W1 - introduction

① What is CV

❖ Automatic understanding of images and video.

it's not:
① Image Processing
② Patteren Recognition
③ computer Graphics

CV emulate human vision
- Process images and extract information using context
- Make decisions using this information

② Human Visual System

❖ But still is not perfect:
- ✧ Suffers from illusions
  - Shadow illusion (Same color, different brightness)
  - Optical illusion (How many something in an image)
  - Length illusion
- ✧ Ignores many details
- ✧ Ambiguous description of the world

③ CV challenges

① Viewpoint variation
② Background Clutter خلفية
③ illumination الإضاءة
④ occlusion اختفاء، انسداد

⑤ Deformation
⑥ Intraclass variation

④ Brief history (started in 1966)

(5) CV tasks

1. ❖ Reorganization (تمييز - إدراك)
   - ✧ Segment (cluster) pixels with similar properties
2. ❖ Reconstruction — complete the image
   - ✧ Recover 3D information from data
3. ❖ Recognition (تعرف)
   - ✧ Detect and identify objects
4. ❖ Understanding
   - ✧ What is happening in the scene?

Levels of CV tasks:-

1

- ❖ Low Level Vision — pixel level
  - ✧ Measurements — Image Resizing
  - ✧ Enhancements (تحسين) — eg. delete noise
  - ✧ Region segmentation — to reduce the complixity for another Algo
  - ✧ Features — identify
    - edges
    - points
    - sub-images

if pexil is red ⇒ it's person

2

- ❖ Mid Level Vision
  - ✧ Reconstruction — recognize the objects
  - ✧ Depth — eg. Car is in front of the person
  - ✧ Motion Estimation

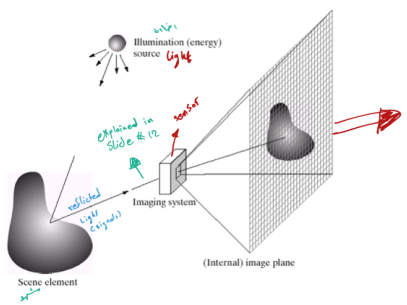
3

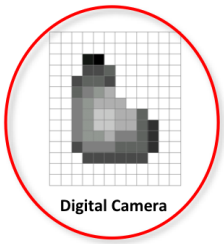
- ❖ High Level Vision
  - ✧ Category detection
  - ✧ Activity recognition — car moving, Person ..., horse ...
  - ✧ Deep understandings
  - ✧ Pose estimation

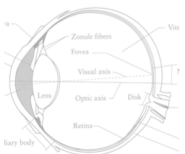
6 CV application

W2- Image Representation (IR)

Image Acquisition (اكتساب) — bring the image to the computer





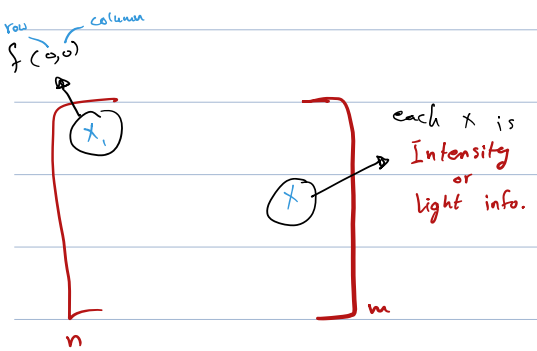


Images are typically generated by *illuminating a scene* and absorbing the energy reflected by the objects in that scene

❖ Image can be represented in two domains:
1. ✧ Spatial domain (مكان)
2. ✧ Frequency domain

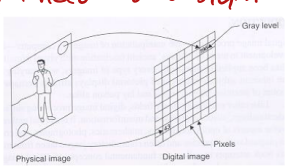
1 IR in the Spatial Domain

❖ A (digital) *image* is a 2D array of intensity values, f(x, y), which represents 2D intensity function discretized (represented) both in spatial coordinates (spatial sampling) and brightness (quantization) values.

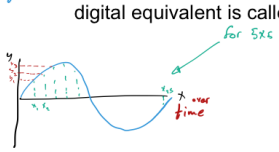


Converting signals into digital form needs two steps:

1. ❖ Sampling: on x axis — # of pixels in the image and their locations
   ✧ Measuring at uniformly spaced locations



2. ❖ Quantization: the values on y axis (How many unique color in the image)
   ✧ The transition between continuous values of the image function and its digital equivalent is called quantization.



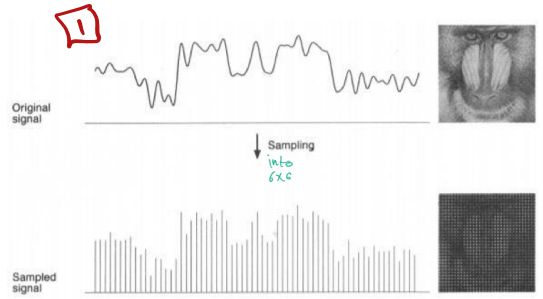
x = # of samples = # of pixels

y = the values of each sample (pixel)

of $x_s \le y_s$

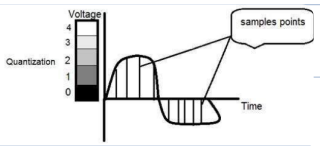
Slide # 12

1



2

No. of Quantization Levels = k = 2^b — * of bits



* Spatial Resolution

- Depends on number of pixels of the image and bit depth

more pixels => more details

space memory to save pixel color

Total bits necessary to represent an image =
No. of rows x No. of cols x Bit depth

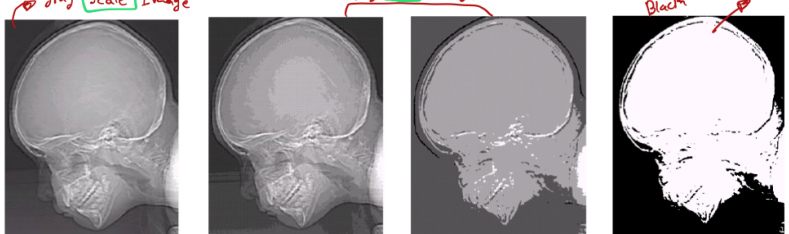
* of pixel

size of pixel to store color value

* Intensity Level Resolution

✧ Intensity level resolution is usually given in terms of the number of bits used to store each intensity level

gray Scale Image — gray Level Image — Black and white image



256 grey levels (8 bits per pixel) — 16 grey levels (4 bits per pixel) — 4 grey levels (2 bits per pixel) — 2 grey levels (1 bit per pixel)

❖ Based on the intensity levels and number of channels, we can categorize images into:

1 bit/pixel ✓ Binary image (0, 1)

8 bit/pixel ✓ Gray-scale (or gray-tone) image (0, 255)

24 bits/pixel ✓ Vector-Valued Images — Color Image

② Image values and Basic Statistics

1. ❖ *Mean* (i.e., the "average gray level") of image *I*:

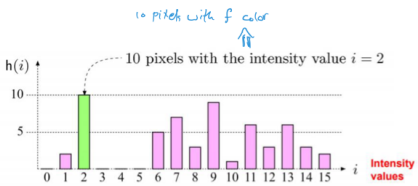
$$\mu_I = \frac{1}{N_{cols} \cdot N_{rows}} \sum_{x=1}^{N_{cols}} \sum_{y=1}^{N_{rows}} I(x, y)$$

❖ Mean is a measure of image *Brightness*

2. ❖ *Variance* of image *I*:

$$\sigma_I^2 = \frac{1}{(N_{cols} \cdot N_{rows} - 1)} \sum_{x=1}^{N_{cols}} \sum_{y=1}^{N_{rows}} [I(x, y) - \mu_I]^2$$

3. ❖ Histogram captures the distribution of gray levels in the image.
   ❖ How frequently each gray level occurs in the image



why we use histograms??

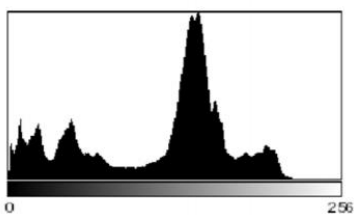
- ❖ Histograms help detect image acquisition issues
- ❖ Problems with image can be identified on histogram
  - ✧ Over and under exposure
  - ✧ Contrast
  - ✧ Dynamic Range
  - ✧ Image defects
  - ✧ Effect of image compression

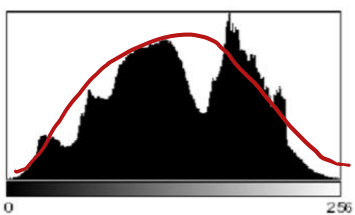
Detecting Bad Exposure using Histograms

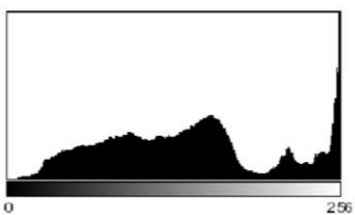
in the color values

- Are intensity values spread (good) out or bunched up (bad)

Image







Histogram

(a) (b) (c)

