# Image Processing

An image is nothing more than a **X-Y** plane.

## Pixel

A Pixel is a smallest element of an image.

Each pixel has a specific value.

In an 8-bit gray scale image, the value of the pixel between 0
and 255.

```
Calculation Pixel Value
Total no: of pixel = no of rows * no of cols
```

### Pixel Value

- Pixel value **zero** means there is no light. So, it is black pixel.
- Pixel value **255** means full white pxiel.

## Bpp - Bits per pixel

The number of different colors in an image depends on the bpp.

- One bit image - 2 colors(1, 0)
- Two bits image - 4 colors(00, 01, 10, 11)
- Three bits image - 8 colors (000, 001, 010, 011, 100, 101, 110, 111)
- n bits image - **2^n colors**

Color images are usually of the 24 bpp format, or 16 bpp.

### White color & Black Color

The color value of black color is always **0**.

The color value that represent **white** can be

```
white color = 2^n - 1;
```

## Image Size

The size of an image depends on 3 factors.

- Number of rows
- Number of cols
- Bits per pixel

To get size of an image

```
size = rows * cols * bpp;
```

Assume a gray-scale image has 1024 rows & 1024 cols.

```
Size = rows * cols * bpp
     = 1024 * 1024 * 8 // (gray-scale is 8-bits)
     = 8388608 bits
```

### 8388608 bits

- in byte, 8388608 / 8 = **1048576** bytes
- in KB, 1048576 / 1024 = **1024** KB
- in MB, 1024/ 1024 = **1**MB

## Types of Image

### Binary Image

It is based on 1bbp. Each pixel may be one of two states - 0 & 1 or black and white.

### Gray-Scale Image

It is based on 8bbp. Each pixel can have 256 pixel states from 0 to 255.

0 is black.
255 is white.
254 is gray.

Format - PGM (Portable Gray Map)

### 16 bits color format

It has 65,536 different colors.
But it is quite complex than gray scale.
Each pixel is built upon **RGB** (Red, Green, Blue) and sometimes **RGBA** (Red, Green, Blue, Alpha)

It has many different distributions.

4 bits for Red, 4 bits for Green, 4 bits fro Blue, 4 bits for Alpha.

OR

5 bits for Red, 5 bits for Green, 5 bits fro Blue, 1 bits for Alpha.

# Color Image to Gray Scale Conversion

Now we will convert a color image into gray scale image.

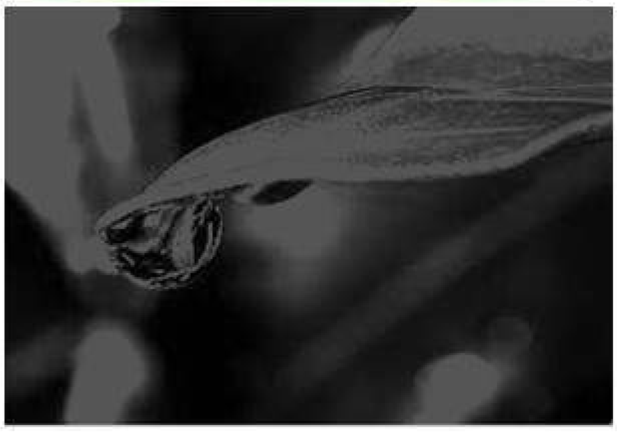
There are two methods for doing this.

- Average Method
- Weighted method or luminosity method

## Average Method

This is simple method. Each pixel has R, G and B value.
Add three values and divide by 3.

```
GrayScale = (R+G+B)/3
```



It works! But something happens. Our image similars to a black image rather than a gray-scale image. **Why?**

3 different colors have different wavelengths & have their own contribution to the image. Thus, we have to average contribution of each color to image. Not average of the total.

What we have done is

```
33% red, 33% green, 33% blue
```

Taking every color has same contribution to image.

## Weighted method or luminosity method

The problem to average method has been solved by weighted method.

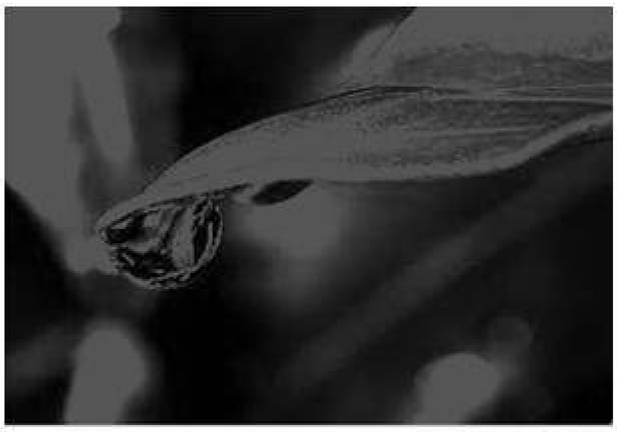
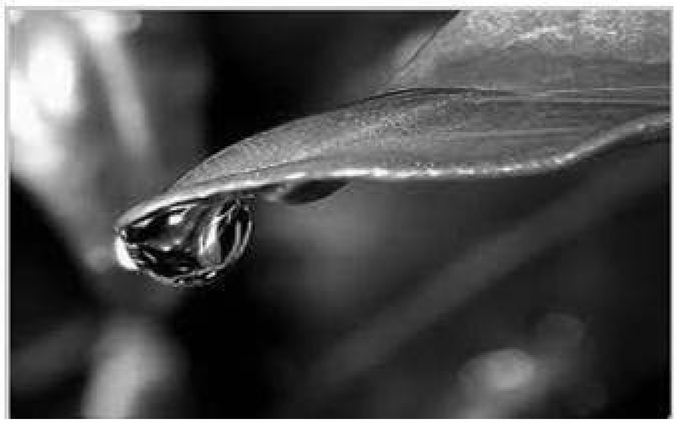
For colors, Red color has more wavelength than others. Green is the most smoothing color to the eye.

That means Red color contribution must be decreased, Green color must be increased and blue color must be adjusted based on two colors.

```
New grayscale image = ( (0.3 * R) + (0.59 * G) + (0.11 * B) )
```

According to this equation, Red has contribute 30%, Green has contributed 59% which is greater in all three colors and Blue has contributed 11%.

Compare Two Images of two methods

### Which is better?# Sampling

Sampling is defined as taking sample.

It is categorized into two types.

- oversampling or upsampling
- undersampling or downsampling

### OverSampling

OverSampling has a very deep application in image processing known as **zooming**.

#### Zooming

Zooming refers to increase the quantity of pixels, so that when you zoom an image, you will see more details.
The increase in the quantity of pixels is done through oversampling.# Resolution

We have defined pixel as smallest element of an image.
The **resolution** can be defined in many ways. Such as pixel resolution, spatial resolution, temporal resolution, spectral resolution.

Now, we are going to focus on pixel resolution.

Computers may have **1024 pixels * 960 pixels**. If an image has **M rows** and **N cols**, then its resolution is *MN** resolution.

```
size = resolution * bpp(bits per pixel)
```

## Aspect Ratio

Aspect ratio is the ratio between width of an image and the height of an image. This ratio differs in different images and different screens.

```
Aspect ratio = cols : rows
```

Most common ratios are

**1.33:1, 1.37:1, 1.43:1, 1.50:1, 1.56:1, 1.66:1, 1.75:1, 1.78:1, 1.85:1, 2.00:1, e.t.c**

Aspect ratio maintains a balance between the appearance of an image on the screen.

For example, we have 1024 rows & 1024 columns. The aspect ratio is 1:1. If we want to make image smaller, reduce to 256 rows and 256 cols, which is same ratio.

#### Problem

An gray-scale image with aspect ratio 4:3 has resolution 480000 pixels.

- Resolve pixel resolution to calculate the dimensions of image
- Calculate the size of the image

```
//Dimensions Calculation

aspect ratio = cols : rows = 4 : 3
resolution =  480000 = cols * rows

cols = 4/3 rows

480000 = 4/3rows * rows
rows^2 = 3/4 *480000

rows = 600 pixels
cols = 800 pixels
```

```
//Size Calculation
size = resolution * bpp
bpp of gray-scale image = 8

size = 480000 * 8 = 3840000 bits
     = 480000 bytes
     = 468 Kb
```

## Zooming

There are three common zooming methods.

- Pixel replication or (Nearest neighbor interpolation)
- Zero order hold method
- Zooming K times

## Pixel replication

Other name - Nearest neighbor interpolation

This method is to replicate the neighboring pixels.
Suppose you want 2 times zooming an image of 2 rows and 2 cols.
It can be done in 2 steps : row-wise or col-wise.

### Row-wise

Suppose in matrix.

1 2
3 4

To zoom 2 times in row-wise, replicate each row-pixel to its adjacent cell.

That means each pixel in a row is replicate twice.

1 1 2 2
3 3 4 4

### Next Step : Col-wise

Each pixel in a column is replicated.

1 1 2 2
3 3 4 4

to

1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4

New Image size will be increased. 2x2 image is now converted into 4x4 image.

```
[newCols,newRows] = [ (cols * zoomFactor),(rows * zoomFactor) ]
```

**Pros**

The method is so simple

**Cons**

As zooming factor increases, the image will get more blurry.

