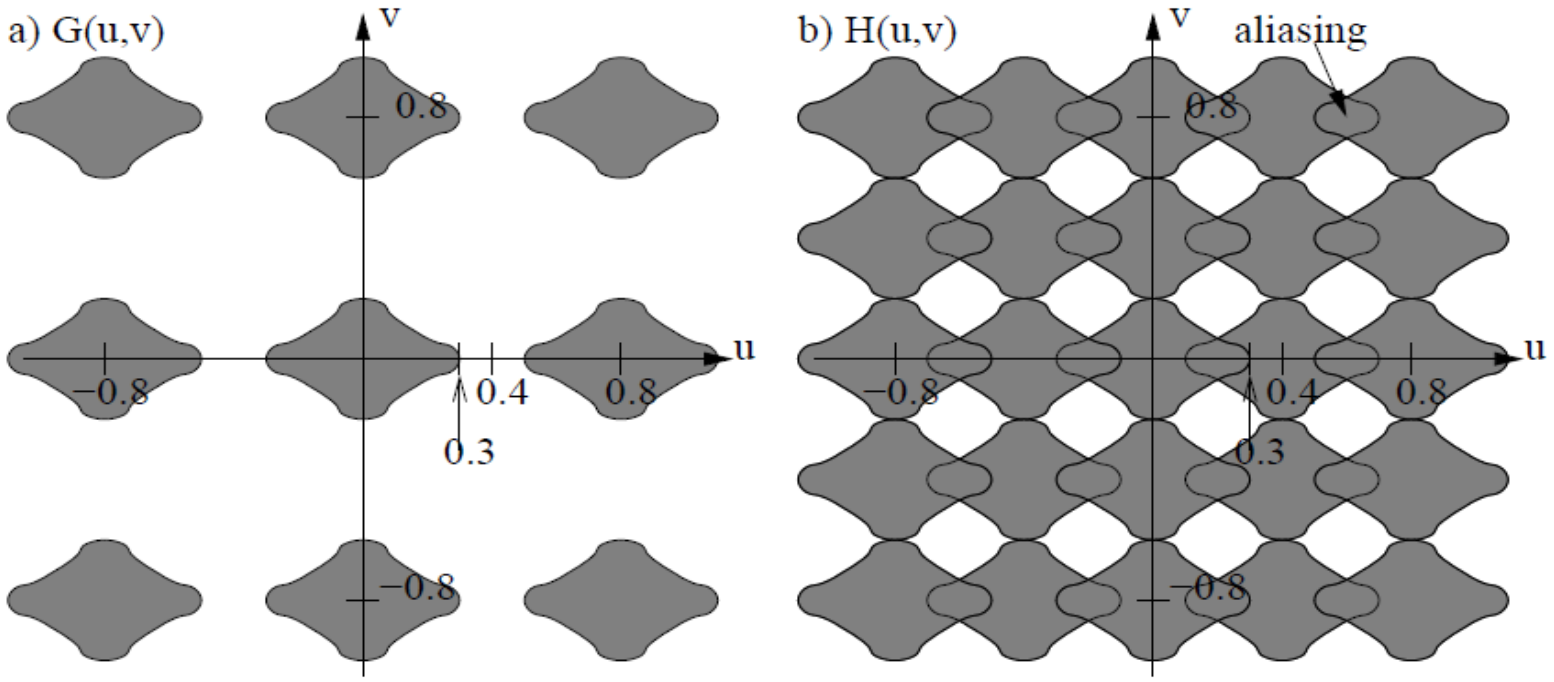
b) Sketch $H(u, v)$ in the (u, v)-plane and grade the axes if function $f(x, y)$ is sampled by another 2D impulse train to

$$h(x,y) = f(x,y)\sum_n \delta(x - n/0.4)\sum_n \delta(y - m/0.4)$$

c) One of the two functions $g(x, y)$ and $h(x, y)$ in a) and b) shows aliasing. Which one?

d) How does aliasing show up in an image? See the pattern below. Will the distance between the stripes be longer or shorter after aliasing? Will the direction of the stripes change or not

2-D sampling / aliasing example (cont.)



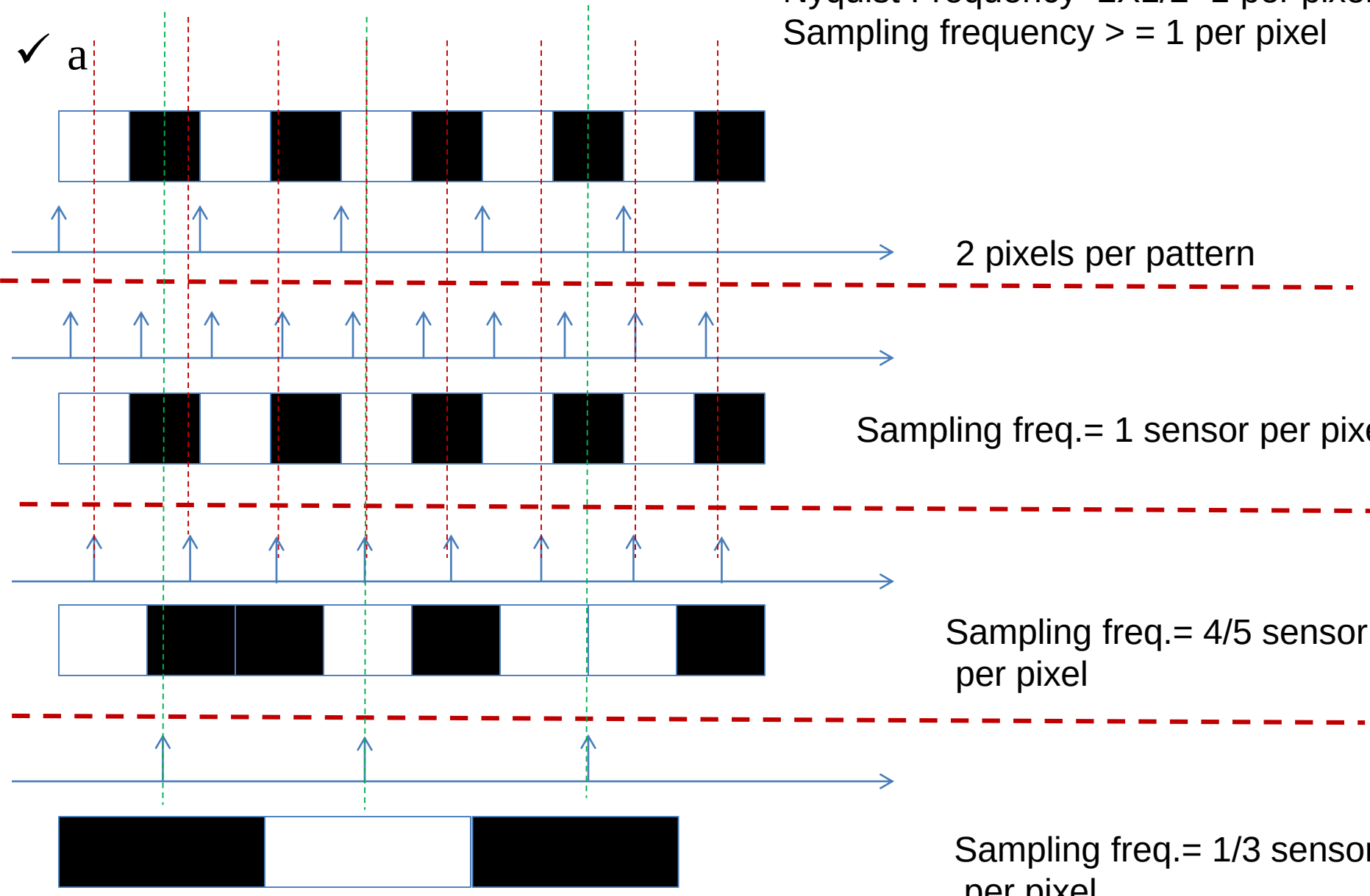
c) There is aliasing in the function $H(u, v)$. There is 20 aliasing areas shown in the figure and one of them is marked with an arrow.

d) After aliasing, the distance between the stripes will be longer. The direction of the stripes may change. The actual direction depends on the original direction and the sampling frequency.

Image Aliasing Example

2 pixels per pattern
Frequency: ½ per pixel
Nyquist Frequency=2X1/2=1 per pixel
Sampling frequency > = 1 per pixel

✓ a



1-D Discrete Fourier transform DFT

- ✓ Given a set of $\{f(x)\}$ consisting of M samples of $f(t)$, Discreet Fourier transform (DFT) yields a sample set $\{F(u)\}$ of M complex discrete value which is defined as

$$F(u) = \sum_{x=0}^{M-1} f(x)\, e^{-j2\pi ux/M} \quad u = 0,1,2,...,M-1$$

- ✓ The Inverse DFT (IDFT) can also be expressed as

$$f(x) = \frac{1}{M} \sum_{u=0}^{M-1} F(u)\, e^{j2\pi ux/M} \quad x = 0,1,2,...,M-1$$

- ✓ Example

t	f(t)
0	2
1	1
2	4
3	3

DFT →

U	F(u)
0	10
1	-2+2j
2	2
3	-2-2j

2-D Discrete Fourier transform (DFT)

- ✓ For 2-D DFT, 1-D DFT cab be expanded as

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi(ux/M+vy/N)} \qquad \begin{matrix} u = 0,1,2,...,M-1 \\ v = 0,1,2,...,N-1 \end{matrix}$$

- ✓ The 2-D inverse DFT can also expressed as

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\, e^{j2\pi(ux/M+vy/N)} \qquad \begin{matrix} x = 0,1,2,...,M-1 \\ y = 0,1,2,...,N-1 \end{matrix}$$

- ✓ For calculating any value of F(u,v), you should use all pixels of f(x,y). F(u,v) are complex values, which can be expressed in polar form

$$F(u,v) = R(u,v) + j\ I(u,v)$$

$$F(u,v) = |F(u,v)|\, e^{j\varphi(u,v)}$$

Polar form

$$|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2}$$

Magnitude, Fourier spectrum

$$\varphi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right]$$

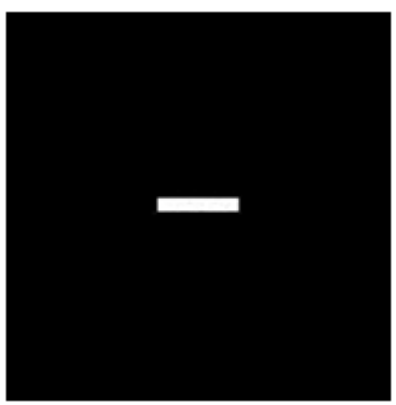
Phase angle

$$P(u,v) = R^2(u,v) + I^2(u,v)$$

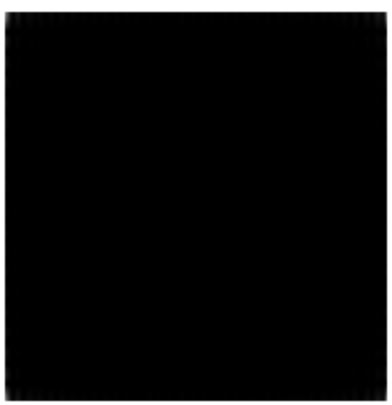
Magnitude, Fourier spectrum

2-D DFT of an image

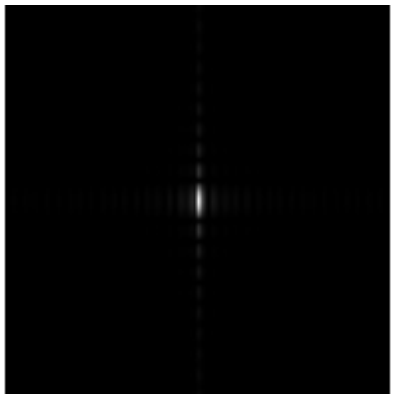
- ✓ F(0,0) is called the DC component since $F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{0}$ calculates the sum of all values in f(x,y)
- ✓ 2D FFT and its inverse in Matlab can be implemented using fft2 and ifft2 function.
- ✓ Some observation after applying fft on a giveen image as shown below
  - ❖ DFT of original image shows some values around the corners of the image.
  - ❖ DFT is periodic and for visualization purposes we shift the DFT results in such a way that zero frequency component goes to the center of the image
  - ❖ The log transformation shows more detail. The log display can be presented as shows $1+\log|F(u,v)|$.



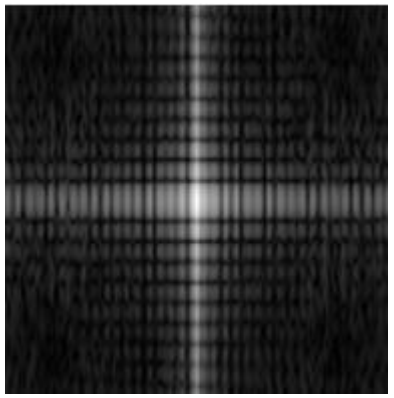
Original image



2D FFT

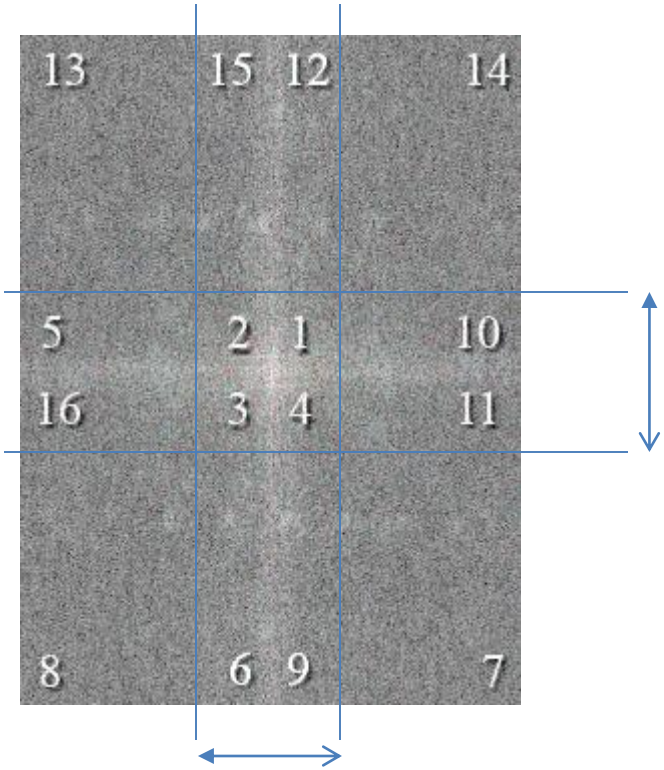


2D FFT- centered

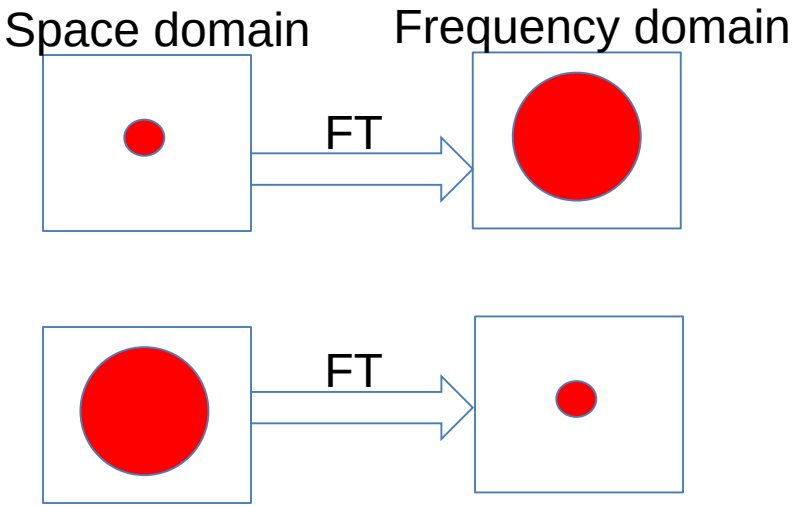


2D FFT- after a Log transformation

Image Frequency domain concept

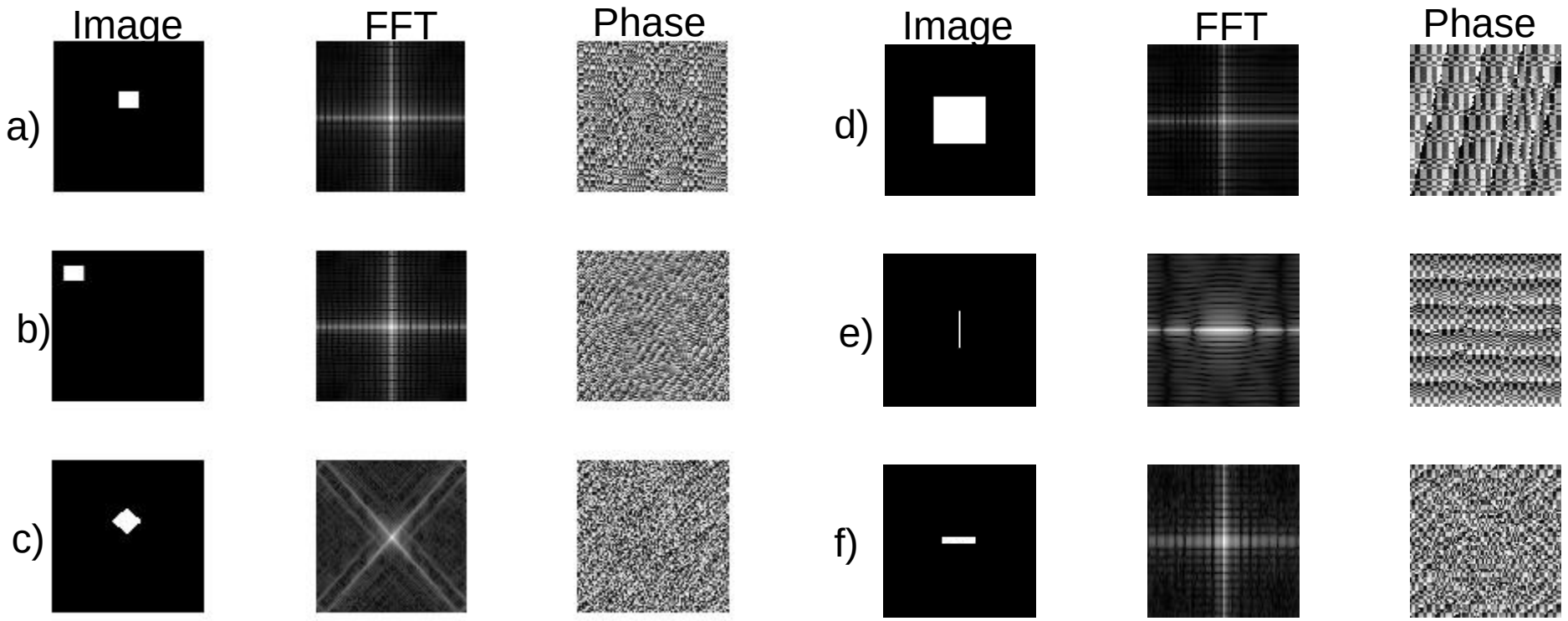


1,2,3,4: low frequency row and column
5,16,10,11: low frequency row , high frequency column
6,9,12,15: high frequency row , low frequency column
7,8,13,14: high frequency row , high frequency column



2-D Fourier transform properties

- ✓ justify the FFT and phase for each of the following images
  - ❖ Image a is the original image consists of a white rectangle and black background
  - ❖ In Image b, the rectangle is translated
  - ❖ In image c, the rectangle is rotated
  - ❖ In image d, the rectangle is enlarged
  - ❖ Image e and f are just thin vertical and horizontal lines



Filtering in Frequency domain

Filtering in frequency domain Examples

- ✓ Low pass filter (LPF):
  - ❖ LPFs attenuate the high frequency components of the Fourier Transform of an image, while leaving the low frequency components unchanged.
  - ❖ The typical overall effect of applying a LPF to an image is a controlled degree of blurring.
  - ❖ The following examples will be discussed
    - ▪ Ideal LPF
    - ▪ Butterworth LPF
    - ▪ Gaussian LPF
- ✓ High pass filter (HPF)
  - ❖ HPFs attenuate the low frequency components of the Fourier Transform of an image, while leaving the high frequency components unchanged.
  - ❖ The typical overall effect of applying a HPF to an image is a controlled degree of sharpening.
  - ❖ The following examples will be discussed
    - ▪ Ideal HPF
    - ▪ Butterworth HPF
    - ▪ Gaussian HPF

Frequency domain filtering Fundamental

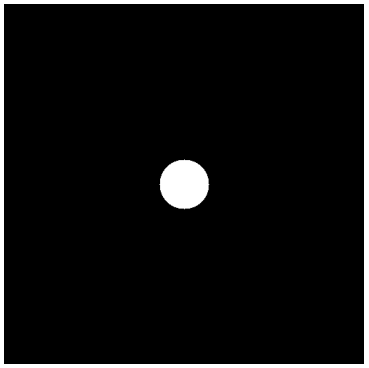
- ✓ Filtering in frequency domain means
  - ❖ Finding the Fourier transform of the image $f(x,y)$ as $F(\mu,\nu)$
  - ❖ Modifying the Fourier transform using a filter function $H(\mu,\nu)$
  - ❖ Computing the inverse Fourier transform to obtain the processed image

$$g(x,y) = \Im^{-1}\left[H(\mu,\nu)\ F(\mu,\nu)\right]$$

- ✓ If image $f(x,y)$, has a size of $M \times N$, then functions $F(\mu,\nu)$, $H(\mu,\nu)$ and the processed image g(x,y) are matrixes of size $M \times N$

Example I- ideal Low Pass Filter (ILPF)

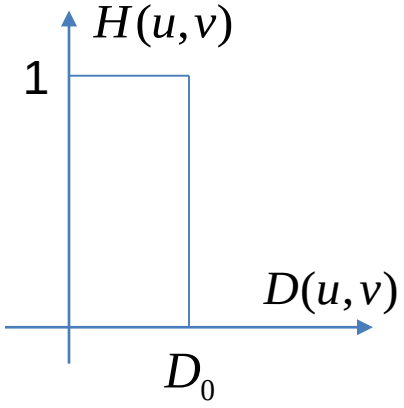
Ideal LPF



- ✓ If P and Q are the sizes of the image, a 2-D ideal LPF which does not attenuate the frequencies within a circle of radius D_0 and filters all frequencies outside this circle can be defined as

$$H(u,v) = \begin{cases} 1 & D(u,v) \le D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

$$D(u,v) = [(u - P/2)^2 + (v - Q/2)^2]$$



Result of LPF filtering
1- Lower values of D0 generates blurrier images
2- it generates ringing artifacts because of sharp transition between pass-band and stop-band in ideal LPF

Original image

$D_0 = 15$

$D_0 = 20$

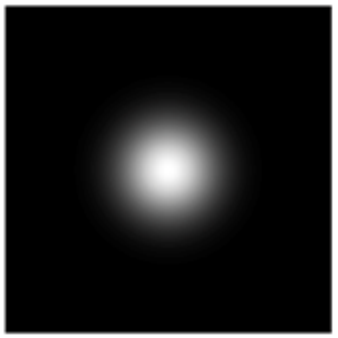
$D_0 = 25$

$D_0 = 30$

$D_0 = 40$

$D_0 = 50$

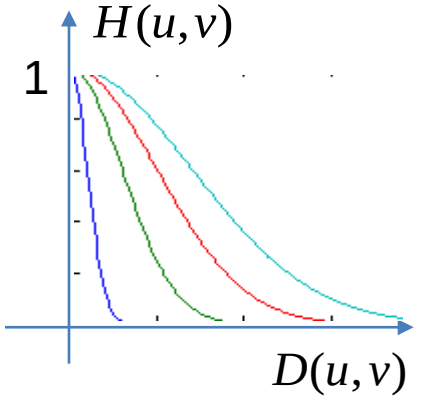
Example II- Butterworth LPF (BLPF)



- ✓ A 2-D BLPF order n with cut-off frequency D_0 from the origin can be defined as

$$H(u,v) = \frac{1}{1+\left[D(u,v)/D_0\right]^{2n}}$$

- ✓ BLPF does not have a sharp transition between pass-band and stop-band. The cut off frequency D_0 is usually defined as the location that $H(u,v)$ is 0.5(50% down its max value 1)
- ✓ Ringing artifact does not happen in BLPF order 1 and it is not significant in BLPF order 2, but can become significant in higher-order filters.
- ✓ BLPF of order 20 or higher is the same sharp transition between stop-band and pass-band as ideal LPF.
- ✓ BLPF order 2 is a good in terms of both low ringing artifacts and being an effective LPF.



Example II- Butterworth LPF (BLPF)-Cont.

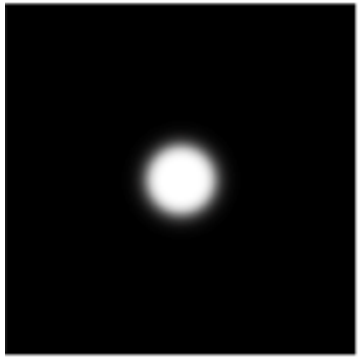
- ✓ The first row shows filtering with BLPF n=2 and various cut-off frequencies
- ✓ The second row shows ideal at the same cut-off frequencies for comparison

Original image

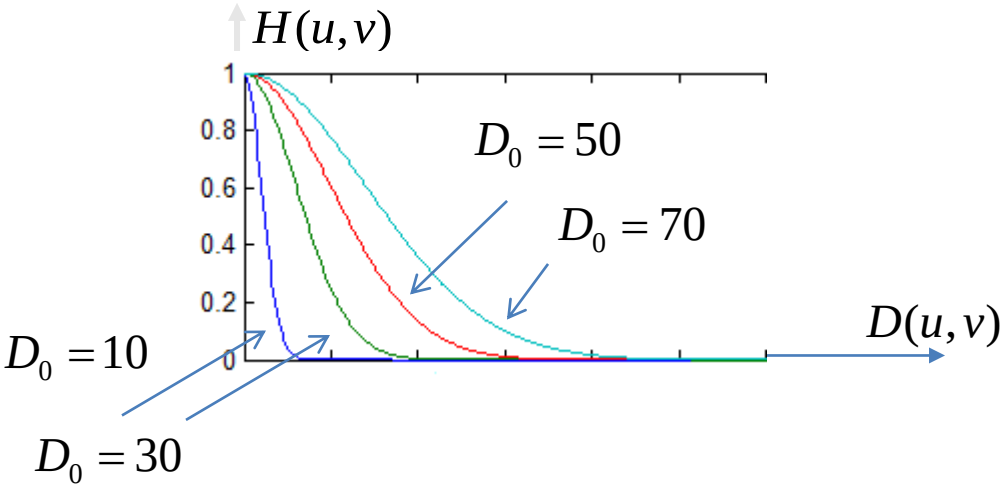
$D_0 = 15$ $D_0 = 20$ $D_0 = 25$ $D_0 = 30$ $D_0 = 40$ $D_0 = 50$

Example III- Gaussian LPF (GLPF)

Gaussian LPF

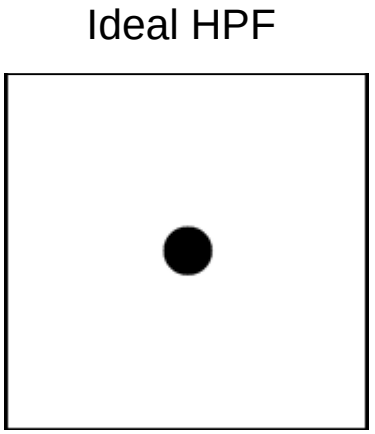


- ✓ A 2-D GLPF can be defined as $H(u,v) = e^{-D^2(u,v)/2D_0^2}$
- ✓ Where $D(u,v)$ is the distance from the center of frequency rectangle and D_0 is the cut-off frequency.



- ✓ GPLF does not produce ringing artifact, since the transition between stop-band and pass-band is not sharp and is controlled the value of by D_0

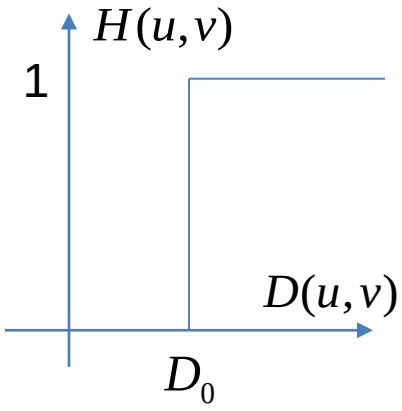
Example VI- ideal High Pass filter (HPF)



- ✓ A 2-D ideal HPF which filters frequencies within a circle of radius D_0 and does not attenuate frequencies outside this circle can be defined as

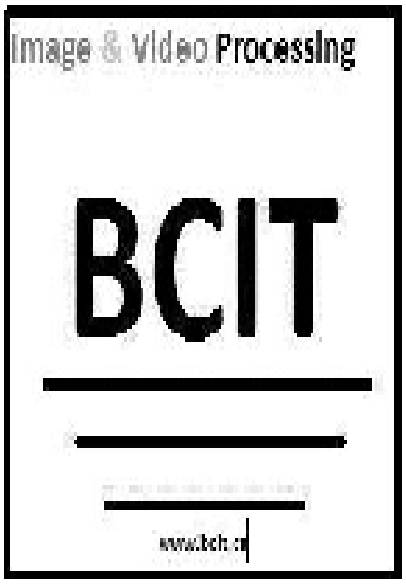
$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_0 \\ 1 & D(u,v) > D_0 \end{cases}$$

- ✓ As ILPF, ideal HPF is not physically realizable.
- ✓ It produces ringing artifacts similar to ideal LPF. The reason for ringing artifact is the sharp transition between pass-band and stop-band.
- ✓ Since low frequency components of the image are strongly attenuated, we may lose information present in large part of the original image.