Underexposed

Properly Exposed

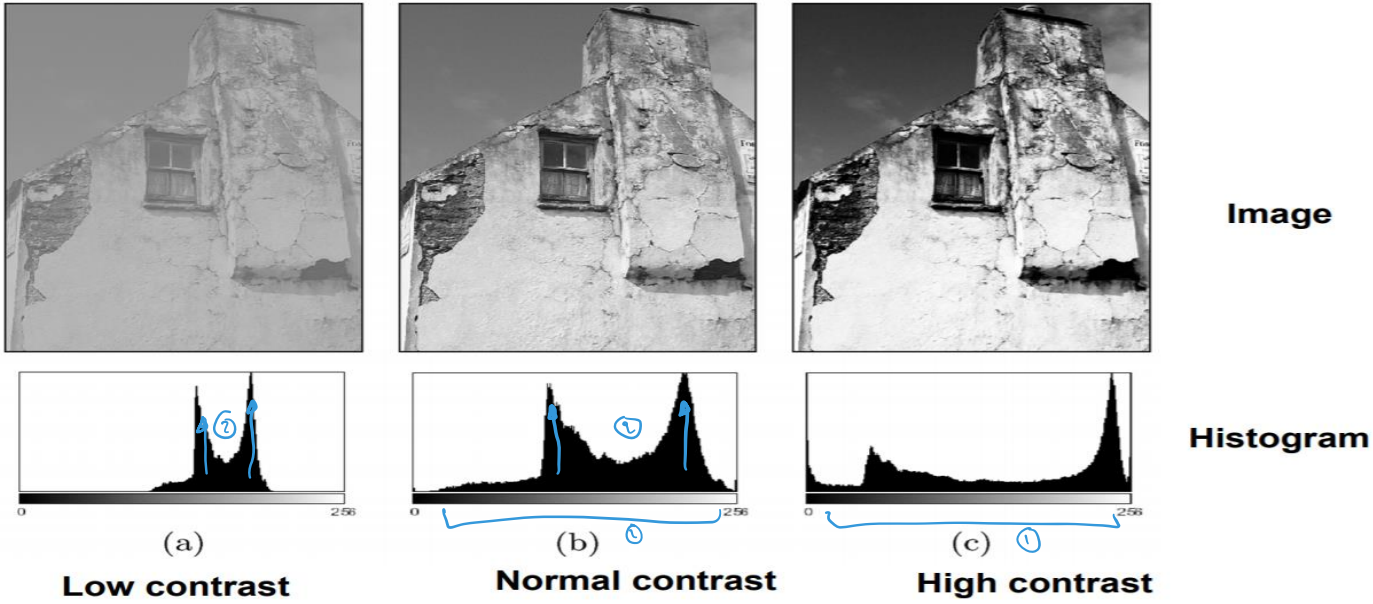
Balanced

nice bell curve

Overexposed

Histograms and Contrast

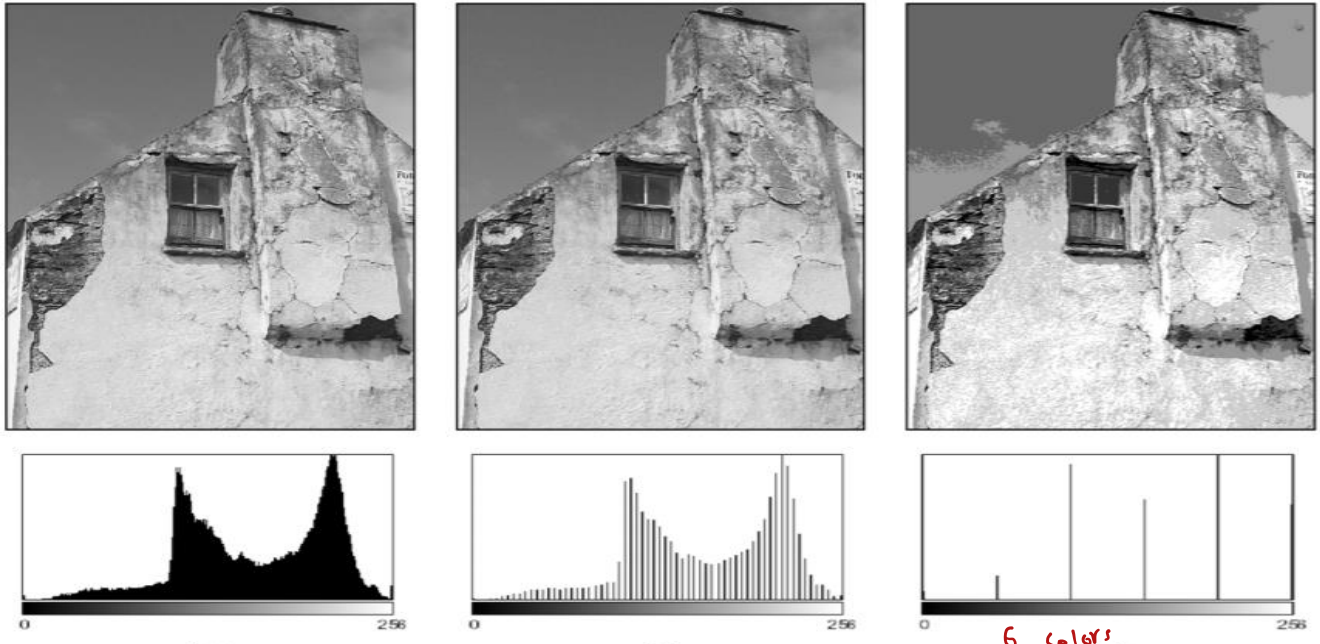
- The contrast of a grayscale image indicates how easily objects in the image can be distinguished
  - Good Contrast: Widely spread intensity values + large difference between min and max intensity values



Histograms and Dynamic Range

How many unique color in image

❖ Dynamic Range: Number of distinct pixels in image



(a) (b) (c)

High Dynamic Range

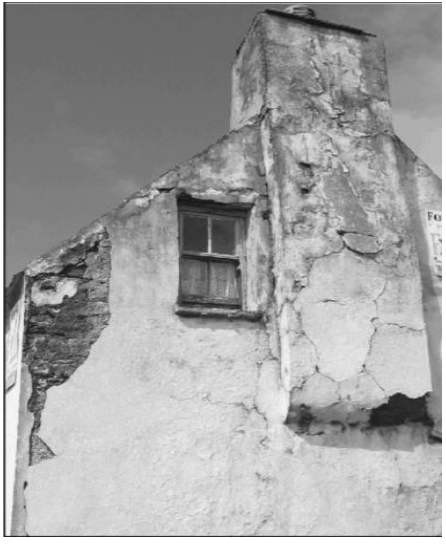
Low Dynamic Range (64 intensities)

Extremely low Dynamic Range (6 intensity values)

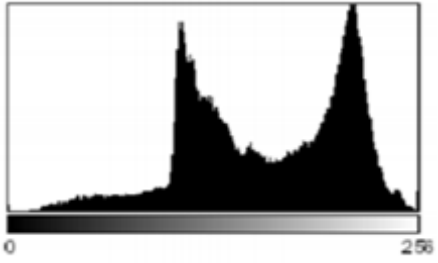
Image Defects: Effect of Image Compression

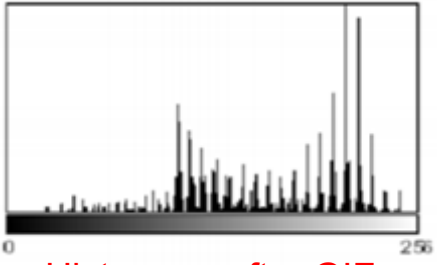
- Example: in GIF compression, dynamic range is reduced to only few intensities (quantization)

Original Image



Original Histogram





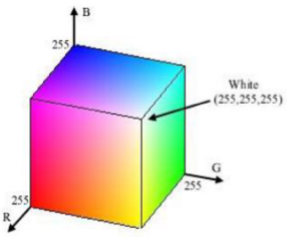
Histogram after GIF

after compression

(3) Color and Color Images

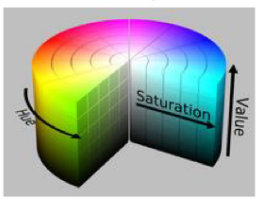
popular models:-

① RGB



- ✧ Can have 256 (R) x 256 (G) x 256 (B) different colors.
- Intensity of a pixel in a RGB space is given by the mean $M=(R+G+B)/3$

② HSV / HSI



center is White

Hue: Which pure color?

Saturation: How white the color is?

Value / Intensity: How Dark the color is? (Brightness)

Hue:

Red = 0° → 60°
Yellow = 61 → 120°
Green
cyan
Blue
Magneta

Saturation:

range [0, 1]

0 = white

1 = primary color

Brightness:

range [0, 100]

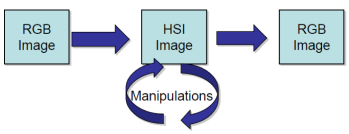
0 = Black

100 = reveals the most color

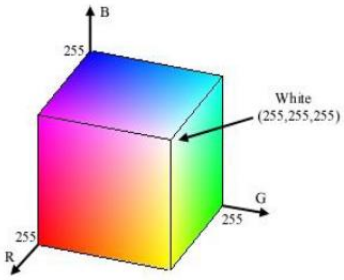
- RGB is great for color generation, but HSV is great for color description

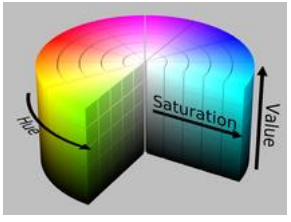
So we use HSV in manipulating

- ❖ In order to manipulate an image under the HSI model we:
  - ✧ First convert it from RGB to HSI
  - ✧ Perform our manipulations under HSI
  - ✧ Finally convert the image back from HSI to RGB



Converting From RGB To HSV/HSI





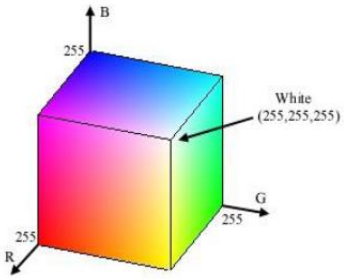
$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & if\ B > G \end{cases}$$

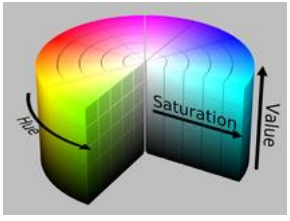
$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right]$$

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

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

Converting From HSI To RGB





– RG sector $(0 <= H < 120^\circ)$

$$R = I\left[1 + \frac{S\cos H}{\cos(60 - H)}\right],\ G = 3I - (R + B),\ B = I(1 - S)$$

– GB sector $(120^\circ <= H < 240^\circ)$

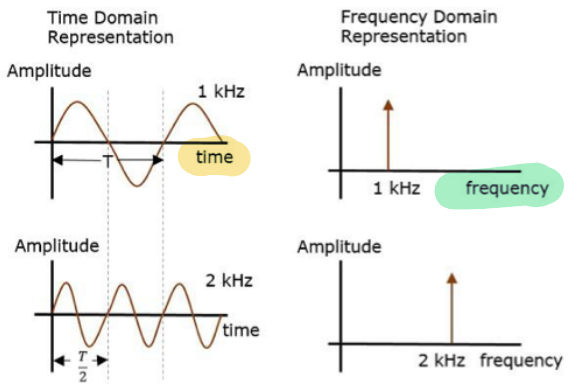
$$R = I(1 - S),\ G = I\left[1 + \frac{S\cos(H - 120)}{\cos(H - 60)}\right],\ B = 3I - (R + G)$$

– BR sector $(240^\circ <= H <= 360^\circ)$

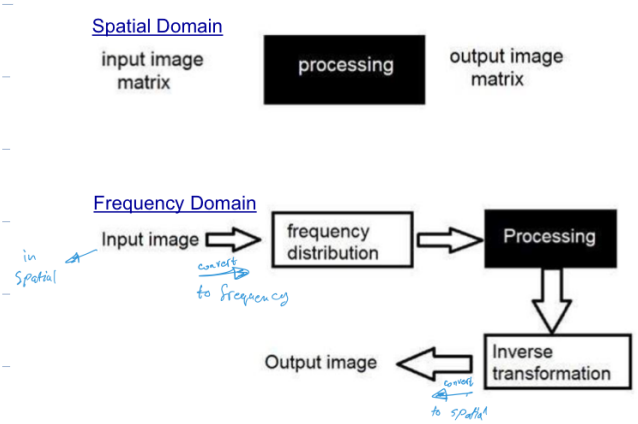
$$R = 3I - (G + B),\ G = I(1 - S),\ B = I\left[1 + \frac{S\cos(H - 240)}{\cos(H - 180)}\right]$$

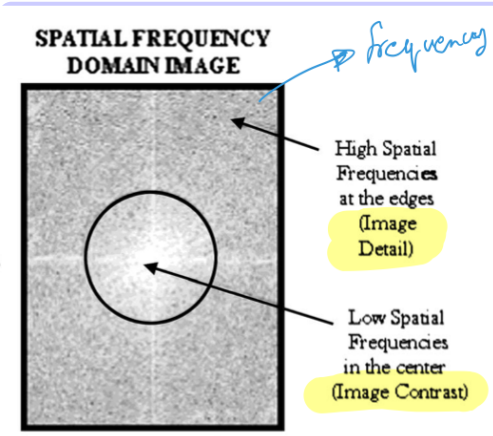
@IR in the Frequency Domain

Spatial



- Whereas in frequency domain, we deal with the rate at which the pixel values are changing in spatial domain.
- The term frequency in an image tells about the rate of change of pixel values. → in spatial domain





❖ *Why we need a domain other than spatial domain ?*

✧ Sometimes image processing/computer vision tasks are best performed in a domain other than the spatial domain.

in time and quality and simplest Algo.

