2019 MMILAB.DIP Seminar Week1(12/31~1/5)

Seong Su Kim

Contents

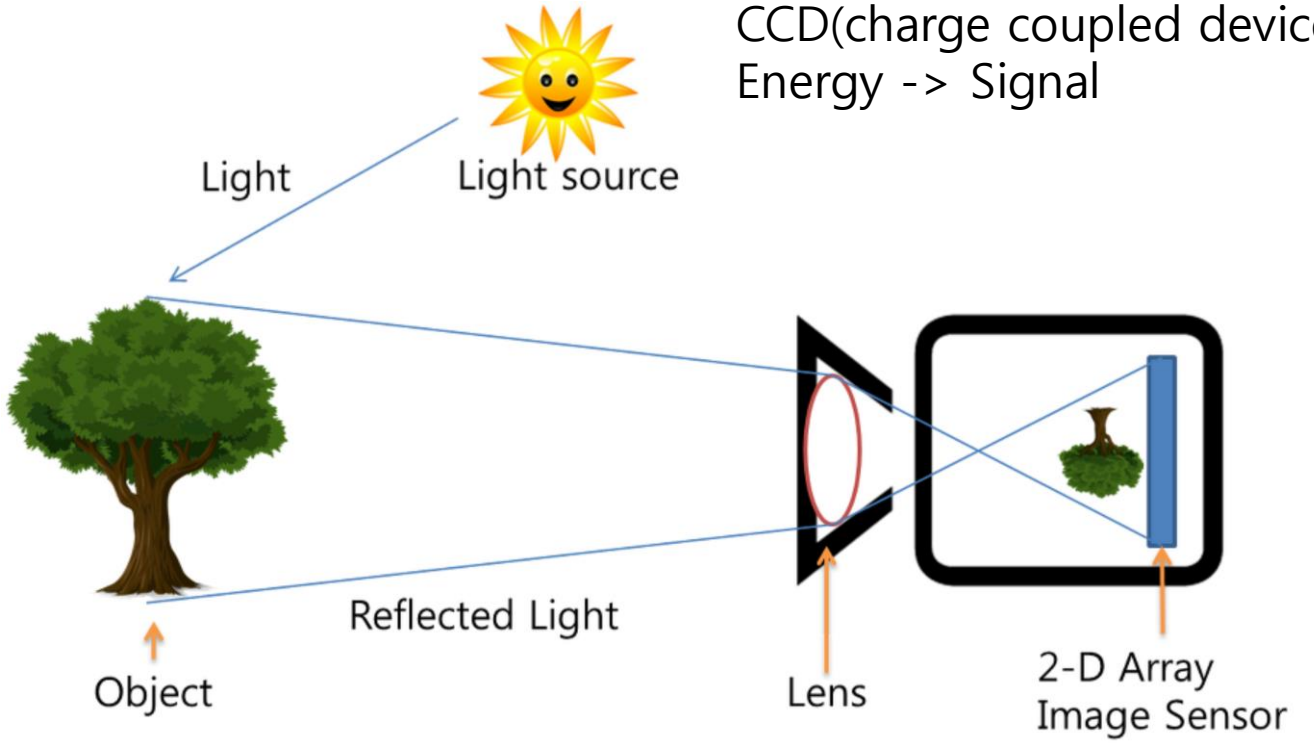
- Digital Image

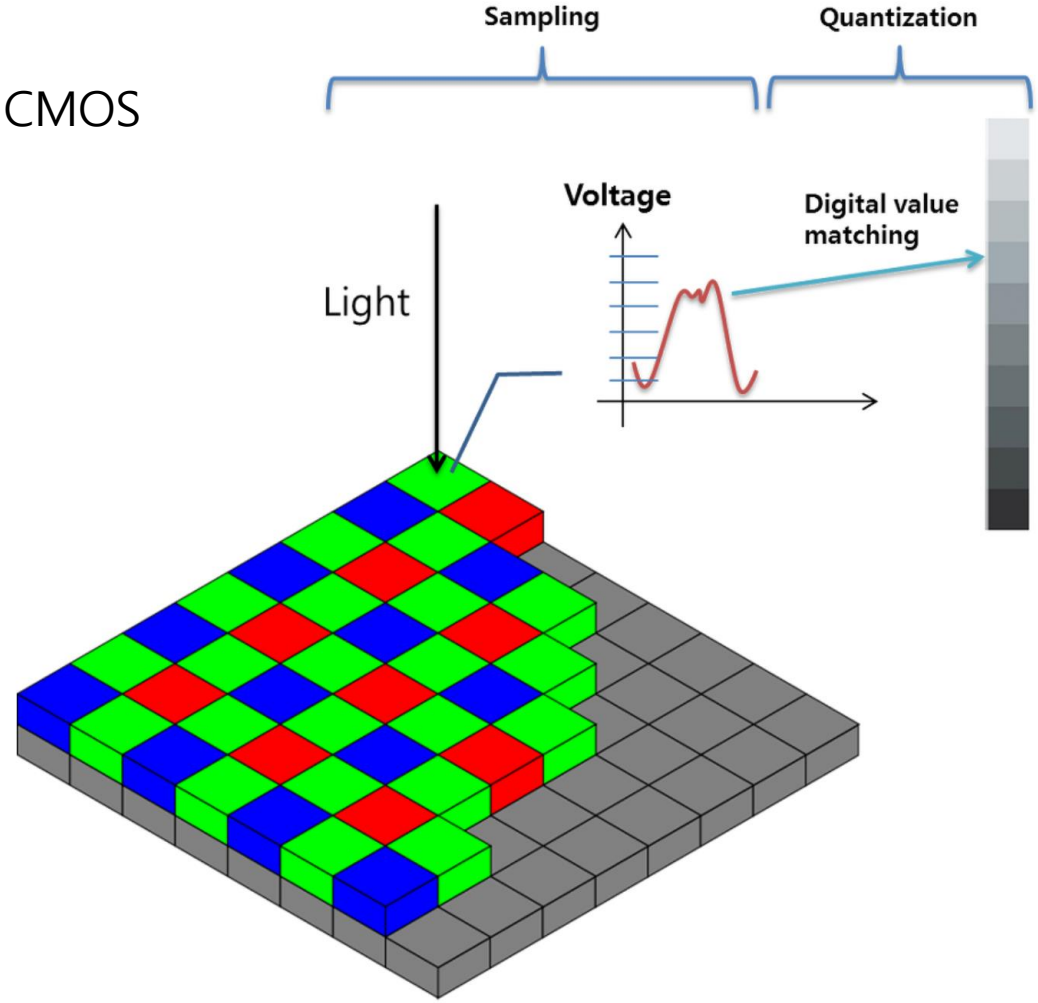
- Intensity Transform
  -negative, log, gamma, contrast stretching, bit slicing