Example VI- ideal High Pass filter (IHPF)

- ✓ The ringing in first image filtered by D_0=10 is severe and has produce distorted edges and boundaries.
- ✓ Edges that are not strong and their intensities are very close to the background would be vanished
- ✓ As D_0 increases , we see filtering in small fonts, but with $D_0 = 10$, we can't see the small fonts
- ✓ As D_0 increases, the edges are less distorted



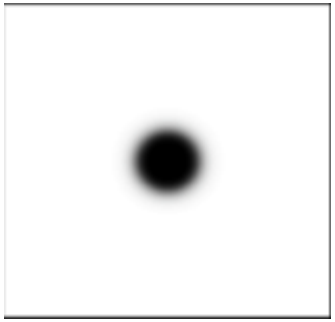
Original image

$D_0 = 10$

$D_0 = 50$

Example V- Butterworth HPF (BHPF)

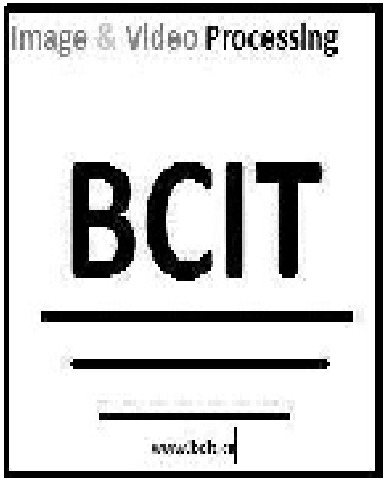
Butterworth HPF



- ✓ A 2-D BLPF order n with cut-off frequency D_0 from the origin can be defined as

$$H(u,v) = \frac{1}{1+\left[D_0 / D(u,v)\right]^{2n}}$$

- ✓ BLPF acts smoother than ideal high pass filter. This means that the edges are less distorted. However, we should not see much difference in filtering small fonts or objects and the objects and fonts with intensities close to the background.



Original image

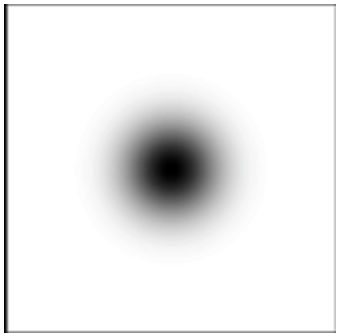
$D_0 = 10, n = 2$

$D_0 = 30, n = 2$

$D_0 = 50, n = 2$

Example VI- Gaussian HPF (GHPF)

Gaussian LPF



- A 2-D GLPF can be defined as

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

- The result of filtering using GHPF would be comparable with the results of BHPF , but somehow more gradual than BHPF. In other word, the edges must be somehow cleaner and less distorted.

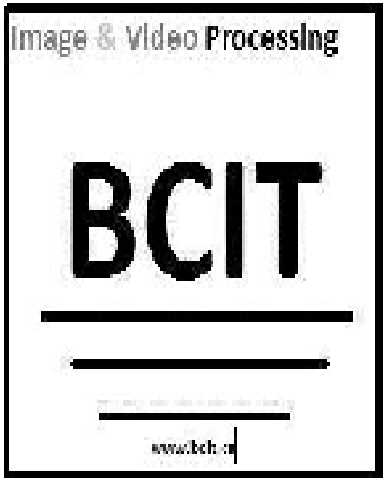
High frequency Emphasis

- ✓ When a high pass filter is applied to an image, the low –frequency data in the image are severely attenuated or completely will be filtered.
- ✓ Filtering low frequency leads to losing info which exist in large part of the image
- ✓ High frequency emphasis is the name of a technique that can be used to keeps the low frequency component, while enhance the high – frequency component.

$$H_{hfe}(u,v) = a + bH(u,v)$$

The transfer function of a high pass filter

The constant for enhancing high frequency component b>a



Original image

$BHPF : D_0 = 30, n = 2$

$H_{hfe}(u,v) = 0.5 + BHPF(u,v)$

$BHPF : D_0 = 30, n = 2$

$a = 0.5 \;\; \& b = 1$

Assignment #4

1) An image has defined as $f(x,y) = \sin(2x+3y)$, find the Fourier transform of this image $F(u,v)$?

2) An image f (x,y) is padded with zeros and the result is called g(x,y). Explain the difference you think will appear in F(u,v) and G(u,v)?

3) Find the filter transfer function H(u,v) for a filter that averages the four immediate neighbors of a point (x,y) , but does not include f(x,y)?

4) A Gaussian low pass filter with $\sigma = 1$ has a the transfer function of $H_{lp}(u,v) = A\exp[\frac{-(u^2+v^2)}{2}]$,Find and $h_{lp}(x,y)$ also find $h_{Hp}(x,y)$

5) Consider a 2D image $f(x,y) = \cos(4\pi x)\times\sin(8\pi y)$

a) determine its Fourier transform $F(u,v)$ and illustrate the spectrum in Fig. (a).

b) suppose this signal is sampled uniformly with sampling interval $\Delta x = \Delta y = \Delta = \frac{1}{6}$. Draw the spectrum of the sampled signal in Fig. (b).

c) suppose the sampled signal is filtered by an ideal low-pass filter $h_1(x,y)$ with frequency response

$$H_1(u,v) = \begin{cases} \Delta^2 & \frac{-f_s}{2} \prec u, v \prec \frac{f_s}{2} \\ 0 & Otherwise \end{cases}$$

Where, $f_s = \frac{1}{\Delta}$. Draw the spectrum of the reconstructed signal in Fig. (c).

Give the spatial representation of the reconstructed signal $f_{r1}(x,y)$.

DCT

1-D DCT and IDCT

- ✓ There have been designed a variety of transformation for transforming the data to frequency domain . So far, we have studied DFT.
- ✓ Example of other frequency domain transformation are Walsh-Hadamard (WHT), Discreet cosine transform(DCT) and KLT
- ✓ DCT has been used widely in image processing and it is somehow similar to DFT
- ✓ 1-D DCT is defined as

Input data

Forward Kernels (basis function)

DCT

$$T(u) = \sum_{x=0}^{n-1} g(x)\, r(x,u)$$

Inverse Kernels

$$g(x) = \sum_{u=0}^{n-1} T(u)\, s(x,u)$$

IDCT

$$r(x,u) = s(x,u) = \alpha(u) \cos\left[\frac{(2x+1)u\pi}{2n}\right]$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{n}} & if \;\; u = 0 \\ \sqrt{\frac{2}{n}} & if \;\; u = 1,2,\ldots,n-1 \end{cases}$$

1-D DCT-Example

Input data

Kernels or basis function

$$T(u) = \sum_{x=0}^{n-1} g(x)\ r(x,u)$$

$$r(x,u) = \alpha(u)\ \cos\left[\frac{(2x+1)u\pi}{2n}\right]$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{n}} & if\ \ u = 0 \\ \sqrt{\frac{2}{n}} & if\ \ u = 1,2,\ldots,n-1 \end{cases}$$

$$n = 4, \quad g(x) = [2 \quad 3 \quad 4 \quad 5]$$

$$T(0) = \sum_{x=0}^{3} g(x)\left[\frac{1}{2}\cos(0)\right] = \frac{1}{2}(g(0)+g(1)+g(2)+g(3)\ ) = 7$$

$$T(1) = \sum_{x=0}^{3} g(x)\left[\frac{\sqrt{2}}{2}\cos(\frac{(2x+1)\pi}{8})\right] = \frac{\sqrt{2}}{2}[g(0)\cos(\frac{\pi}{8}) + g(1)\cos(\frac{3\pi}{8}) + g(2)\cos(\frac{5\pi}{8}) + g(3)\cos(\frac{7\pi}{8})\ ] = -2.23$$

$$T(2) = \sum_{x=0}^{3} g(x)\left[\frac{\sqrt{2}}{2}\cos(\frac{(2x+1)2\pi}{8})\right] = \frac{\sqrt{2}}{2}[g(0)\cos(\frac{\pi}{4}) + g(1)\cos(\frac{3\pi}{4}) + g(2)\cos(\frac{5\pi}{4}) + g(3)\cos(\frac{7\pi}{4})\ ] = 0$$

$$T(3) = \sum_{x=0}^{3} g(x)\left[\frac{\sqrt{2}}{2}\cos(\frac{(2x+1)3\pi}{8})\right] = \frac{\sqrt{2}}{2}[g(0)\cos(\frac{3\pi}{8}) + g(1)\cos(\frac{9\pi}{8}) + g(2)\cos(\frac{15\pi}{8}) + g(3)\cos(\frac{21\pi}{8})\ ] = -0.15$$

$$T(u) = [7 \quad -2.23 \quad 0 \quad -0.15\ ]$$

1-D inverse DCT-Example

$$n=4,\quad T(u)=[7\quad -2.23\quad 0\quad -0.15]$$

$$g(0)=\sum_{u=0}^{3}T(u)\left[\alpha(u)\cos(\frac{(2x+1)u\pi}{2n})\right]=[\frac{1}{2}\times 7\cos(0)+\frac{\sqrt{2}}{2}(-2.23)\cos(\frac{\pi}{8})+\frac{\sqrt{2}}{2}(0)\cos(\frac{2\pi}{8})+\frac{\sqrt{2}}{2}(-0.15)\cos(\frac{3\pi}{8})\ )=2$$

$$g(1)=\sum_{u=0}^{3}T(u)\left[\alpha(u)\cos(\frac{(2x+1)u\pi}{2n})\right]=[\frac{1}{2}7\cos(0)+\frac{\sqrt{2}}{2}(-2.23)\cos(\frac{3\pi}{8})+\frac{\sqrt{2}}{2}(0)\cos(\frac{6\pi}{8})+\frac{\sqrt{2}}{2}(-0.15)\cos(\frac{9\pi}{8})\ )=3$$

$$g(2)=\sum_{u=0}^{3}T(u)\left[\alpha(u)\cos(\frac{(2x+1)u\pi}{2n})\right]=\frac{1}{2}7\cos(0)+\frac{\sqrt{2}}{2}(-2.23)\cos(\frac{5\pi}{8})+\frac{\sqrt{2}}{2}(0)\cos(\frac{10\pi}{8})+\frac{\sqrt{2}}{2}(-0.15)\cos(\frac{15\pi}{8})\ )=4$$

$$g(3)=\sum_{u=0}^{3}T(u)\left[\alpha(u)\cos(\frac{(2x+1)u\pi}{2n})\right]=\frac{1}{2}7\cos(0)+\frac{\sqrt{2}}{2}(-2.23)\cos(\frac{7\pi}{8})+\frac{\sqrt{2}}{2}(0)\cos(\frac{14\pi}{8})+\frac{\sqrt{2}}{2}(-0.15)\cos(\frac{21\pi}{8})\ )=5$$

$$g(x)=[2\quad 3\quad 4\quad 5]$$

$$n=4,\quad T(u)=[7\quad -2.23\quad 0\quad 0]$$

In MATLAB

```
f=[7 -2.23 0 0]
p=idct(f)
```

p =

2.0432 2.8966 4.1034 4.9568

1-D DCT Kernel (basis function)

Forward Kernels (basis function)

Input data

$$T(u) = \sum_{x=0}^{n-1} g(x)\, r(x,u)$$

DCT

Inverse Kernels

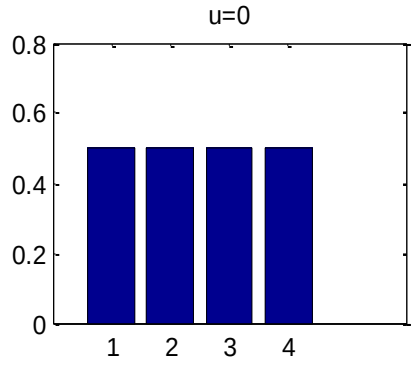
$$g(x) = \sum_{u=0}^{n-1} T(u)\, s(x,u)$$

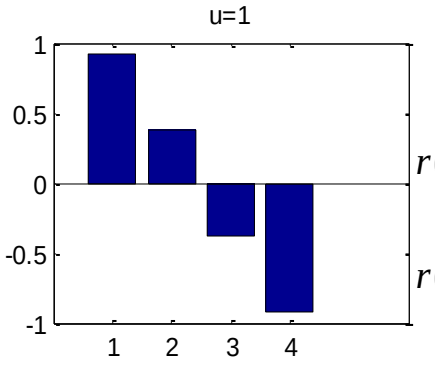
IDCT

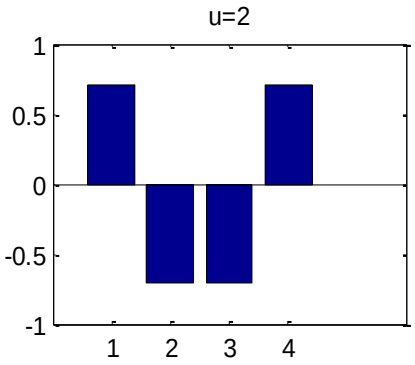
$$r(x,u) = s(x,u) = \alpha(u)\cos\left[\frac{(2x+1)u\pi}{2n}\right]$$

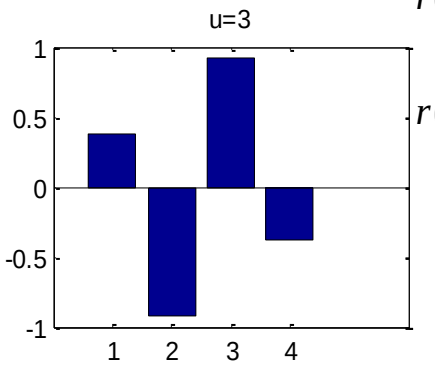
$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{n}} & \text{if } u = 0 \\ \sqrt{\frac{2}{n}} & \text{if } u = 1,2,\ldots,n-1 \end{cases}$$

For n=4









$$r(x,0) = h_0 = \frac{1}{2}[1 \quad 1 \quad 1 \quad 1]$$

$$r(x,1) = h_1 = \frac{\sqrt{2}}{2}[\cos(\frac{\pi}{8}) \quad \cos(\frac{3\pi}{8}) \quad \cos(\frac{5\pi}{8}) \quad \cos(\frac{7\pi}{8})]$$

$$r(x,2) = h_2 = \frac{\sqrt{2}}{2}[\cos(\frac{\pi}{8}) \quad \cos(\frac{6\pi}{8}) \quad \cos(\frac{10\pi}{8}) \quad \cos(\frac{14\pi}{8})]$$

$$r(x,3) = h_3 = \frac{\sqrt{2}}{2}[\cos(\frac{3\pi}{8}) \quad \cos(\frac{9\pi}{8}) \quad \cos(\frac{15\pi}{8}) \quad \cos(\frac{21\pi}{8})]$$

2-D DCT and IDCT

- ✓ 2-D DCT has been used widely in image processing and is more efficient than DFT

Forward Kernels (basis function)

DCT

Input data

$$T(u,v) = \sum_{x=0}^{n-1}\sum_{y=0}^{n-1} g(x,y)\, r(x,y,u,v)$$

IDCT

Inverse Kernels

$$g(x,y) = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\, s(x,y,u,v)$$

$$r(x,y,u,v) = s(x,y,u,v) = \alpha(u)\,\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2x+1)v\pi}{2n}\right]$$

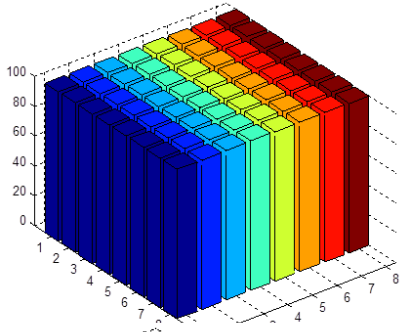
$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{n}} & if\ \ u = 0 \\ \sqrt{\frac{2}{n}} & if\ \ u = 1,2,\ldots,n-1 \end{cases}$$

2-D DCT Examples

✓ 2-D DCT has been used widely in image processing and is more efficient than DFT

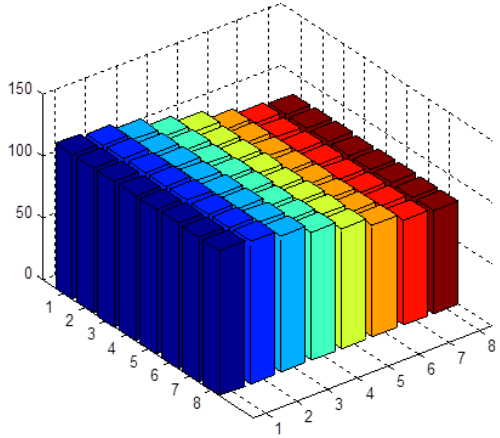
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100

800	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



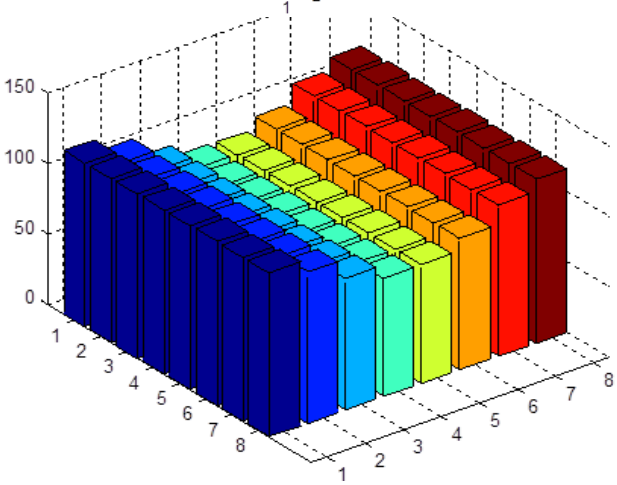
117	115	110	103	97	90	85	83
117	115	110	103	97	90	85	83
117	115	110	103	97	90	85	83
117	115	110	103	97	90	85	83
117	115	110	103	97	90	85	83
117	115	110	103	97	90	85	83
117	115	110	103	97	90	85	83
117	115	110	103	97	90	85	83

800	100	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



116	107	93	84	84	93	107	116
116	107	93	84	84	93	107	116
116	107	93	84	84	93	107	116
116	107	93	84	84	93	107	116
116	107	93	84	84	93	107	116
116	107	93	84	84	93	107	116
116	107	93	84	84	93	107	116
116	107	93	84	84	93	107	116

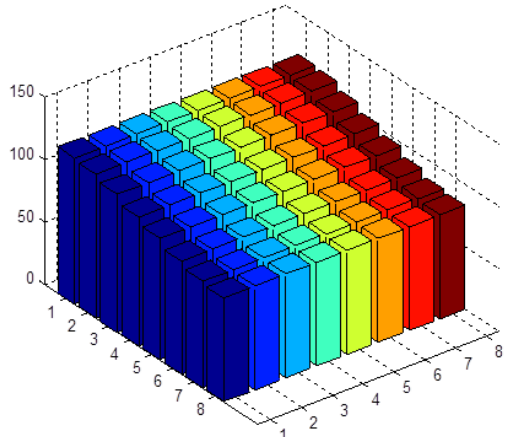
800	0	100	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



2-D DCT Examples

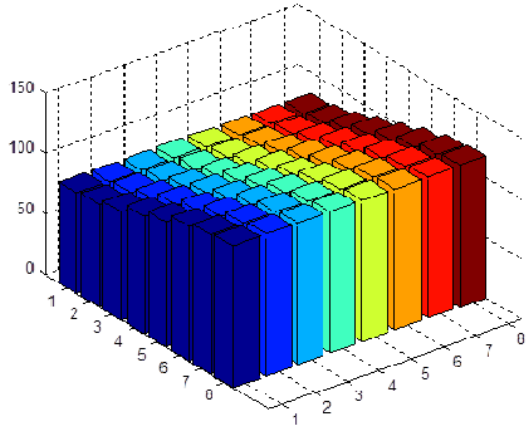
117	117	117	117	117	117	117	117
115	115	115	115	115	115	115	115
110	110	110	110	110	110	110	110
103	103	103	103	103	103	103	103
97	97	97	97	97	97	97	97
90	90	90	90	90	90	90	90
85	85	85	85	85	85	85	85
83	83	83	83	83	83	83	83

800	0	0	0	0	0	0	0
10 0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



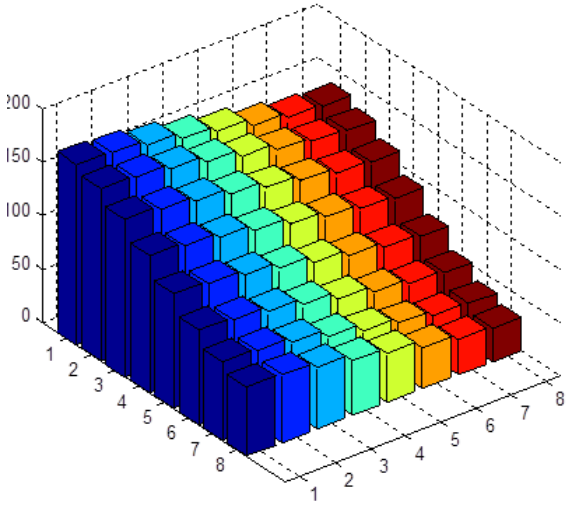
83	83	83	83	83	83	83	83
85	85	85	85	85	85	85	85
90	90	90	90	90	90	90	90
97	97	97	97	97	97	97	97
103	103	103	103	103	103	103	103
110	110	110	110	110	110	110	110
115	115	115	115	115	115	115	115
117	117	117	117	117	117	117	117

800	0	0	0	0	0	0	0
-100	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



169	167	162	155	149	142	137	135
161	159	154	148	141	134	129	127
147	144	139	133	126	120	115	112
128	125	120	114	107	101	96	93
107	104	99	93	86	80	75	72
88	85	80	74	67	61	56	53
73	71	66	59	52	46	41	39
65	63	58	51	45	38	33	31

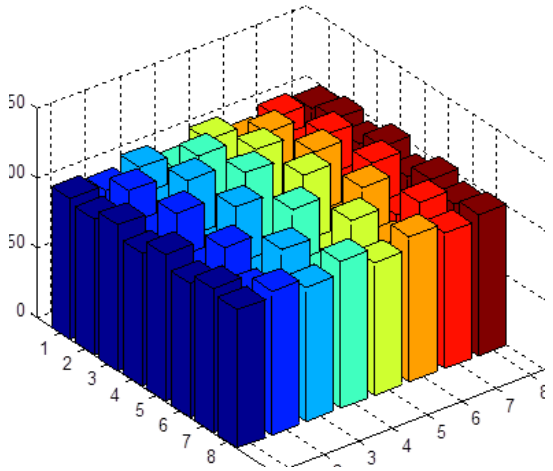
800	100	0	0	0	0	0	0
300	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



2-D DCT Examples

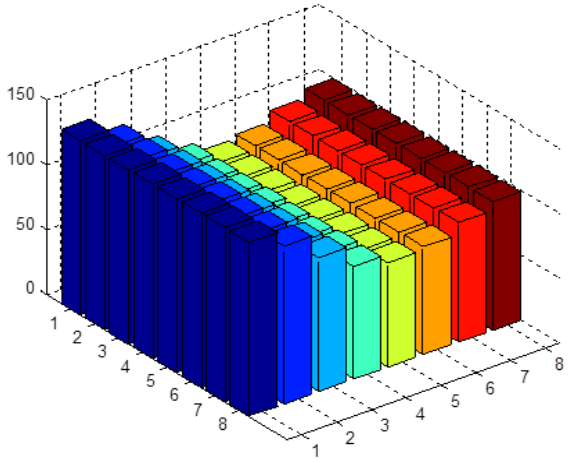
101	97	104	95	105	96	103	99
97	108	88	114	86	112	92	103
104	88	117	80	120	83	112	96
95	114	80	124	76	120	86	105
105	86	120	76	124	80	114	95
96	112	83	120	80	117	88	104
103	92	112	86	114	88	108	97
99	103	96	105	95	104	97	101

800	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	10 0



134	121	103	87	80	83	92	99
134	121	103	87	80	83	92	99
134	121	103	87	80	83	92	99
134	121	103	87	80	83	92	99
134	121	103	87	80	83	92	99
134	121	103	87	80	83	92	99
134	121	103	87	80	83	92	99
134	121	103	87	80	83	92	99

800	100	100	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



- ✓ 2-D DCT, is often used in signal and image processing, especially for lossy data compression, because it has a strong "energy compaction" property
- ✓ most of the signal information tends to be concentrated in a few low-frequency components of the DCT
- ✓ The DCT is used in JPEG image compression and MPEG video compression standards

2-D DCT Kernel

$$r(x,y,u,v)=\alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2x+1)v\pi}{2n}\right]$$

- ✓ How to build 2-D Kernel from a set of 1-D kernels
  - ❖ If there are some 1-D kernels such as h_0 ,h_1,h_2 ,h_3 and they are orthonormal, then you can calculate a set of 2-D kernel from them by $H_{ij}=(h_i)'h_j$
  - ❖ Orthonormality condition means $(h_i,h_j)=h_i(h_j)'=\begin{cases}1 & i=j\\ 0 & i\neq j\end{cases}$

To show orthonormality for 2-D DCT kernels, we should for all possible values of i and j. As an example show orthonormality for $(h_0,h_0),(h_0,h_1)$ and calculate H_{00}, H_{01}

1-D DCT

$$h_0=\frac{1}{2}[1\ \ 1\ \ 1\ 1]$$

$$h_1=\frac{\sqrt{2}}{2}[\cos(\frac{\pi}{8})\ \ \cos(\frac{3\pi}{8})\ \ \cos(\frac{5\pi}{8})\ \ \cos(\frac{7\pi}{8})]$$

$$h_2=\frac{\sqrt{2}}{2}[\cos(\frac{2\pi}{8})\ \ \cos(\frac{6\pi}{8})\ \ \cos(\frac{10\pi}{8})\ \ \cos(\frac{14\pi}{8})]$$

$$h_3=\frac{\sqrt{2}}{2}[\cos(\frac{3\pi}{8})\ \ \cos(\frac{9\pi}{8})\ \ \cos(\frac{15\pi}{8})\ \ \cos(\frac{21\pi}{8})]$$

$$(h_0,h_0)=\frac{1}{2}[1\ \ 1\ \ 1\ 1]\times\frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}=\frac{1}{4}\times 4=1\ ,(h_0,h_1)=\frac{1}{2}[1\ \ 1\ \ 1\ 1]\times\frac{\sqrt{2}}{2}\begin{bmatrix}\cos(\pi/8)\\ \cos(3\pi/8)\\ \cos(5\pi/8)\\ \cos(7\pi/8)\end{bmatrix}$$

$$(h_0,h_1)=\frac{\sqrt{2}}{4}\times[\cos(\pi/8)+\cos(3\pi/8)+\cos(5\pi/8)+\cos(7\pi/8)]=0$$

$$H_{00}=\frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}\times\frac{1}{2}[1\ \ 1\ \ 1\ \ 1]=\frac{1}{4}\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix}$$

$$H_{01}=(h_0)'(h_1)=\frac{\sqrt{2}}{4}\begin{bmatrix}\cos(\pi/8)&\cos(3\pi/8)&\cos(5\pi/8)&\cos(7\pi/8)\\ \cos(\pi/8)&\cos(3\pi/8)&\cos(5\pi/8)&\cos(7\pi/8)\\ \cos(\pi/8)&\cos(3\pi/8)&\cos(5\pi/8)&\cos(7\pi/8)\\ \cos(\pi/8)&\cos(3\pi/8)&\cos(5\pi/8)&\cos(7\pi/8)\end{bmatrix}$$

Building 2-D kernel- Example

✓ You are given two 1-D vectors below $h_0 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}, \quad h_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$

(a) Show that they are orthonormal vectors.

(b) Derive four 2-D basis images using these 1-D vectors.

(c) Calculate the transform of the image using the basis images derived in step (b).

(d) Find the reconstructed image $\hat{F}$ obtained with the largest coefficients (in magnitude). What can you say about the physical meaning of $\hat{F}$?

$$F = \begin{bmatrix}1 & 2\\3 & 5\end{bmatrix}$$

Solution

a)

$$(h_0, h_0) = \frac{1}{2}\begin{bmatrix}1 & 1\end{bmatrix}\times\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{2}\times 2 = 1 \ , (h_0, h_1) = \frac{1}{2}\begin{bmatrix}1 & 1\end{bmatrix}\times\begin{bmatrix}1\\-1\end{bmatrix} = 0$$

$$(h_1, h_0) = \frac{1}{2}\begin{bmatrix}1 & -1\end{bmatrix}\times\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{2}\times 02 = 0 \ , (h_1, h_1) = \frac{1}{2}\begin{bmatrix}1 & -1\end{bmatrix}\times\begin{bmatrix}1\\-1\end{bmatrix} = 1$$

Building 2-D kernel- Example (cont.)