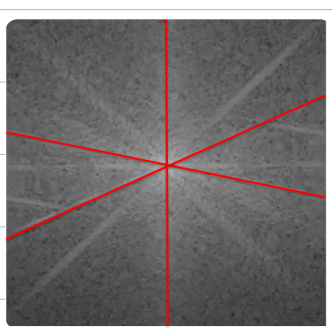
High frequencies correspond to the smooth areas in the image?

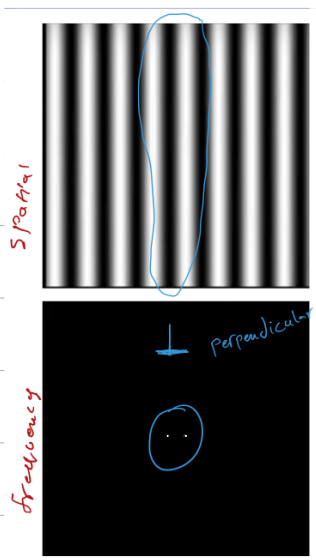
False

Black

⇒ high freq = sharp areas and edges

⇒ low freq = smooth areas





to convert signal from time domain into frequency domain we use transforms

Some Kind of transforms:-

① Fourier Series

❖ Fourier series

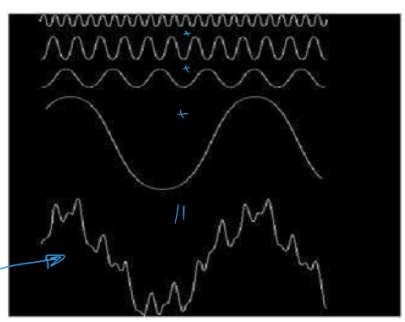
✧ Periodic signals can be represented into sum of Sines and Cosines when multiplied with a certain weight.

❖ Fourier series

✧ The signals are sines and cosines.

✧ The last signal is actually the sum of all the above signals.

✧ This was the idea of the Fourier.



weighted sum of all signals

Sum the amplitude

② Fourier transform

❖ **Fourier transform**: *the non periodic signals whose area under the curve is finite can also be represented into integrals of the sines and cosines after being multiplied by a certain weight.*

✧ Images are non – periodic, so Fourier transform is used to convert them into frequency domain.

Discrete Fourier Transform

❖ The DFT is the sampled Fourier Transform and therefore does not contain all frequencies forming an image, but only a set of samples which is large enough to fully describe the spatial domain image.

❖ 2D Discrete Fourier Transform

spatial

freq.

mean

complex

$$\mathbf{I}(u,v) = \frac{1}{N_{cols} \cdot N_{rows}} \sum_{x=0}^{N_{cols}-1} \sum_{y=0}^{N_{rows}-1} I(x,y) \cdot \exp\left[-i2\pi\left(\frac{xu}{N_{cols}} + \frac{yv}{N_{rows}}\right)\right]$$

Base function

❖ Inverse Discrete Fourier Transform maps Fourier transform **I** back into the spatial domain:

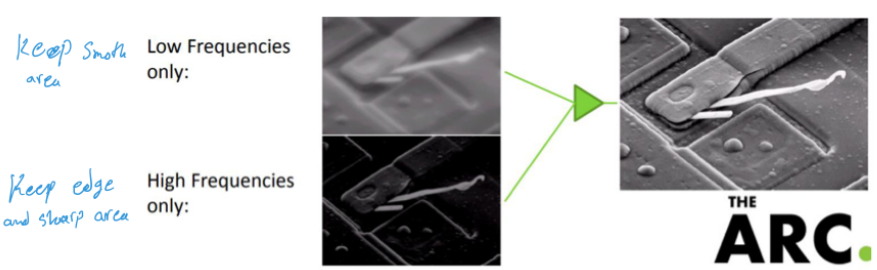
Spatial

freq.

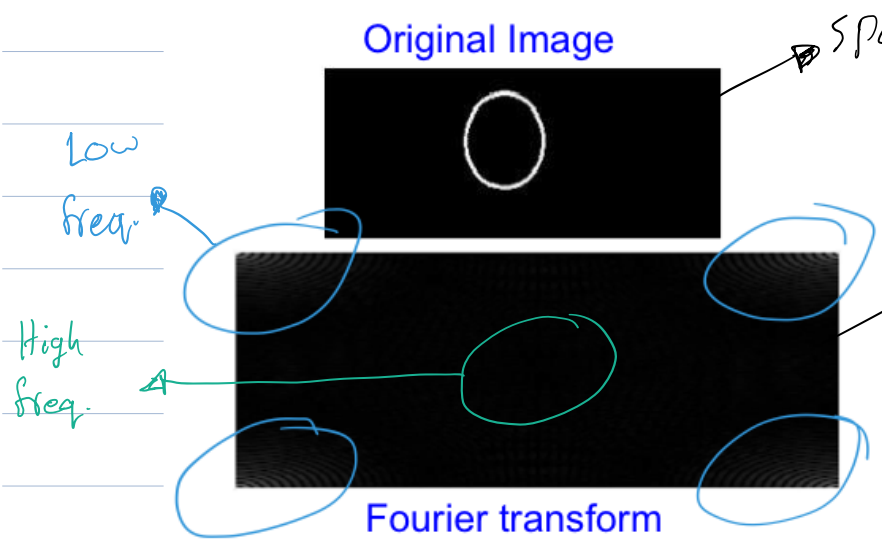
$$I(x,y) = \sum_{u=0}^{N_{cols}-1} \sum_{v=0}^{N_{rows}-1} \mathbf{I}(u,v) \exp\left[i2\pi\left(\frac{xu}{N_{cols}} + \frac{yv}{N_{rows}}\right)\right]$$

* notes about DFT:-

- F(0,0) is called dc component
- the values of DFT are complex
- to visually analyze a DFT, we compute Fourier spectrum (the magnitude of F(u,v))



❖ Origin in image *I* and Fourier transform ***I*** is in upper left corner



Spatial

Low freq.

High freq.

freq.

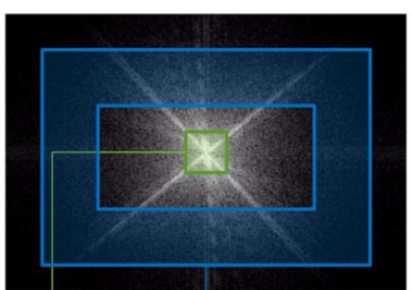
to get ⇒ Low freq. in the origin

$$I(x,y)\cdot(-1)^{x+y}$$

Spatial

⇓

then we convert it into freq.



✧ Allows to plot ordered pairs of the form (a,b), where a and b are real numbers

$$z = a + ib$$

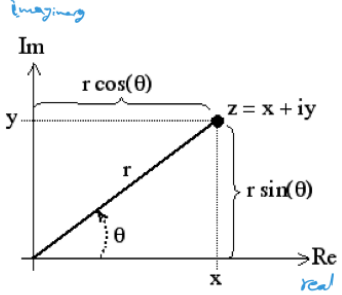
- Also called "Rectangular Coordinates"

❖ Polar Form: Rather than using **a** and **b**, the polar coordinates use **r** and **θ** in their ordered pairs.

for real — for imaginary

✧ The general form for polar numbers is as follows: $re^{i\theta}$

imaginary



$$r = \sqrt{a^2 + b^2}$$

$$\theta = \arctan\left(\frac{b}{a}\right)$$

Discrete Fourier transform (TF = fft2())

❖ **Magnitude** (spectrum) of the FT

abs(TF)

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

❖ **Power spectrum** (spectral density)

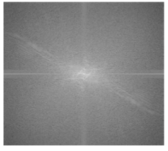
$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v)$$

❖ **Phase angle** (phase spectrum)

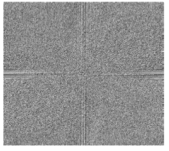
angle(TF)