## Zero order hold

It is another method of zooming, also known as zooming twice.
Because it can only zoom twice.

### How it works

First, we take two adjacent pixels from a row and add them. Divide the result by 2. Put the output as a new pixel between the original adjacent pixels. We do row-wise first and then col-wise.

Suppose in matrix.

1 2
3 4

First row-wise. Add 1 & 2. Divide result by 2, which is 1.5. Then, take nearby value approximately as 1. Put 1 between 1 & 2. So does second row.

1 1 2
3 3 4

Then col-wise. Take 1 & 3, and add and divide by 2 is 2.
Put 2 between 1 & 3.

1 1 2
2 2 3
3 3 4

```
[newCols,newRows] = [(cols * 2)-1,(rows * 2)-1]
```

**Pros**

It does not create as blurry image as Pixel replication.

**Cons**

It can only zoom 2 times & 2 power times.
Here, 2x2 is zoomed into 3x3. 3x3 will be into 5x5. And then 5x5 will be into 9x9.

## K-Times zooming

K-times zooming solves the problem of pixel-replication and zero-order zooming.

**K = zooming factor**

### Working Algorithm

1. First take two adjacent pixels
2. Subtract smaller from larger pixels
3. Result is difference
4. Divide the difference with zooming factor **(K)**
5. Add the output(OP) to smaller value
6. Put the result between two values
7. Add OP to previous result you put & place it next to previous pixel
8. Do till we have (k-1) number of values

Sound Confused? Look Example!

Zooming factor = 3

15 30 15
30 15 30

First row-wise.

First take two adjacent pixels => 15 & 30

Subtract smaller from larger pixels => 30 - 15 = difference = 15

Divide the difference with zooming factor => 15/3 = 5 = output

Add the result to smaller value => 15 + 5 = 20

Put the result between two values.
Add OP to previous result you put & place it next to previous pixel => 20 + 5 = 25

Now, we have two values : 20,25. We stop here. Because we need (k-1) numbers of values. **k-1=2 (k=3)**

15 20 25 30 25 20 15
30 25 20 15 20 25 30

Then col-wise. Same as row-wise.

Result is

15 20 25 30 25 20 15
20 21 21 25 21 21 20
25 22 22 20 22 22 25
30 25 20 15 20 25 30

```
[newCols,newRows] = [k(cols-1)+1,k(rows-1)+1]
```

**Pros**

Solve blurry problem of pixel manipulation & 2power problem of zero-order.

**Cons**

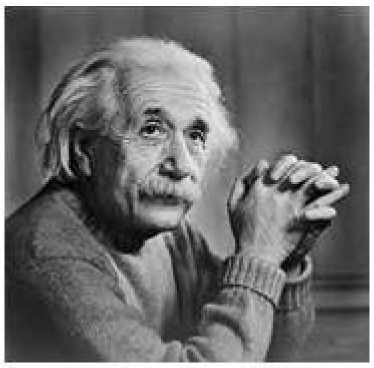
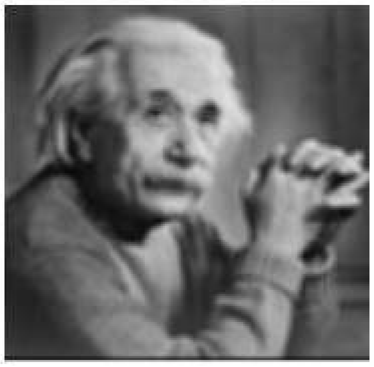
It costs a lot of computation power.# Spatial Resolution

We have discussed pixel resolution. Now, let's study spatial resolution.

Spatial resolution states that the number of pixels in an image does not matter.

It is defined as **the number of independent pixels values per inch**.

In spatial resolution, we cannot compare clarity of two images of different sizes. We have to compare the clarity of same size images.

Compare two images which are same size.
1st picture has more spatial resolutin than 2nd one.

Spatial resolution refers to clarity that different devices measure it differnetly.

- Dots per inch (mostly in monitor)
- Lines per inch (mostly in laser printer)
- Pixels per inch (Tablets, mobile phone etc)

## Pixels per inch (Pixel density)

It is a measure of spatial resolutin in devices like mobile phones and tablets.
The higher the pixel density, the higher the quality.

### Calculating PPI

Consider a device, a mobile phone has 1080 x 1920 pixels. **(cols * rows)** ***or*** **(width * height)**

Diagonal resolution in pixel is calculated in Pythagora's theorem.

```
Diagonal Resolution = Sqrt( width^2 + height^2 )
```

Thus Diagonal Resolution is 2202.90717 pixels;

To get PPI, divide diagonal resolution by diagonal size in inches.

```
pixels per inch = Diagonal Resolution/diagonal size in inches
```

Suppose device has 5 inches diagonal size. So, **PPI** is 440.58 ~ 441.

Therefore, pixel density of this device is 441 PPI.

## Gray Level Resolution

Gray level resolution refers to the predictable or deterministic change in the shades or levels of gray in an image.

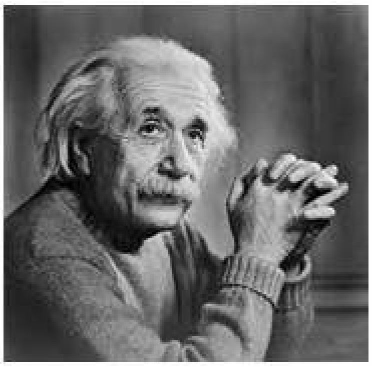
Confuse?

Simply, gray level resolution refers to the number of bits per pixel (bpp).

Remember bpp?

Bpp value means the number of available colors of an image.

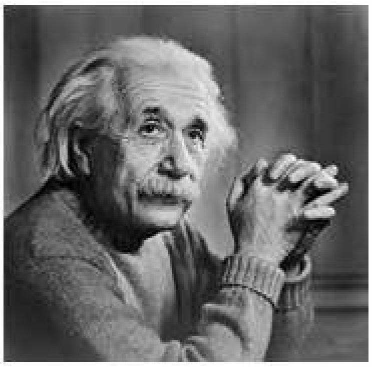
```
Gray Level Resolution = 2^bpp
```



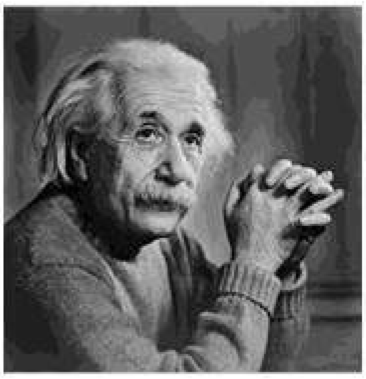
This image is an gray-scale image. Thus, it is bpp is 8 and Gray Level Resolution is 256.

### Contouring

If we reduce the gray scale level of image, we will see a special effect.



*256 Level*



*16 Level*

When compare two photos, we can see obviously special effects (some false colors, or edges) in 16 Level than original 256 Level.
This feature is called **contouring**.
The answer to this effect lies in iso-preference curves.# Isoperference curves

The nature of Isopreference curves shows that contouring effect not only depends on decreasing the gray scale level but also on the image details.

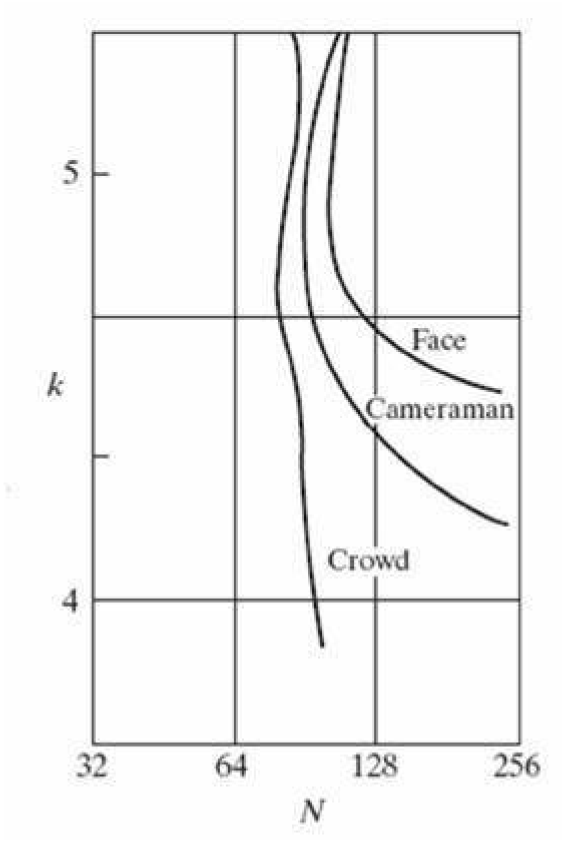
If an image has **more detail**, the effect of contouring would start appear on this image **later**, as compare to an image which has less detail, when the gray levels are descreasd on gray-scale level.

According to the original research, the researchers took these three images and they vary the Gray level resolution, in all three images.

First image is least detailed since it contains a face. Second is less detailed than third. But third image is most detailed since it contains lots of people.