(b)

$$H_{00} = \frac{1}{2}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix},\ H_{01} = \frac{1}{2}\times\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & -1\\1 & -1\end{bmatrix}$$

$$H_{10} = \frac{1}{2}\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & 1\\-1 & -1\end{bmatrix},\ H_{11} = \frac{1}{2}\times\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix}$$

(C)

$$t_{00} = F.H_{00} = \frac{1}{2}\begin{bmatrix}1 & 2\\3 & 5\end{bmatrix}.\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} = \frac{11}{2},\ t_{01} = F.H_{01} = \frac{1}{2}\begin{bmatrix}1 & 2\\3 & 5\end{bmatrix}.\begin{bmatrix}1 & -1\\1 & -1\end{bmatrix} = \frac{-3}{2}$$

$$t_{10} = F.H_{10} = \frac{1}{2}\begin{bmatrix}1 & 2\\3 & 5\end{bmatrix}.\begin{bmatrix}1 & 1\\-1 & -1\end{bmatrix} = \frac{-5}{2},\ t_{11} = F.H_{11} = \frac{1}{2}\begin{bmatrix}1 & 2\\3 & 5\end{bmatrix}.\begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix} = \frac{1}{2}$$

(d)

$$\hat{F} = t_{00}H_{00} = \frac{11}{4}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$$

11/4 is the average of all the pixels in the image. So building the image using t_{00}, produces an image that all the pixels has the average value of the original image.

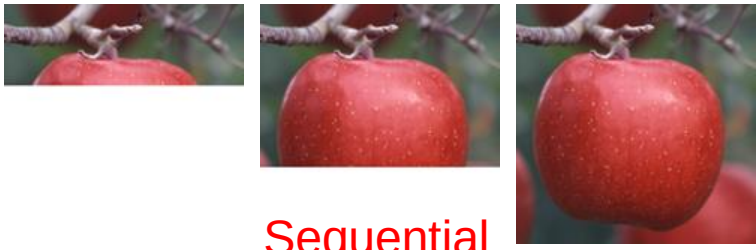
2-D DCT vs. DFT

- ✓ DCT express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. So DCT acts as any other Fourier-related transform.
- ✓ Similar to DFT, a DCT operates on a function at a finite number of discrete data points.
- ✓ DFT uses both cosines and sines (in the form of complex numbers). While DCT uses only cosine functions which is the real part of DFT.
- ✓ it is clear that any function with less transition and sharp discontinuities should be represented by fewer terms in its DFT or DCT and therefore compressed more effectively.
- ✓ Efficacy of a transformation scheme can be directly gauged by its ability to pack input data into as few coefficients as possible. This allows the quantizer to discard coefficients with relatively small amplitudes without introducing visual distortion in the reconstructed image. DCT exhibits excellent energy compaction for highly correlated images.

JPEG

- ✓ One of the most popular Image compression standards is the JPEG (Joint Photographic Experts Group) standard which uses DCT in one of its modes.
- ✓ JPEG uses block transform coding, in which the input image is first will be divided into small non-overlapping blocks, and then each small blocks will be processed independently.

JPEG modes of operation



Sequential

Progressive

- ✓ JPEG offers adjustable compression/quality and has 4 different modes of operations:
  - ❖ Sequential (line-by-line) (also called baseline implementation)
    - ▪ In sequential encoding, the whole image is encoded and decoded in a single run. It allows decoding with immediate presentation, but in a top-to-bottom sequence
  - ❖ Progressive (blur-to-clear) (also called layer coding)
    Progressive mode encodes and reconstructs the image with a very rough representation, and refines it during successive steps.
  - ❖ Hierarchical (multiple resolutions)
    down-sample by factors of 2 in both directions, (reduce 640×480 to 320×240 ,..)
    Repeat the following process recursively until the full resolution image is compressed. Initially, encode the smallest image. Then at each level, decode and up-sample the smaller image and then encode the difference between the up-sampled and the original images.
  - ❖ Lossless (pixel-for-pixel)
    A special case of JPEG where there is no loss in the encoding process. In this mode, image processing and quantization use a predictive technique instead of transformation encoding

JPEG

- ✓ JPEG processing steps in base-line mode:
  - ❖ 1- subdivide the input image $f(x,y)$ into non-overlapping blocks of size 8×8. These blocks need to be processed from left to right and top to bottom.
  - ❖ 2- process each 8×8 block, by first shifting the level of a block and then calculating its 2-D cosine transform. Lets call the DCT coefficient as $T(u,v)$. A block of 64 pixels can be level shifted by subtracting 2^{m-1}, when the number of gray levels in the image is 2^m. For instance, in 8-bit mode, the range [0,255] needs to be shifted to [-128,127] range.
  - ❖ 3- The DCT coefficients of each block, $T(u,v)$ are then de-normalized by using an normalization array $Z(u,v)$ and quantized as follows

$$\hat{T}(u,v) = round\ [\frac{T(u,v)}{Z(u,v)}]$$

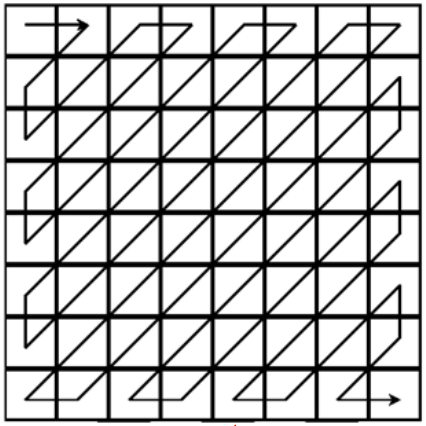
  - ❖ 4- the coefficient of $\hat{T}(u,v)$ are then arranged in a zigzag pattern as a 1-D array
  - ❖ 5- The 1-D array in step 4 is then coded using variable length coding

JPEG block diagram

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

$Z(u,v)$ Normalization array

$$\hat{T}(u,v) = round\,[\frac{T(u,v)}{Z(u,v)}]$$



$f(x,y)$

$T(u,v)$

Image → Build blocks of 8×8 → Level shift & Transform each block → De-normalization & Quantization → Zig-zag pattern and Encoding → Compressed Image

Compressed Image → Decoding → Inverse Tranform → Merging small blocks → De- compressed Image

Encoding Process

size	Amplitude	code
1	1,-1	1=1, -1=0
2	-3,-2,2,3	-3=00, -2=01, 2=10,3=11

- ✓ DC component encoded using differential encoding
  - ❖ DC coefficient determines the average intensity of the block.
  - ❖ Between adjacent blocks, the variation is (usually) small.
  - ❖ DC coefficient is calculated by finding the difference between the current DC coefficient and the one of the previous block
  - ❖ Since DC are differentially encoded, its range is [-2048,2047].
  - ❖ DC values is encode to (size) and (amplitude). Size is number of bits needed to represent the amplitude and is 4 bits. The Amplitude is the DC value.

- ✓ AC components are
  - ❖ The 63-number stream has lots of zeros in it.
  - ❖ AC coefficients are calculated as **(skip , size) amplitude**, where skip is the number of zeros and amplitude is the next non-zero component. And size is the number of bits that is needed to represent the non-zero component.
  - ❖ Skip must be in range of 0 to 15. if the number of zeros are more than 15, then more (skip, size) pair should be used. For example, for 28 zeros, we can write (15,0)(13,size) amplitude
  - ❖ Amplitude is the amplitude of the nonzero AC coefficient and it is in the range of [-1024,+1023]. So the size needs maximum of 10 bits
- ✓ The DCT coefficients are then converted into a compact binary sequence using Huffman table

JPEG-Example

140	144	147	140	139	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

Block of 8x8

12	16	19	12	11	27	51	47
16	24	12	19	12	20	39	51
24	27	8	39	35	34	24	44
40	17	28	32	24	27	8	32
34	20	28	20	12	8	19	34
19	39	12	27	27	12	8	34
8	28	-5	39	34	16	12	19
20	27	8	27	24	19	19	8

Level shifted block

185	-17	14	-9	23	-9	-13	-19
20	-34	26	-9	-11	10	13	6
-10	-23	-1	5	-18	3	-20	-1
-8	-5	14	-14	-8	-2	-3	8
-3	9	8	1	-11	18	18	15
3	-2	-18	8	8	-3	0	-6
8	0	-2	3	-1	-7	-1	-2
0	-7	-2	1	1	4	-6	0

DCT coefficients

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Normalization and quantization matrix

61	-3	2	0	2	0	0	-1
4	-4	2	0	0	0	0	0
-1	-2	0	0	-1	0	-1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Normalized DCT coefficients

61	-3	4	-1	-4	2	0	2	-2	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0
-1	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Zig-zag pattern of Normalized DCT coefficients

(6) (61), (0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2), (5,2)(2),(3,1)(1), (6,1)(-1),(2,1)(-1), (4,1)(-1),(7,1)(-1),(0,0)

Encoded ,Zig-zag pattern of Normalized DCT coefficients

JPEG-Example (cont.)

(6) (61), (0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2), (5,2)(2),(3,1)(1),
(6,1)(-1),(2,1)(-1), (4,1)(-1),(7,1)(-1),(0,0)

Encoded ,Zig-zag pattern of Normalized DCT coefficients

Partial Huffman encoding tables

(0,0) EOB	1010	(3,1)	111010
(0,1)	00	(4,1)	111011
(0,2)	01	(5,2)	11111110111
(0,3)	100	(6,1)	1111011
(1,2)	11011	(7,1)	11111010
(2,1)	11100	(6)	1110

size	Amplitude	code
1	1,-1	1=1, -1=0
2	-3,-2,2,3	2=10 -2=01 3=11 -3=00
3	-7,-6,-5,-4 4, 5, 6, 7	4=100 -4=011 5=101 -5=010 6=110 -6=001 7=111 -7=000

(1110)(111101) (01)(00)(100)(100)(00)(0)(100)(011)(01)(10)(11011)(10)(01)(01)
(1111110111)(10)(111010)(1)(1111011)(0)(11100)(0)(111011)(0)(11111010)(0)(1010)

JPEG binary format based on Huffman coding

JPEG- Hierarchical Mode

- ✓ The block diagram of JPEG in hierarchical mode has been shown here

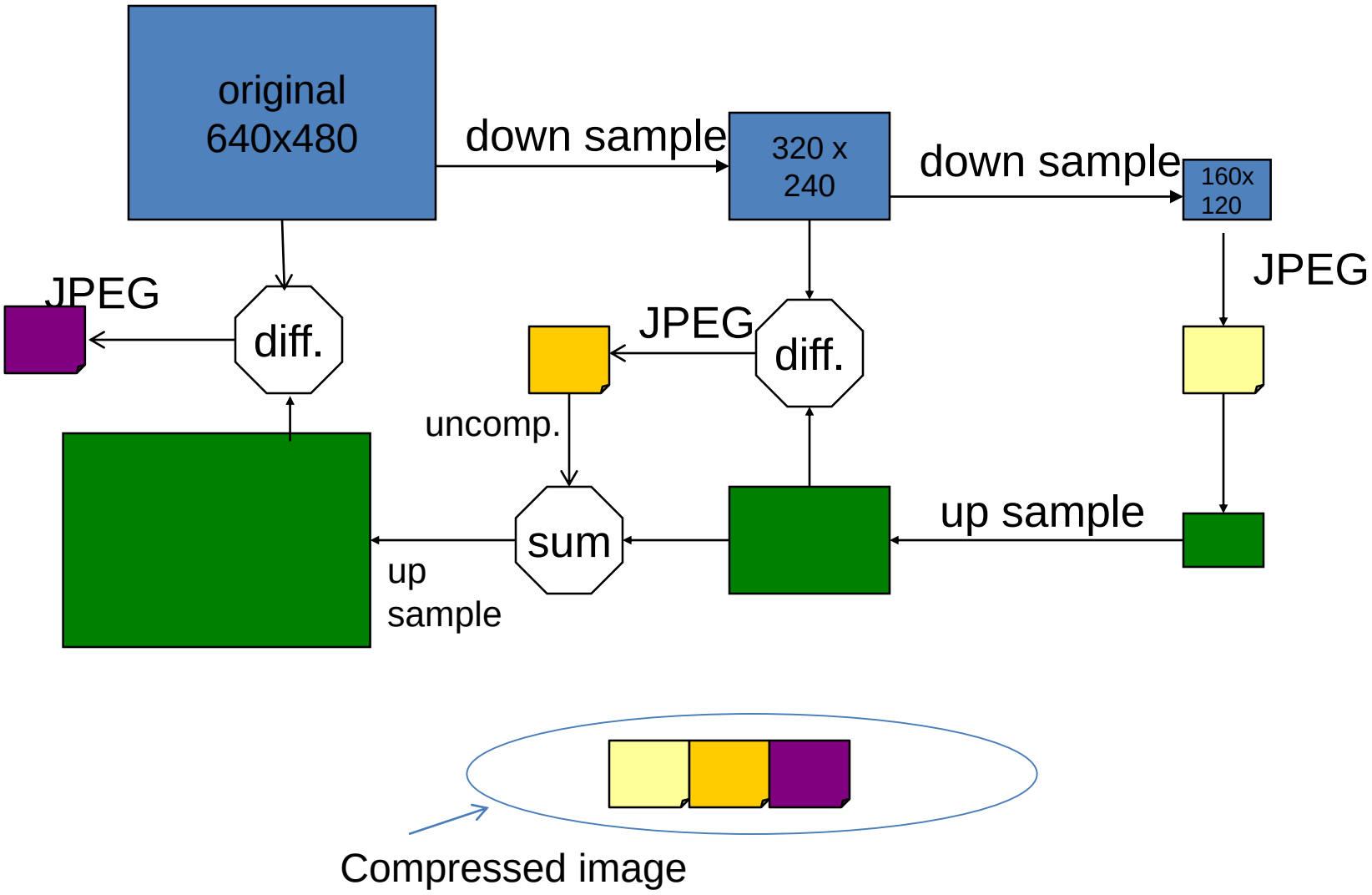


Image restoration and reconstruction

Image restoration and reconstruction

- ✓ Image restoration is a process which tries to recover an image that has been degraded.
- ✓ The restoration techniques usually model the degradation and apply the inverse process to recover the original image.
- ✓ To do the image restoration, we should model the degradation function and therefore, we should have a knowledge of how the image is degraded.
- ✓ Thus, Image restoration techniques enhance the degraded image in general.
- ✓ Difference between image enhancement process and restoration process
  - ❖ Image enhancement techniques tries to manipulate an image to be suited more for human eyes , while restoration techniques tries to find the source of degradation and formulate a process to recover the signal
  - ❖ Image enhancement is usually a subjective process, where as image restoration is an objective process

Point processing → Image enhancement

De-blurring → Image restoration

Image restoration model

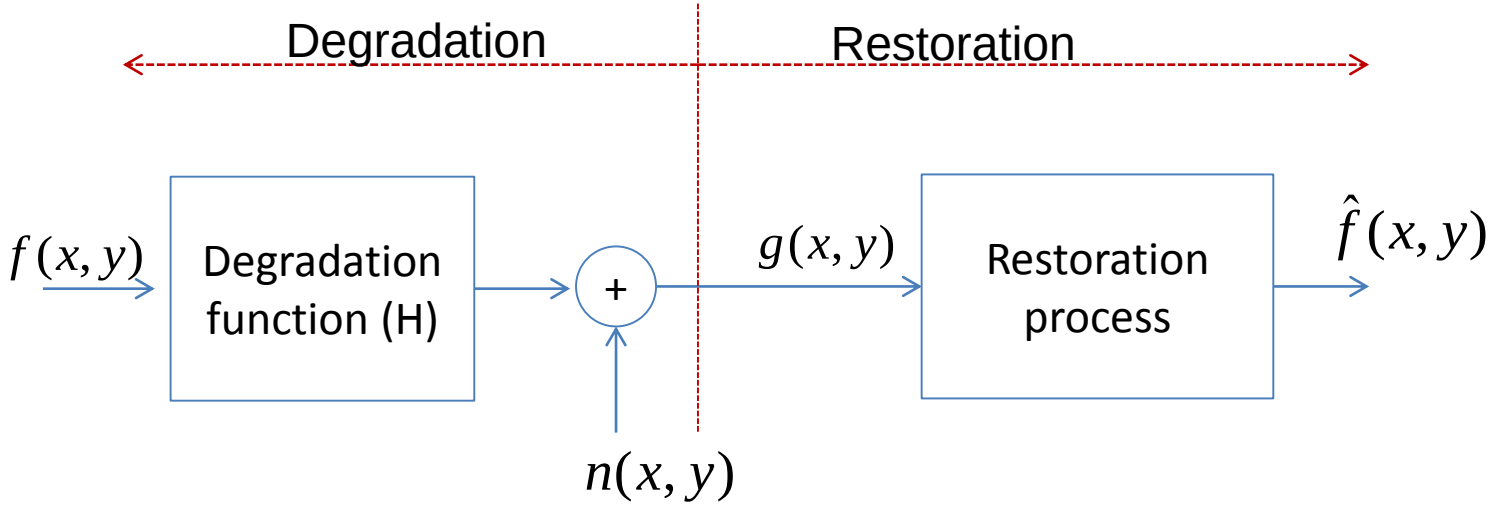
- ✓ The degradation / restoration processes can be modeled as follows
  - ❖ An image $f(x,y)$ is degraded by a degradation function h(x,y) and additive noise $n(x,y)$
  - ❖ The degraded image is $g(x,y)$ which is

$$g(x,y) = h(x,y) * f(x,y) + n(x,y)$$

  - ❖ The Fourier transform of $g(x,y)$

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

  - ❖ Having $g(x,y)$ as the degraded image and some knowledge about the degradation function H(u,v) and the type of noise , the restoration process tries to find an estimate of the original image $\hat{f}(x,y)$
  - ❖ if H is an identity operator, the degradation is just dependent to noise.
  - ❖ When the estimated image is close to the original image ?



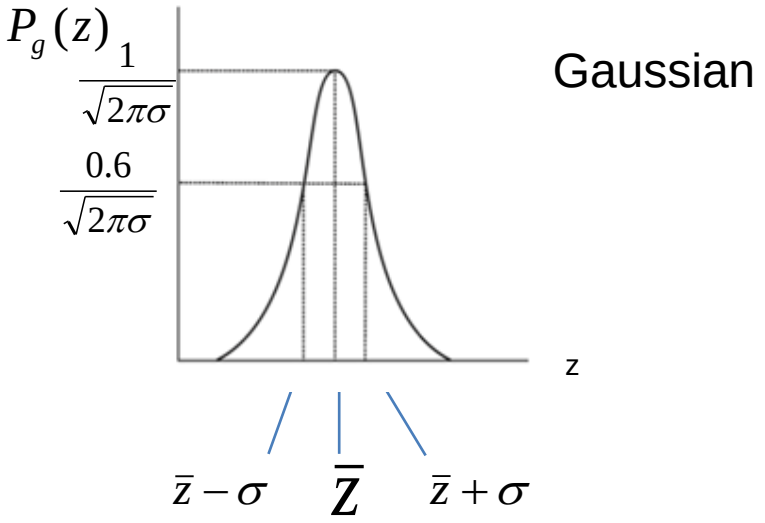
Noise and noise model

- ✓ Noise can be defined as an undesired artifact that has been generated during image acquisition and/ or transmission
- ✓ Noise can be considered as a random variable whose probability density function (PDF) describes its shape and distribution across the of intensity values.
- ✓ Not always true, but for simplicity purpose, we assume that the noise is uncorrelated with respect to the image
- ✓ Example of some noise model
  - ❖ Gaussian noise : (sensor noise , electronic circuit noise)
  - ❖ Uniform noise : (not very practical)
  - ❖ Rayleigh noise (range imaging)
  - ❖ Erlang (Gamma) noise: (laser imaging)
  - ❖ Exponential noise : (laser imaging)
  - ❖ Impulse noise(salt and pepper) : (quick transients situation, faulty switching)
  - ❖ Periodic noise (electrical and electromagnetic interference)

Selected noise PDF

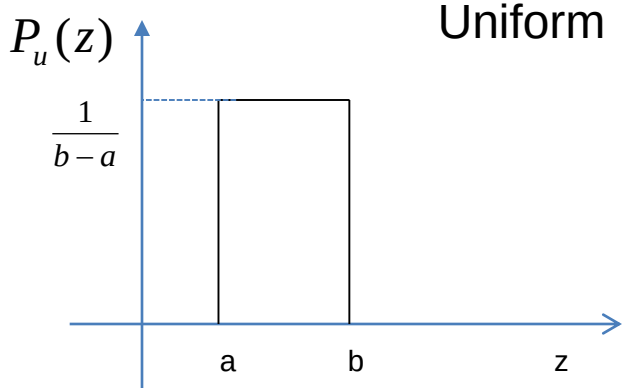
✓ **Gaussian noise**: (z is intensity value, $\bar{z}$ is the mean , σ is the standard deviation and σ^2 is the variance of z)

$$P_g(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\bar{z})^2/2\sigma^2}$$



✓ **Uniform noise** :

$$P_u(z) = \begin{cases} \frac{1}{b-a} & a \le z \le b \\ 0 & otherwise \end{cases}$$



- The value of the mean is $\bar{z} = \frac{a+b}{2}$
- The value of variance is

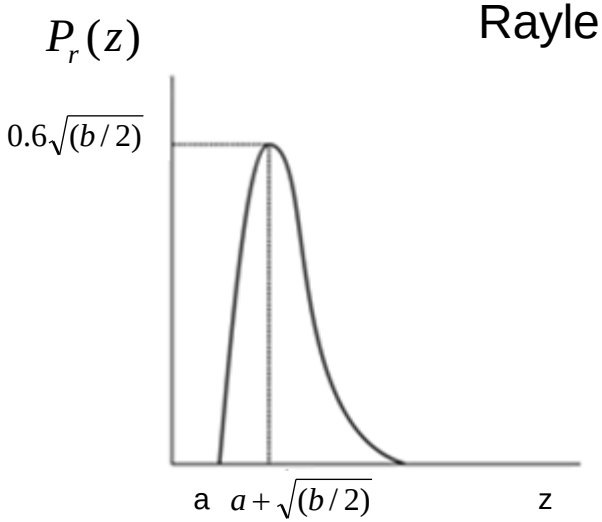
$$\sigma^2 = \frac{(b-a)^2}{12}$$

Selected noise PDF

✓ Rayleigh noise

$$P_r(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & z \geq a \\ 0 & otherwise \end{cases}$$

Rayleigh

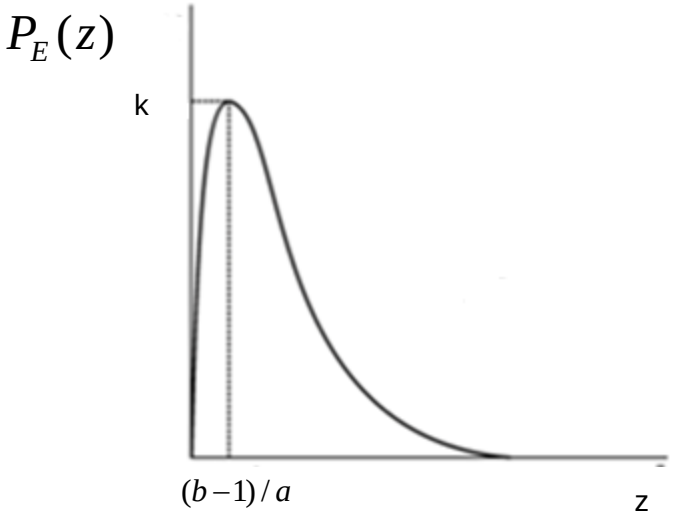


- There is a displacement from the origin and the density is skewed to the right. This is quite useful for estimating skewed histograms

✓ **Erlang** (Gamma) noise

$$P_E(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!}e^{-az} & z \geq 0 \\ 0 & otherwise \end{cases}$$

Erlang



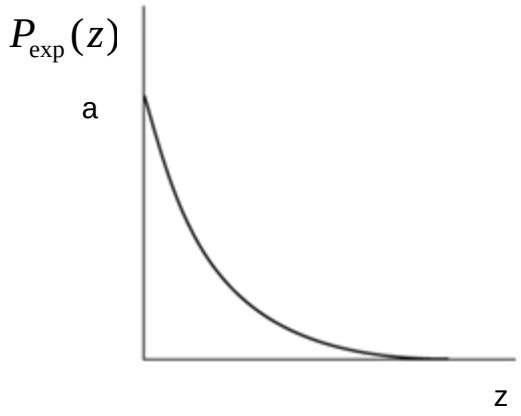
The value of the mean is $\bar{z} = b/a$

The value of variance is $\sigma^2 = \frac{b}{a^2}$

$$k = \frac{a(b-1)^{b-1}}{(b-1)!}e^{-(b-1)}$$

Selected noise PDF

Exponential



✓ **Exponential noise**

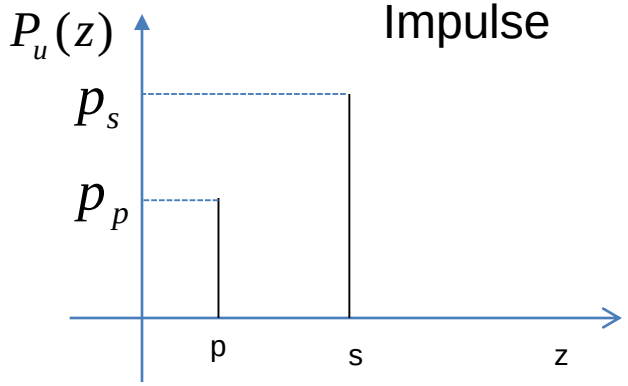
$$P_{\exp}(z) = \begin{cases} ae^{-az} & z \geq 0 \\ 0 & otherwise \end{cases}$$

- ❖ The value of the mean is $\bar{z} = 1/a$
- ❖ The value of variance is $\sigma^2 = \frac{1}{a^2}$

✓ **Impulse noise**(salt and pepper)

- ❖ p_p and p_s are the probability of occurrence of pixels whose values are equal to P (for pepper) and s for salt.

$$P_{sp}(z) = \begin{cases} P_p & z = p \\ P_s & z = s \\ 0 & otherwise \end{cases}$$



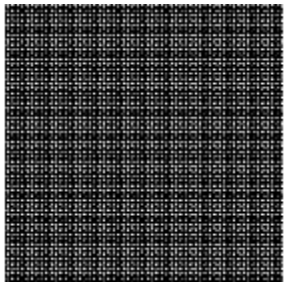
Periodic noise

- ✓ If the image is subject to a periodic, rather than a random disturbance, an image corrupted by periodic noise is generated.
- ✓ The function **imnoise** in can be used to make some random noise such as (Gaussian or Salt and Pepper noise), but it does not support periodic noise
- ✓ The periodic noise can be added to an image in MATLAB by adding a periodic matrix(using a trigonometric function) to the image:

```
[M,N]=size(f);
[x,y]=meshgrid(1:M,1:N);
p= 0.25*(2*sin(x)+2*sin(y)+sin((x+y)/2)+sin((x-y)/2)) ;
f_pn=(im2double(f)+p/2)/2;
```

- ✓ The random noises are usually be removed using spatial filtering. The best way to remove periodic noise is using frequency-domain filtering.

Original Image



Periodic noise

Degraded image

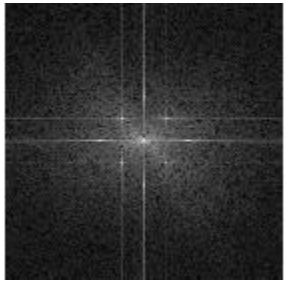


Image spectrum

A pair of impulse for each sine wave

Restoration due to random noise

- ✓ We can use spatial filtering to remove additive random noise. Spatial filtering needs a window as mask. If S_{xy} is this window and the size of this window is $m \times n$ and the window is centered at location (x, y), the spatial filtering calculates the value of restored image at coordinates (x, y). As $\hat{f}(x, y)$. To apply spatial filtering to whole image The window would be moved over $g(x, y)$ the degraded image.
- ✓ There are various filtering methods that can be used for this purpose. Examples are
  - ❖ Mean filters
    - Arithmetic mean filter
    - Geometric mean filter
    - Harmonic mean filter
    - Contra-harmonic mean filter
  - ❖ Order statistic filters
    - Median filter
    - Max and Min filter
    - Midpoint filter
    - Alpha-trimmed filter

Restoration due to random noise- Mean filters

✓ Mean filters

❖ Arithmetic mean filter

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

- calculates the average value of the corrupted image in the neighborhood S_{xy}. So it has smoothing effect on the image.