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

Original



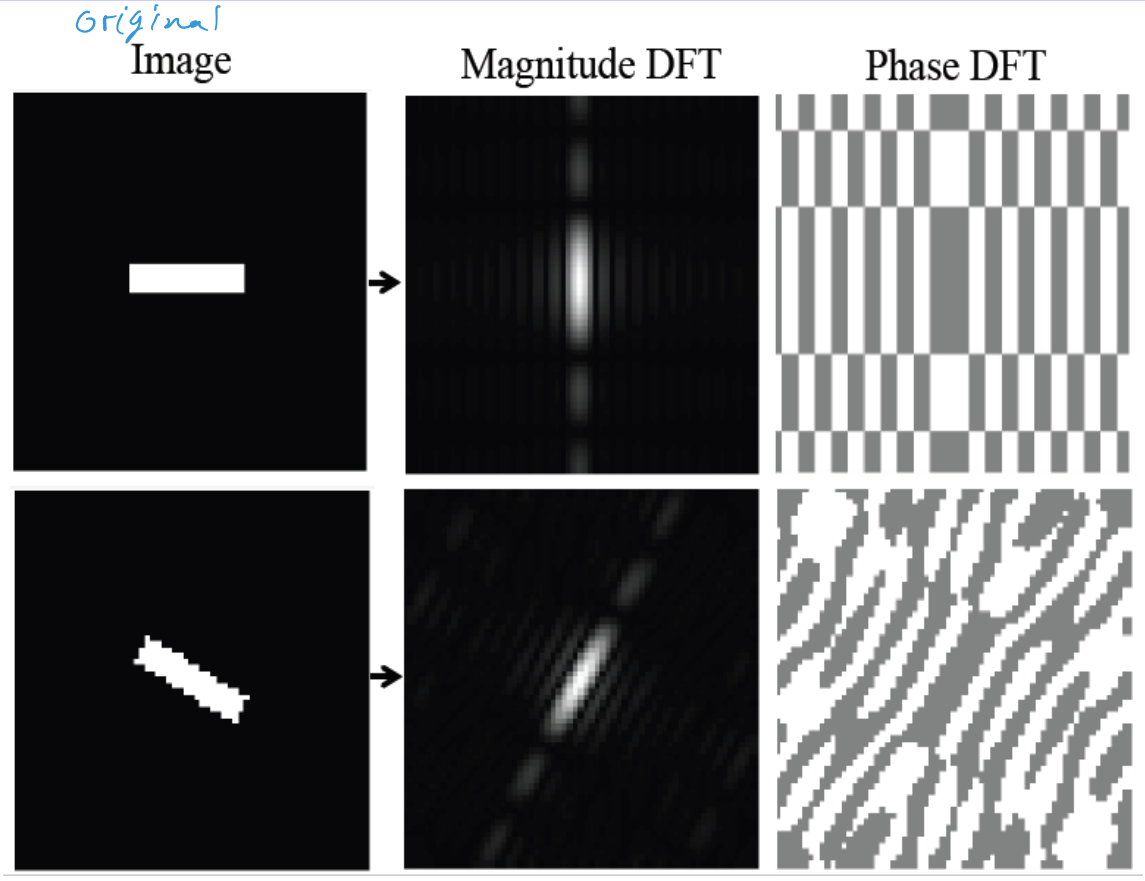
Amplitude



Phase

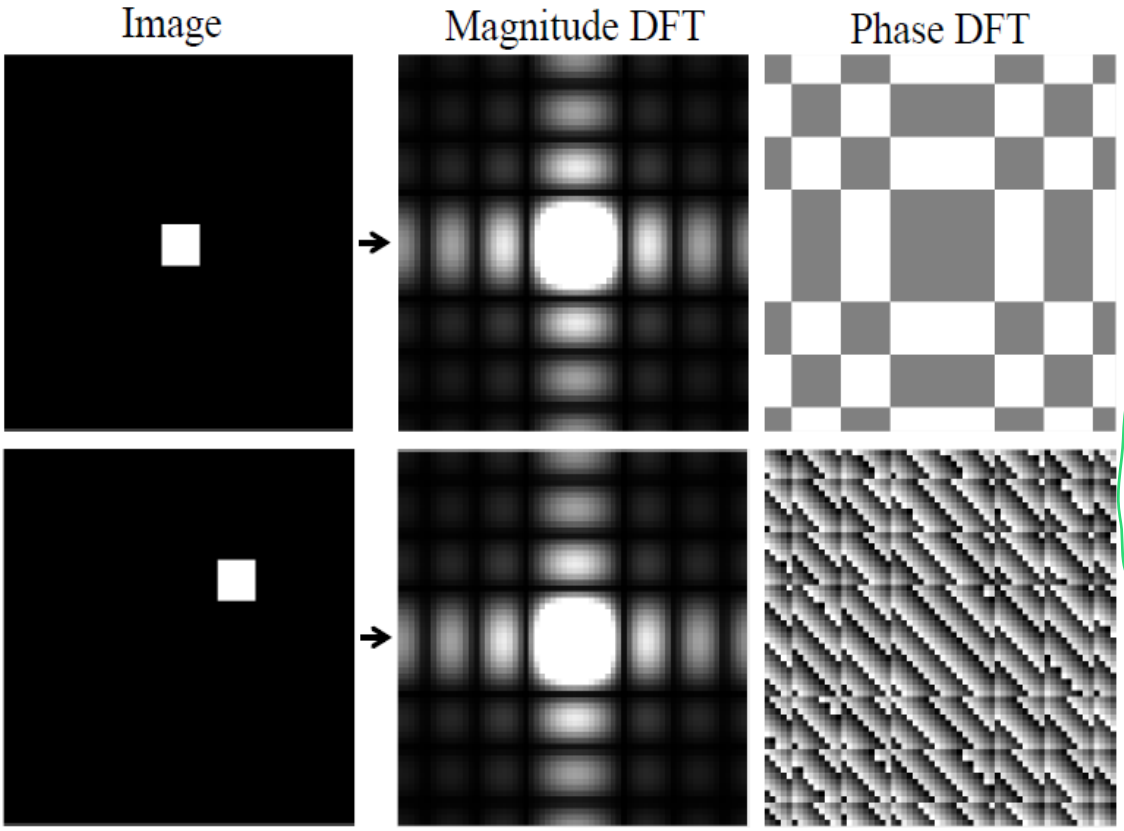
Note: R(u, v) and I(u, v) are real and imaginary parts of F(u, v), respectively.

Fourier Transform Examples



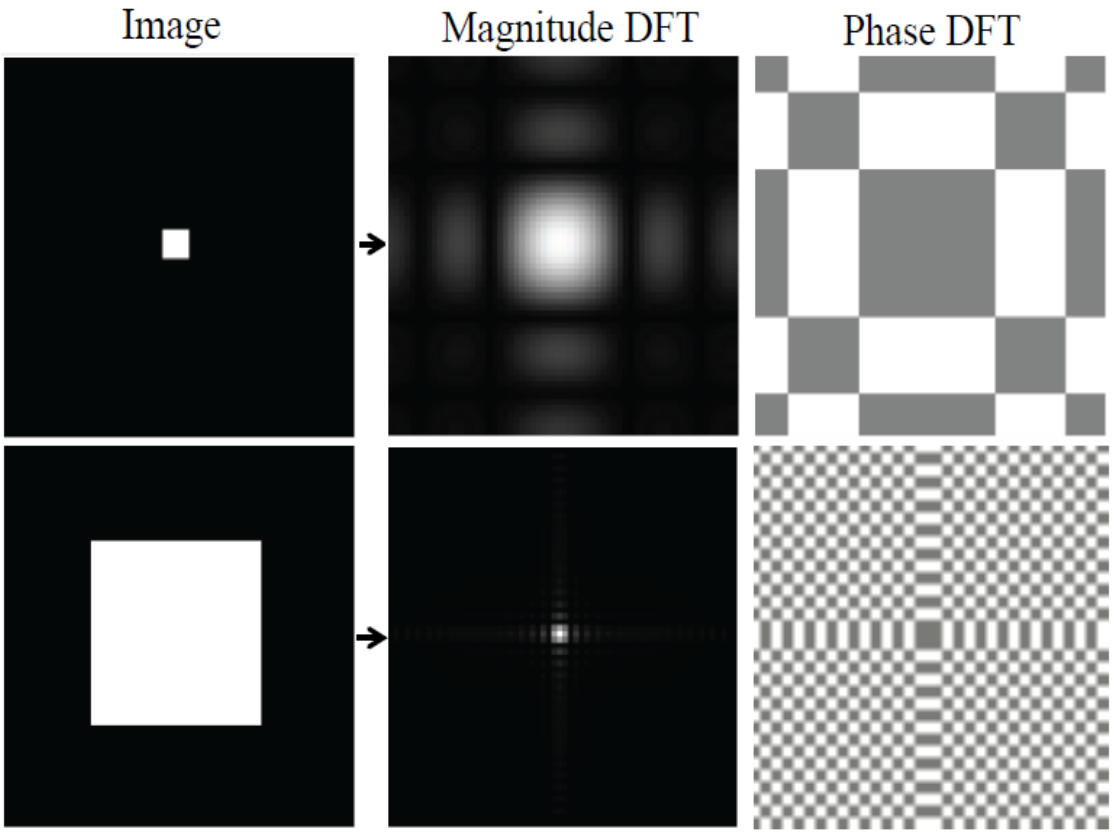
Orientation
A line transforms to a line oriented perpendicularly ⊥ to the first

Fourier Transform Examples



Translation
Shifts of an image only produce changes on the phase of the DFT

Fourier Transform Examples



Scale
Small image details produce content in high spatial frequencies

⑤ Basic Image Operators

① Thresholding

what?

→ Colored IMg → binary IMg

Does not work well with natural scenes.

why?

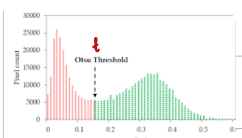
- document analysis (eg. read digits)
- Industrial inspection
- Medical imaging

How

→ histogram

Otsu's Method → assumes the histogram is bimodal and tries to minimize the variation within the same group of values

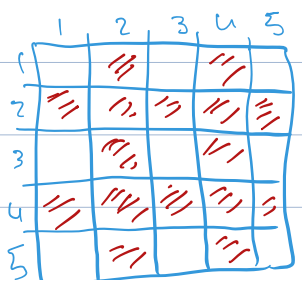
Method: find the threshold t that minimizes the weighted sum of within-group variances for the two groups that result from separating the gray tones at value t.



In practice, this operator works very well for true bimodal distributions and not too badly for others.

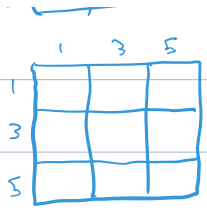
② Resampling (Resize the image)

① reduce the size = sub-Sampling



delete even rows and columns

Throw away every other row and column to create a ½ size image



(2) increase the size = up-sampling

(1) ❖ Simplest approach:

repeat each row and column 10 times

(2) ❖ More efficient approach:

Interpolation (estimate a new value)

❖ We can do this by looking at the values nearby. Methods include:

[1] ✧ Nearest neighbor – just take the value of the closest neighbor

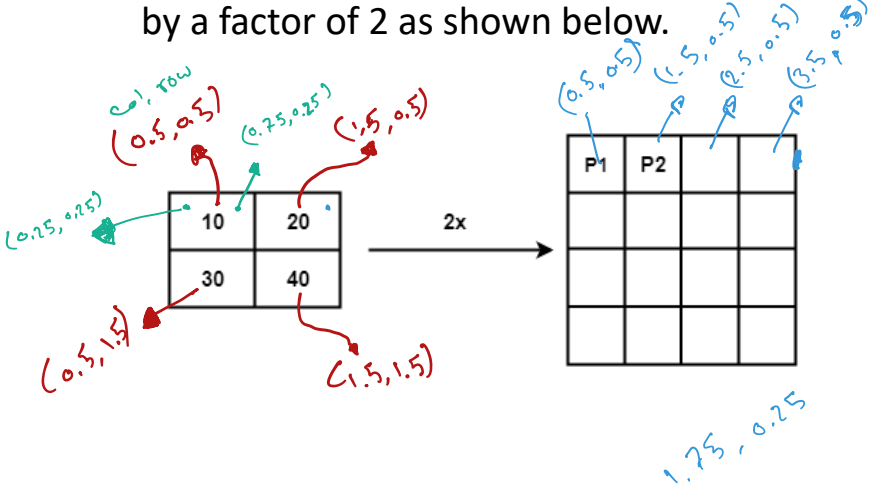
[4] ✧ Bilinear – take a combination of the four closest neighbors

[16] ✧ Bicubic – use the closest 16 neighbors (most computationally expensive, but best results)

Review: Nearest Neighbor Interpolation

Estimation value from one point to one point

- We assign the unknown pixel to the nearest known pixel.
  - Example: Suppose, we have a 2×2 image and let's say we want to upscale this by a factor of 2 as shown below.