## Experiment

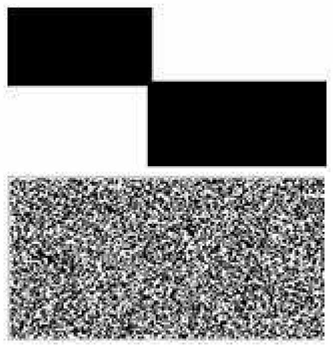
The gray level was varied on all images and ask audience to rate the images. After rating, result came out as graph.



N = number of gray level,
k = bits per pixel

The point of interest is that isopreference curve seems to be more vertical in more-detailed image.
That means, less-detaied images are more suspcetable to contouring.# Dithering

Dithering is the process by which we create illusions of the color that are not present actually. It is done by the random arrangement of pixels.



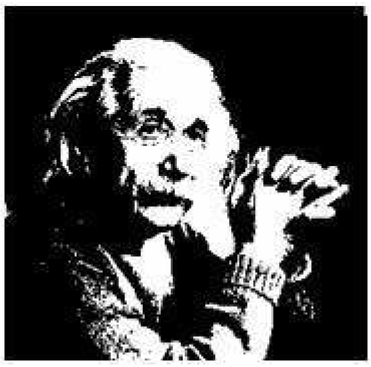
Compare two images. They have same quantity of pixel but different arrangement of pixels.

## Why Dithering?

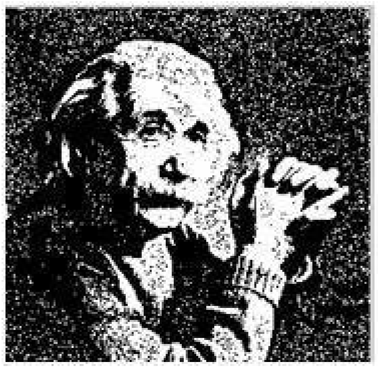
To get more slight image.

## Preforming Process

First we will work on thresholding. In thresholding, we take a constant value. All pixels that are above constant value are set as white(1) and below are black(0).



After that, we got a black and white image. But it is not so slight to see. We perform some random dithering effect on this B&W image.



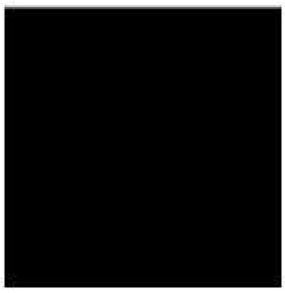
Now we got slighter image by random dithering.# Brightness & Contrast

## Brightness

Brightness can be defined as energy outputs of sources of light when comparing things. Thus, it is relative property.

### Increase brightness of an image

This is a black image.



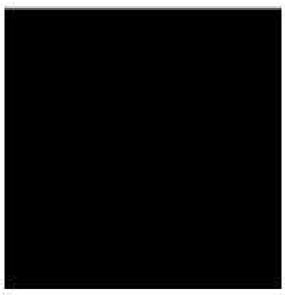
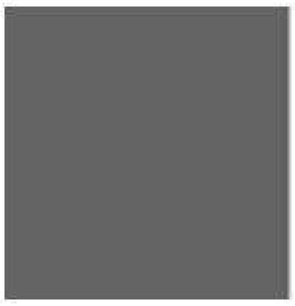
This is matrix of a black image.

0 0 0  
0 0 0  
0 0 0

Now, add 100 to each pixel of the black image.

100 100 100  
100 100 100  
100 100 100

Compare original and result image



Though we change pixels, what result is change in brightness.
The real working process is that pixel value is gradaully increased into white.

## Contrast

Contrast is simply defined as difference between maximum and minimum pixel values of an image.

100 100 100
100 100 100
100 100 100

For this image,

```
contrast = 100 - 100
         = 0
```

Thus, this image has contrast 0.# Image Transformation

Transformation is a function that convert a set of input into a set of output.

Image Transformation is a function that convert input image into output image.

```
output = Trasformer(input);
```

**where**

input = pixel value of input image at position (x,y)

output = pixel value of output image at position (x,y)

## Gray Level Transformation

### Image enhancement

An enhanced image provides better contrast and more detailed than a non-enhanced image.
It has lots of applications.

- Medical Image enhancement
- Satellite Image enhancement
- Sensor Image enhancement

Image enhancement can be done with gray level transformation.

- Linear
- Lograthimic
- Power - law

### Linear Transformation

It has two parts : simple identity and negative transformation.

**Simple Identity Transformation**

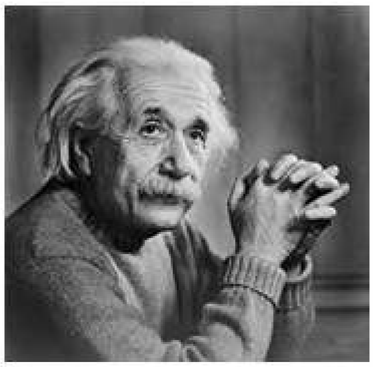
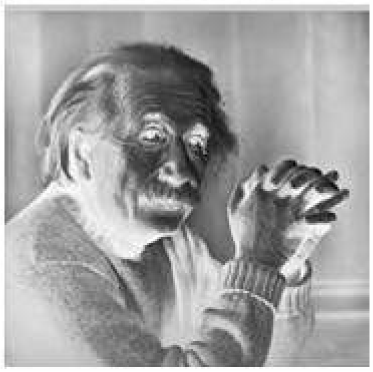
It is transformation that each pixel of image as direct ouput. Input and output images are the same.

**Negativ e Transformation**

It is a function that each value of image is subtracted from (L-1) and mapped as output.

```
output = (L-1) - input
```

where
L is the number of gray-scale level of image.

Since original image is a gray-scale image, bpp = 8
L = 2^8 = 256;

output = (256-1)-input

Lighter pixel becomes dark and darker pixel becomes light.
Thus it is called negative transformation.

## Lograthimic Transformation

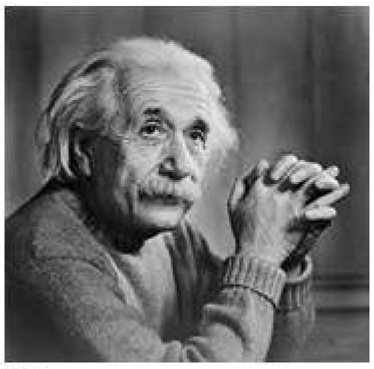
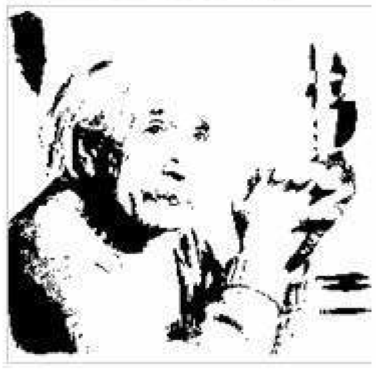
It also contains two types : Log transformation and Inverse Log Transformation

**Log Transformation**

```
output = c log(input+1)
```

where c = constant
1 is added because when input is 0, log 0 would be infinity.
Therefore log 1 is 0.

Dark pixels are expanded into larger values & Larger values compressed into lograthmic value. c is the value of enhancement.

Inverse log transformation is opposite to log transform.

### Power-Law Transformation

```
output = (c*input)^γ
```

γ is called gamma and hence this transformation is called gamma ransformation. The gamma of different display devices is different. That's why different devices show images at their own enhancement.# Histogram

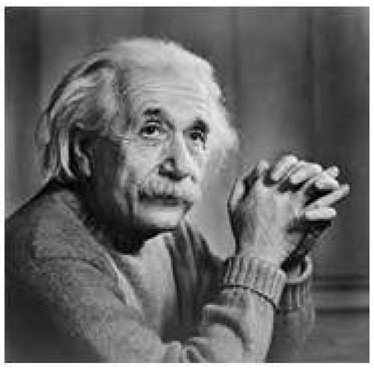
Histogram is a graph that shows frequency of anything.

A histogram usually contains x and y axis.
X is the whose frequency and Y is frequency.