❖ Geometric mean filter

$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

- It has smoothing effect comparable to arithmetic filters, but less image detail is lost

---

❖ Harmonic mean filter

- As a mean filter has a smoothing effect. It works well with various random noise except for the pepper noise. It does well for the salt noise.

$$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

❖ Contra-harmonic mean filter

- It uses a parameter called Q that
  - Q=0 → the filter acts as Arithmetic mean filter
  - Q=-1 → the filter acts as harmonic mean filter
  - Q>0 → the filter reduces the pepper noise
  - Q< 0 → the filter works well for filtering salt noise

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

Restoration due to random noise- Order statistic filters

✓ Order statistic filters

❖ Median filter

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{median}\{g(s,t)\}$$

- Modifies the value of a pixel by the median of the intensity value in the neighborhood window of that pixel (50 percentile of the set)
- Very effective in reducing of impulse noise (salt and pepper)

❖ Max and Min filter

$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}}\{g(s,t)\} \qquad \hat{f}(x,y) = \min_{(s,t)\in S_{xy}}\{g(s,t)\}$$

- Max filter works well with pepper noise (pepper noise has low values)
- Min filter works well with salt noise (salt noise has high value)

❖ Midpoint filter

$$\hat{f}(x,y) = \frac{1}{2}\left(\max_{(s,t)\in S_{xy}}\{g(s,t)\} + \min_{(s,t)\in S_{xy}}\{g(s,t)\}\right)$$

- This filter calculates the average of min and max values
- Works well with uniform and Gaussian noise

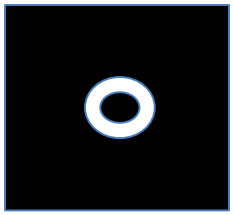
❖ Alpha-trimmed filter

$$\hat{f}(x,y) = \frac{1}{m\times n - d}\sum_{(s,t)\in S_{xy}} g_r(s,t)$$

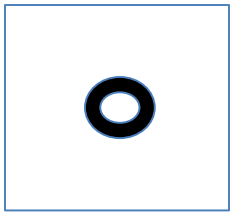
- This filter , first erase $d/2$ highest and $d/2$ lowest values of in neighborhood S_{xy}. We can call the remaining (mn-d) values as $g_r(s,t)$. The filter then calculates the average of the values in The values in $g_r(s,t)$

Restoration due to periodic noise

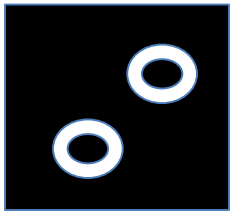
- ✓ Frequency –domain filtering can effectively filters the periodic noise
  - ❖ The FFT of the Periodic noise shows bright sections at locations corresponding to the frequencies of the periodic noise. Filtering does frequency location, will eliminate the periodic noise.
  - ❖ Various filters can be used for filtering periodic noise
    - Band-reject filters
      - Attenuates frequency components within a certain area , while leaving all other areas untouched.
    - Band-pass Filters
      - The opposite of band- reject filter. $H_{BP}(u,v) = 1 - H_{BR}(u,v)$
    - Notch filters
      - This filter rejects (or passes) frequencies in predefined neighborhood about the center frequency.



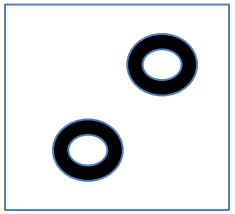
Band reject



Band pass



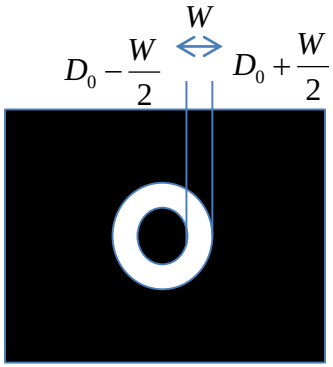
Notch-reject



Notch-pass

Restoration due to periodic noise- Frequency filtering

✓ Band-Reject filters (W is the width of the band)



Band reject

$$H_{i-br}(u,v) = \begin{cases} 1 \quad if \quad D(u,v) < (D_0 - \frac{W}{2}) \\ 0 \quad if \quad (D_0 - \frac{W}{2}) \le D(u,v) \le (D_0 + \frac{W}{2}) \\ 1 \quad if \quad D(u,v) < (D_0 + \frac{W}{2}) \end{cases}$$

Ideal band –reject filter

$$H_{B-br}(u,v) = \frac{1}{1 + (\frac{D(u,v) \times W}{D^2(u,v) - D_0^{\ 2}})^{2n}}$$

Butterworth band –reject filter

$$H_{G-br}(u,v) = 1 - \exp[-0.5(\frac{D^2(u,v) - D_0^{\ 2}}{D(u,v) \times W})^2]$$

Gaussian band –reject filter

✓ Band –pass filters

$$H_{BP}(u,v) = 1 - H_{BR}(u,v)$$

Restoration due to periodic noise- Frequency filtering – cont.

✓ Notch filter : considering the notch filter of radius D_0 with center at (u_0, v_0)

$$D_1(u,v) = \sqrt{(u - M/2 - u_0)^2 + (v - N/2 - v_0)^2} \qquad D_2(u,v) = \sqrt{(u - M/2 + u_0)^2 + (v - N/2 + v_0)^2}$$

$$H_{i-Nr}(u,v) = \begin{cases} 0 & if \quad D_1(u,v) < D_0 \\ & or \quad D_2(u,v) < D_0 \\ 1 & elsewhere \end{cases}$$

Ideal notch–reject filter

For Notch pass filter

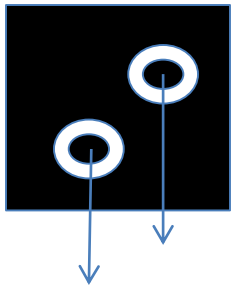
$$H_{Np}(u,v) = 1 - H_{Nr}(u,v)$$

$$H_{B-Nr}(u,v) = \frac{1}{1 + \left(\frac{D_0^{\,2}}{D_1(u,v)D_2(u,v)}\right)^{2n}}$$

Butterworth notch–reject filter

$$H_{G-Nr}(u,v) = 1 - \exp\left[-0.5\left(\frac{D_1(u,v)D_2(u,v)}{D_0^{\,2}}\right)\right]$$

Gaussian notch–reject filter



Noise reduction – Special case

- ✓ If the degradation is just dependent to noise, and the noise is uncorrelated and has a average of zero, then the noise can be reduced by averaging several noisy image of the same image.

$$g(x,y) = f(x,y) + n(x,y)$$ ← The degradation is only dependent to noise

$$g_i(x,y) = f(x,y) + n_i(x,y)$$ ← several noisy image of the same image.

$$E\{n_i(x,y)\} = 0$$ ← noise has a average of zero,

$$\sigma_n^{\,2} = \sigma_{ni}^{\,2}$$

$$\sigma_f^{\,2} = 0$$ ← values of f(x,y) are constant and do not change

---

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

$$\sigma_f^{\,2} = 0$$

$$E\{\bar{g}(x,y)\} = \frac{1}{K}\sum_{i=1}^{K} E\{f_i(x,y)\} + \frac{1}{K}\sum_{i=1}^{K} E\{n_i(x,y)\}$$

$$\sigma_{\bar{g}}^{\,2} = \frac{1}{K}\sigma_n^{\,2}$$

$$E\{\bar{g}(x,y)\} = \frac{1}{K}(K)\, f(x,y) + \frac{1}{K}(0) = f(x,y)$$

$$\sigma_{\bar{g}}^{\,2} = \sigma_f^{\,2} + \frac{1}{K^2}[\sigma_{n1}^{\,2} + \sigma_{n2}^{\,2} + \ldots \sigma_{nk}^{\,2}] = \frac{1}{K}\sigma_n^{\,2}$$ ← noise is uncorrelated

Degradation Function Estimation

- ✓ The methods to estimate degradation function
  - ❖ By image observation which tries to estimate from the image itself. For instance, a sub-image of the degraded image can be considered $g_s(s,t)$. Try to improve the quality of the sub-image to find $\hat{f}_s(x,y)$. So you can find the transfer function of the degradation function for the sub-image as

$$H_s(u,v) = \frac{G_s(u,v)}{\hat{F}_s(u,v)}$$

Then try to find H(u,v) in larger scale

  - ❖ By experiment . In this method , if we have the device that has been used to take the picture , we try to find the setting that produces the same degraded picture. Now, under this setting, try to find the impulse response

$$H(u,v) = \frac{G(u,v)}{A}$$

$G(u,v)$: FT of the observed image under impulse input

A: Constant value showing the strength of impulse function

  - ❖ By modeling. In this method, we try to find the mathematical formula depending on the situation

Degradation Function Estimation by modeling - Example

- ✓ Assume that a picture is taken by a perfect camera, but the scene is moving by a planner motion of $x_0(t)$ and $y_0(t)$ in the x and y directions. If $f(x,y)$ is the image without any motion and $g(x,y)$ is the image considering the motion. Then

$$g(x,y) = \int_0^T f(x - x_0(t), y - y_0(t))dt$$

$$G(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)\, e^{-j2\pi(ux+vy)}\, dx\, dy =$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[\int_0^T f(x - x_0(t), y - y_0(t))dt\right] e^{-j2\pi(ux+vy)}\, dx\, dy$$

$$G(u,v) = \int_0^T \left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x - x_0(t), y - y_0(t))\, e^{-j2\pi(ux+vy)}\, dx\, dy \right] dt$$

$$F(u,v)\, e^{-2j\pi[ux_0(t)+vy_0(t)]}$$

$$G(u,v) = F(u,v)\int_0^T e^{-2j\pi[ux_0(t)+vy_0(t)]}dt \qquad H(u,v)$$

Degradation Function Estimation by modeling – Example

- ✓ If we know that the motion is $x_0(t) = at/T$ & $y_0(t) = 0$, model the distortion under the motion

$$G(u,v) = F(u,v)\ H(u,v)$$

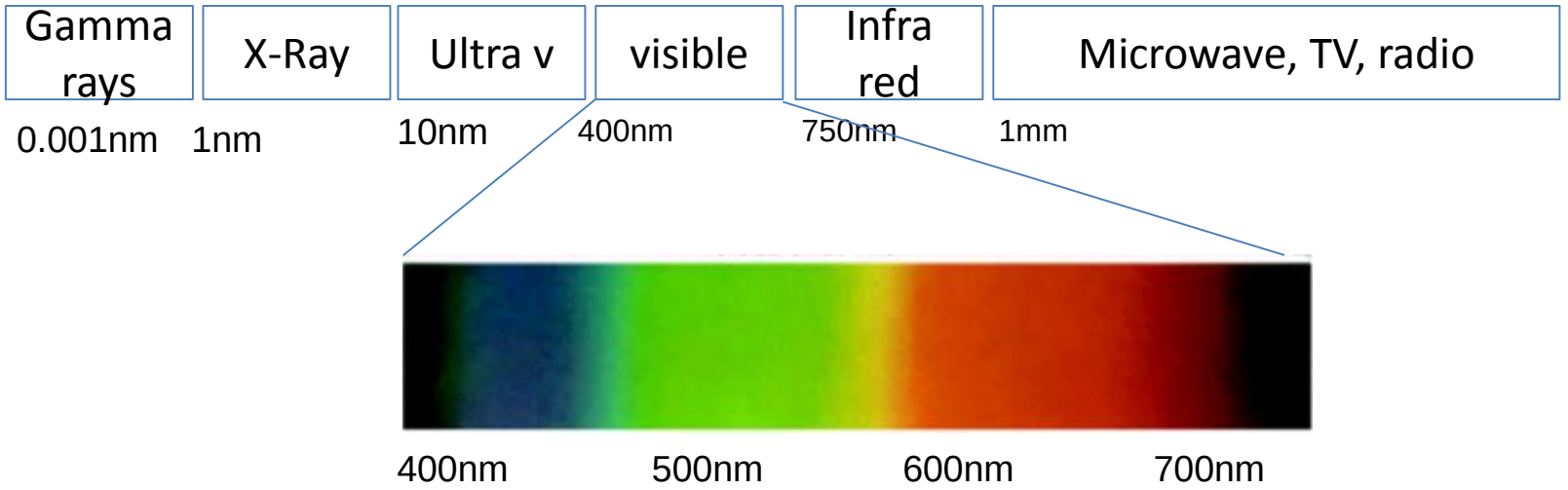
$$H(u,v) = \int_0^T e^{-2j\pi[ux_0(t)+vy_0(t)]}dt$$

$$H(u,v) = \int_0^T e^{-2j\pi uat/T}dt = \frac{T}{\pi ua}\sin(\pi ua)e^{-j\pi ua}$$

Color Image Processing

Electromagnetic Spectrum

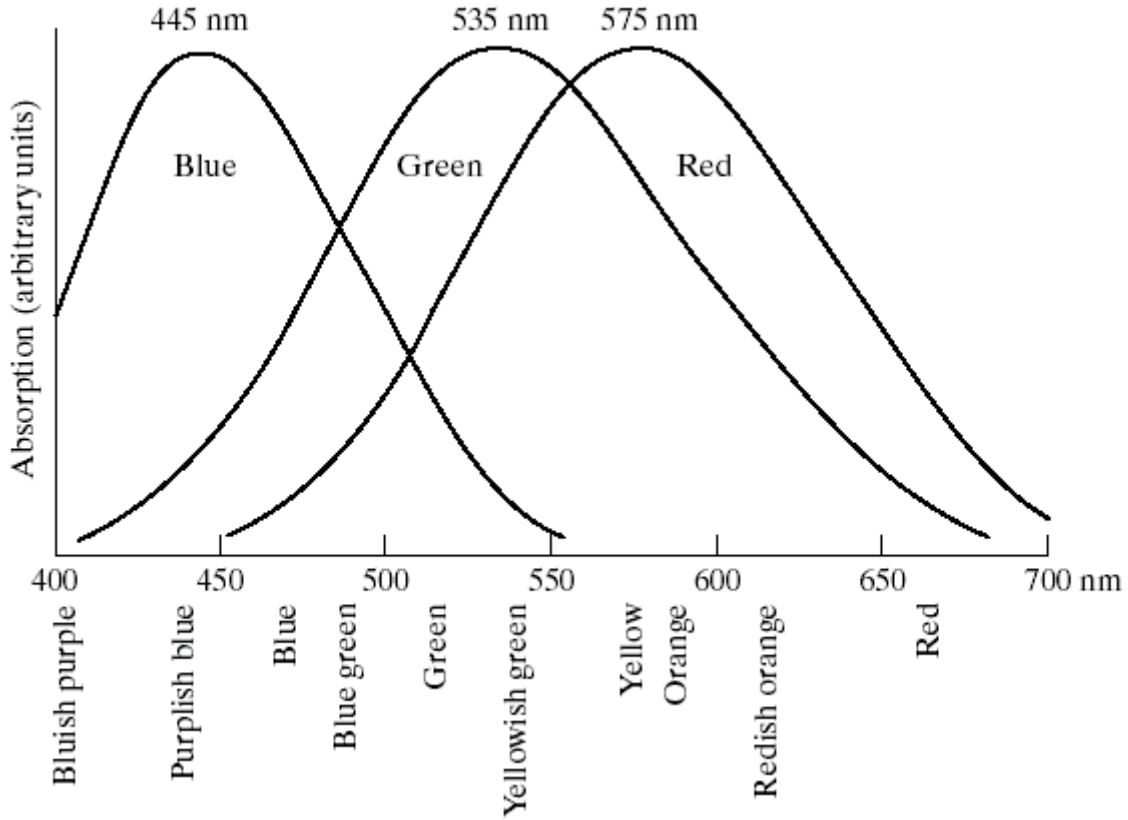
- ✓ Visible light wavelength: from around 400 to 700 nm



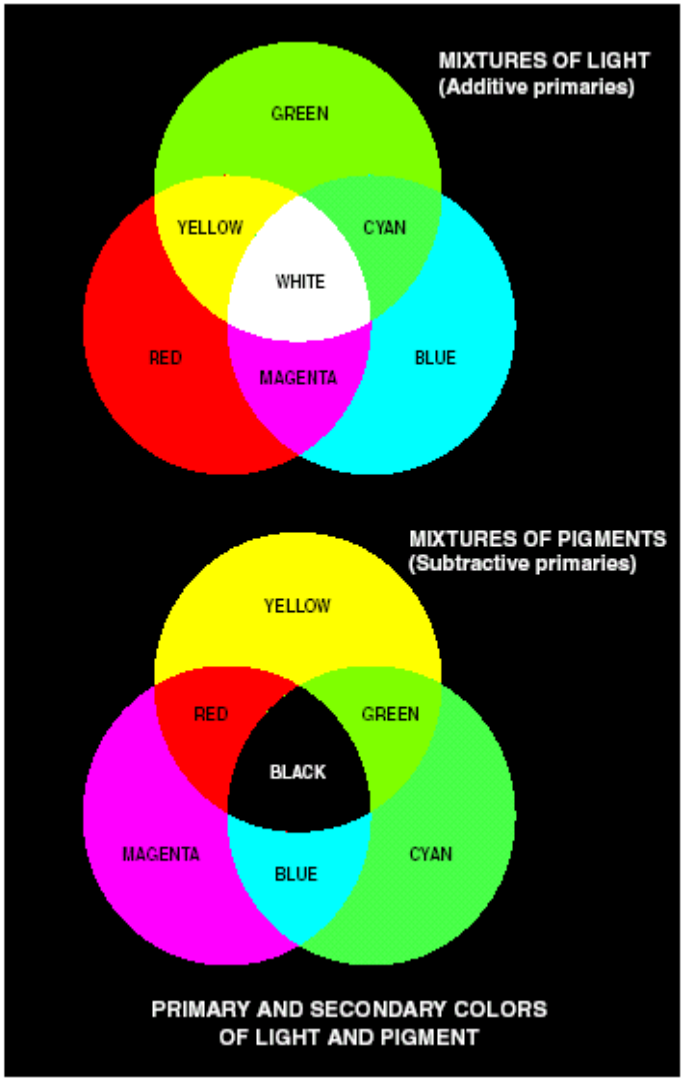
- ✓ How to describe the quality of an light source
  - ❖ For an achromatic (monochrome) light source, the attribute to describe the quality is intensity
  - ❖ For a chromatic light source, there are 3 attributes to describe the quality:
    - Radiance= total amount of energy flow from a light source (Watts)
    - Luminance= amount of energy received by an observer (lumens)
    - Brightness= intensity (subjective descriptor like intensity)

Sensitivity of Cones in the Human Eye

- ✓ 6-7 millions cones in a human eye
  - ❖ 65% sensitive to Red light
  - ❖ 33% sensitive to Green light
  - ❖ 2 % sensitive to Blue light
  - ❖ Primary colors: Defined by CIE (The International Commission on Illumination)
    - Red = 700 nm Green = 546.1nm Blue = 435.8 nm



Primary and Secondary Colors



- ✓ Additive primary colors: RGB
  - ❖ use in the case of light sources (i.e color monitors)
  - ❖ The primary colors can be added to make the secondary colors
    - (Magenta= Blue + red), Cyan (green + blue), Yellow (red+ green)
  - ❖ Mixing three primary colors or a secondary color with its opposite primary color makes white
    - White= Green+Magenta =Blue+Yellow= red+Cyan
- ✓ Subtractive primary colors: CMY
  - ❖ use in the case of pigments in printing devices
  - ❖ The primaries are the ones that absorbs the primary color of light and reflects the other two colors
    - Yellow absorbs blue, Cyan absorbs red, Magenta absorbs green
  - ❖ Mixing colors
    - White – (**C**+**M**+**Y**) = Black
    - White – (**C**+**Y**)= green
    - White – (**M**+**Y**) = red

(Image from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 3rd Edition

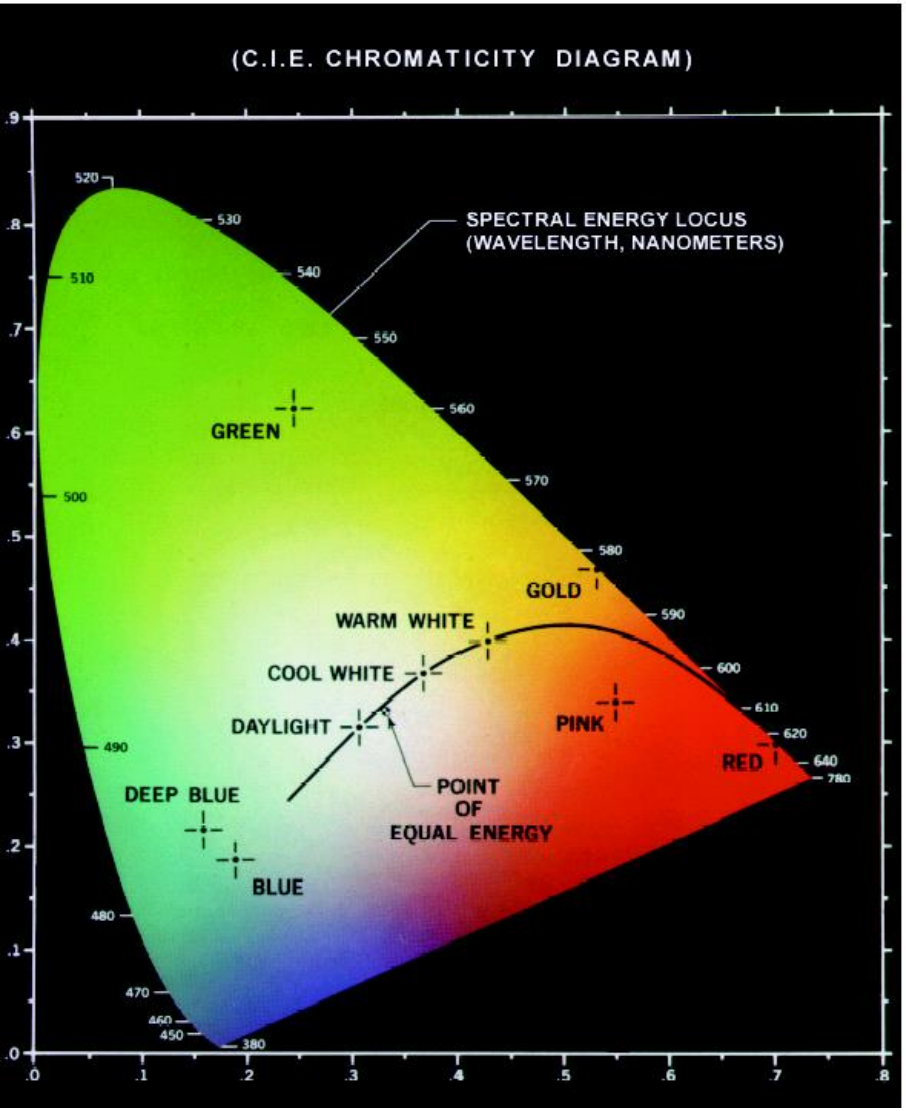
Color differentiation

- ✓ The amount of red (X), green (Y) and blue (Z) to form any particular color is called *tri-stimulus. We can define x, y, z as normalized value representing X, Y and Z respectively.*

$$x = \frac{X}{X + Y + Z} \qquad y = \frac{Y}{X + Y + Z} \qquad z = \frac{Z}{X + Y + Z}$$

$$x + y + z = 1$$

Chromaticity diagram



- ✓ The amount of red (X), green (Y) and blue (Z) to form any particular color is called tri-stimulus. We can define x, y, z as normalized value representing X, Y and Z respectively.

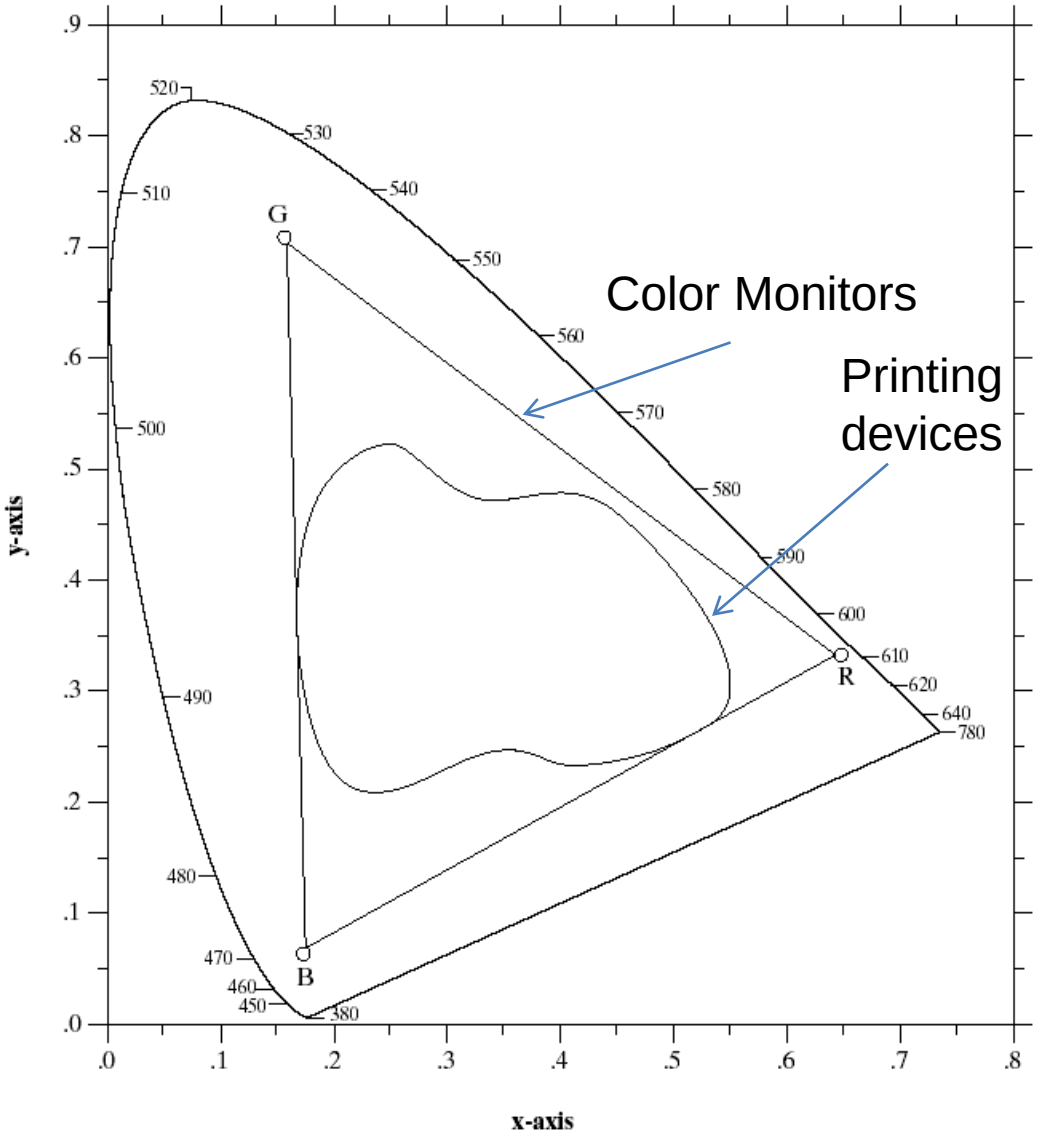
$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z}$$

$$x + y + z = 1$$

- ✓ Chromaticity diagram is useful in color mixing
- ✓ The horizontal axes is x and vertical axes is y. For each value of x and y, the z value is fixed. (z=1-x-y)
- ✓ Each point in the diagram shows a specific color.
- ✓ A straight line from connecting any two points in this diagram, gives all the combination that can be obtained by mixing these two colors additively.
- ✓ A line from the point of equal energy to any point on the boundary of the chart, will define all the shades of that particular spectrum color.

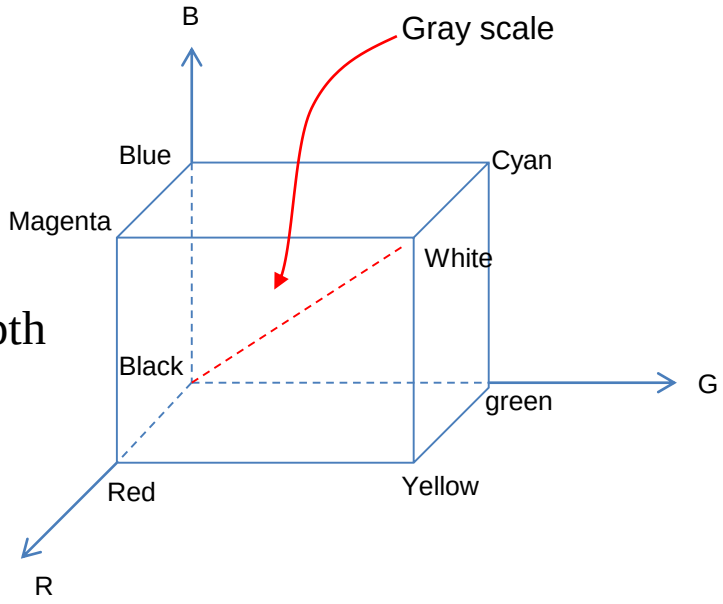
Image from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 3rd Edition

Color Gamut of Color Monitors and Printing Devices



- ✓ Connecting three points of RGB makes a triangle which represents all the colors that can be generated by a RGB monitor.
- ✓ The irregular region inside the triangle is the colors that high-quality printers can generate.