$\frac{2}{4}$ → original size
→ interpolated size

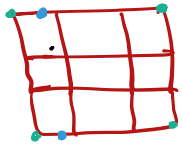
$P_1 = (0.5, 0.5) \times \frac{1}{2} = (0.25, 0.25)$

From original the value = 10

$P_2 = (1.5, 0.5) \times \frac{1}{2} = (0.75, 0.25)$

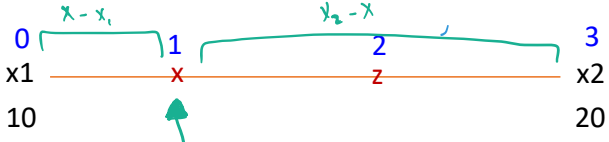
From original the value = 10

apply twice (vertically, horizontally)



Review: Bi-Linear interpolation

- Let us first review the linear interpolation where more weight is given to the nearest value
  - Assume we want to find the pixel value at locations 1and 2
  - Let us call these points: x and z



for vertical replace x by y

$$f(x) = \frac{x2-x}{x2-x1} \times f(x1) + \frac{x-x1}{x2-x1} \times f(x2)$$
$$= \frac{1}{3} \times 20 + \frac{2}{3} \times 10 = 13.3$$

$$f(z) = \frac{x_2 - z}{x_2 - x_1} \times f(x_1) + \frac{z - x_1}{x_2 - x_1} \times f(x_2)$$

W3-P1 - Image filtering

① Image enhancment

- make it more suitable for specific application
- depends on application
- methods may different for each image

- Common reasons for enhancement include
  - Improving visual quality,
  - Improving recognition accuracy.

- it can be done in:

① Spatial domain
② Frequency domain بعدين بنيجي
③ combinations of method from both domains

* by Spatial domain: (Point or Kernal operators) بعدين بنيجي

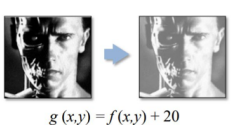
* Point operations:

deals with pixel intensity values individually

Intensity Transformation function:-

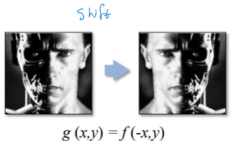
general → * Gray-level $S = T(r)$ ← original pixel

- Linear, Logarithmic: s = c log(r + 1), Power-Law: S = C r^{γ}

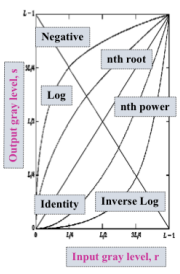


$g(x,y) = f(x,y) + 20$

shift



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



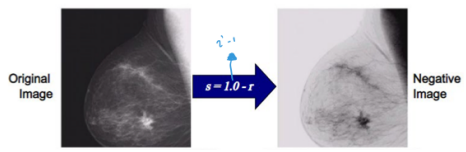
① Image negative

of bits
in gray L=256 <= 2^{bits} <= maximum Intensity value

Image negative is produced by subtracting each pixel from the maximum Intensity value

$$s = L - 1 - r$$ (r: original)

- Suitable for enhancing white or gray detail embedded in dark regions of an image.



- expand the intensities of dark pixel
- compress the bright (higher-level) intensities

② Log Transformations

dark pixel → darker
wight pixel → wighter

Constant, original pixel

$$s = c \log(r + 1)$$

③ Power-Law (Gamma) Transformations

Light

Constant is used to correct image's luminance.

$$s=cr^{\gamma} \text{ or } s=c(r+\varepsilon)^{\gamma}$$

$\gamma < 1$ increase Brightness

$\gamma > 1$ increase Darkness

④ Piecewise-Linear Transformation

Commonly types:

① Contrast Stretching (تباين)

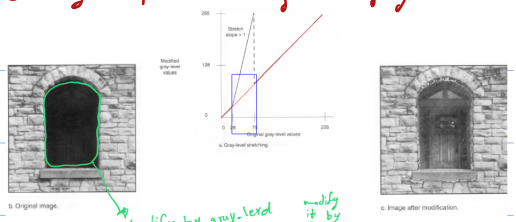
Contrast stretching expands the range of intensity levels in an image.

Contrast stretching is done in three ways:

1. Multiplying each input pixel intensity value with a constant scalar.
   - Example: s=2*r
2. Using Histogram Equivalent
3. Applying a transform which makes dark portion darker by assigning slope of < 1 and bright portion brighter by assigning slope of > 1.

② Gray-level Slicing

identify a specific range of gray levels in an image



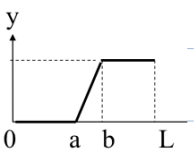
identify by gray-level slicing → modify it by another technique

③ Clipping (set two thresholds)

reassign some pixels

$$y = \begin{cases} 0 & 0 \le x < a \\ \beta(x-a) & a \le x < b \\ \beta(b-a) & b \le x < L \end{cases}$$

you can replace them



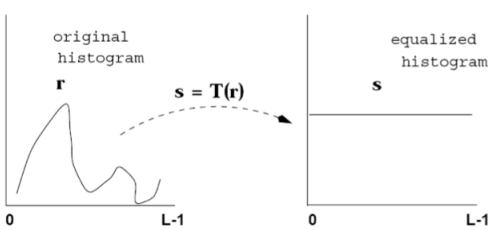
② Histogram processing

usually if the histogram of an image is uniformally distributed

it will have a high contrast ⇒ more details

- The main idea is to redistribute the gray-level values uniformly.

to use all colors



to achive that

⇓

Histogram equalization example

- Histogram equalization algorithm

0	0	1	0	2	0
1	0	7	7	7	0
0	7	0	0	7	0
1	0	0	7	2	0
0	0	7	1	0	1
1	0	7	7	7	0

frequencies

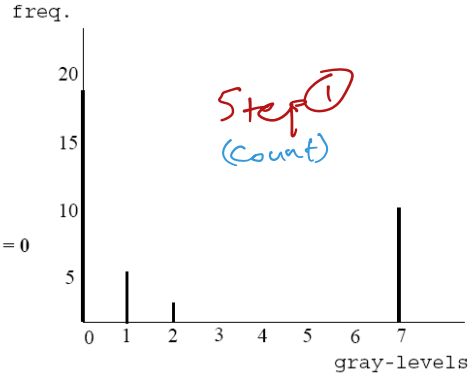
f(0)= 18

f(1)= 6

f(2)= 2

f(3)=f(4)=f(5)=f(6) = 0

f(7) = 10



- Divide frequencies by total number of pixels to represent as probabilities.

$$p_k = n_k / N$$

Step 2 (normalize)

$$P(0) = \frac{f(0)}{36} = \frac{1}{2} \qquad P(1) = \frac{f(1)}{36} = \frac{1}{6}$$

$$P(2) = \frac{f(2)}{36} = \frac{1}{18} \qquad P(3) = P(4) = P(5) = P(6) = 0$$

$$P(7) = \frac{f(7)}{36} = \frac{5}{18}$$

never used

Histogram equalization example

step 3

$$T(r_k) = \sum_{j=0}^{k} p(r_j)$$

$$P(0) = \frac{f(0)}{36} = \frac{1}{2} \qquad P(1) = \frac{f(1)}{36} = \frac{1}{6}$$

$$P(2) = \frac{f(2)}{36} = \frac{1}{18} \qquad P(3) = P(4) = P(5) = P(6) = 0$$

$$P(7) = \frac{f(7)}{36} = \frac{5}{18}$$

- Compute intensity transformation function

since they =0

$T(r_0) = \sum_{j=0}^{0} p_r(r_k) = p_r(r_0) = \frac{1}{2} = 0.5$

$T(r_1) = \sum_{j=0}^{1} p_r(r_k) = \frac{1}{2} + \frac{1}{6} = 0.5 + 0.17 = 0.66$

$T(r_2) = \sum_{j=0}^{2} p_r(r_k) = \frac{1}{2} + \frac{1}{6} + \frac{1}{18} = 0.5 + 0.17 + 0.06 = 0.72$

$T(r_3) = T(r_4) = T(r_5) = T(r_6) = 0.72$

$T(r_7) = \sum_{j=0}^{7} p_r(r_k) = 0.72 + \frac{5}{18} = 1.01$

$T(2) = P(0) + P(1) + P(2)$

$= \frac{1}{2} + \frac{1}{6} + \frac{1}{18}$

$\Rightarrow 0.72$

step 4

- Multiple intensity transformation function by intensity level – 1

8-1=7

step 5

$S_0 = (L-1) * T(r_0) = 7*0.5 = \lfloor 3.5 \rfloor = 3$

$S_1 = (L-1) * T(r_1) = 7*0.66 = \lfloor 4.6 \rfloor = 4$

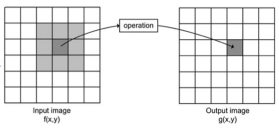
$S_2 = (L-1) * T(r_2) = 7*0.72 = \lfloor 5.04 \rfloor = 5$

$S_3 = S_4 = S_5 = S_6 = 7*0.72 = \lfloor 5.04 \rfloor = 5$

$S_7 = (L-1) * T(r_7) = 7*1.01 = \lfloor 7.01 \rfloor = 7$

* Kernal / filter / Mask operators.

template / window



- in spatial domain
- it consider the value of neighboors

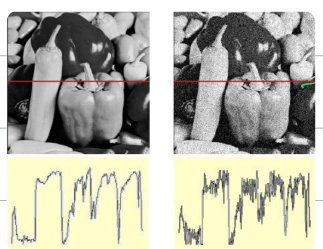
What Point Operations Can't Do

- Combining multiple pixels needed for certain operations:
  - Enhance an image, e.g., denoise, Blurring, Sharpening.
  - Extract information, e.g., texture, edges.
  - Detect patterns, e.g., template matching.

noise:

- Noise sources
  - Light Variations
  - Camera Electronics
  - Surface Reflectance
  - Lens

→ Common types:-



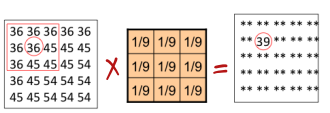
① Salt and pepper (random)

② Gaussian noise

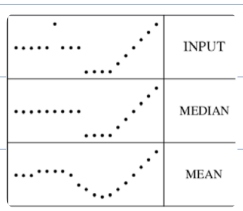
→ How to denoise ?!

by filters

- The values in a filter sub-image are referred to as coefficients, rather than pixels.
  - A mask of coefficients is centered on a pixel.
  - The mask coefficients are multiplied by the pixel values in its neighborhood and the products are summed.
  - The result goes into the corresponding pixel position in the output image.