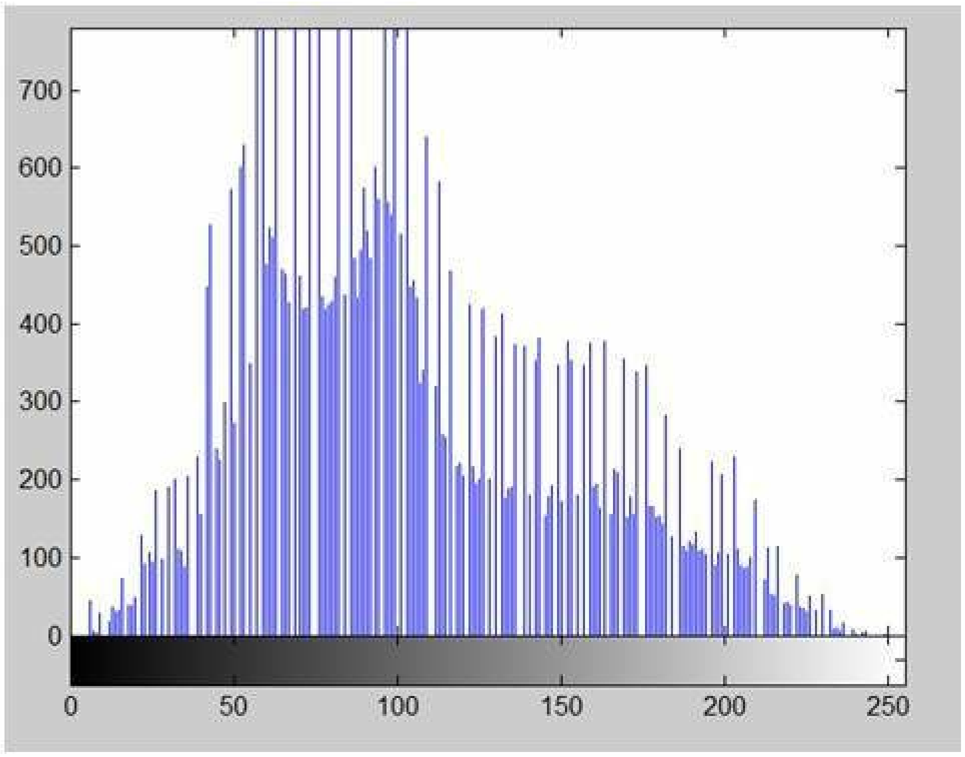
histogram

## Histogram of an image

Image histogram also shows frequency but it shows frequencies of pixel values.



The histogram of above eistein picture is below.



histogram has many uses in image processing. For example,

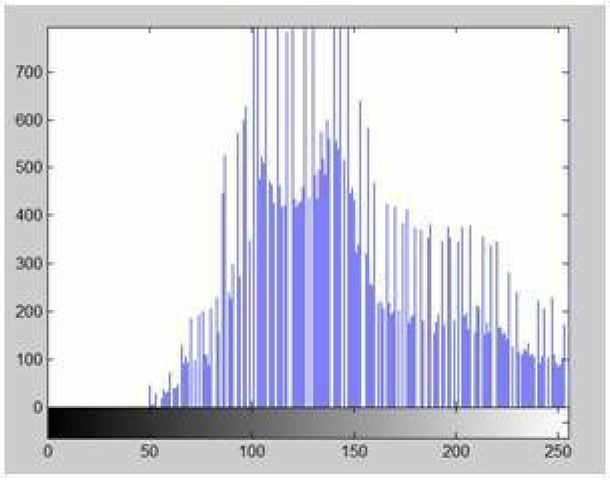
- brightness and contrast
- equalizing the image
- computer vision

## Histogram Sliding

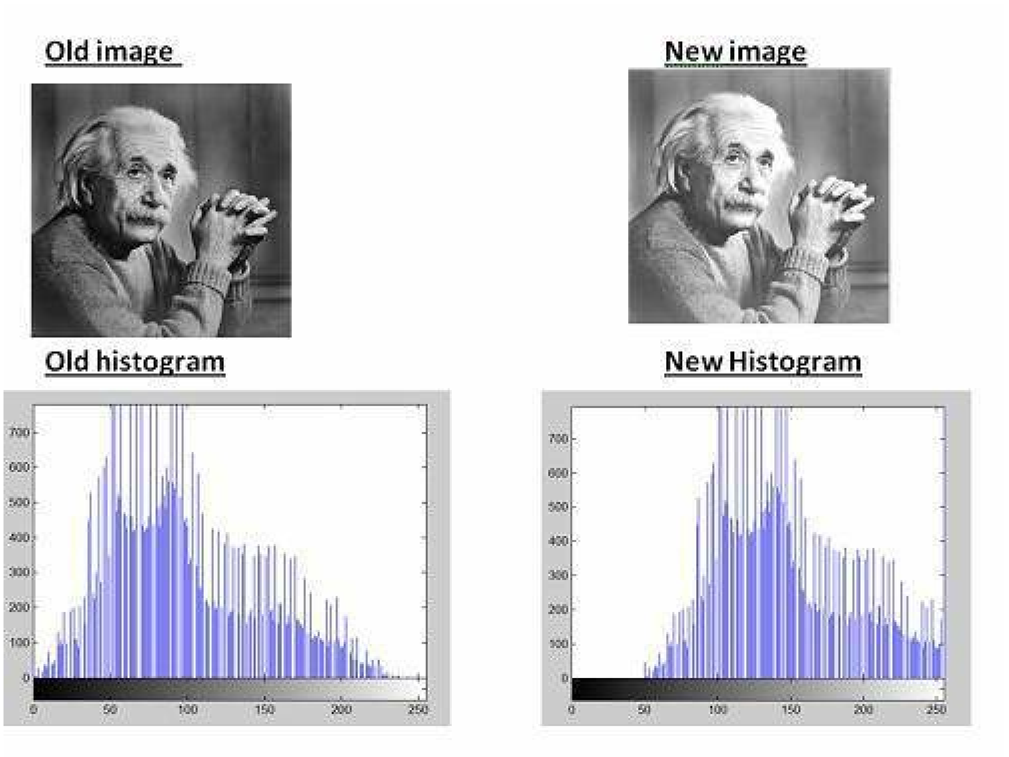
histogram sliding is shifing a complete histogram leftwards or rightwards. Due to shifting, clear changes in image can be seen.

For example,

shift right to 50 pixels will increase the brightness. Our image has highest frequency between 50 and 100. Shifting 50 pixels to right means add 50 to each pixel. Now, hight frequency will occcurs between 100 and 150.



And compare the two images.



Shifting left will decrease brightness.

## Histogram Streching & Equalization

We have discuss about brightness which is histogram shifting. histograms can also be used for changing contrast. There are two ways of performing constrast on image histogram.
One is **histogram stretching** & another is **histogram equalization**.

### Contrast

Contrast is the difference between maximum and minimum pixel intensity. For einstein iamge, contrast is 255.

### Histogram Stretching

Now, we will increase the contrast of image.

```
output = [(input- min pixel)/contrast] * gray level
gray level = 2^bpp
```

For einstein, min pixel = 0, contrast = 255, gray level = 256
Thus

```
output ~ input
```

So no contrast effect would occurs here.
This is the failure of histogrm streching.# Probability

Histrogram equalization is built upon two concepts : probability mass function(PMF) & cumulative distributive function(CDF).

## Probability Mass Function (PMF)

The probability that a discrete random variable X takes on a particular value x, that is, P(X = x), is frequently denoted f(x). The function f(x) is typically called the probability mass function.

### Calculating PMF

For example, an outcome of a dice has only 6 possible values.

Hence, by calculating probability of each value a random variable attains you effectively calculating a PMS for that variable.

e.g. In case of a dice each value [1, 2, 3, 4, 5, 6] has probability 1/6. Hence PMS if X belongs [1, 2, 3, 4, 5, 6] is 1/6 for every other value of X it is 0.

**Reference taken from Quora** (https://www.quora.com/How-can-you-calculate-probability-mass-function)

## Cumulative Distributive Function (CDF)

The PMF is one way to describe the distribution of a discrete random variable.

### Definition

The cumulative distribution function (CDF) of random variable X is defined as
FX(x)=P(X≤x), for all x∈R.

### Calculating CDF

We already computed that the PDF of X is given by Pr(X = k) = 1/6 for k = 1,2,...,6. The CDF can be computed by summing these probabilities sequentially; we summarize as follows:

Pr(X ≤ 1) = 1/6

Pr(X ≤ 2) = 2/6

Pr(X ≤ 3) = 3/6

Pr(X ≤ 4) = 4/6

Pr(X ≤ 5) = 5/6

Pr(X ≤ 6) = 6/6 = 1

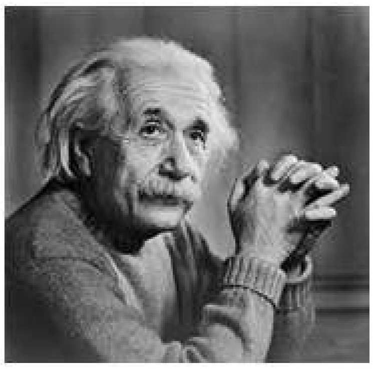
Notice that Pr(X ≤ x) = 0 for any x < 1 since X cannot take values less than 1. Also, notice that Pr(X ≤ x) = 1 for any x > 6. Finally, note that the probabilities Pr(X ≤ x) are constant on any interval of the form [k,k + 1) as required.

## PMF and CDF usage in histogram equalization

In histrogram equalization, first and second steps are PMF and CDF. As the name implies, we have to eqaulize every pixel of image. PMF helps us calculating the probability of each pixel. CDF gives us cumulative sum of these values. Then what?

Read on next chapter!# Histogram Equalization

In this chapter, we will see how to contrast with histogram equalization. In this case, histogram is not always not need for contrast. In some cases, histogram is worse for contrast.



From Last chapter, in histogram equalization, first and second steps are PMF and CDF. As the name implies, we have to eqaulize every pixel of image. PMF helps us calculating the probability of each pixel. CDF gives us cumulative sum of these values.