Color Spaces or Models



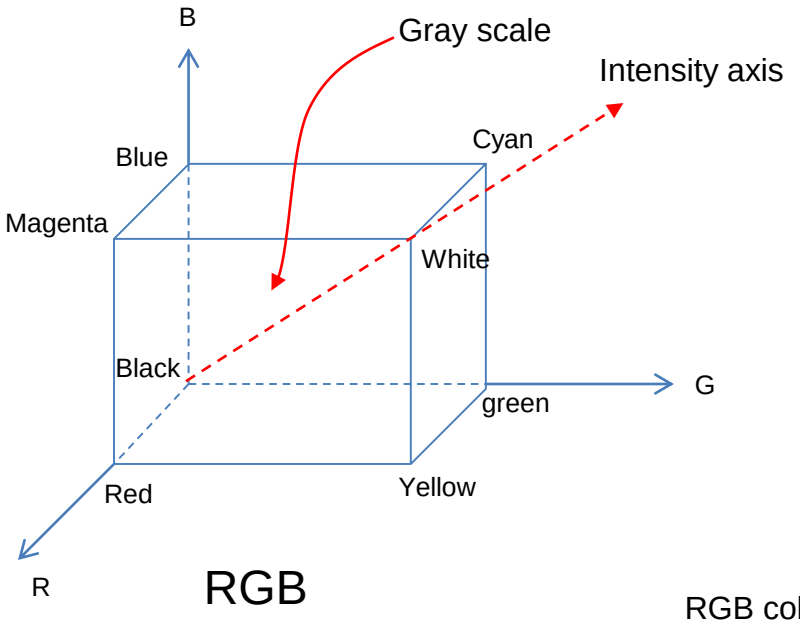
- ✓ Purpose of color models is to facilitate the specification of colors in some standards
  - ❖ RGB
    - R, G and B color, each has 8 bits. The color depth is 24 bits = almost 16 million different colors
    - The model is based on Cartesian coordinate system.
  - ❖ Safe RGB
    - A subset of all possible RGB combination. This subset has 256 members including 216 fixed member and 40 that can be processed by OS or application program.
  - ❖ CMYK
    - Most printing devices use CMYK color model

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

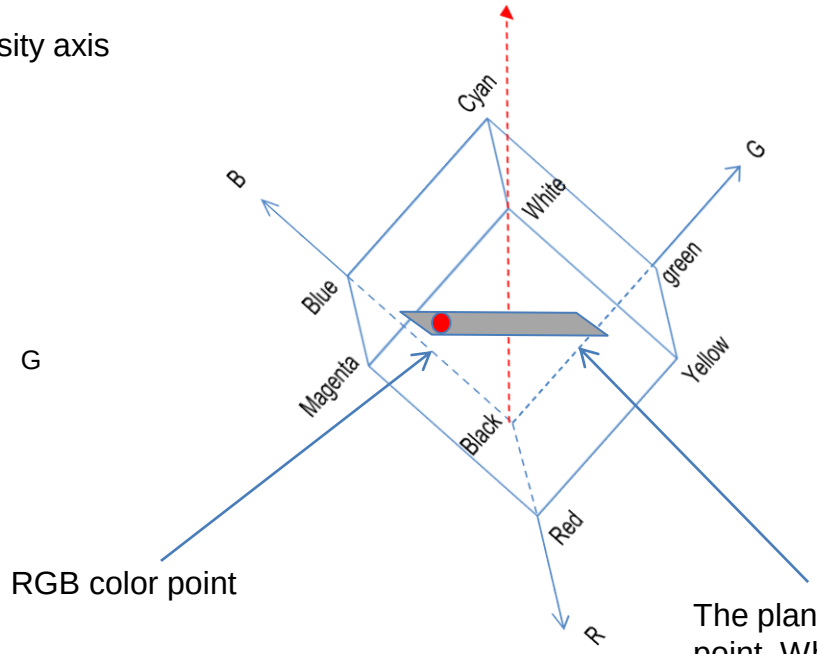
  - ❖ HSI
    - RGB, CMY models are not good for describing colors as human interpretation, no one describes a color based on three RGB values.
    - Human describe the color in terms of Hue, saturation and brightness.
      - Hue: Dominant color
      - Saturation: Relative purity (inversely proportional to amount of white light added)
      - Intensity: grayscale image

HSI Color Model- intensity value

- ✓ To determine the intensity component of any RGB color point, pass a plane perpendicular to the intensity axis which contains the color point. The intersection of this plane with the intensity axis, is a point. The value of this point in intensity axis is the intensity value of HSI model.



RGB



RGB color point

The plane containing the RGB point Which is perpendicular to intensity axis
The plane can be defined to be triangular, circular or have a shape of hexagon

HSI Color Model- Hue and Saturation Planes

- ✓ Circular and triangular plane are shown here. A point is the plane is an arbitrary RGB color.
- ✓ To find the Saturation, calculate the distance from the center of the plane to the RGB point.
- ✓ To find the Hue, connect the RGB point to the center of the plane . Also connect a line from the pure red point to the center of the plane. The angle between these two lines is Hue.

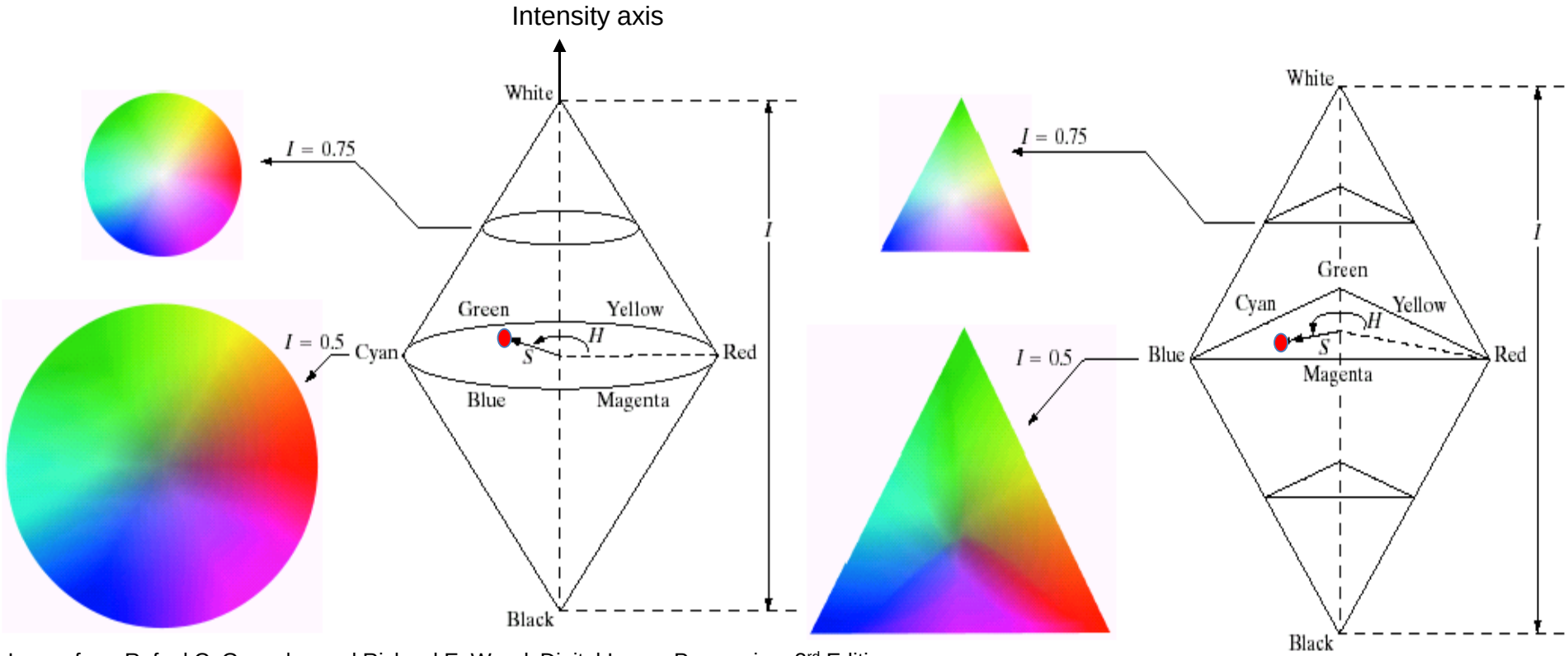


Image from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 3rd Edition

Converting Colors from RGB to HSI and vice versa

✓ To convert from RGB to HSI

$$H = \begin{cases} \theta & if \;\; B \le G \\ 360 - \theta & if \;\; B. > G \end{cases}$$

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}(R-G)+(R-B)}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right\}$$

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right]$$

$$I = \frac{1}{3}(R+G+B)$$

✓ To convert from HSI to RGB

$0^\circ \le H < 120^\circ$	$120^\circ \le H < 240^\circ$	$240^\circ \le H < 360^\circ$
$B = I(1-S)$	$H = H - 120$	$H = H - 240$
$R = I\left[1 + \frac{S\cos(H)}{\cos(60^\circ - H)}\right]$	$R = I(1-S)$	$G = I(1-S)$
$G = 3I - (R+B)$	$G = I\left[1 + \frac{S\cos(H)}{\cos(60^\circ - H)}\right]$	$B = I\left[1 + \frac{S\cos(H)}{\cos(60^\circ - H)}\right]$
	$B = 3I - (R+G)$	$R = 3I - (B+G)$

RGB to HSI conversion-example

✓ To compute HIS value of a pixel that its RGB values are (100, 150, 200)

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}}\right\} = \cos^{-1}\left\{\frac{\frac{1}{2}[(100-150)+(100-200)]}{\left[(100-150)^2+(100-200)(150-200)\right]^{1/2}}\right\} = 150$$

$$H = \begin{cases}\theta & if \;\; B \le G \\ 360-\theta & if \;\; B.>G\end{cases} \Rightarrow H = 360-150 = 210 \longrightarrow$$ 210 in [0 , 360} range

$$I = \frac{1}{3}(R+G+B) = \frac{100+150+200}{3} = 150 \longrightarrow$$ 150 in [0 , 255] range

$$S = 1-\frac{3}{(R+G+B)}\left[\min(R,G,B)\right] = 1-\frac{3}{100+150+200}[100] = 0.33 \longrightarrow$$ 33 in [0 , 100] range

---

✓ To convert from HSI (210, 150, 33) to RGB

$$120^\circ \le H < 240^\circ$$

$$H = H - 120 \;\; \Rightarrow \;\; H = 210-120 = 90$$

$$R = I(1-S) \;\; \Rightarrow 150(1-0.333) = 100$$

$$G = I\left[1+\frac{S\cos(H)}{\cos(60^\circ - H)}\right] = 150\left[1+\frac{0.33\cos(90)}{\cos(60^\circ - 90)}\right] = 150$$

$$B = 3I-(R+G) \;\; \Rightarrow 3\times 150-(100+150) = 200$$

Example: HSI Components of RGB Colors

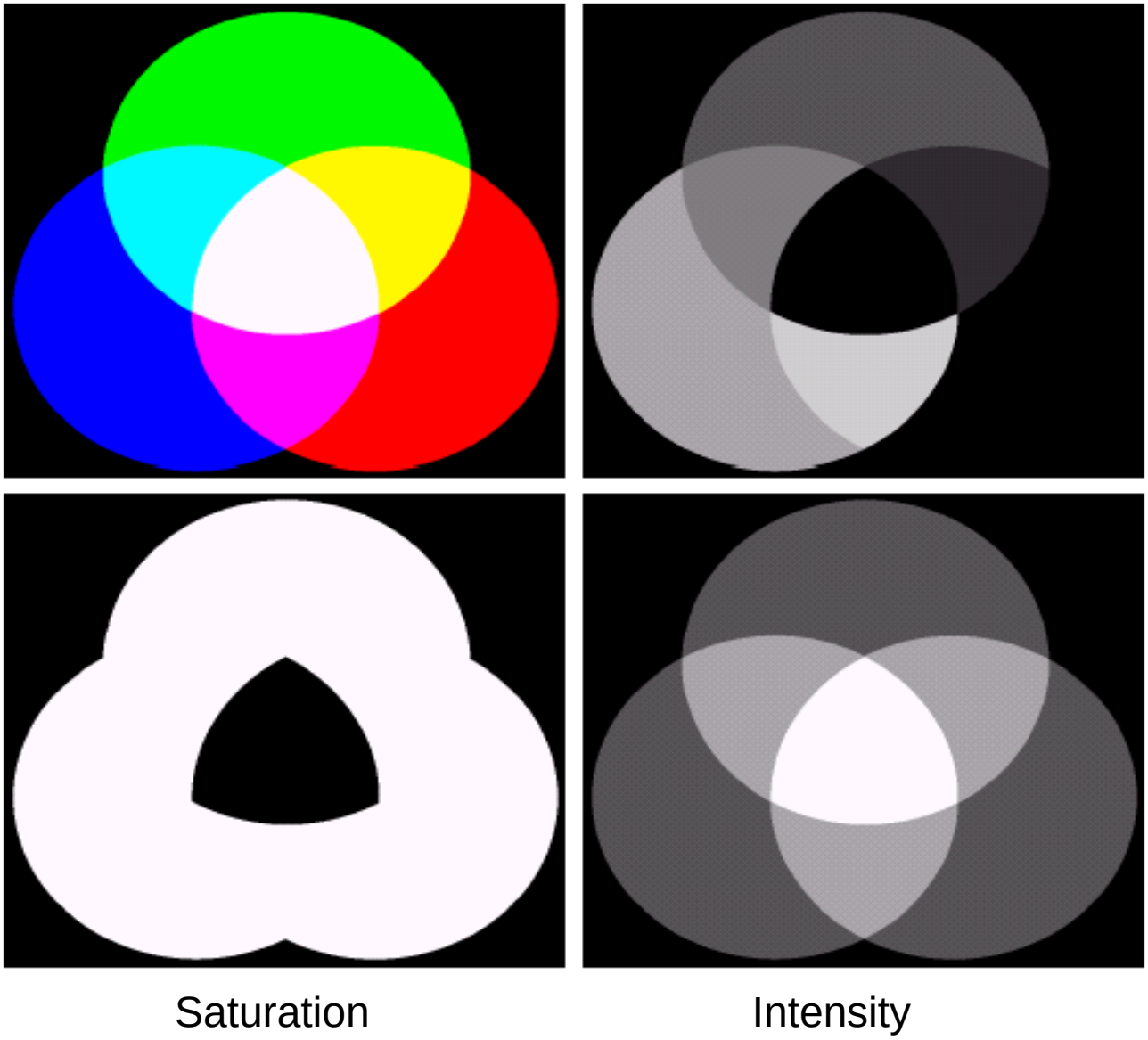


Image from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 3rd Edition

Color image processing

- ✓ To process color images, any color model can be used. Some processing can be done in a specific model easier or more efficiently. HSI model is one of the model that has been used widely for color image processing.

  - ❖ One way to process the color image expressed in HSI model, is to apply the processing to the intensity values and then Hue and Saturation to be added to the modified intensity image to make the processed color image. This method can be used in many processes such as color image smoothing, color image sharpening, segmentation, noise removal, edge detection and so on.

  - ❖ Another way to process the color image expressed in RGB model, is to apply the processing to each color separately and then combine the modified R ,G and B images to make the color image. This method can be used in many processes such as color image smoothing, color image sharpening, segmentation, histogram equalization, edge detection and so on.

Color Image Smoothing

✓ Image smoothing per-color-plane method:

❖ 1- smooth each color plane (R , G, B) using moving averaging and the combine back to RGB

$$\bar{c}(x,y) = \frac{1}{K}\sum_{(x,y)\in S_{xy}} c(x,y) = \left[\frac{1}{K}\sum_{(x,y)\in S_{xy}} R(x,y),\ \frac{1}{K}\sum_{(x,y)\in S_{xy}} G(x,y),\ \frac{1}{K}\sum_{(x,y)\in S_{xy}} B(x,y)\right]$$

R

G

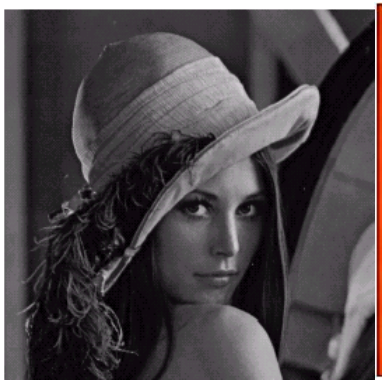
B

Smoothed image by 1

❖ 2- Smooth only Intensity component while leaving H and S unmodified

Hue

Saturation



Intensity

Smoothed image by method 2

Color Image Sharpening

- ✓ Image sharpening can be done in the same manner as color image smoothing:
  - ❖ 1. Per-color-plane method for RGB,CMY images
  - ❖ 2. Sharpening only I component of a HSI image

Sharpening all RGB components

Sharpening Intensity component of HSI

The difference between two methods

Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 3rd Edition

Example of other color Image processing

- ✓ Histogram equalization
  - ❖ Histogram equalization of a color image can be performed by adjusting color intensity uniformly in HSI domain while leaving hue and saturation unchanged. The HSI model is suitable for histogram equalization where only Intensity image is equalized.
- ✓ Pseudo-color Image Processing
  - ❖ In some application, it is needed to change gray scale image to color one. Pseudo color is a false color. In some case there is no "color" concept for a gray scale image but we can assign "false" colors to an image. The reason for this processing is that Human can distinguish different colors better than different shades of gray.
    - Binary Intensity slicing
    - Multi level intensity slicing
- ✓ Gradient of color image
  - ❖ The following formulas can be used to define the gradient of a color image.

$$F(\theta)=\left\{\frac{1}{2}(g_{xx}+g_{yy})+(g_{xx}-g_{yy})\cos(2\theta)+2g_{xy}\sin(2\theta)\right\}^{1/2} \qquad \theta=\frac{1}{2}\tan^{-1}\left[\frac{2g_{xy}}{(g_{xx}-g_{yy})}\right],$$

$$g_{xx}=\left|\frac{\partial R}{\partial x}\right|^2+\left|\frac{\partial G}{\partial x}\right|^2+\left|\frac{\partial B}{\partial x}\right|^2 \qquad g_{yy}=\left|\frac{\partial R}{\partial y}\right|^2+\left|\frac{\partial G}{\partial y}\right|^2+\left|\frac{\partial B}{\partial y}\right|^2 \qquad g_{xy}=\frac{\partial R}{\partial x}\frac{\partial R}{\partial y}+\frac{\partial G}{\partial x}\frac{\partial G}{\partial y}+\frac{\partial B}{\partial x}\frac{\partial B}{\partial y}$$

Assignment #5

1) An image has 200 X 200 pixels. The image shows a white rectangle of size 10 x 10 pixels in middle of the image and the background is black. If Gaussian noise with mean of 0.5 and variance of 0.01 is added to the image.

a) In Gaussian noise, what percentage of noise value is in the range of $\bar{z}-2\sigma$, $\bar{z}+2\sigma$ and what percentage in the range of $\bar{z}-\sigma$, $\bar{z}+\sigma$

b) What is the histogram of image before and after applying noise?

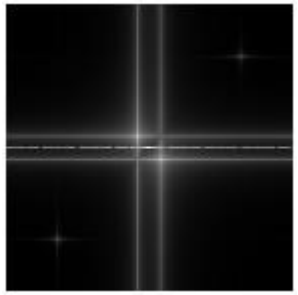
2) An image f(x,y) and its FFT , F(u,v) are shown below. The following noise functions (a-d) are added to the image and the Fourier transform of the degraded function due to noise are shown below. Explain which one corresponds to the noise function of a-d

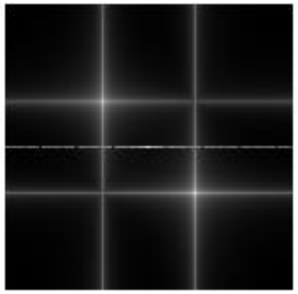
a) sin (x+y)+sin(x+2*y) b) sin(x+y)

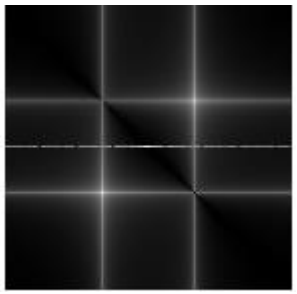
c) sin(x-y) d) sin(x/4+y/4)+sin(2x-2y)

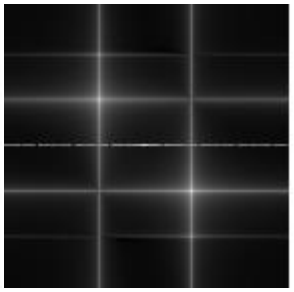
f(x,y)

F(u,v)









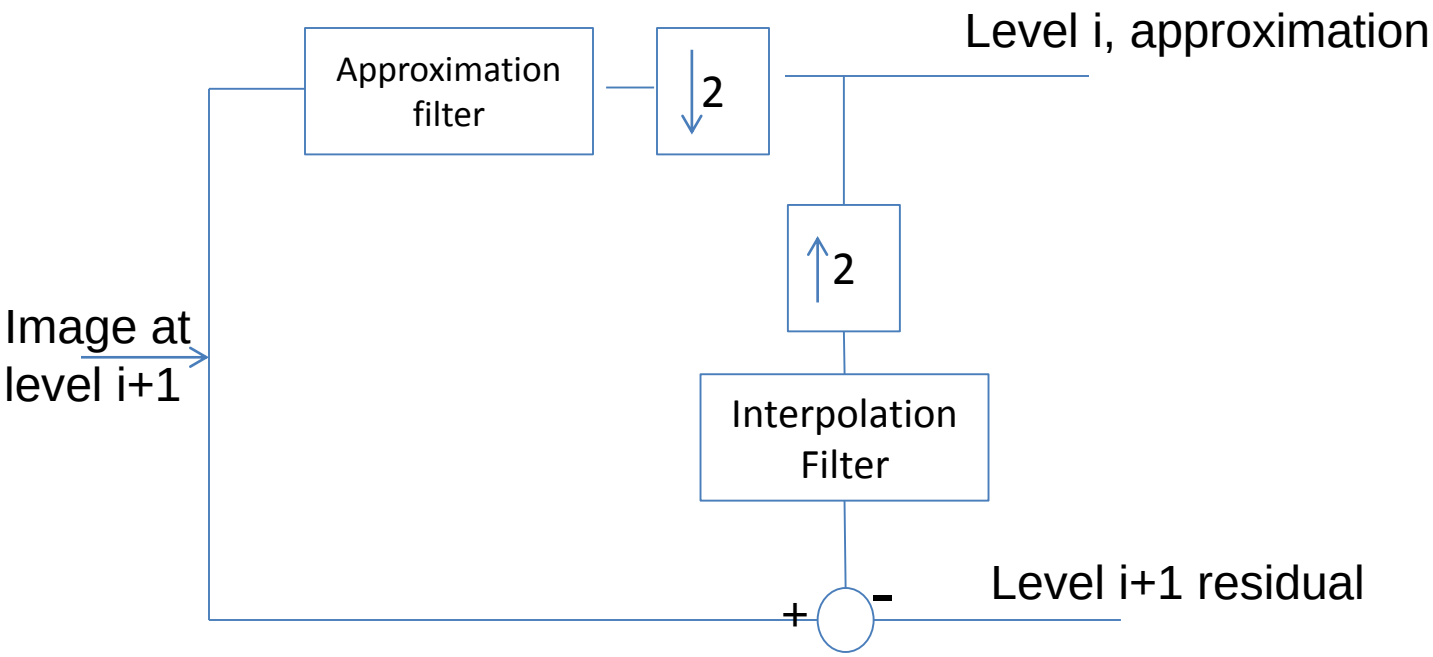
3) Explain how contra-harmonic filter operates if the area has constant intensity values of P?

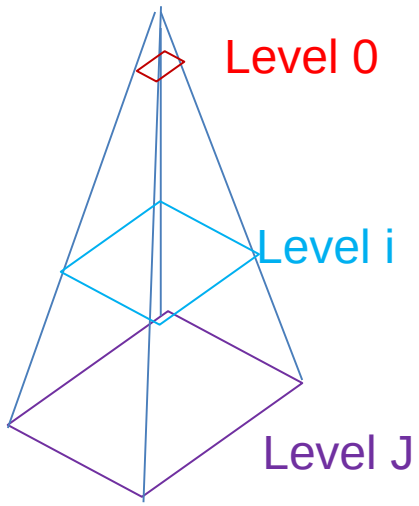
4) What is the difference between salt noise and pepper noise. Why contra-harmonic filters are effective in elimination of salt noise when Q= negative and pepper noise, when Q is positive?

Multi-resolution processing

Image Pyramid

- ✓ A simple way for showing an image in multi resolution is pyramid representation with J+1 level
  - ❖ Level J is the base of the pyramid . The image in this level has a size of $2^J \times 2^J = N \times N$
  - ❖ Level 0 is the top of the pyramid . The image in this level has a size of $2^0 \times 2^0 = 1 \times 1$
  - ❖ Level i is a level between 0 and J. The image in this level has the size of $2^i \times 2^i$
- ✓ To build an image in level i





- ✓ Usually we keep level P to J, not 0 to J . Why?
- ✓ What is the total number of pixels from level P to J ? (in terms of N)

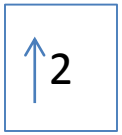
Image Pyramid(Cont.)

Approximation filter

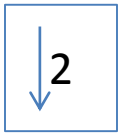
- ✓ Different types of approximation filtering can be used
  - ❖ No filtering → subsampling pyramid
  - ❖ Averaging filtering→ mean pyramid
  - ❖ Gaussian LPF filtering→ Gaussian pyramid

Interpolation Filter

- ✓ Different types of interpolation filtering can be used
  - ❖ Nearest neighbor
  - ❖ Bilinear



- ✓ **Up-sampling by 2**. (inserting a sample after any sample both in row and column directions)

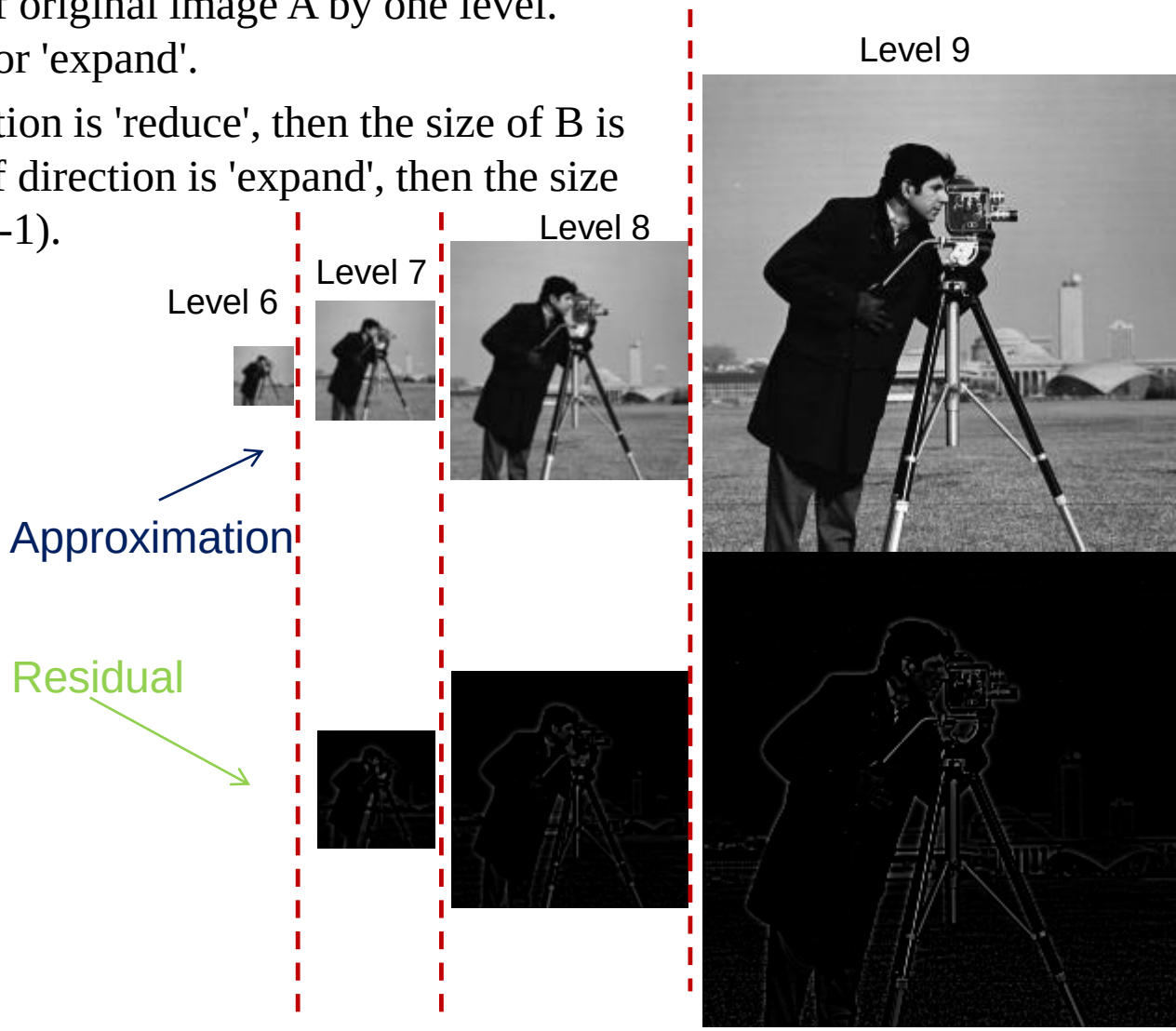


- ✓ **Down-sampling by 2**. (discarding every other sample both in row and column directions)

Image pyramid- MATLAB

- ✓ MATLAB uses **impyramid** function for Image pyramid reduction and expansion
  - ❖ B = impyramid (A, direction) computes a Gaussian pyramid reduction or expansion of original image A by one level. direction can be 'reduce' or 'expand'.
  - ❖ If A is M-by-N and direction is 'reduce', then the size of B is ceil(M/2)-by-ceil(N/2). If direction is 'expand', then the size of B is (2*M-1)-by-(2*N-1).