Input Image filter Output Image



- Some simple neighborhood operations include:
  - Min: Set the pixel value to the minimum in the neighborhood
  - Max: Set the pixel value to the maximum in the neighborhood
  - Average: Set the pixel value to the average of the neighborhoods
  - Median: The median value of a set of numbers is the midpoint value in that set (e.g. from the set [1, 7, 15, 18, 24] 15 is the median). Sometimes the median works better than the average

- Image filtering is used to:
  - Remove noise
  - Sharpen contrast
  - Highlight contours
  - Detect edges
  - Other uses?

Filters
- linear — convolution
- non-linear — (eg. median filter)

W3-P2 - Image filtering

Some properties of linear filters:-

Key properties of linear filters

1. Linearity: filter(f1 + f2) = filter(f1) + filter(f2)
2. Shift invariance: same behavior regardless of pixel location.
   - The value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood

   filter(shift(f)) = shift(filter(f))
   - Any linear, shift-invariant operator can be represented as a convolution
3. Commutative: a * b = b * a

Key properties of linear filters

4. Associative: a * (b * c) = (a * b) * c
   - Often apply several filters one after another: (((a * b1) * b2) * b3)
   - This is equivalent to applying one filter: a * (b1 * b2 * b3)
5. Distributes over addition: a * (b + c) = (a * b) + (a * c)
6. Scalars factor out: ka * b = a * kb = k (a * b)
7. Identity: unit impulse e = [0, 0, 1, 0, 0], a * e = a
8. Differentiation: $\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$

Some types of linear filters:-

1. Average

for Smoothing

increasing Kernal size = more blur

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Average filter

Special type:

Weighted Smoothing Filter

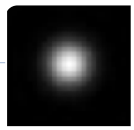
- More effective smoothing filters can be generated by assigning different pixels in the neighborhood different weights in the averaging function
  - Pixels closer to the central pixel are more important
  - Reduce smoothing effect
  - Often referred to as a weighted averaging

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

② Gaussian

the largest value in the center



0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

for Smoothing

σ = Scale / width / spread of the distribution

σ ↗ ⇒ Smoother = more blurer

σ ↙ ⇒ sharper

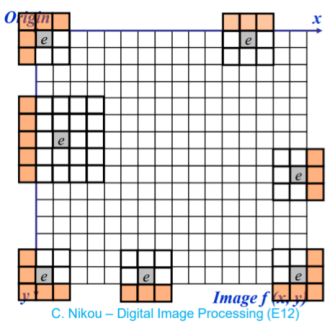
* Median filter (non-linear)

Selecting the median intensity in the kernal

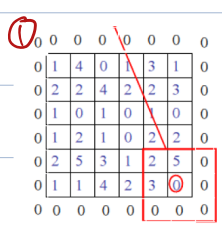
good for impules and salt & pepper noise

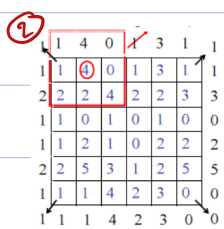
Boundary issues

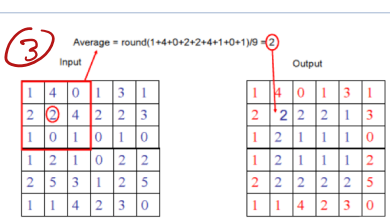
- There are a few approaches to deal with missing edge pixels:
  1. Pad the image (padding)
     - Typically with either all white or all black pixels
  2. Replicate border pixels
  3. Truncate the image Keep the values of border
  - Allow pixels wrap around the image
    - Can cause some strange image artefacts



C. Nikou – Digital Image Processing (E12)







W4-P1-Edge detection

- Edges are sharp change in brightness (discontinuities).
- Goal: Identify sudden changes in an image

Characteristics of an Edge

- Edge: A sharp change in brightness
- Edge models/types:
  - Step
  - Ramp
  - Roof

in Real world

Ideal

Step Ramp Roof

a b c

FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles

Edge detection

- Basic idea: look for a neighborhood with strong signs of change.

81	82	26	24
82	33	25	25
81	82	26	24

- Problems:
  - Neighborhood size
  - How to detect change
- Differential operators:
  - Attempt to approximate the gradient at a pixel via masks.
  - Threshold the gradient to select the edge pixels.

- Main Steps in Edge Detection

in W3-P2

W3-P2

Linear filter (Last slides)

1) **Smoothing:** suppress as much noise as possible, without destroying true edges.

take dirivitive

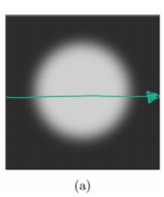
2) **Enhancement:** apply differentiation to enhance the quality of edges (i.e., sharpening)
3) **Thresholding:** determine which edge pixels should be discarded as noise and which should be retained (i.e., threshold edge magnitude).
4) **Localization:** determine the exact edge location. you can use erosion to connect them

Edge detection using derivatives: (two ways)

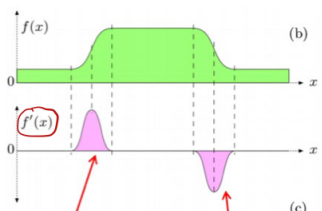
(1)

1st order derivative:

scanning from L to R



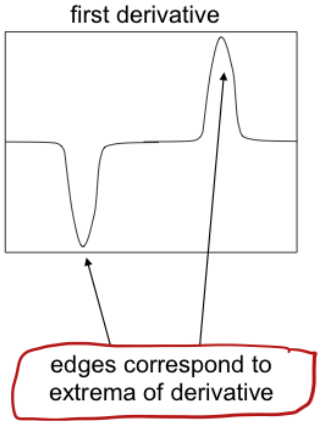
(a)



tells us the location of edges

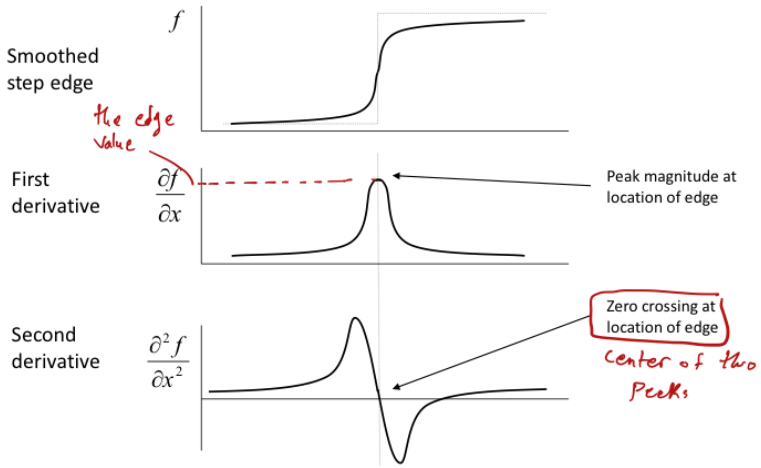
Rising slope causes positive + high value first derivative

Falling slope causes negative + high value first derivative

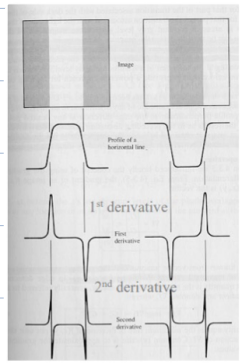


(2)

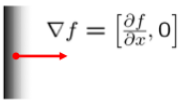
- 2nd order derivatives enhance fine detail much better



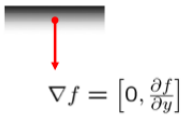
- Often, points that lie on an edge are detected by:
  1. Detecting the local maxima or minima of the first derivative.
  2. Detecting the zero-crossings of the second derivative.



Since images are 2D, we use gradient



no change on y direction



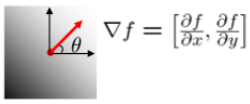


Image gradient

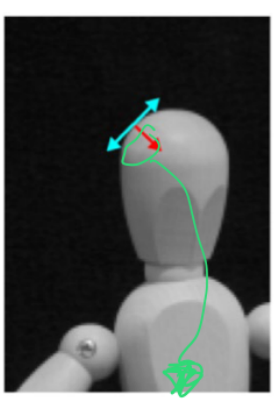
always

- Gradient is perpendicular to edge contour
- The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- The edge strength is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



always to the Right side

for 1D

1D discrete derivate filters

- Backward filter: H = [0 1 -1]

next current Prev

current Prev

$$f(x) - f(x-1) = f'(x)$$

- Forward: H = [-1 1 0]

current next

$$f(x) - f(x+1) = f'(x)$$

- Central: H = [1 0 -1]

next Prev

$$f(x+1) - f(x-1) = f'(x)$$

Note: Derivative kernels sum to zero

$\Sigma H = 0$

2D filter

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \xrightarrow{\cdot T} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

in x direction

in y direction

Effect of noise:

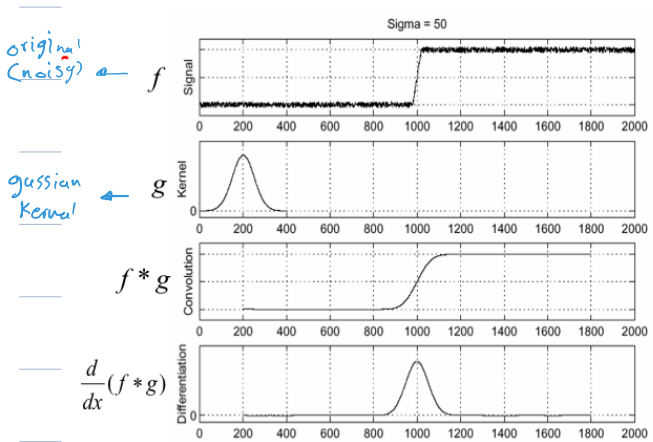
derivative is very sensitive for noise (since it's measure the rate of change)

Solution:

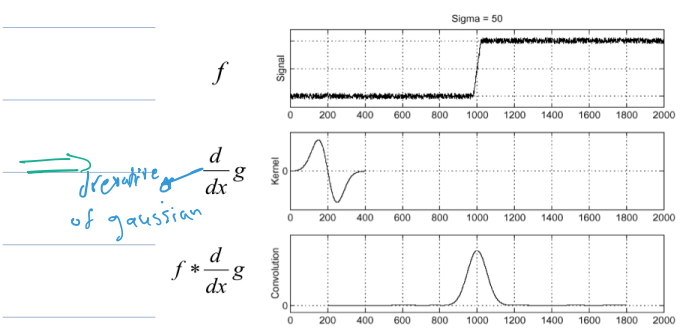
Smoothing first

small σ → sharper

large σ → smoother (detect large scale edge)



derivative of gaussian

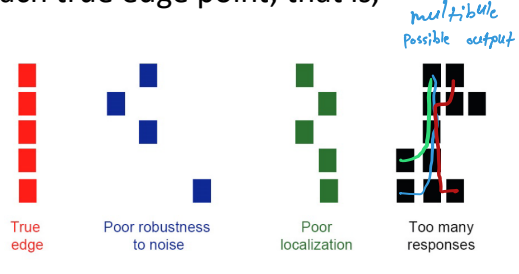


Smoothed derivatives removes noise and finds edges at different "scales"

But it blurs edges

Designing an edge detector

- Criteria for an “optimal” edge detector:
  1. **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  2. **Good localization**: the edges detected must be as close as possible to the true edges
  3. **Single response**: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Sobel Operator:

- Smoothing + differentiation

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [+1 \quad 0 \quad -1]$$

Gaussian smoothing — differentiation

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Problems in Sobel:

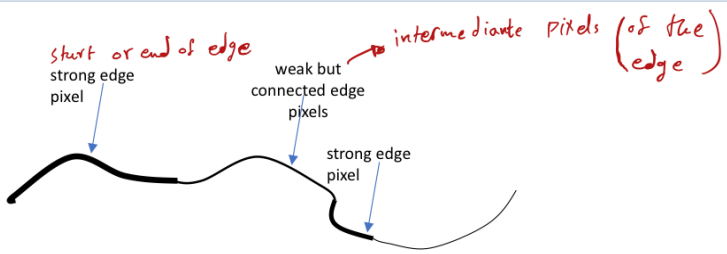
① poor localization

② Too many responses

Canny Edge operator:

multi-Stage Algo.