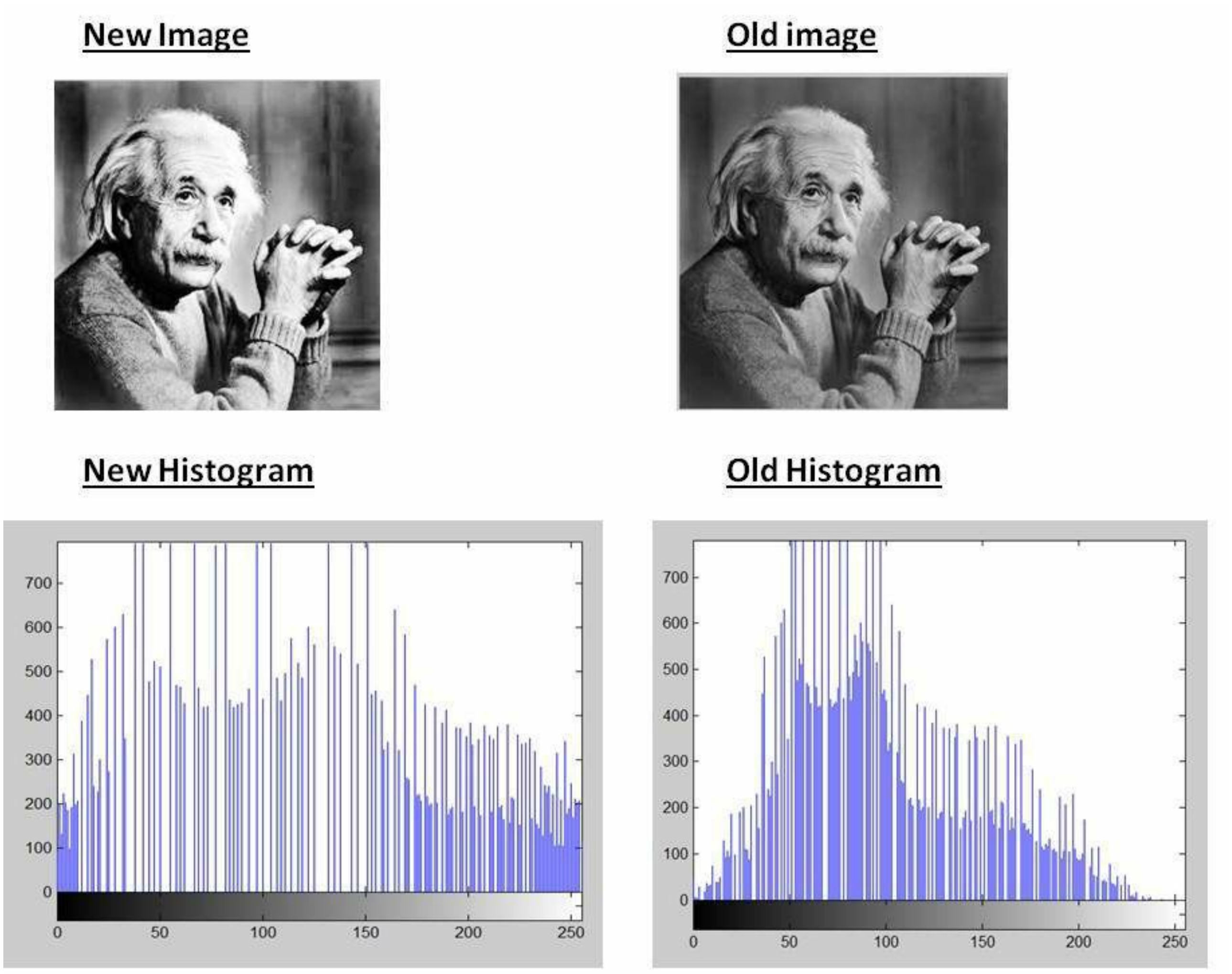
Calculate CDF according to the gray scale level.

| Gray Level Value | CDF |
|---|---|
| 0 | 0.11 |
| 1 | 0.22 |
| 2 | 0.55 |
| 3 | 0.66 |
| 4 | 0.77 |
| 5 | 0.88 |
| 6 | 0.99 |
| 7 | 1 |

Next Step is to multiply CDF values with (Gray Level - 1).
Consider we have 3bbp image, then Gray Level is 8.

| Gray Level Value | CDF | CDF*(Gray Level -1) |
|---|---|---|
| 0 | 0.11 | 0 |
| 1 | 0.22 | 1 |
| 2 | 0.55 | 3 |
| 3 | 0.66 | 4 |
| 4 | 0.77 | 5 |
| 5 | 0.88 | 6 |
| 6 | 0.99 | 6 |
| 7 | 1 | 7 |

Then map old value to new value. 0 to 0, 1 to 1, 2 to 3,etc.



There is also one important thing to be note here that during histogram equalization the overall shape of the histogram changes, where as in histogram stretching the overall shape of histogram remains same.# Convolution

Where have we been?

We have find two different ways of manipulation.

- Transformation
- Histogram

Another way of dealing image is **convolution**.

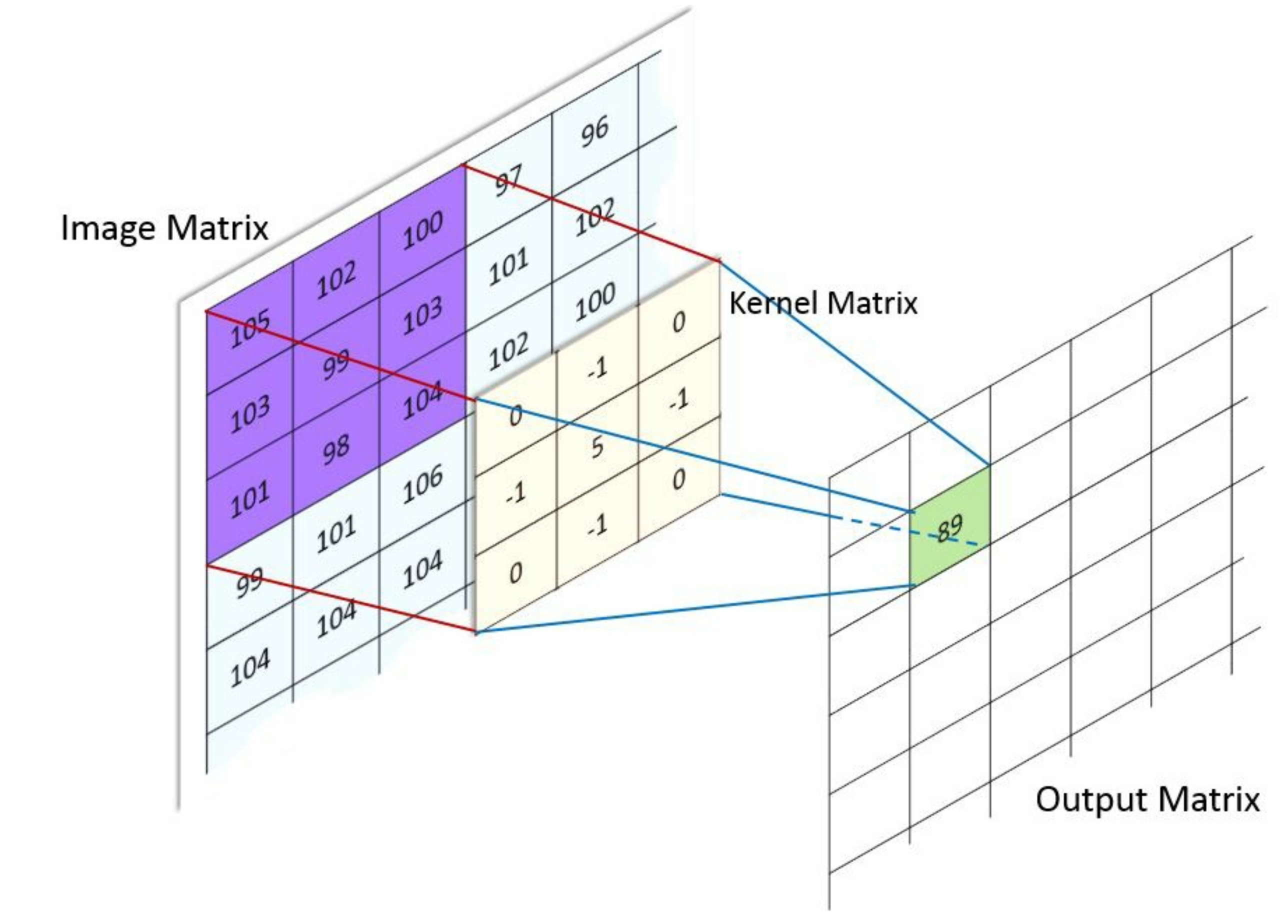
Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel. *(wikipedia)*

Each convolution operation has a kernel which could be a any matrix smaller than the original image in height and width. (http://machinelearninguru.com)