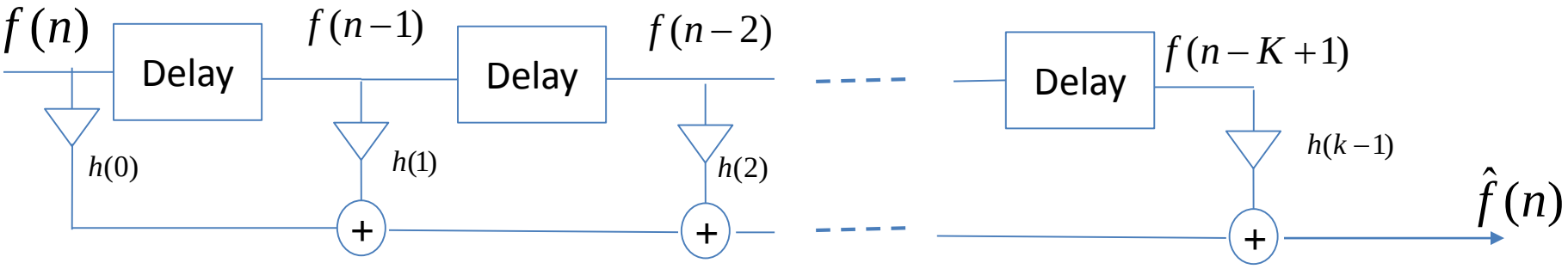
```
I9 = imread('cameraman.tif');

I8 = impyramid(I9, 'reduce');
I9e = impyramid(I8, 'expand');
R9=imsubtract(I9(1:255,1:255),I9e);

I7 = impyramid(I8, 'reduce');
I8e = impyramid(I7, 'expand');
R8=imsubtract(I8(1:127,1:127),I8e);

I6 = impyramid(I7, 'reduce');
 I7e = impyramid(I6, 'expand');
 R7=imsubtract(I7(1:63,1:63),I7e);
```



Sub-band Coding

- ✓ In this technique, an image is decomposed into a set of sub-bands at source and the sub-bands can be reassembled to reconstruct the original image at destination.
- ✓ Decomposition and reconstruction can be implemented using digital filters.
- ✓ An order K digital filter can be shown as

Order K digital filter



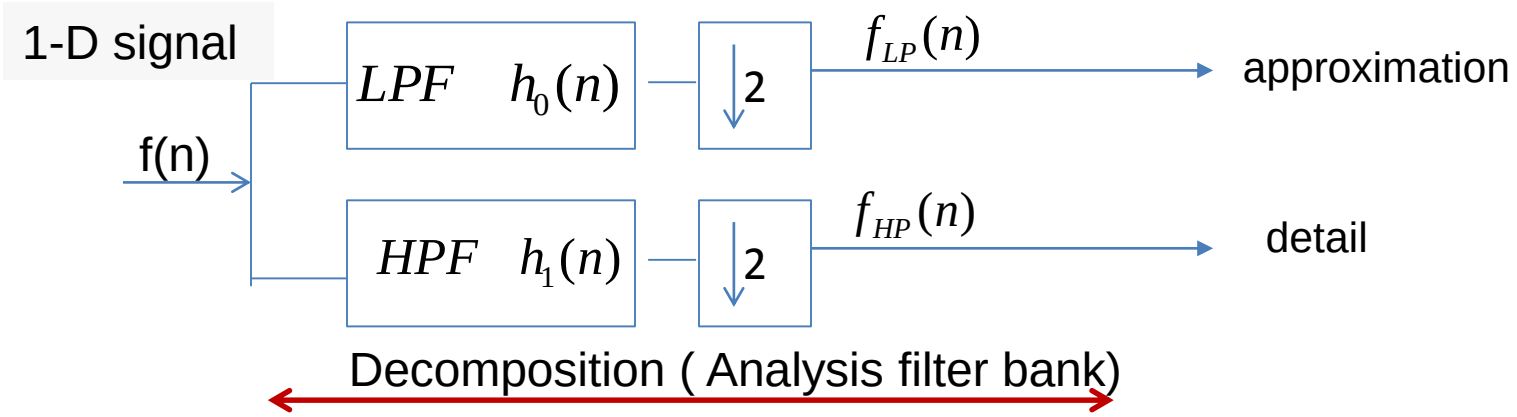
$$\hat{f}(n) = h(0)\,f(n) + h(1)\,f(n-1) + h(2)\,f(n-2) + \ldots + h(K-1)\,f(n-K+1)$$

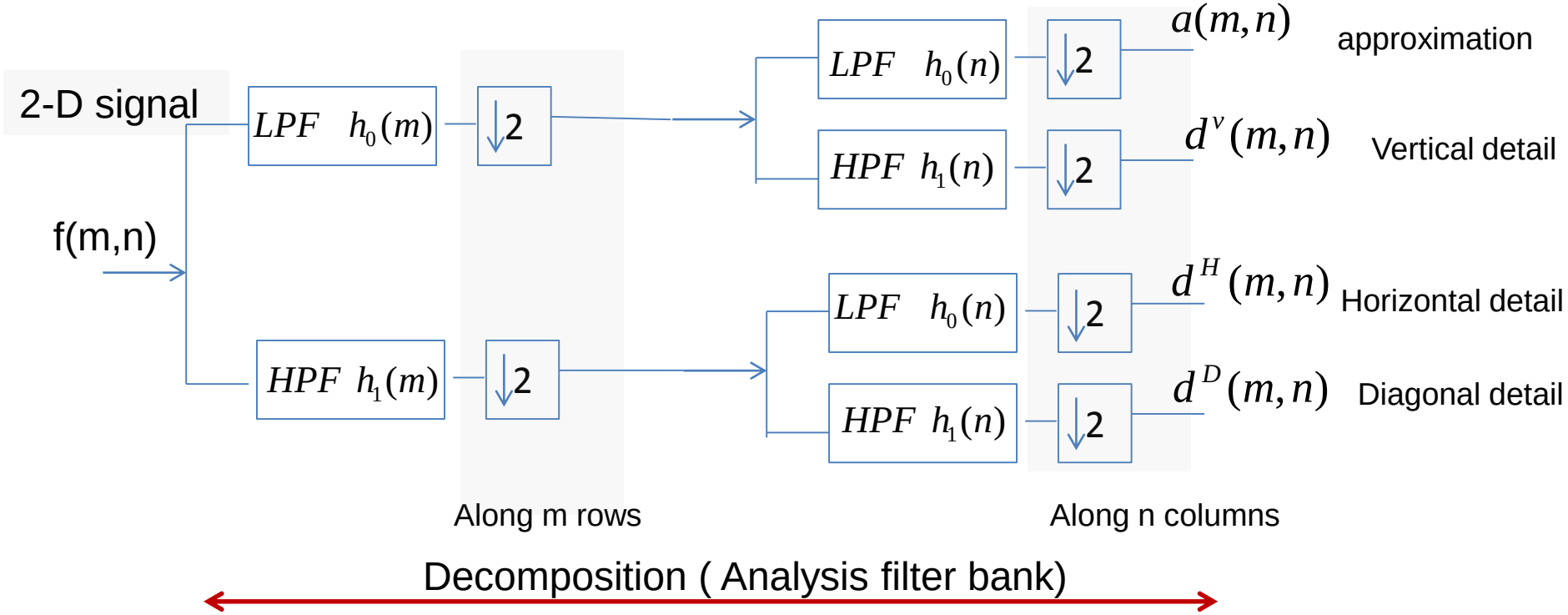
$$\hat{f}(n) = \sum_{i=0}^{K-1} h(i)\,f(n-i) = f(n) * h(n)$$

- ✓ In special case if $f(n) = \delta(t) \Rightarrow \hat{f}(n) = h(n)$

Sub-band Coding- Decomposition

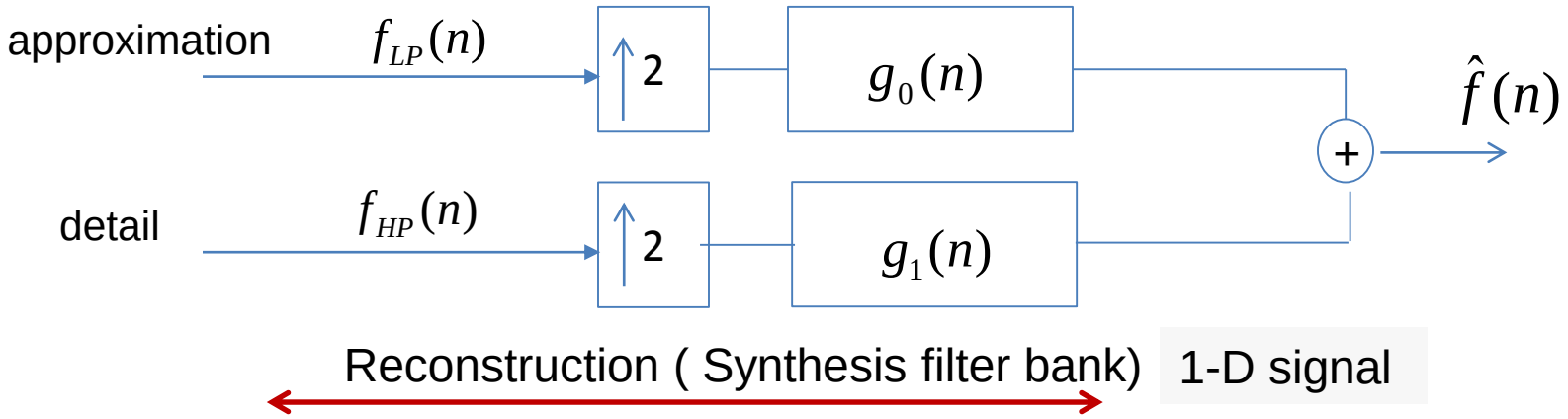
✓ Decomposition for 1-D and 2-D signal can be implemented as follows



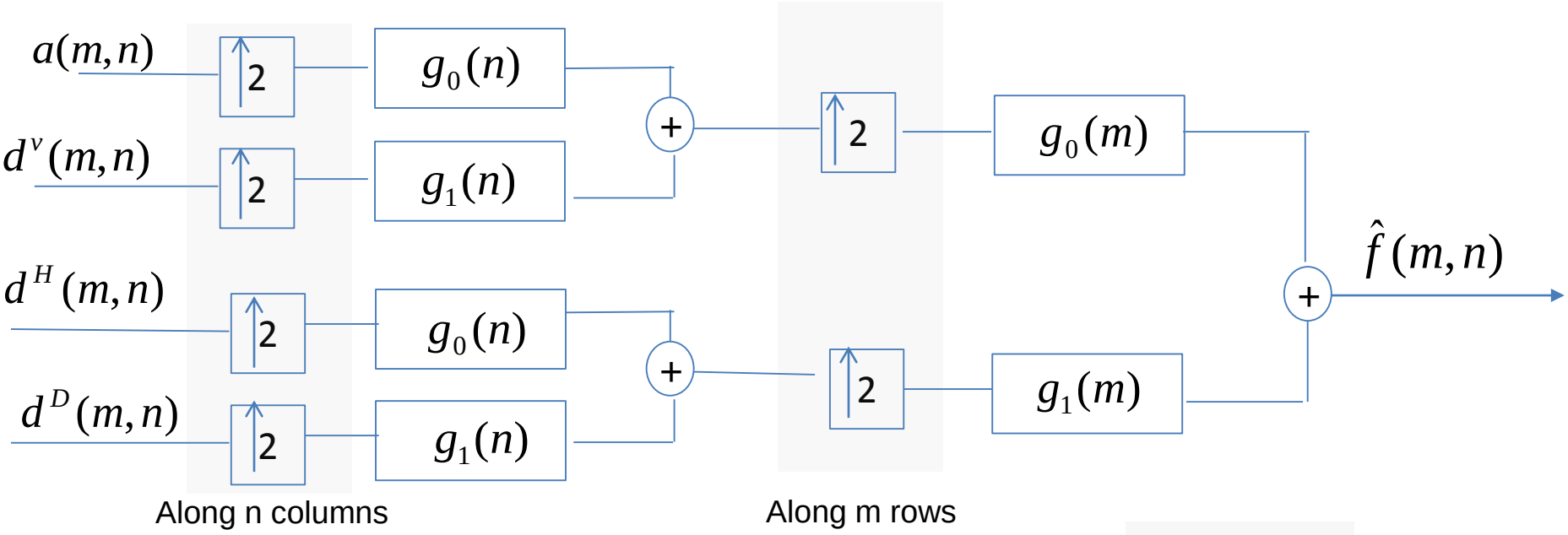


Sub-band Coding- Reconstruction

✓ Reconstruction for 1-D and 2-D signal can be implemented as follows



Reconstruction (Synthesis filter bank) 1-D signal



Reconstruction (Synthesis filter bank) 2-D signal

Sub-band Coding- Design

- ✓ In sub-band coding the filters $h_0(n),\ h_1(n)\ , g_0(n), g_0(n)$ are selected in such a way that

$$\hat{f}(n) = f(n)$$

- ✓ There are many ways to design the filters to satisfy the above condition, such as
  - ❖ 1- All filters are designed based on filter $g_0(n)$ which is called the prototype filter. If K is the number of filter coefficients for $g_0(n)$, and K is even, the rest of filters can be designed as

$$\begin{cases} g_1(n) = (-1)^n\ g_0(K\ -1-n), & K\ must\ be\ even \\ h_i(n) = g_i(K_{even} - 1 - n) & i = 0,1 \end{cases}$$

  - ❖ 2- the filters $g_0(n), g_0(n)$ are designed based on having filter $h_0(n)$ and $h_1(n)$

$$\begin{cases} g_0(n) = (-1)^n\ h_1(n) \\ g_1(n) = (-1)^{n+1}\ h_0(n) \end{cases} \quad \text{Or} \quad \begin{cases} g_0(n) = (-1)^{n+1}\ h_1(n) \\ g_1(n) = (-1)^n\ h_0(n) \end{cases}$$

Sub-band Coding- Example

✓ Design using Prototype filter $g_0(n)$: 8 Tap Daubechies Filter

g_0 =[0.2304 0.7148 0.6309 -0.0280 -0.1870 0.0308 0.0329 -0.0106]

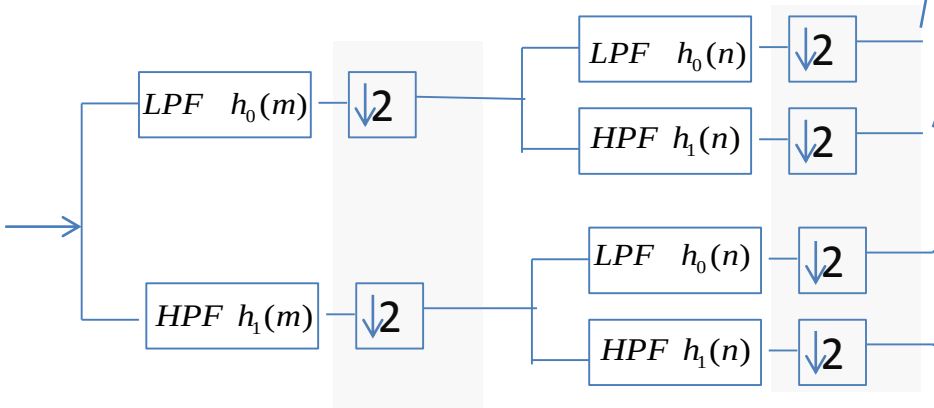
Other filters can be designed using

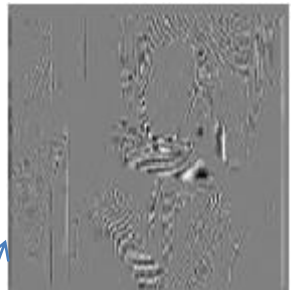
$$\begin{cases} g_1(n) = (-1)^n \; g_0(K \; -1-n), \;\; K \; must \; be \; even \\ h_i(n) = g_i(K_{even} - 1 - n) \quad i = 0,1 \end{cases}$$

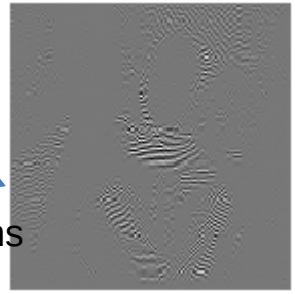
g1= [-0.0106 -0.0329 0.0308 0.1870 -0.0280 -0.6309 0.7148 -0.2304]

h0= [-0.0106 0.0329 0.0308 -0.1870 -0.0280 0.6309 0.7148 0.2304]

h1= [-0.2304 0.7148 -0.6309 -0.0280 0.1870 0.0308 -0.0329 -0.0106]

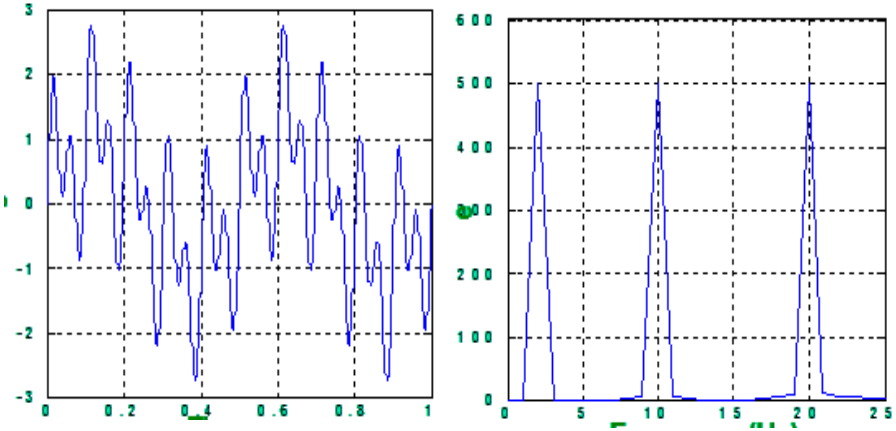






Wavelet Transform- why there is a need for another transformation

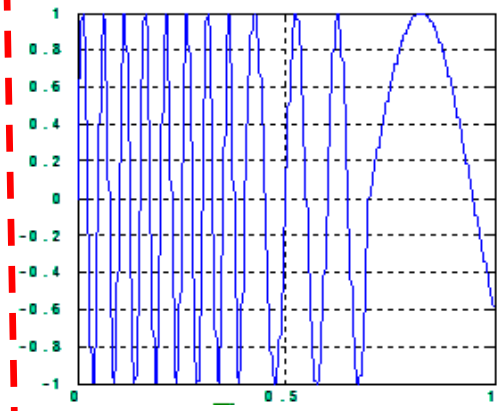
- ✓ In Fourier Transformation (FT) , a signal is represented in terms of the sum of indefinitely long sine waves. So, Fourier transformation is good for finding the spectrum of stationary signals in which all frequency components exist at all time.
- ✓ For the non- stationary signals in which their spectral characteristics vary with time, FT does not give any information of when a frequency component exist. FT only gives what frequency components exist in the Signal. The Time and Frequency information can not be seen at the Same Time.
- ✓ For non-stationary signals, Time-frequency Representation of the signal is needed.



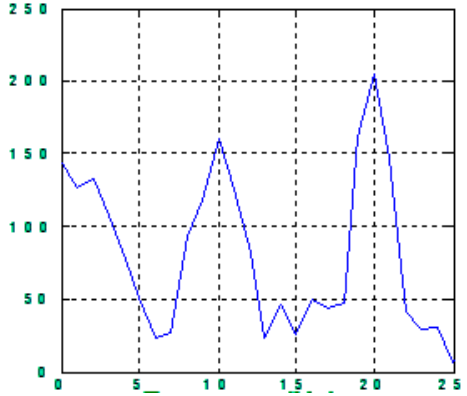
Time

Frequency

A stationary signal
2Hz + 10 Hz + 20 Hz



Time

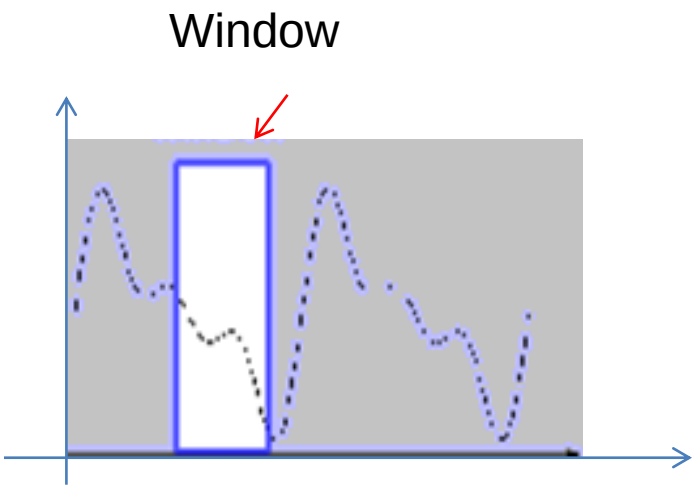


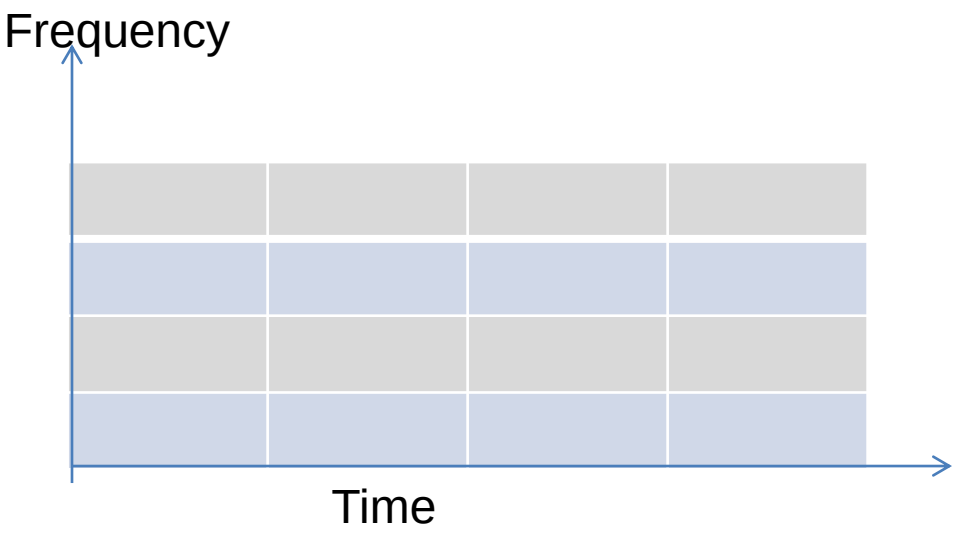
Frequency

A non- stationary signal
2Hz (0-0.4 sec) +10 Hz (0.4 sec-0.7 sec)
+ 20 Hz (0.7 sec-1 sec)

STFT (short time Fourier Transform)

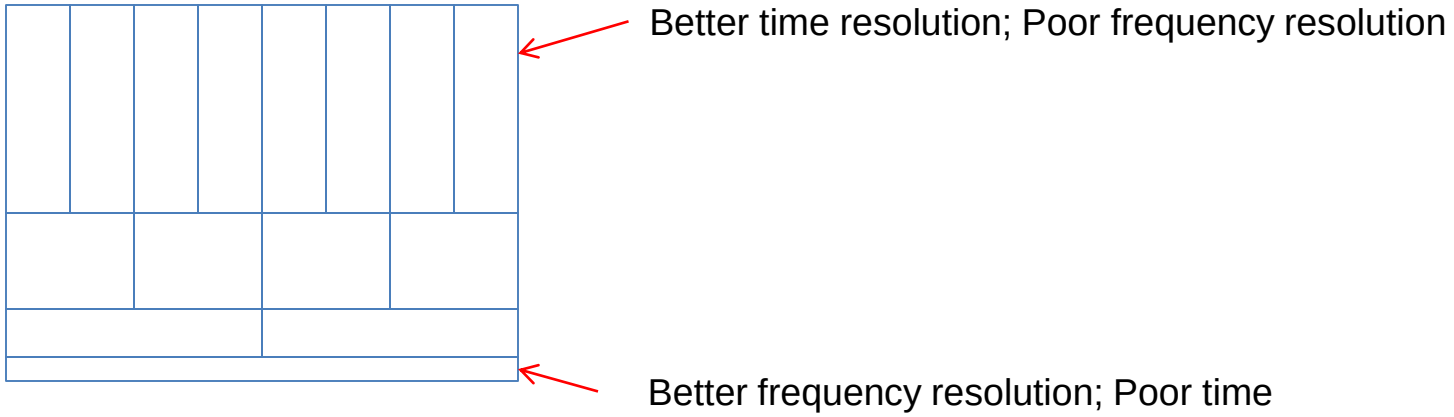
- ✓ STFS is a variation of FT that calculates the Fourier transforms in a short-time called a window .
  - ❖ The assumption is that the signal is stationary in each window.
- ✓ There is a dilemma in choosing the size of the window
  - ❖ Narrow window -> poor frequency resolution
  - ❖ Wide window -> poor time resolution
- ✓ Uncertainty principle
  - ❖ It is not possible to find what frequency exists at what time intervals





Wavelet Transform

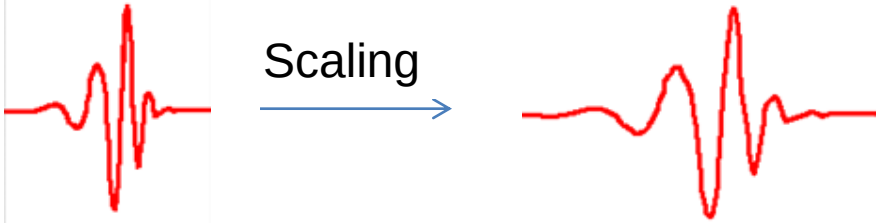
- ✓ Wavelet transform is an alternative approach to the STFT to overcome the resolution problem. Similar to STFT, the signal is multiplied with a function
- ✓ Multi-resolution Analysis
  - ❖ Analyze the signal at different frequencies with different resolutions
  - ❖ Good time resolution and poor frequency resolution at high frequencies
  - ❖ Good frequency resolution and poor time resolution at low frequencies
  - ❖ More suitable for short duration of higher frequency; and longer duration of lower frequency components
- ✓ Split Up the Signal into a Bunch of Signals
- ✓ Representing the Same Signal, but all Corresponding to Different Frequency Bands
- ✓ Only Providing What Frequency Bands Exists at What Time Intervals



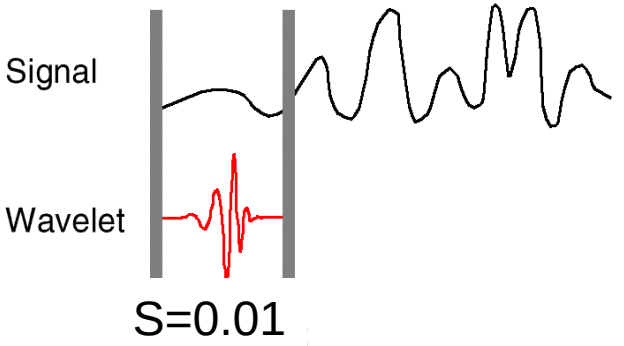
Wavelet of a continuous signal

Example of a Wavelet function

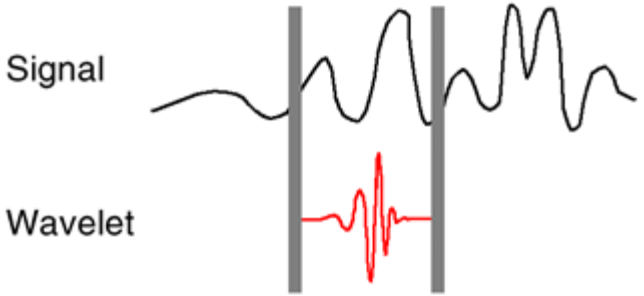
- ✓ A wavelet is a waveform of effectively limited duration that has an average value of zero.
- ✓ The wavelet function can be scaled using a scaling function

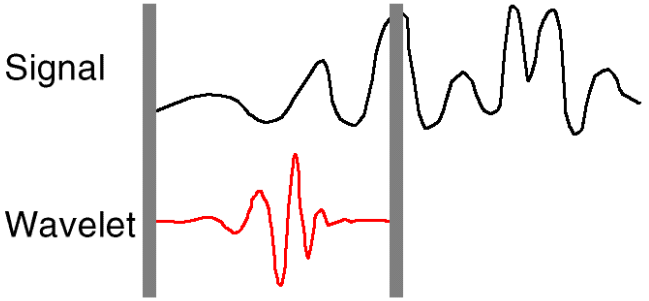


- ✓ Continuous wavelet transform is calculated by the sum over all time of the signal, multiplied by scaled and shifted versions of the wavelet function
- ✓ 1- Take a Wavelet and compare it to a section at the start of the original signal
- ✓ 2- Calculate a number, S, that represents how closely correlated the wavelet is with this section of the signal. The higher S is, the more the similarity.
- ✓ 3-Shift the wavelet to the right and repeat step 2 until you've covered the whole signal
- ✓ 4- Scale (stretch) the wavelet and repeat steps 2-3



S=0.01





S=0.2

Discrete Wavelet Transform

- ✓ For a sequence of f(n), where the sequence has M elements, the sequence can be represented in wavelet domain using the scaling function and the wavelet function $\varphi_{j_0,k}(n)$ as

$$f(n) = \frac{1}{\sqrt{M}} \sum_k W_\varphi(j_0,k)\, \varphi_{j_0,k}(n) + \frac{1}{\sqrt{M}} \sum_{j=j_0}^{\infty} \sum_k W_\psi(j,k)\, \psi_{j,k}(n)$$