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation (direction) of gradient
3. Non-maximum suppression: → Edge occurs where gradient reaches Maxima → thinning the edge to get 1 response
   - ❑ Thin multi-pixel wide "ridges" down to single pixel width
   - Non-maximum suppression can help to suppress all the gradient values (by setting them to 0) except the local maxima
4. Thresholding and linking (hysteresis):
   - ❑ Define two thresholds: low and high
   - ❑ Use the high threshold to start edge curves and the low threshold to continue them



W4-P2 - Edge detection AND Mathematical Morphology

Hough Transform (parametric function, Accumulator array)

- Main idea:
  - Record all possible lines on which each edge point lies.
  - Look for lines that get many votes.

fitting

- **Line detection:** Hough transform
  - Simple parametric model, two parameters m, b

$$y = mx + b$$

- **Voting strategy**
  - Hard to generalize to higher dimensions

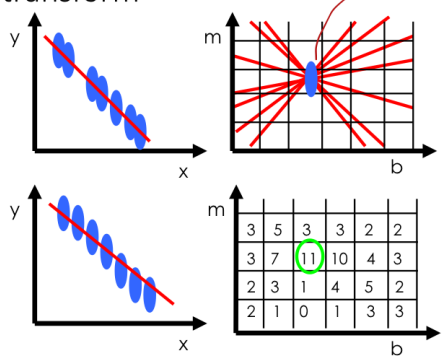
Image space — Hough Space

line ⟷ point

point ⟷ line

Hough transform

highest voting



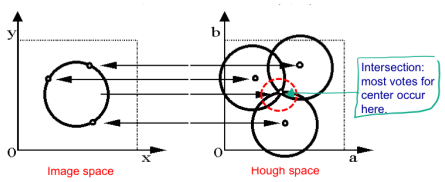
finding circles: (a, b, R) — center, radius

$a = x + R\cos(\theta)$

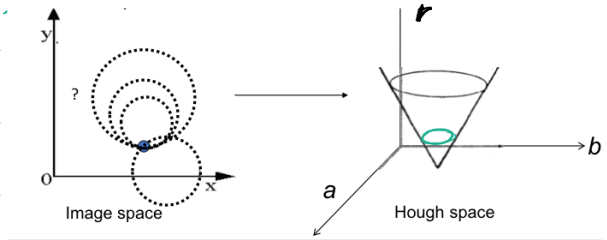
$b = y + R\sin(\theta)$

① R is known (a,b) is unknown

- Search can be reduced to 2D



② R is unknown (a,b) is known



Hough transform: pros and cons

- Pros
  - All points are processed independently, so can cope with occlusion
  - Some robustness to noise: noise points unlikely to contribute consistently to any single bin
  - Can detect multiple instances of a model in a single pass (Provides multiple good fits)
- Cons
  - Complexity of search time increases exponentially with the number of model parameters (Not suitable for more than a few parameters - grid size grows exponentially)
  - Some sensitivity to noise.
  - Quantization: hard to pick a good grid size

Summary

- The Hough transform and its variants can be used to find line segments or circles.
- It has also been generalized to find other shapes
- The original Hough method does not work well for line segments, but works very well for circles.

circles are better than lines

Mathematical Morphology

form and structure (shape of a region)

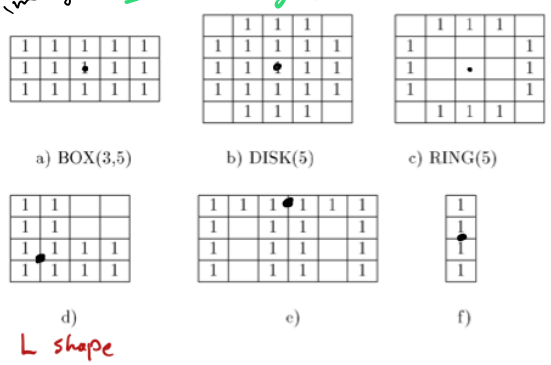
Morphological operations are a set of operations that process images based on shapes. They apply a structuring element to an input image and generate an output image.

as long as it has an original point

consist of two operations:

applied on binary images

Structuring Elements



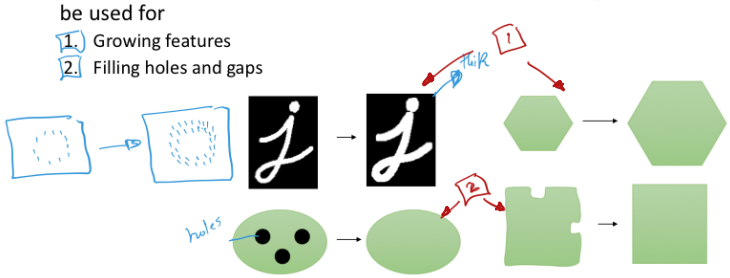
OR ① Dilation Expand ⊕

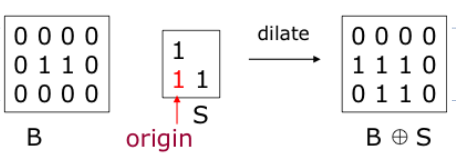
AND ② Erosion Shrink ⊖

origin = where the value in the image will be changed

Dilation

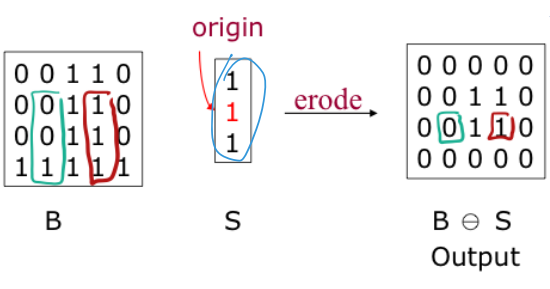
- Dilation expands the connected sets of **1s** of a binary image. It can be used for
  1. Growing features
  2. Filling holes and gaps



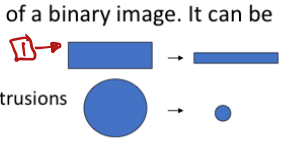


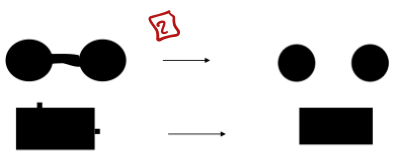
Erosion

inverse of dilation not exactly



- Erosion shrinks the connected sets of 1s of a binary image. It can be used for:
  1. Shrinking features
  2. Removing bridges, branches and small protrusions

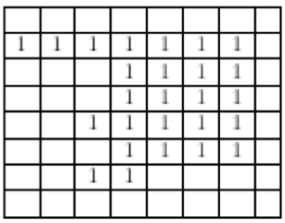




Closing : OR dilation → AND erosion

Opening : AND erosion → OR dilation

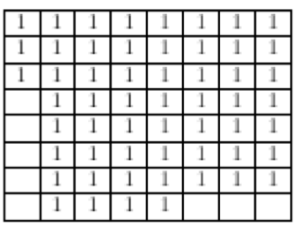
} with SAME structuring element



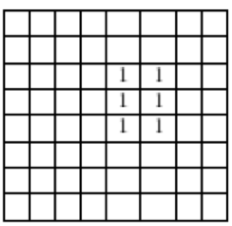
a) Binary image B

mask R, filter, Kernal, Structring element

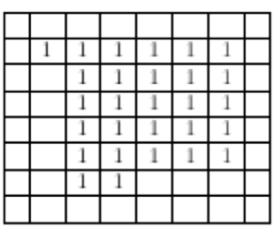
b) Structuring Element S



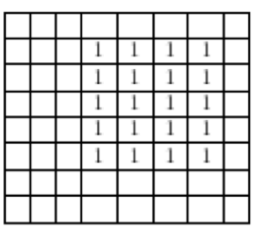
c) Dilation $B \oplus S$



d) Erosion $B \ominus S$



e) Closing $B \bullet S$

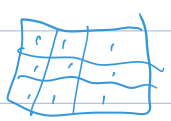


f) Opening $B \circ S$

We can Extract boundary using:

Image A

$A - (A \ominus B)$ → 3X3 square structuring element



Example: Boundary extraction

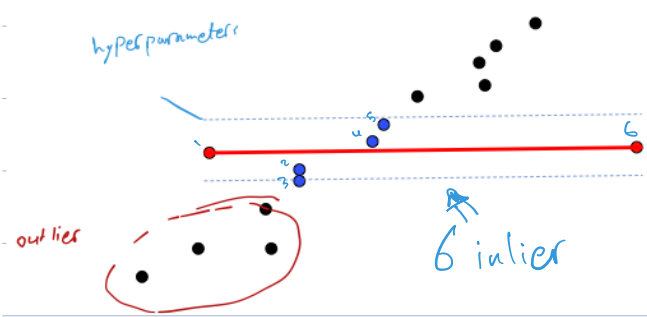
W5 - P1 - Feature detection and matching

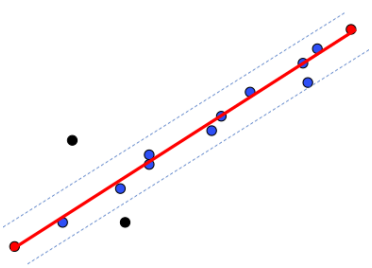
fitting (Data association) methods:

1. Hough Transform
2. RANSAC

RANdom SAmple Consensus (RANSAC)

- A model fitting method for edge detection
  - RANSAC
- Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use only those.