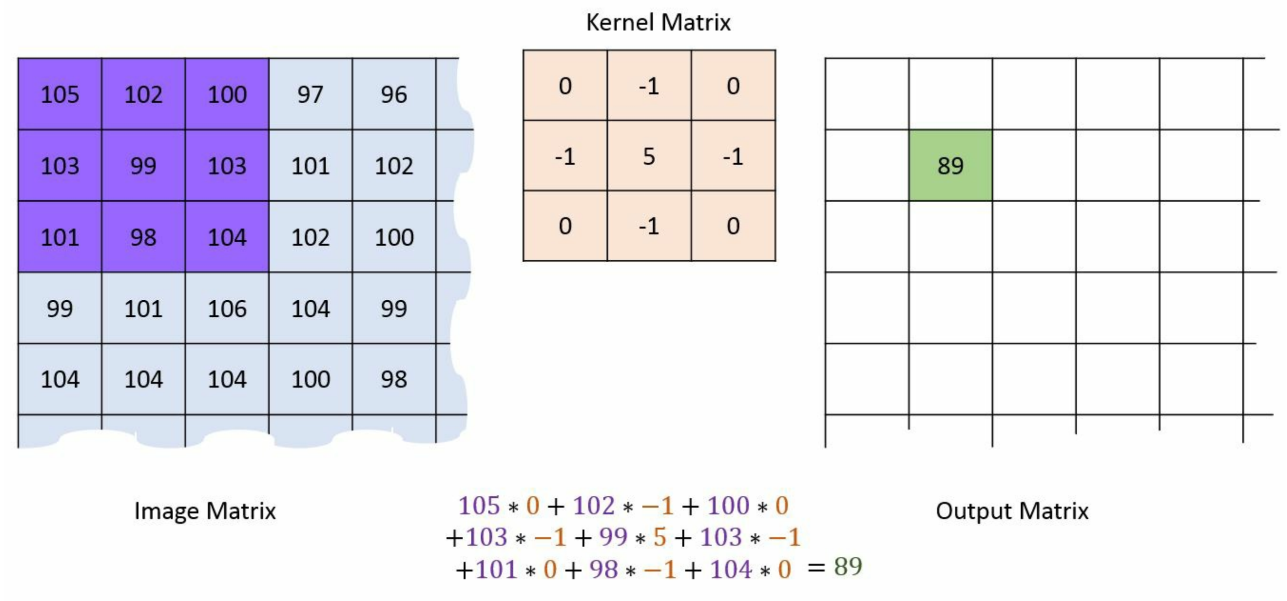
## Convolution Process

In order to perform convolution on an image, following steps should be taken.

- Flip the kernel both horizontally and vertically. As our selected kernel is symetric, the flipped kernel is equal to the original.
- Put the first element of the kernel at every pixel of the image (element of the image matrix). Then each element of the kernel will stand on top of an element of the image matrix.



- Multiply each element of the kernel with its corresponding element of the image matrix (the one which is overlapped with it)
- Sum up all product outputs and put the result at the same position in the output matrix as the center of kernel in image matrix.
-

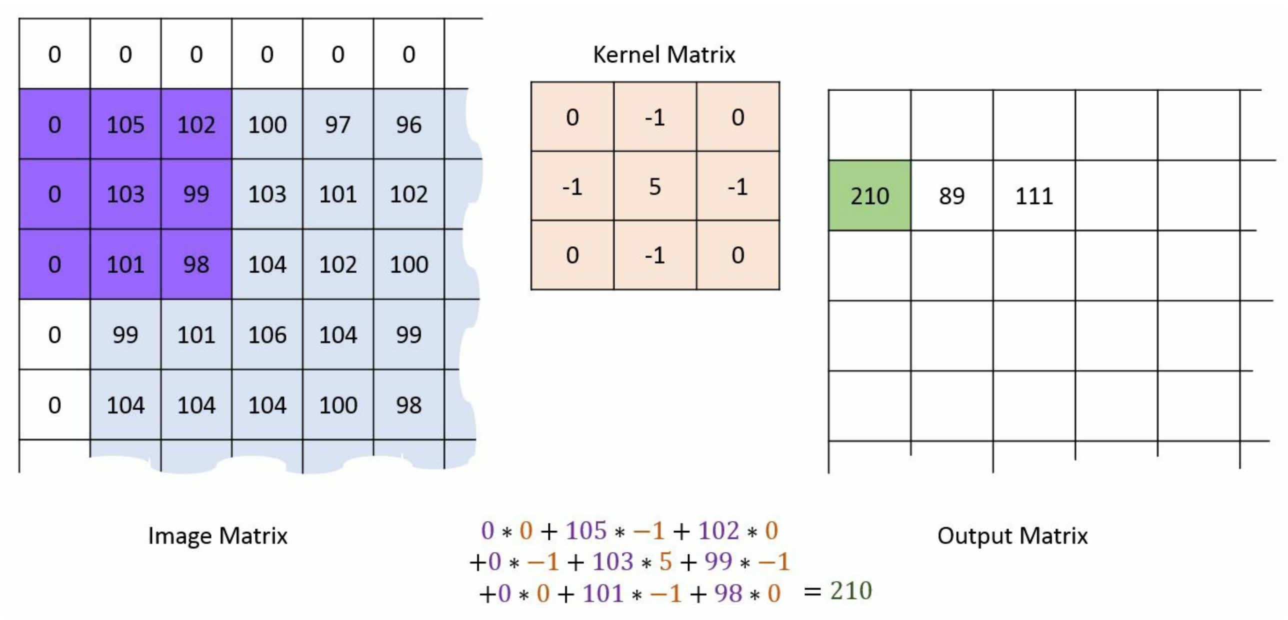


| | | | | |
|---|---|---|---|---|
| 105 | 102 | 100 | 97 | 96 |
| 103 | 99 | 103 | 101 | 102 |
| 101 | 98 | 104 | 102 | 100 |
| 99 | 101 | 106 | 104 | 99 |
| 104 | 104 | 104 | 100 | 98 |

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$$105 * 0 + 102 * -1 + 100 * 0 \\ +103 * -1 + 99 * 5 + 103 * -1 \\ +101 * 0 + 98 * -1 + 104 * 0 \;\; = 89$$

- For the pixels on the border of image matrix, some elements of the kernel might stands out of the image matrix and therefore does not have any corresponding element from the image matrix. In this case, you can eliminate the convolution operation for these position which end up an output matrix smaller than the input (image matrix) or we can apply padding to the input matrix (based on the size of the kernel we might need one or more pixels padding, in our example we just need 1 pixel padding):



| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 105 | 102 | 100 | 97 | 96 |
| 0 | 103 | 99 | 103 | 101 | 102 |
| 0 | 101 | 98 | 104 | 102 | 100 |
| 0 | 99 | 101 | 106 | 104 | 99 |
| 0 | 104 | 104 | 104 | 100 | 98 |

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

| | | |
|---|---|---|
| 210 | 89 | 111 |

$$0*0+105*-1+102*0$$
$$+0*-1+103*5+99*-1$$
$$+0*0+101*-1+98*0 \quad =210$$

Why Convolution
Convolution can achieve something, that the previous two methods of manipulating images can’t achieve. Those include the blurring, sharpening, edge detection, noise reduction e.t.c.

## List of common kernels

https://en.wikipedia.org/wiki/Kernel_(image_processing)#Details

Reference & Images of this chapter
http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html# JPEG Compression

JPEG - Joint Photography Experts Group

## Image compression

Image compression is the method of data compression on digital images. Purpose is to store and transmit data efficiently.

JPEG compression is one of the first methods used in image compression. It could be lossy as well as lossless .

## Working Process

Firstly, divide an image into blocks of 8x8 dimensions.

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

We will subtract 128 from each pixel.

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

We will compute using this formula.

$$G_{u,v} = \alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

$$\alpha_p(n) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } n = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases}$$

The result comes from this is stored in let's say A(j,k) matrix.

There is a standard matrix that is used for computing JPEG compression, which is given by a matrix called as Luminance matrix.

**Luminance matrix**

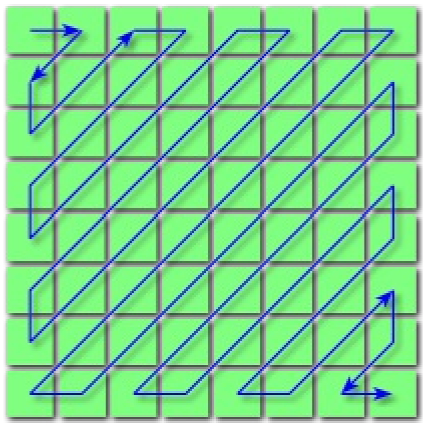
$$Q_{j,k} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Then, we will apply this formula.

$$B_{j,k} = \text{round}\left(\frac{A_{j,k}}{Q_{j,k}}\right)$$

## Entropy coding

Entropy coding is a special form of lossless data compression. It involves arranging the image components in a "zigzag" order employing run-length encoding (RLE) algorithm that groups similar frequencies together, inserting length coding zeros, and then using Huffman coding on what is left.



Zigzag ordering of JPEG image components

−26

−3 0

−3 −2 −6

2 −4 1 −3

1 1 5 1 2

−1 1 −1 2 0 0

0 0 0 −1 −1 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0

0 0 0 0

0 0 0

0 0

0

This chapter takes reference from https://en.wikipedia.org/wiki/JPEG#JPEG_compression.# Concept of Blurring

An image with a face, looks clear when we are able to identify eyes, ears, nose, lips, forehead e.t.c very clear. This shape of an object is due to its edges. So in blurring, we simple reduce the edge content and makes the transition form one color to the other very smooth.