Scaling coefficient — Scaling function — Wavelet coefficient — Wavelet function

$J_0 = starting\ \ point\ usually \quad j_0 = 0$

$n = 0,1,2,\ldots,M-1 \quad M = 2^J$

$j = 0,1,\ldots,J-1$

$$k = 0,1,2,\ldots,2^j - 1 \Rightarrow \begin{cases} j=0 & k=0 \\ j=1 & k=0,1 \\ \vdots & \\ j=J-1 & k=0,1,\ldots,\frac{M}{2}-1 \end{cases}$$

$$W_\varphi(j_0,k) = \frac{1}{\sqrt{M}} \sum_n f(n)\, \varphi_{j_0,k}(n)$$

$$W_\psi(j,k) = \frac{1}{\sqrt{M}} \sum_n f(n)\, \psi_{j,k}(n) \qquad j \geq j_0$$

Discrete Wavelet Transform

- ✓ Find the wavelet coefficients for f{3, 4, 5, 6}, if

$$\varphi_{0,0}(n) = \{1, 1, \ 1, \ 1\}$$
$$\psi_{0,0}(n) = \{1, 1, -1, -1\}$$
$$\psi_{1,0}(n) = \{\sqrt{2}, -\sqrt{2}, \ 0, \ 0\}$$
$$\psi_{1,1}(n) = \{0, 0, \sqrt{2}, -\sqrt{2}\}$$

$$W_{\varphi}(j_0, k) = \frac{1}{\sqrt{M}} \sum_{n} f(n)\, \varphi_{j_0,k}(n) \qquad W_{\psi}(j, k) = \frac{1}{\sqrt{M}} \sum_{n} f(n)\, \psi_{j,k}(n) \qquad j \geq j_0$$

$M = 4 \Rightarrow J = 2$

$j = 0,1$

$k = 0,1,2,\ldots\ldots, 2^{j} - 1 \Rightarrow \begin{cases} j = 0 & k = 0 \\ j = 1 & k = 0,1 \end{cases}$

$$W_{\varphi}(0,0) = \frac{1}{\sqrt{4}}[3*1 + 4*1 + 5*1 + 6*1] = 9$$

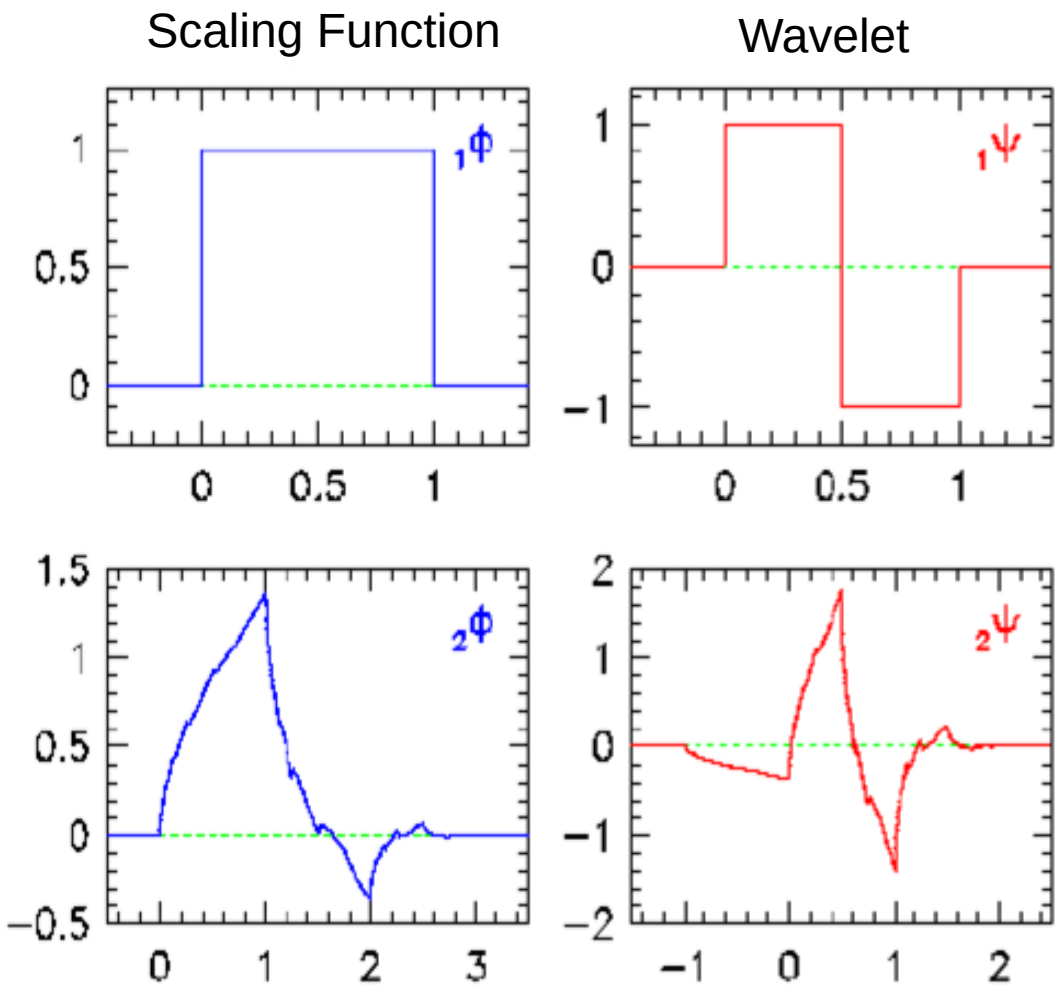
$$W_{\psi}(0,0) = \frac{1}{\sqrt{4}}[3*1 + 4*1 + 5*-1 + 6*-1] = -2$$

$$W_{\psi}(1,0) = \frac{1}{\sqrt{4}}[3*\sqrt{2} + 4*-\sqrt{2} + 5*0 + 6*0] = -0.70711$$

$$W_{\psi}(1,1) = \frac{1}{\sqrt{4}}[3*0 + 4*0 + 5*\sqrt{2} + 6*-\sqrt{2}] = -0.70711$$

Wavelet function examples

- Haar function
- Daubechies function



JPEG 2000

- ✓ An image compression standard that uses wavelet transformation and provides Low bit-rate compression performance
- ✓ Besides, it offers several interesting features
  - ❖ Progressive transmission by quality, resolution, component, or spatial locality
  - ❖ support for the tiling of images
    - Images can be compressed in rectangular tiles of any size. Only those tile parts of interest or those parts that can fit on the display screen need be decompressed for viewing. Each tile can be compressed and decompressed by resolution or quality.
  - ❖ Support for region of interest
    - Coding different regions of the image with different quality

JPEG

JPEG 2000

JPEG 2000 is generating a better quality image as compared to JPEG
(for the same image size)

Assignment #6

1) In a pyramid image representation, the approximation process use a 2x2 averaging filter and the interpolation process repeats the same value. For 4x4 image shown below, build the pyramid and show the approximation and prediction residual frame ?

$$f(x,y) = \begin{bmatrix} 4 & 3 & 0 & 1 \\ 6 & 2 & 2 & 1 \\ 1 & 0 & 1 & 3 \\ 0 & 1 & 2 & 9 \end{bmatrix}$$

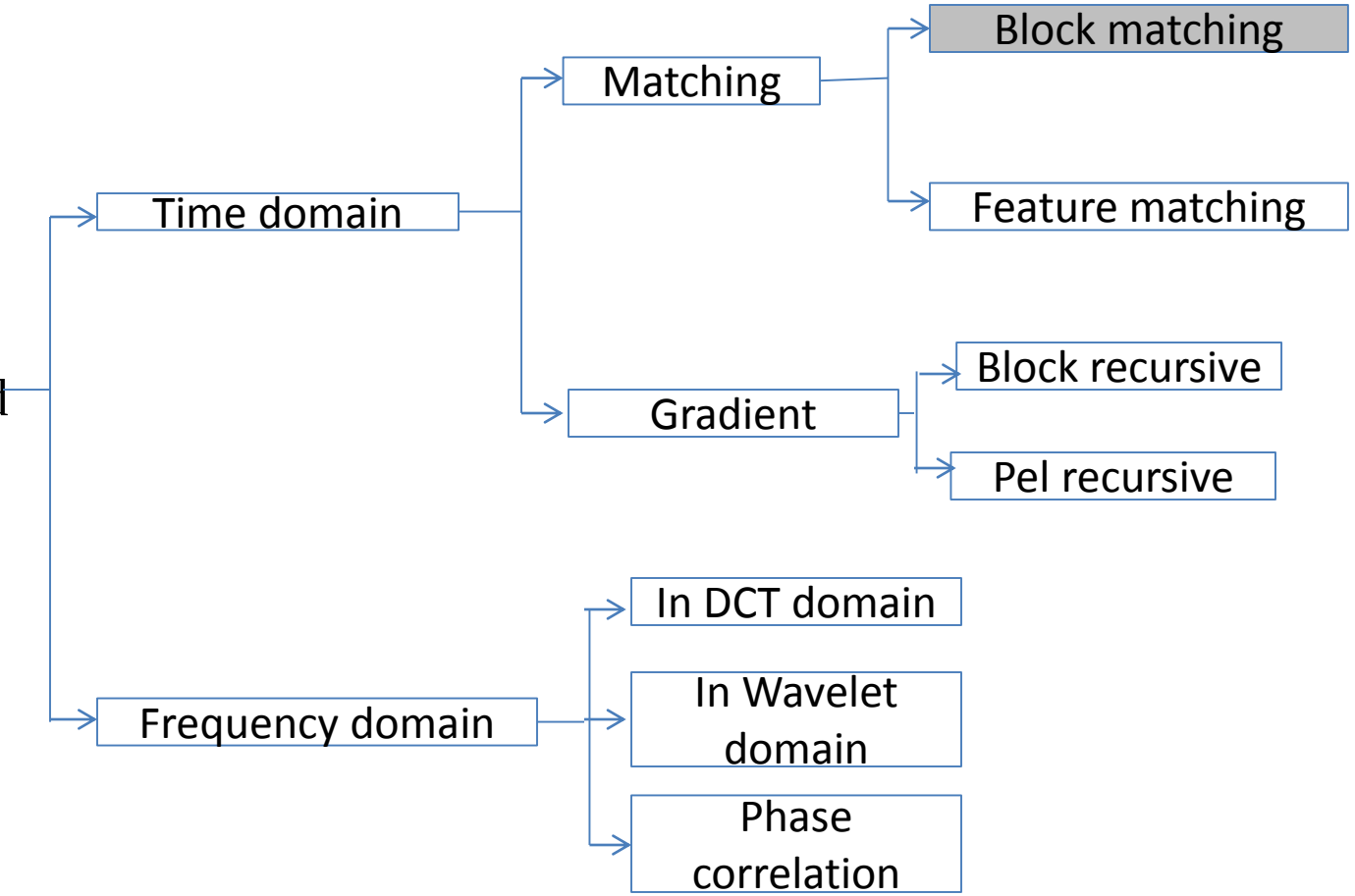
2) In image, what is the percentage of increase in data size in pyramid representation?

Video Processing

Motion Estimation

- ✓ There are many ways to estimate the motion within frames. Block based motion estimation techniques are very popular. We will study some of the existing block-matching methods in this course.

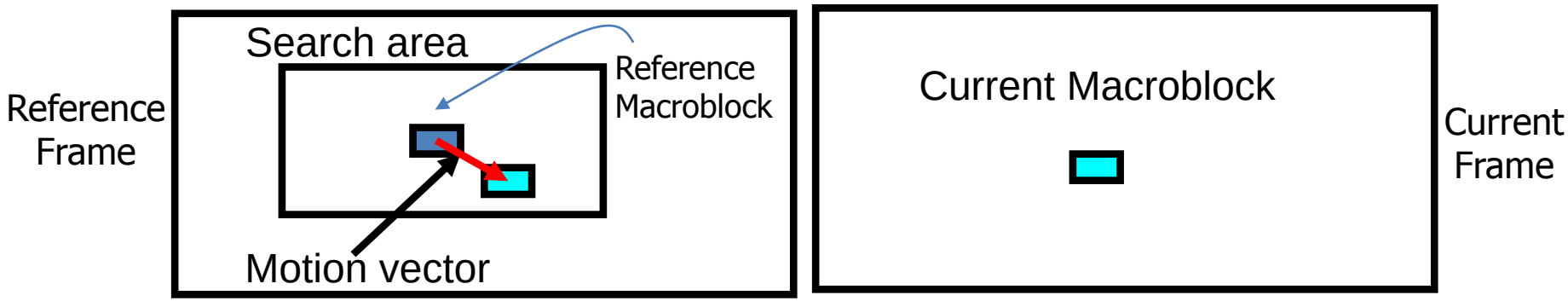
- ✓ Block-based method



- ✓ Mesh-based method

Block-Matching Ttechnique (BMT)

- ✓ Current Frame: Frame which is being analyzed to derive motion vectors
- ✓ Reference Frame : Frame in past (or future) used to predict in current frame
- ✓ Motion vector : The displacement of the closest matching block in reference frame for a block in current frame
- ✓ Motion estimation is the process of finding the values of motion vectors for each frame
- ✓ Block matching techniques assume that all pixels within a block have the same uniform motion (no rotational motion)
- ✓ Range of motion vector is constrained by 'Search area'
- ✓ To do block matching, for each block in current Frame
  - ❖ Search in the reference frame within search area
  - ❖ Find the closest matching block (One with the least distortion with respect to the current block)
  - ❖ Determine the displacement as motion vector



Overview of BMT

- ✓ Search Area of size $(2P+1)\times(2q+1)$ represents displacement of the blocks in reference frame in the range (-p, p) and (-q, q) in x and y directions.
  - ❖ the distortion (distance) between current block and all possible displacements of the blocks in reference frame within the search area needs to be calculated based on a distortion measure.
  - ❖ the position of the block in search area with the least distortion will be chosen as best matched block.
  - ❖ The Vector represents the relative position of the best-matched block with respect to current block is the motion vector.

Distortion Measures

- ✓ distortion criterion for measuring distance between previous block and search area block. Various Criterions are
  - ❖ Mean Square Error (MSE)
  - ❖ Mean Absolute Error(MAE) =Mean Absolute difference (MAD)
  - ❖ Cross Correlation Function (CCF)
  - ❖ Sum of Absolute Difference (SAD)

- ✓ MSE and MAD are commonly employed due to their simplicity in hardware implementation. If C_{ij} and R_{ij} are the pixels the current and reference block respectively, and macro-block is $N \times N$, then

$$MAD = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| C_{ij} - R_{ij} \right|$$

$$MSE = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left(C_{ij} - R_{ij} \right)^2$$

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| C_{ij} - R_{ij} \right|$$

$$CCF = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C_{ij} \, R_{ij}}{\sqrt{\left(\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C^2{}_{ij} \right) * \left(\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} R^2{}_{ij} \right)}}$$

Exhaustive Search (ES)

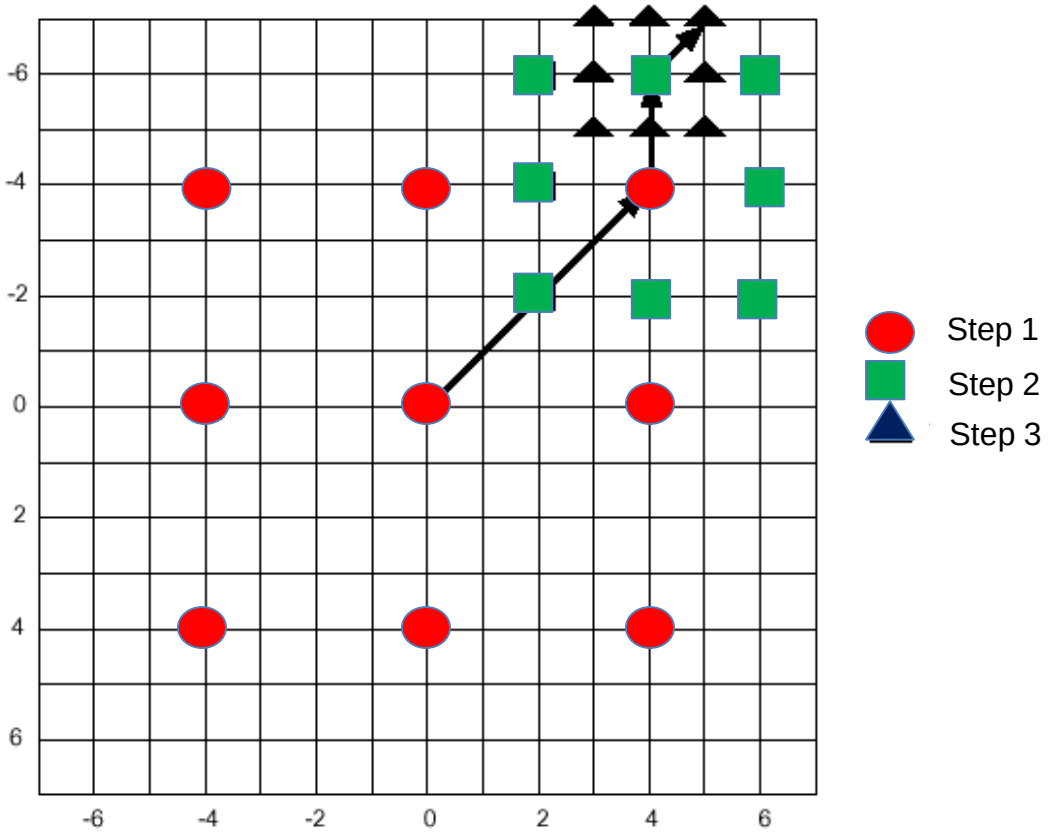
- ✓ This method is also called exhaustive block matching algorithm (EBMA)
- ✓ In this method all possible displacements in Search area are evaluated. And as a result finds the best match possible, as compared to other block matching algorithms.
- ✓ The disadvantage of this method is its computational complexity
  - ❖ For a search area of (-p, p), (-q, q) around the current position, $(2P+1)\times(2q+1)$ distortion values need to be computed.
  - ❖ Simplest algorithm, but computationally most expensive
  - ❖ The obvious disadvantage to ES is that the larger the search window gets the more computations it requires.
- ✓ Fast block matching algorithms try to achieve almost the same result by doing as little computation as possible. Example of fast methods:
  - ❖ Three step search (TSS)
  - ❖ 2D Logarithmic Search (2DLS)
  - ❖ Orthogonal Search (OS)
  - ❖ New Three Step Search (NTSS)
  - ❖ Hierarchical block matching algorithm(HBMA)

Three Step Search (TSS or 3SS)

For each step, nine checking points are matched and the minimum distorion point of that step is chosen as the starting center of the next step.

A search paths are shown here, which needs (9 + 8 + 8)=25 checking points.

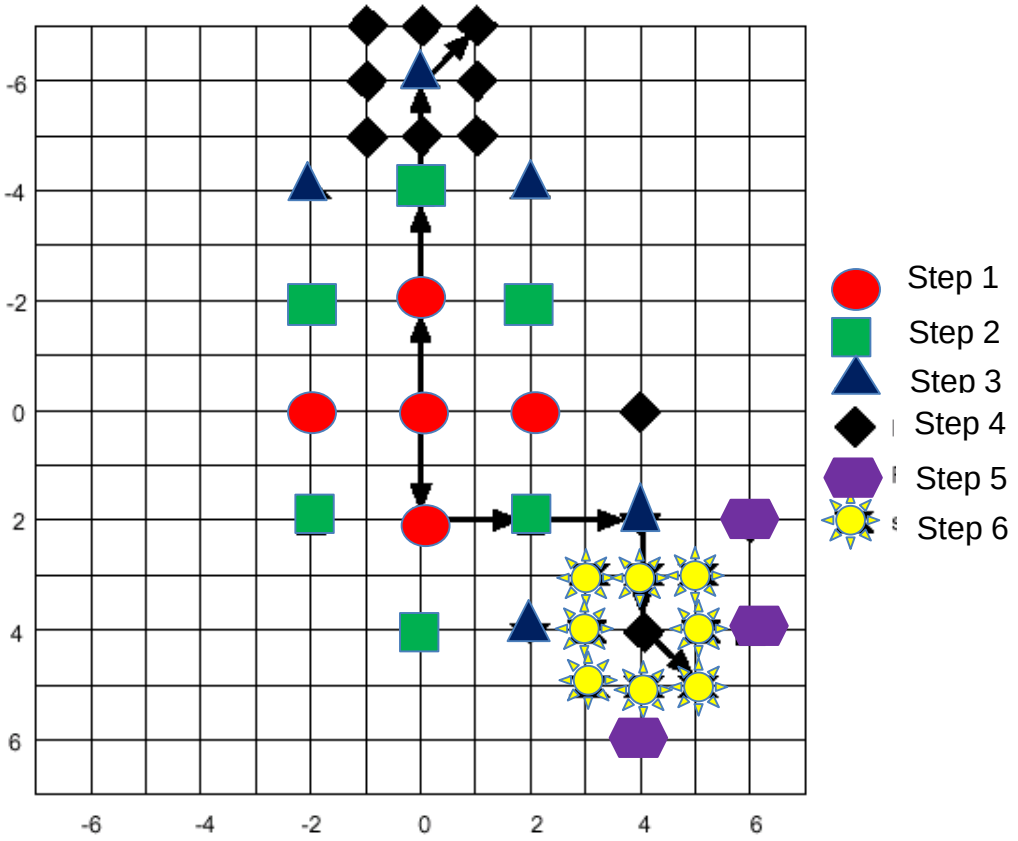
For larger search window, this method can be easily be extended to n-steps using the same searching strategy with the number of checking points required equals to [1 + 8 log2(d + 1)].



2D Logarithmic Search (2DLS)

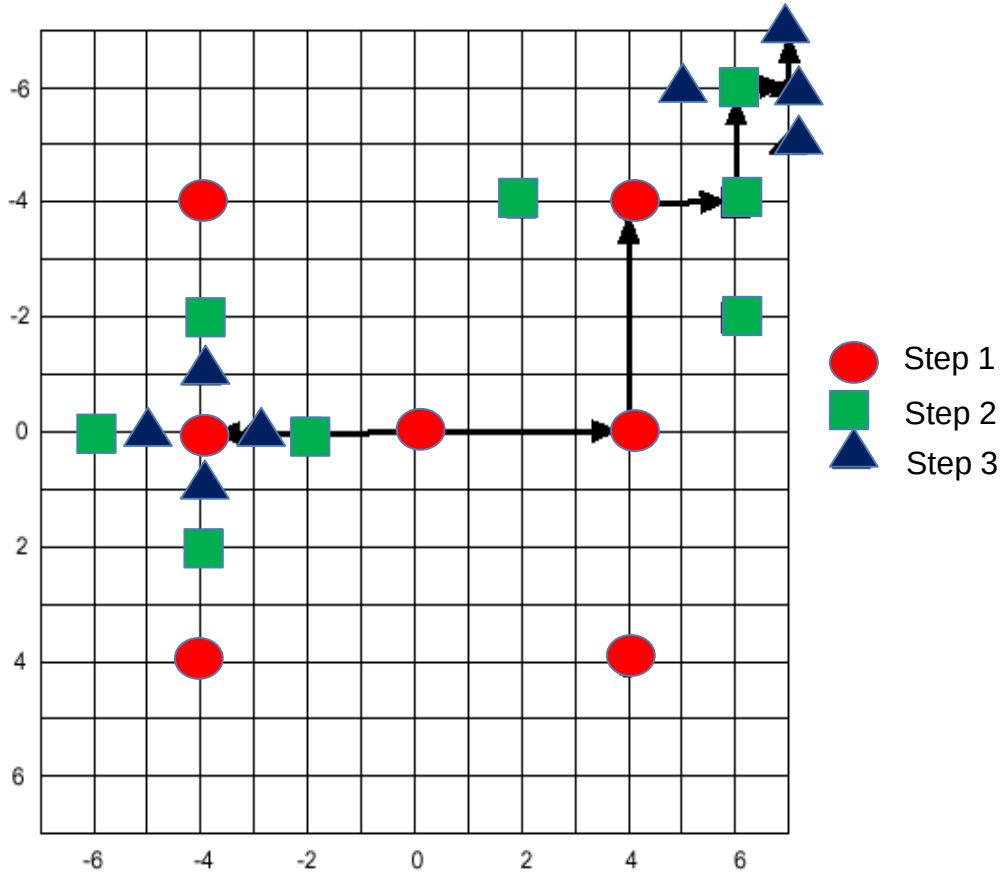
- ✓ This fast method, starts from the zero displacement and computes the error of blocks in the five locations points which are arranged in a diamond shape with an initial step size. It select the location corresponding to minimum error and uses that location as a center of next step.
- ✓ The step size is reduced by half only when the selected points for the next step is the center one or the current minimum point reaches the search window boundary. Otherwise, the step size remains the same.

- ✓ When the step size reduced to 1, all the 8 checking points adjacent to the center checking point of that step are searched.
- ✓ Two different search paths are shown here,
  - The top search path requires (5 +3 +3 +8) = 19 checking points.
  - The lower-right search path requires (5+3+2+3+2+8) =23 checking points.



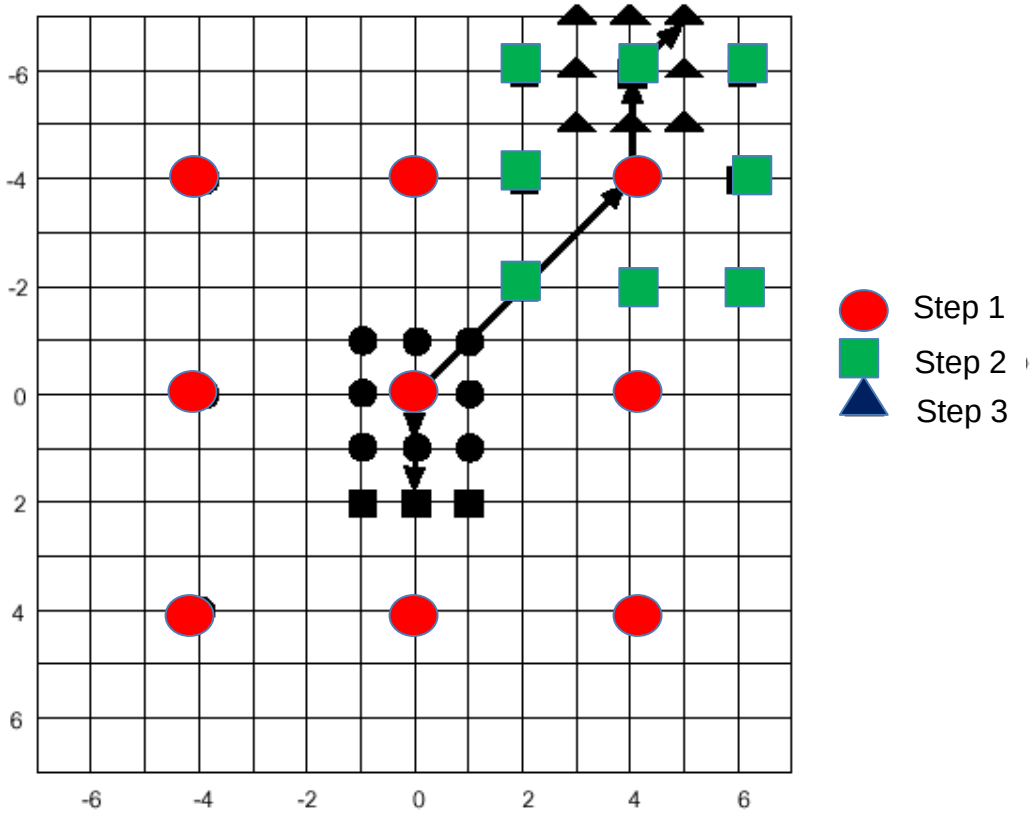
Orthogonal Search (OS)

- ✓ It consists of pairs of horizontal and vertical steps with a logarithmic decreasing in step size
- ✓ The search paths of OSA are shown in Starting from the horizontal searching step, three checking points in the horizontal direction are searched.
- ✓ The minimum checking point then becomes the center of the vertical searching step which also consists of three checking points.
- ✓ Then the step size decreases by half and using the same searching strategy. The algorithm ended with step size equals to one.
- ✓ Two different search paths are shown here.
- ✓ For d = 7, the OSA algorithm requires a total of (3 + 2 + 2 +2 + 2 +2)=13 checking points. For the general case, the OSA algorithm requires (1 + 4 log2(d + 1)) checking points.