12 "inlier" points

want the highest

success rate (given)

# of iteration we need to get p

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

Sample size

% of inliers

⇒ # of inliers / total # of samples

How many iterations (k) if the desired succes rate (p) is 0.99 ?

Sample size	Proportion of inliers (w)						
n	95%	90%	80%	75%	70%	60%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

RANSAC: Pros and Cons

- **Pros**:
  - General method suited for a wide range of model fitting problems
  - Easy to implement and easy to calculate its failure rate
- **Cons**:
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)

Matching Problem:

can be solved by fitting [مواءمة]

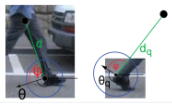
difficulties:-

1. different zoom
2. differen orientation or angle

So we cannot compair pixel by pixel

we use smart map (local features, interest points)

leads to

- Increased robustness to
  - Occlusions 
  - Articulation 
  - Intra-category variations 

Interest points

- 0D structure: single points
  - not useful for matching

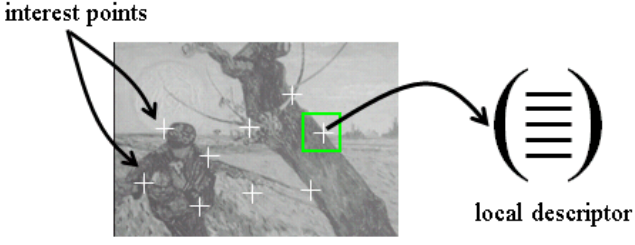
- 1D structure: lines
  - edge, can be localized in 1D, subject to the aperture problem*

- 2D structure: corners
  - corner or interest point, can be localized in 2D, good for matching
  - Interest Points have 2D structure.

More details: https://youtu.be/i2ETv2N1z5E?t=526

We can use matching to generate Panorama images

General Interest Detector/Descriptor Approach



1) Extraction of interest points
2) Computation of local descriptors
3) Determining correspondences between two images
4) Selection of similar images

Common Requirements

- Detect the *same* point *independently* in both images

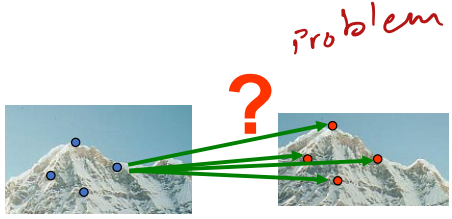
no chance to match!

We need a repeatable detector

Sometimes we got the problem in next slide

Common Requirements

- For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor

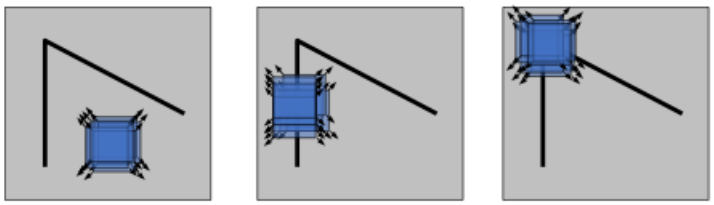
Detectors

to detect interest points

1. Harris
2. Laplacian
3. Difference of Gaussians (DoG)

Harris

1. Small window
2. move this window on the image



"flat" region: no change as shift window in all directions (in intensities)
0D

"edge": no change as shift window along the edge direction
just 1D

"corner": significant change as shift window in all directions
2D

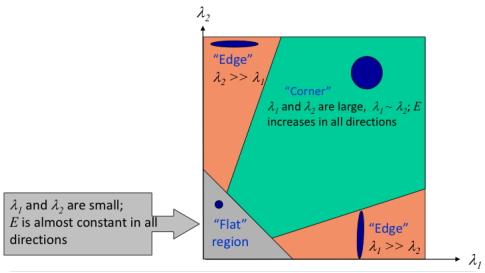
$\sum I_x^2$ → Large
$\sum I_y^2$ → Large
Corner (large change in both sides)

$\sum I_x^2$ → Small
$\sum I_y^2$ → Large
Edge

$\sum I_x^2$ → Small
$\sum I_y^2$ → Small
Nothing

- This measure of change (in the shift between two windows) can be approximated by:

$$E(u,v) = (u \quad v) M \begin{pmatrix} u \\ v \end{pmatrix}$$



$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

both small => flat
one large => edge
both large => corner

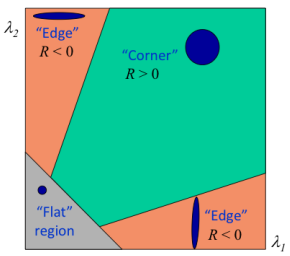
We can use another way to avoid computing Eigenvalues ⇒ Respone func.

- Measure of cornerness in terms of λ_1 and λ_2

response function → $\theta = \det(M) - \alpha \operatorname{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region (of small positive)

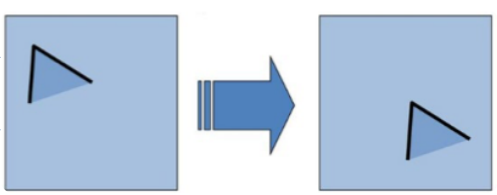
- Fast approximation
  - Avoid computing the Eigenvalues
  - α: empirical constant (0.04 to 0.06)



We can Smooth the window with Gaussian to be rotation invariant

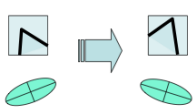
Harris properties:-

(1) It is translation invariant (Same information will be detected)



Corner location is equivariant w.r.t. translation

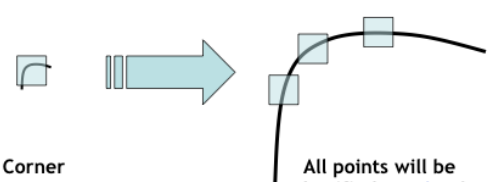
(2) It is rotation invariant (Same information will be detected)



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response R is invariant to image rotation

(3) It is NOT Scale invariant (NOT Same information will be detected)



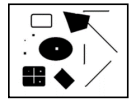
Not invariant to image scale!

Summary: Harris Detector

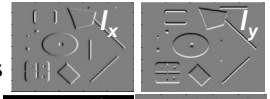
- Compute second moment matrix (autocorrelation matrix)

$$M(\sigma_I,\sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

σ_D: for Gaussian in the derivative calculation
σ_I: for Gaussian in the windowing function



1. Image derivatives



2. Square of derivatives

3. Gaussian filter $g(\sigma_I)$

4. Cornerness function – two strong eigenvalues

$$R = \det[M(\sigma_I,\sigma_D)] - \alpha[\text{trace}(M(\sigma_I,\sigma_D))]^2$$

$$= g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

not part of harris!

5. Perform non-maximum suppression

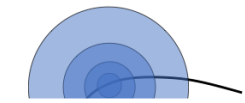
Slide credit: Krystian Mikolajczyk

W5 - P2 - Feature detection and matching

we need to solve scale invariant issue

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images

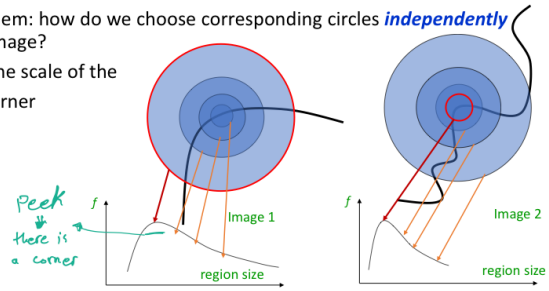
unfit! it satisfied the right result



① ▪ A "good" function for scale detection has one stable sharp peak

② 

- The problem: how do we choose corresponding circles ***independently*** in each image?
- Choose the scale of the "best" corner



Important: this scale invariant region size is found in each image independently!

- Choices of Kernel: $f = \text{Kernel} * \text{Image}$

2 times derivative

choices of Kernels

① (Laplacian) highlight high intensity (like log)

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

② (Difference of Gaussians)

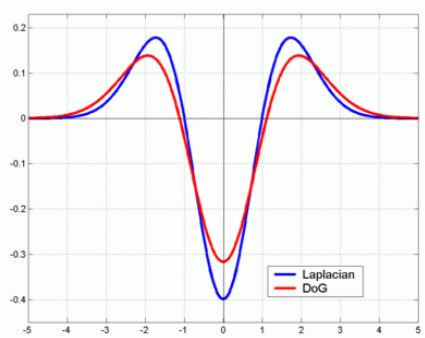
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

Smooth Image − Smooth Image (different sigma)

I ↓ → GI, G'I

where G is the Gaussian function

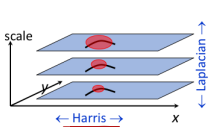
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$



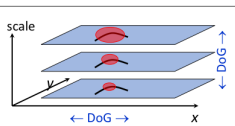
Note: both kernels are invariant to *scale* and *rotation*

Scale Invariant Detectors

- Harris-Laplacian[1] → combination
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



- DoG (from SIFT by Lowe)[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



Scale Invariant Detection: Summary

- Given: two images of the same scene with a large *scale difference* between them
- Goal: find *the same* interest points *independently* in each image
- Solution: search for *maxima* of suitable functions in *scale* and in *space* (over the image)

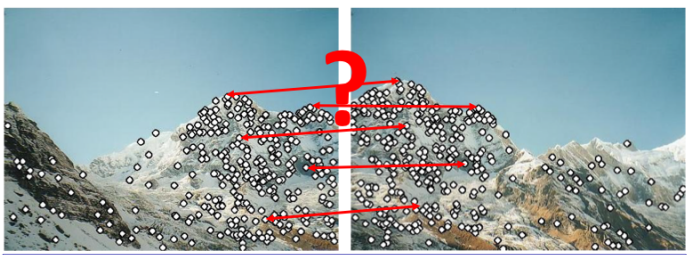
the region size or circle

Methods:

1. Harris-Laplacian [Mikolajczyk, Schmid]: maximize Laplacian over scale, Harris' measure of corner response over the image
2. SIFT [Lowe]: maximize Difference of Gaussians over scale and space

After we extract intrest point we need to describe them

- Descriptor: Vector that summarizes the content of the keypoint neighborhood.



Point descriptor should be: (1-to-1 mapping)
1. Invariant
2. Distinctive

① Simple descriptors:

interest point		neighborhood around interest point		Normalized neighborhood around interest point
201	− (substract)	45 56 200 46 201 200 85 101 105	=	156 145 1 155 0 1 116 100 96

Properties:-

- Translation Invariant
- Not scale invariant
- Not rotation invariant
- Somewhat invariant to lighting changes

② SIFT Algo. :

used for:

① Detect

② Describe

③ Match local features between images

Steps involve in SIFT Algorithm

1. Scale-space Extrema Detection
2. Key point Localization
3. Orientation Assignment
4. Keypoint Descriptor
5. Key point Matching