## Blurring vs Zooming

In zooming, as the zooming factor increase, the blurring will also occurs. This is due to increasing the pixel of the zoomed part, but in blurring, the number of pixels in blurred image and normal image remains the same.

## How to Blur

Blurring can be achieved in many ways. There are three common ways to perform blurring.

- Mean filter
- Weighted average filter
- Gaussian filter

## Mean Filter

Mean Filter is also known as box blurring and average blurring.

A mean filter has followig properties.

1. It is odd order.
2. Sum of all elements must be 1.
3. All the elements must be same value.

3x3 kernel will not blur. As soon as increase the nxn mask, blurring will increase.

## Weighted Average Filter

We will give more weight to center.

Following properties must be in WAF.

- It must be odd ordered
- The sum of all the elements should be 1
- The weight of center element should be more then all of the other elements

1 1 1
1 2 1
1 1 1

But second property is not satisfied. In order to solve this, divide the whole matrix by the total of all elements.
Here, dividing factor is 11.# Concept of Edge Detection

## Edges?

Sudden changes of discontinuities in an image are called as edges. Significant transitions in an image are called as edges.

### Types of edges

- Horizontal edges

- Vertical Edges
- Diagonal Edges

### Why we detect edges?

Most of the shape information of an image is enclosed in edges.

Then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

### Some of the masks for edge detection

- Prewitt Operator
- Sobel Operator
- Robinson Compass Masks
- Krisch Compass Masks
- Laplacian Operator

### Prewitt Operator

Prewitt operator is used for detecting edges horizontally and vertically.

### Sobel Operator

The sobel operator is very similar to Prewitt operator. It is also a derivate mask and is used for edge detection. It also calculates edges in both horizontal and vertical direction.

### Robinson Compass Masks

This operator is also known as direction mask. In this operator we take one mask and rotate it in all the 8 compass major directions to calculate edges of each direction.

### Kirsch Compass Masks

Kirsch Compass Mask is also a derivative mask which is used for finding edges. Kirsch mask is also used for calculating edges in all the directions.

### Laplacian Operator

Laplacian Operator is also a derivative operator which is used to find edges in an image. Laplacian is a second order derivative mask. It can be further divided into positive laplacian and negative laplacian.

All these masks find edges. Some find horizontally and vertically,some find in one direction only and some find in all the directions. The next concept that comes after this is sharpening which can be done once the edges are extracted from the image.

---

## Sharpening

Sharpening is opposite to the blurring. In blurring, we reduce the edge content and in Sharpening, we increase the edge content. So in order to increase the edge content in an image, we have to find edges first.

Reference - https://www.tutorialspoint.com/dip/concept_of_edge_detection.htm

# Prewitt Operator

Prewitt Operator is used for image processing.

It detects two types of edges : horizontal & vertical.

## Process

Edges are calculated by using difference between corresponding pixel intensities of an image. Since images are signals, changes in signals or pixels can be calculated by differentiation. Thus, these operators are called Dertivative Operator.

All derative operators follow 3 properties.

1. Opposite sign should be present in the mask.
2. Sum of mask should be equal to zero.
3. More weight means more edge detection.

Prewitt operator provides us two masks one for detecting edges in horizontal direction and another for detecting edges in an vertical direction.

### Vertical edge detecting kernels

-1 0 1
-1 0 1
-1 0 1

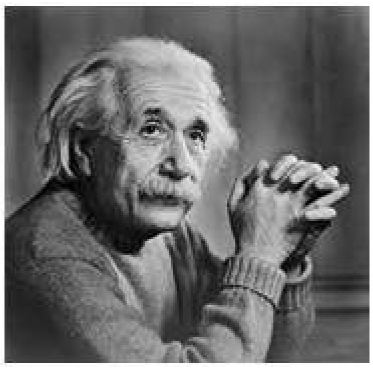
As the center column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge.

### Horizontal edge detecting kernels

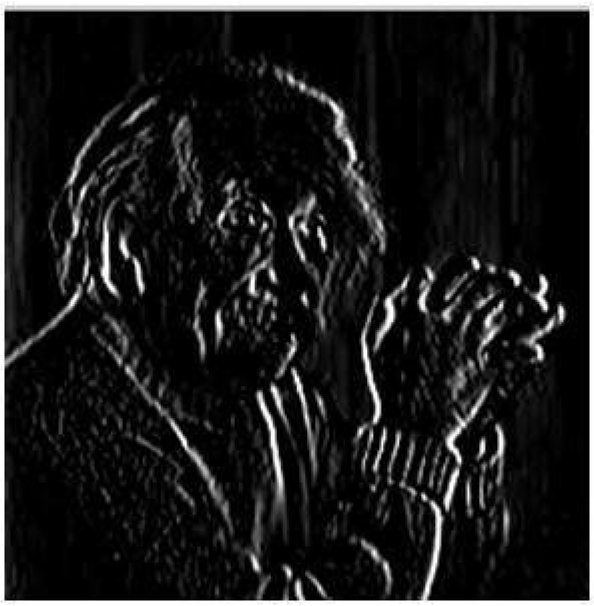
-1 -1 -1
0 0 0
1 1 1

It works as the same principle with vertical.

When apply,



### Vertical Edge Detection



### Horizontal Edge Detection

By comparing two images, we can get edges.# Sobel Operator

It is very similar to Prewitt Operator.

In sobel operator the coefficients of masks are not fixed(same) and they can be adjusted according to our requirement unless they do not violate any property of derivative masks.

## Vertical Edge Kernel

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

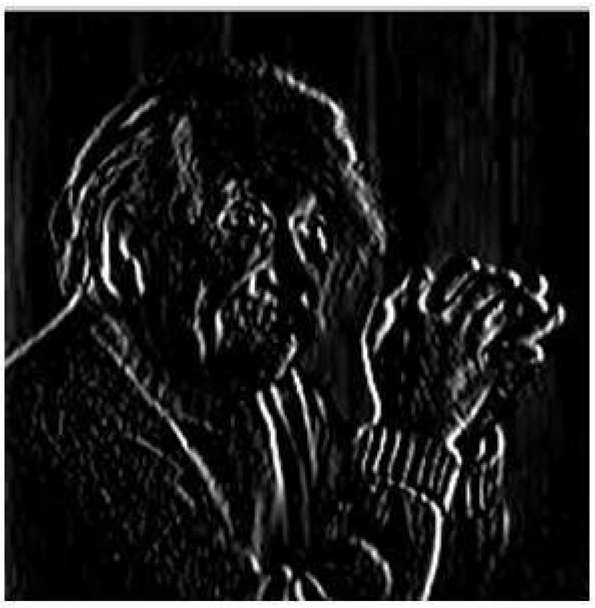
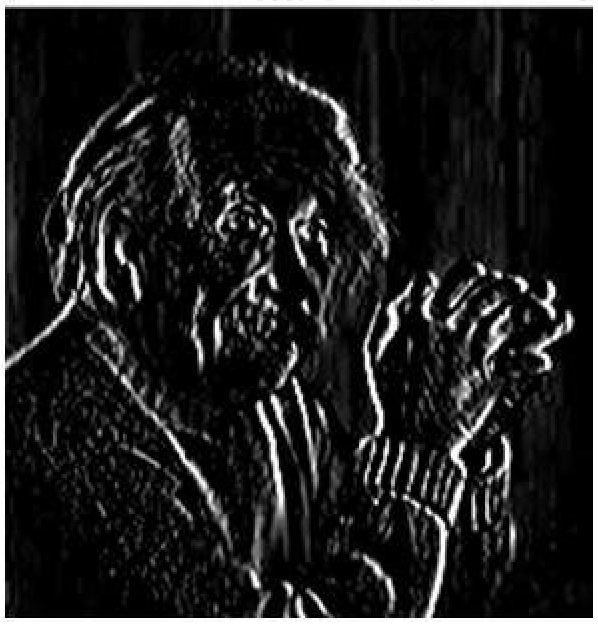
This mask works exactly same as the Prewitt operator vertical mask. There is only one difference that is it has “2” and “-2” values in center of first and third column. When applied on an image this mask will highlight the vertical edges.

This give more weight age to the pixel values around the edge region. This increase the edge intensity and it become enhanced comparatively to the original image.

## Horizontal Mask of Sobel Operator

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

## Comparison between Prewitt & Sobel Operators

First image is Prewitt Vertical & second is Sobel Vertical.

Sobel operator finds more edges or make edges more visible as compared to Prewitt Operator.
This is because in sobel operator we have allotted more weight to the pixel intensities around the edges.

### Adding More Weights

-1 0 1
-5 0 5
-1 0 1

Applying more weight will give more edges.# RobinSon Compass Mask

As the name, it is an edge detector that can work in all major 8 directions in compass.

- North
- North West
- West
- South West
- South
- South East
- East
- North East

## Process

Take one Sobel or Prewitt Operator. Rotate the Mask.