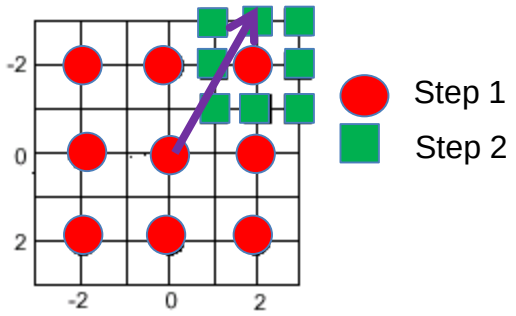
New Three Step Search (NTSS)

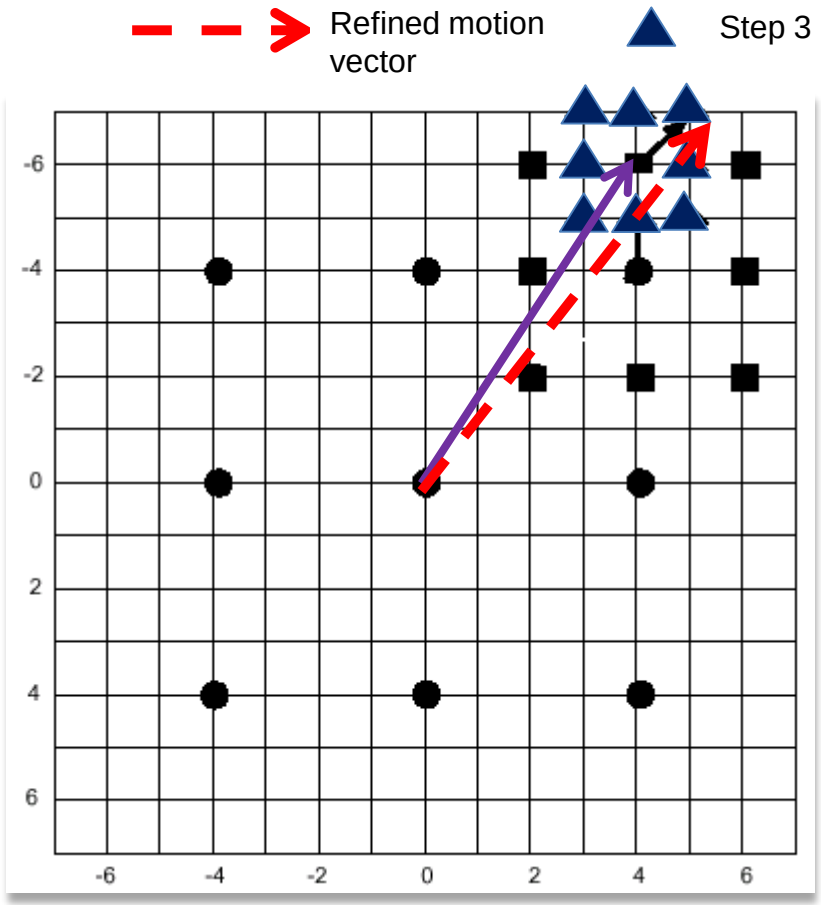
- ✓ For those video sequences where the motion vector distribution is highly center biased, an additional 8 neighbor checking points are searched in the first step of N3SS
- ✓ Two search paths with d = 7 are shown.
  - The center path shows the case of searching small motion. In this case, the minimum distortion point of the first step is one of the 8 neighbor checking points. The search is halfway-stopped with matching three more neighbor checking points of the first step's minimum distortion point. The number of checking points required is (17 + 3) = 20.
  - The upper-right path shows the case of searching large motion. In this case, the minimum distortion point of the first step is one of the outer eight checking points. Then the searching procedures proceed the same as the 3SS algorithm. The number of checking points required is(17 + 8 + 8)=33.



Hierarchical block matching algorithm (HBMA)

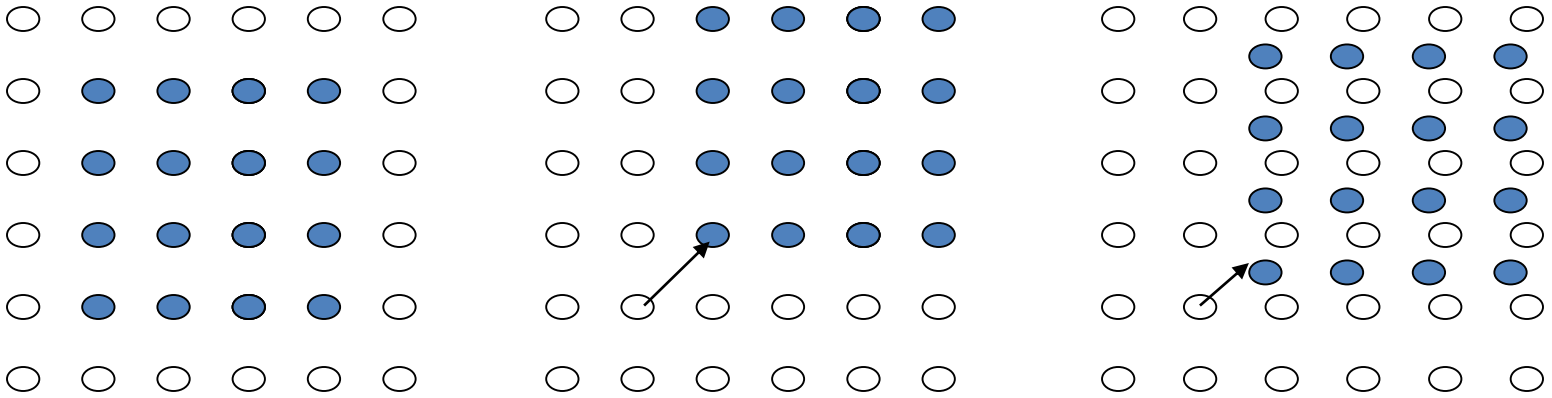
- ✓ The basic idea of hierarchical (multi-resolution) block matching is to perform motion estimation at each level successively, starting with the lowest resolution level .
- ✓ The estimate of the motion vector at a lower resolution level is then passed onto the next higher resolution level as an initial estimate. The motion estimation at higher level refine the motion vector of the lower one. At higher levels, relatively smaller search window can be used as it starts with a good initial estimate.
- ✓ For each level, one could use fast BMAs such as 3SS, 2DLOG for fast motion estimation.
- ✓ Suppose there is a HBMA with two levels as shown. The lower level is formed by sub-sampling, the higher level by a factor of two in both horizontal and vertical directions. One pixel displacement at the lower level corresponds to two pixels displacement at the higher level. That is, the search window size in pixel is fourth of the one at higher level.





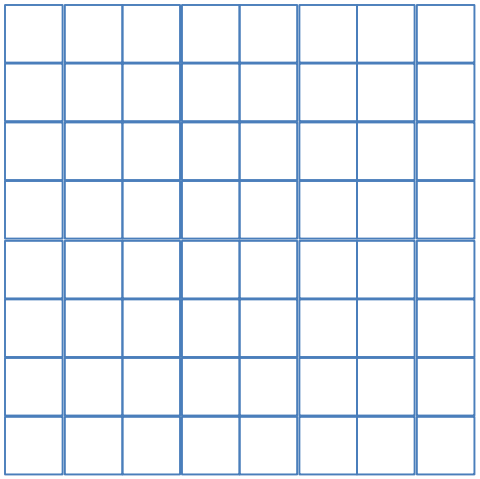
Sub-Pixel Motion Vector

- ✓ If the horizontal and vertical components of a motion vector are integers the relevant block in the reference frame actually exist
- ✓ If one or both components are fractional values, the relevant block is virtually generated by processing and interpolation
- ✓ Sub-pixel motion compensation can provide significantly better compression performance than the integer-pixel compensation
- ✓ The half pixel positions are generated first and are interpolated from neighbouring integer-pixel samples using 6-tap FIR filter. Once all the half-pixel samples are available, each quarter-pixel is produced using bilinear interpolation between neighbouring half or integer pixels.

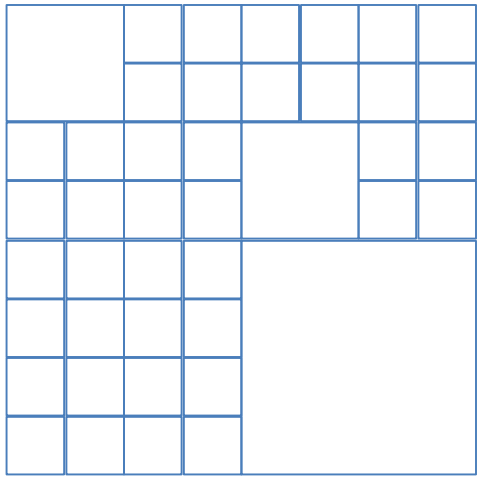


Motion Compensation

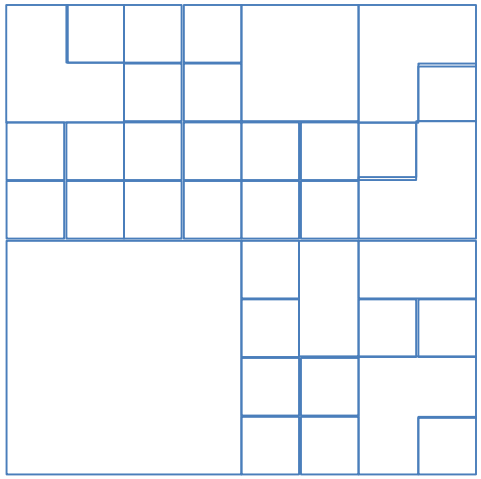
- ✓ Components of a Typical Motion-Based Video Coding System are shown avove.
- ✓ Block-based motion compensation scheme
  - ❖ Fixed-Size Block Motion Compensation (FSBMC)
    - Divides a frame into non-overlapping blocks of equal size. It is simple and does not require any partitioning structure since, the location of each block is fixed based on the block size
  - ❖ Variable-Size Block Motion Compensation (VSBMC)
    - A frame is partitioned into variable size square. This scheme needs a quad-tree decomposition structure. One bit corresponding to a node in this structure indicates whether or not the corresponding block has been split into four sub-blocks
  - ❖ Region-Wise Motion Compensation (RWMC)
    - a frame is partitioned into variable-size rectangular and L-shaped regions according to the motion information of a frame. It utilizes a quad-tree to represent the partitioning structure



FSBMC



VSBMC



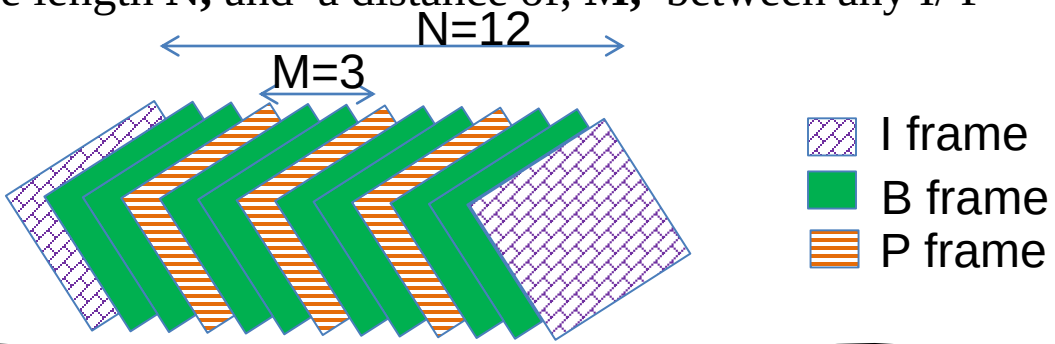
RWMC

Video Coding Standards

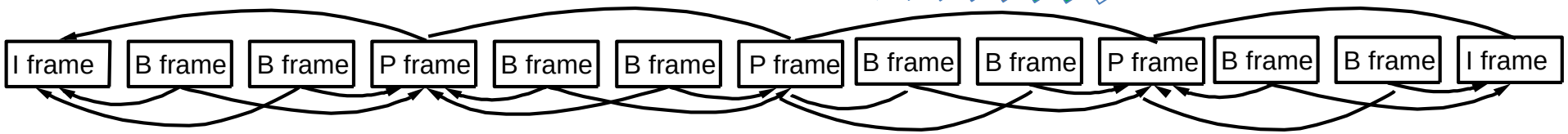
- ✓ ITU-T standards, developed by video coding expert Group (VCEG)
  - ❖ **H.261** (1990)
    - Video compression over ISDN network with bit rates up to 128 kbps and supporting CIF (352x240) and QCIF (176x144) video formats
  - ❖ **H.263** (1995), (H.263+, 1997), (H.263++, 2000)
    - video coding standard targeted for visual telephone over PSTN with bit rates of 18- 24 kbps but with higher quality as compared to H.261 or over the Internet
- ✓ ISO/IEC standards, developed by Moving Pictures Expert Group (MPEG)
  - ❖ **MPEG-1** (1990)
    - AV compression and coding , video compression based on block-based motion estimation, bit rate up to 1.5 Mbps and CIF video format 352x240
  - ❖ **MPEG-2** (1994)
    - the same as MPEG-1 , but bit rate 2-8 Mbps for TV quality and 18-45 Mbps for HDTV and and video format of 704x480 and HDTV
  - ❖ **MPEG-4**：the same as MPEG2 but object based
  - ❖ **MPEG-7**：structures for describing and annotating audio-visual (AV) content
- ✓ ITU-T/ISO/IEC standard, developed by Joint Video Team (JVT){MPEG & VCEG}
  - ❖ **H.264/MPEG-4 AVC** (2003 and today)
    - one of the most commonly used formats for the recording, compression, and distribution of high HD video and It is also widely used by streaming internet sources, such as Youtube

MPEG-1/2 Structure

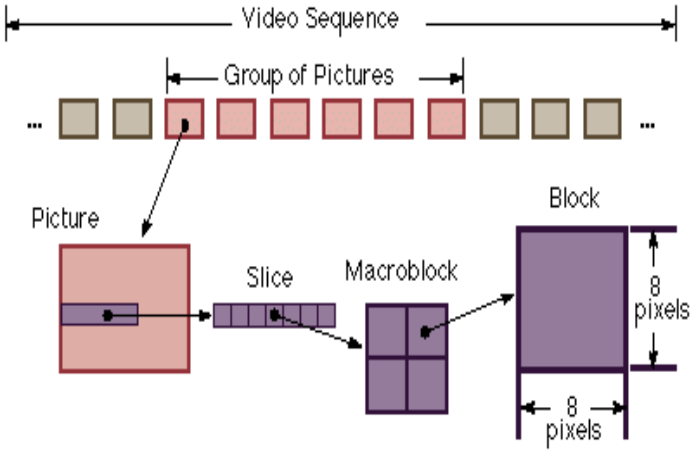
- ✓ In MPEG 1/2 standards the coded video has a structure called GOP(Group of pictures)
  - ❖ GOP is composed of I, P and B frames.
    - I frame : compressed similar to JPEG (intra-frame coded using 8X8 DCT, zigzag scan, Differential coding of DC-coefficients, Uniform quantization, Entropy coding)
    - P Frame: contains motion-compensated information from the preceding I or P frame (residual frame compressed as in I picture)
    - B Frame: Motion-compensated frame from two consecutive P or I pictures (residual frame compressed as in I picture)
      - either only forward prediction, only backward prediction or average of forward and backward prediction
  - ❖ A GOP starts with an I frame and must contain at least one I.
  - ❖ A GOP can defined as having the length **N,** and a distance of, **M,** between any I/ P frame and next P frame.

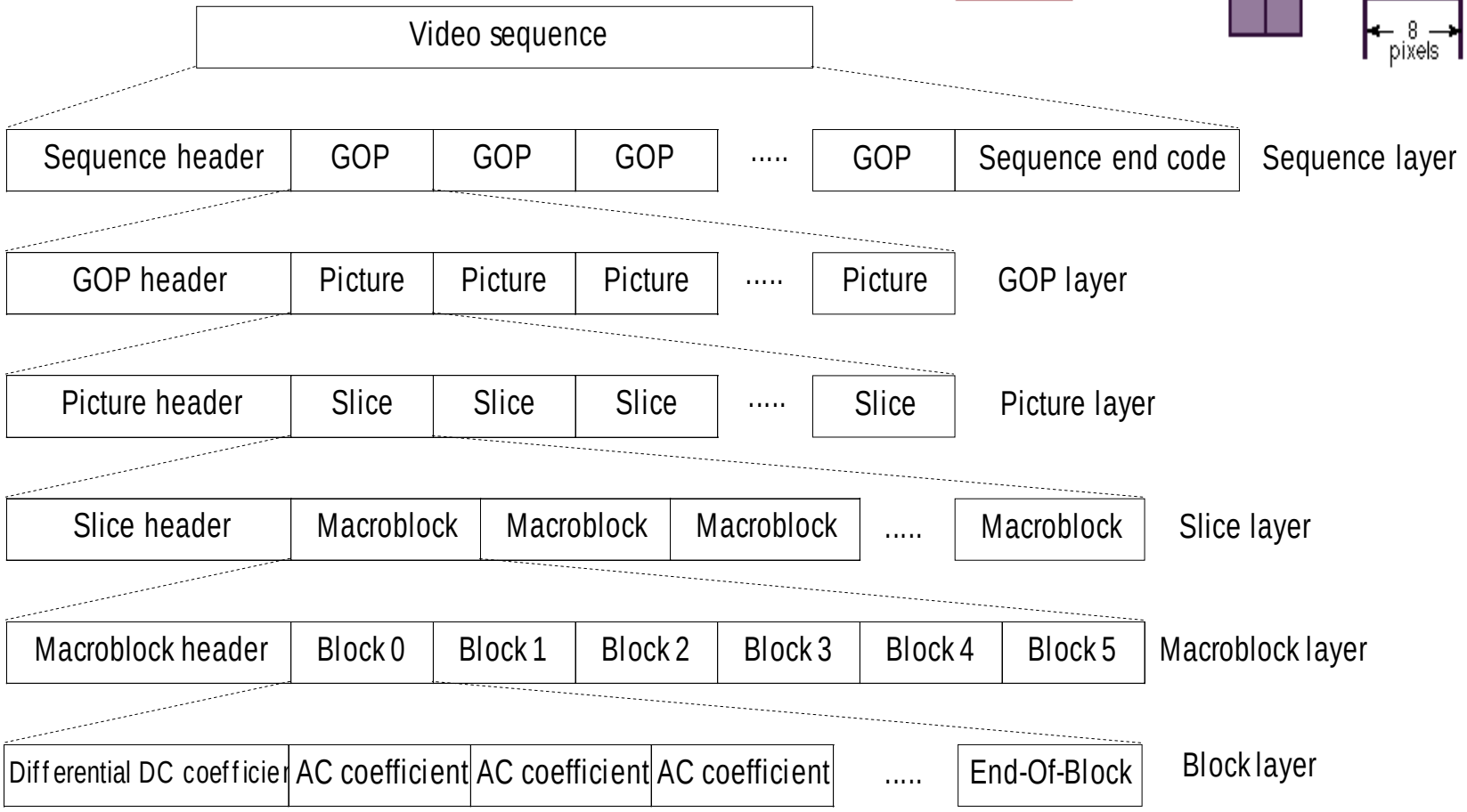


Compressed video stream



MPEG 1 /2 video structure

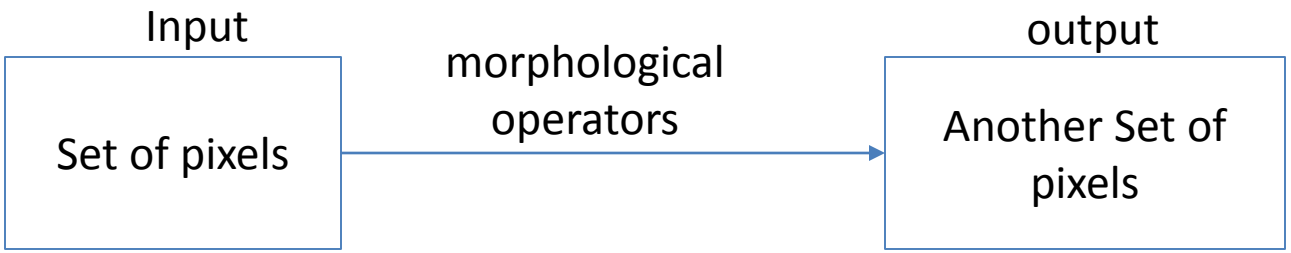




Morphological image processing

Morphological image processing

- ✓ Morpho= shape
  - ❖ Morphological image processing = shape-based image processing
  - ❖ So it is not related to colors, but related to shape
  - ❖ This filed of image processing provides a tool for extracting image components such as boundaries & skeleton, and removing or connecting isolated shapes in the images.
  - ❖ Morphological image processing is usually applies to binary images (which can be generated after thresholding of a grayscale image
  - ❖ Morphological image processing is based on set theory. It introduces some morphological operators that takes a set of pixels and produces another set of pixels.
  - ❖ A set of pixels is a list of (x,y) coordinates of pixels



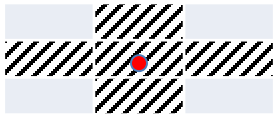
Set Theory operator

- ✓ If B is a set of coordinates, and b $= (b_1, b_2)$ is an element of B then, (b $\in B$)
- ✓ If B is a set with no element , B is called Null or empty set and represented by $\emptyset$
- ✓ If B is a set of elements w , such that w is formed by multiplying each of elements of set C by 2, then we can say $B = \{w | \ w = 2 * c, \ \ c \in C\}$
- ✓ If every element of set B is also an element of set C , then set B is said to be a subset of C and shown as $B \subseteq C$
- ✓ The complement of a set B is a set of elements that are not in B and it is represented as $B^c = \{w \mid w \notin B\}$
- ✓ The union of two sets of B and C is shown as $B \cup C$ and it is a set of elements belonging to either B or C or both.
- ✓ The intersection of two sets of B and C is shown as $B \cap C$ and it is a set of elements belonging to both B and C.
- ✓ The difference of two sets of Band C is shown as B - C and it is a sets that its elements belong to B, but not to C.
- ✓ The translation of set B by a vector z is shown as B_z. If B is a set of coordinates, then $B_{z=} \{(x + z, y + z \,|(x, y) \in B\}$ *or* $B_z = \{c \mid c = b + z, \ b \in B\}$
- ✓ The reflection of set B is shown as $\hat{B}$ *where* , $\hat{B} = \{(-x, -y)|(x, y) \in B\}$

Structuring Elements

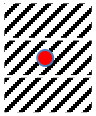
- ✓ The key idea in morphological image processing is defining a structuring element.
- ✓ The structuring element is a small pixel template that we use to apply to the input image using a specific morphological operator to generate an output image of the same size as input image.
- ✓ In Matlab, **strel** function can be used to define various structuring elements.
- ✓ Each structuring element should have an origin
- ✓ Example of structure elements:

0	1	0
1	1	1
0	1	0

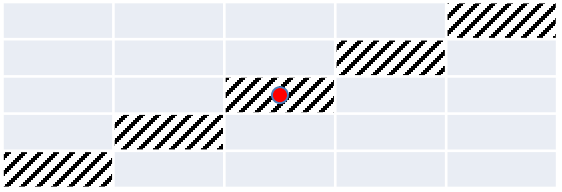


1	1	1	1	1

1
1
1



0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0



Basic Morphological operations

- ✓ If A is an image and B is the structuring element, then the following morphological operators can be expressed as:

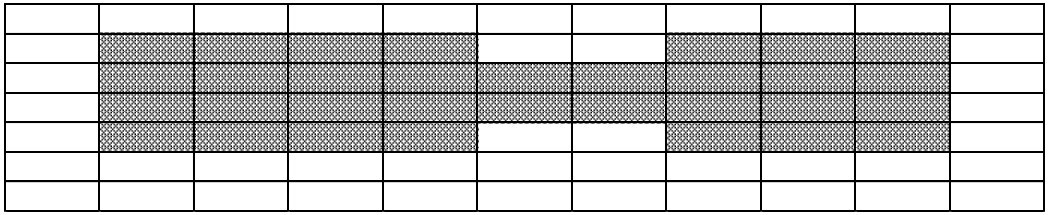
- ✓ Erosion $A \ominus B = \{ z \mid B_z \subseteq A \}$
  - ❖ Set of z, such that the structure element translated by z fits fully inside A
- ✓ Dilation $A \oplus B = \{ z \mid \widehat{B}_z \cap A \subseteq A \}$
  - ❖ Set of z, such that the shifted structuring element has any overlap with A
- ✓ Opening $A \circ B = (A \ominus B) \oplus B$
  - ❖ Erode and then dilate (same structuring element)
- ✓ Closing $A \cdot B = (A \oplus B) \ominus B$
  - ❖ Dilate and then erode

Erosion

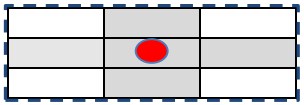
✓ $A \ominus B = \{ z \mid B_z \subseteq A \}$

- ❖ Set of z, such that the structure element translated by z fits fully inside A

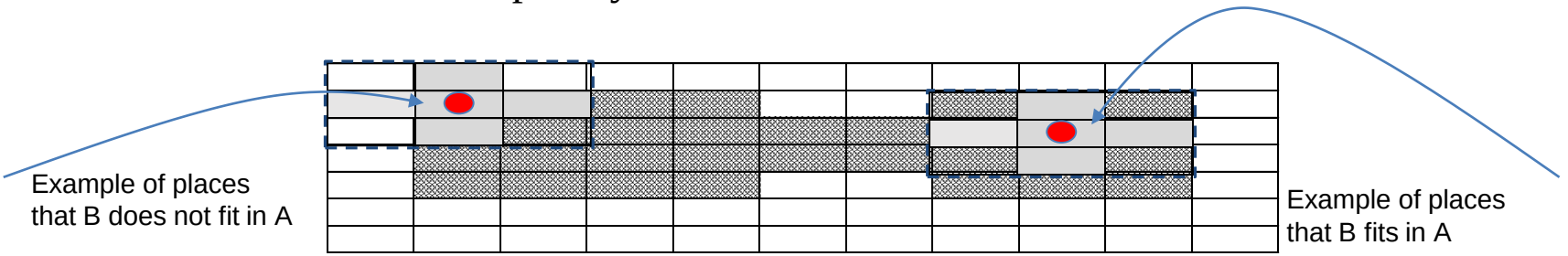
A



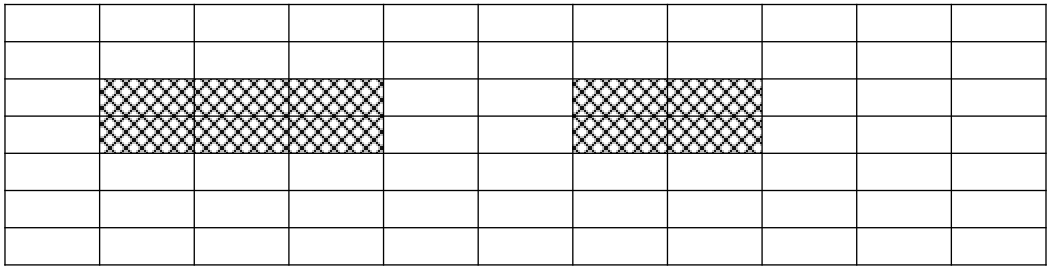
B



To find the erosion of A and B, we should move B over A and label all locations that structure B fits in A completely.



$A \ominus B$



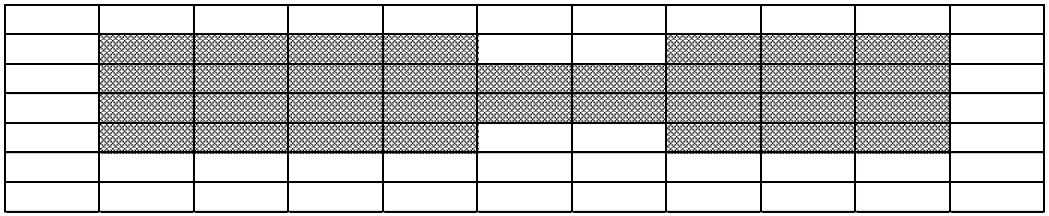
In a white background, the black shapes becomes thinner, breaking objects that are connected using thin black links

Dilation

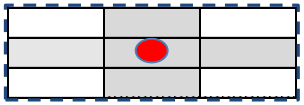
✓ Dilation $A \oplus B = \{z \mid \widehat{B}_z \cap A \subseteq A\}$

❖ Set of z, such that the shifted structuring element has any overlap with A

A

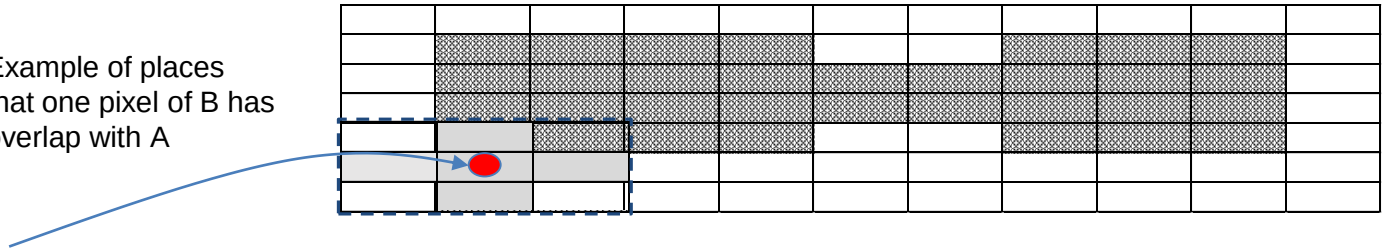


B

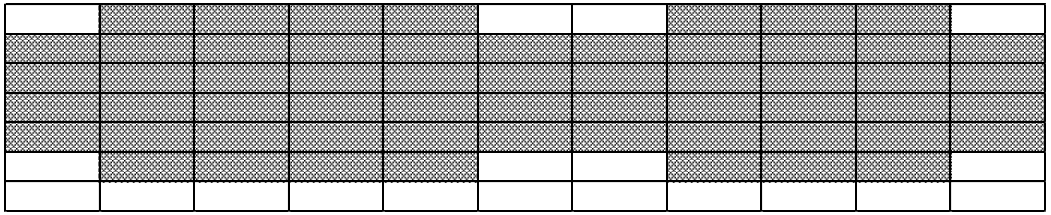


To find the dilation of A and B, we should move B over A and label all locations that structure B has any overlap with A .

Example of places that one pixel of B has overlap with A



$A \oplus B$



In a white background, the black shapes become fatter, connects objects that are seprated