### North Direction

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

### North West Direction

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

### West Direction

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

This is Horizontal edge mask.

$$\begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\-1 & -1 & -1\end{bmatrix}\overset{\uparrow}{N}\quad\begin{bmatrix}1 & 1 & 0\\1 & 0 & -1\\0 & -1 & -1\end{bmatrix}\begin{matrix}N\\W\end{matrix}\quad\begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\-1 & -1 & -1\end{bmatrix}\overset{\leftarrow}{W}\quad\begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix}\begin{matrix}/\\S\\W\end{matrix}$$

$$\begin{bmatrix}-1 & -1 & -1\\0 & 0 & 0\\1 & 1 & 1\end{bmatrix}\overset{\downarrow}{S}\quad\begin{bmatrix}-1 & -1 & 0\\-1 & 0 & 1\\0 & 1 & 1\end{bmatrix}\begin{matrix}S\\E\end{matrix}\quad\begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\-1 & -1 & -1\end{bmatrix}\overset{\rightarrow}{E}\quad\begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix}\begin{matrix}/\\N\\E\end{matrix}$$

Suppose there is an image, which do not have any North East direction edges so then that mask will be ineffective.# Krisch Compass

The only difference between Robinson and kirsch compass masks is that in Kirsch we have a standard mask but in Kirsch we change the mask according to our own requirements.

## North Direction

-3 -3 5
-3 0 5
-3 -3 5

# Laplacian Operator

Unlike other operators, laplacian operator is second order derivative. It can be classified into two types.

- Positive Operator
- Negative Operator

Laplacian does not take out edges in direction. It takes out edges as **inward** and **outward**.

## Positive Laplacian Operator

Center element must be negative and corner elements must be zero.

0 1 0
1 -4 1
0 1 0

This takes outward edges.

## Negative Laplacian Operator

Center must be positive, all corners must be zero and all other elements must be negative.

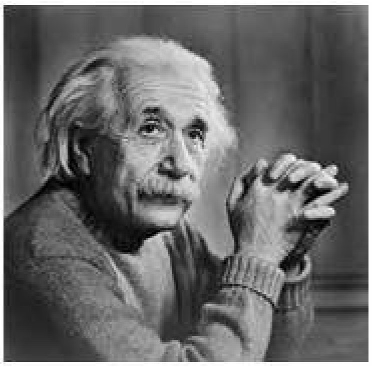
0 -1 0
-1 4 -1

0 -1 0

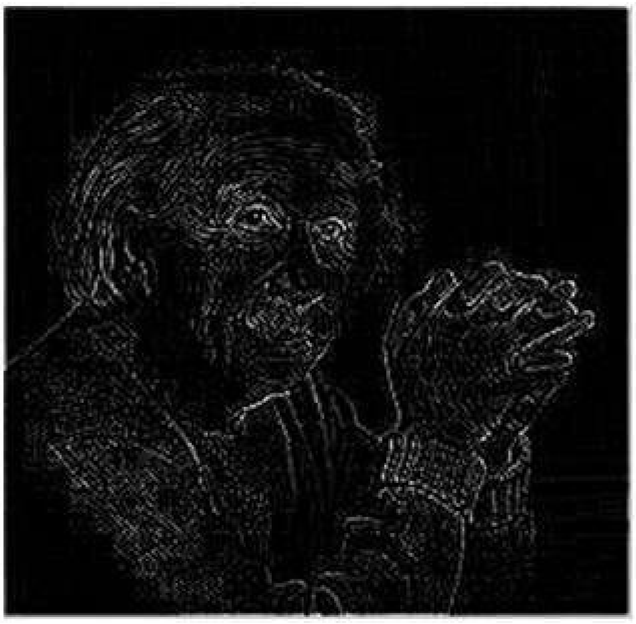
## How it works

Laplacian is a derivative operator; its uses highlight gray level discontinuities in an image and try to deemphasize regions with slowly varying gray levels. This operation in result produces such images which have grayish edge lines and other discontinuities on a dark background. This produces inward and outward edges in an image

The important thing is how to apply these filters onto image. Remember we can’t apply both the positive and negative Laplacian operator on the same image. we have to apply just one but the thing to remember is that if we apply positive Laplacian operator on the image then we subtract the resultant image from the original image to get the sharpened image. Similarly if we apply negative Laplacian operator then we have to add the resultant image onto original image to get the sharpened image.



**Positive Laplacian Op**



**Negative Laplacian Op**

Reference : https://www.tutorialspoint.com/dip/laplacian_operator.htm# Frequency Domain Analysis

We have manipulate in many ways.

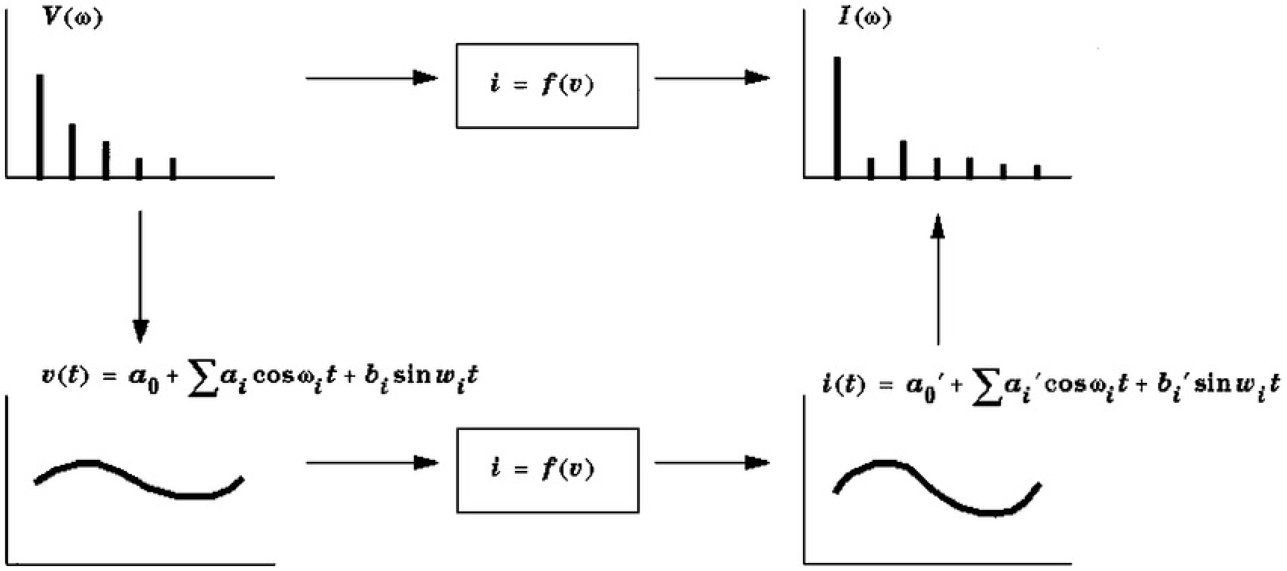
- Transformation
- Histrogram
- Convolution

Now we will see Frequency Domain Analysis.

In frequency domain analysis, we analyze signals with respect to frequency.

## How?

First, we transform an image into its frequency distribution. It will be processed into a transformation. After performing inverse transformation, it will back into the normal image.



$V(\omega)$
$i = f(v)$
$I(\omega)$
$v(t) = a_0 + \sum a_i \cos\omega_i t + b_i \sin w_i t$
$i = f(v)$
$i(t) = a_0' + \sum a_i' \cos\omega_i t + b_i' \sin w_i t$

## Transform

A signal can be converted between the time and frequency domains with a pair of mathematical operators called a transform.

### Some common transfroms

- Fourier Series
- Fourier transformation
- Laplace transform

- Z transform

### High frequency components

High frequency components correspond to edges in an image.

### Low frequency components

Low frequency components in an image correspond to smooth regions.# Fourier Series & Transform

## Fourier

Fourier was a mathematician in 1822. He give Fourier series and Fourier transform to convert a signal into frequency domain.

## Fourier Series

Fourier series simply states that, periodic signals can be represented into sum of sines and cosines when multiplied with a certain weight. It further states that periodic signals can be broken down into further signals with the following properties.

The signals are sines and cosines
The signals are harmonics of each other

## How?

To perform image in frequency domain, we must first convert it into frequency domain and takes inverse of output to convert it back into spatial domain.

### formula

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)}\, dx\, d$$

### Inverse formula

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)}\, du\, dv.$$

## Fourier transform

The Fourier transform simply states that that 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.

The Fourier transform has many wide applications that include, image compression (e.g JPEG compression), filtering and image analysis.

## Difference between Fourier series and transform

Although both Fourier series and Fourier transform are given by Fourier , but the difference between them is Fourier series is applied on periodic signals and Fourier transform is applied for non periodic signals.

## Which one is applied on images?

Now the question is that which one is applied on the images , the Fourier series or the Fourier transform. Well, the answer to this question lies in the fact that what images are. Images are non – periodic. And since the images are non periodic, so Fourier transform is used to convert them into frequency domain.

# Essential Notes

```
for (var y=0; y<height; y++) {
    for (var x=0; x<width; x++) {
      var dstOff = (y*width+x)*4;
    }
}
```

Leave 4 pixels every x increses.For sample codes;

https://www.html5rocks.com/en/tutorials/canvas/imagefilters/

---

For javascript image processing library based on this book;

https://github.com/MinSiThu/agamotto-eye/

---

## References

https://www.tutorialspoint.com/dip

https://en.wikipedia.org/wiki/Kernel_(image_processing)#Details

http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html