- Affine Transform

- Color Transform
  - Reason of various Color model, RGB, Ycbcr, CMY, HSI

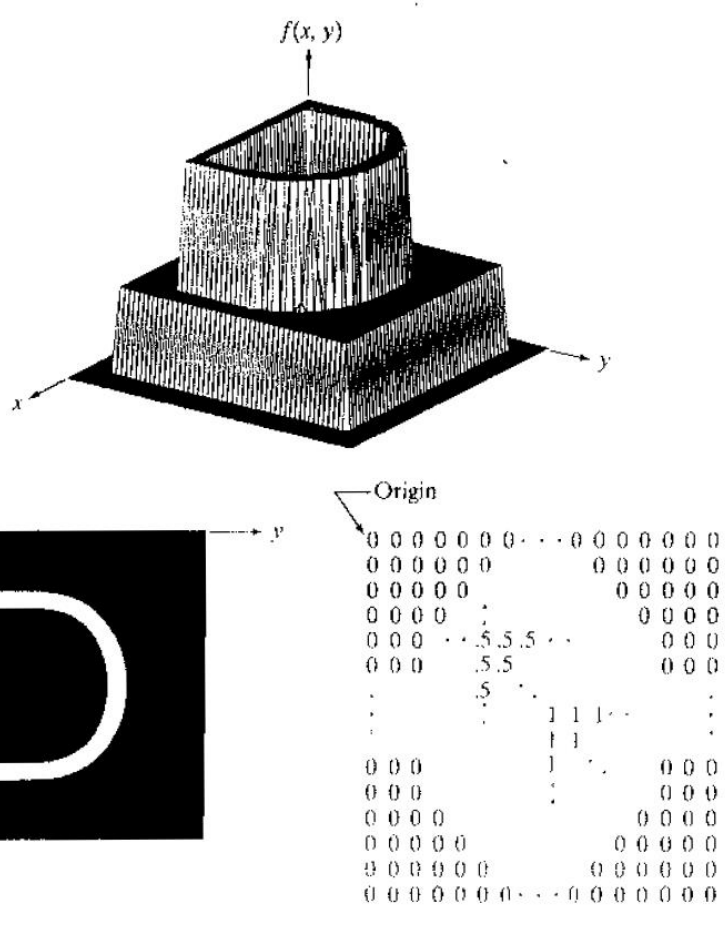
Digital Image

CCD(charge coupled device), CMOS
Energy -> Signal





Digital Image



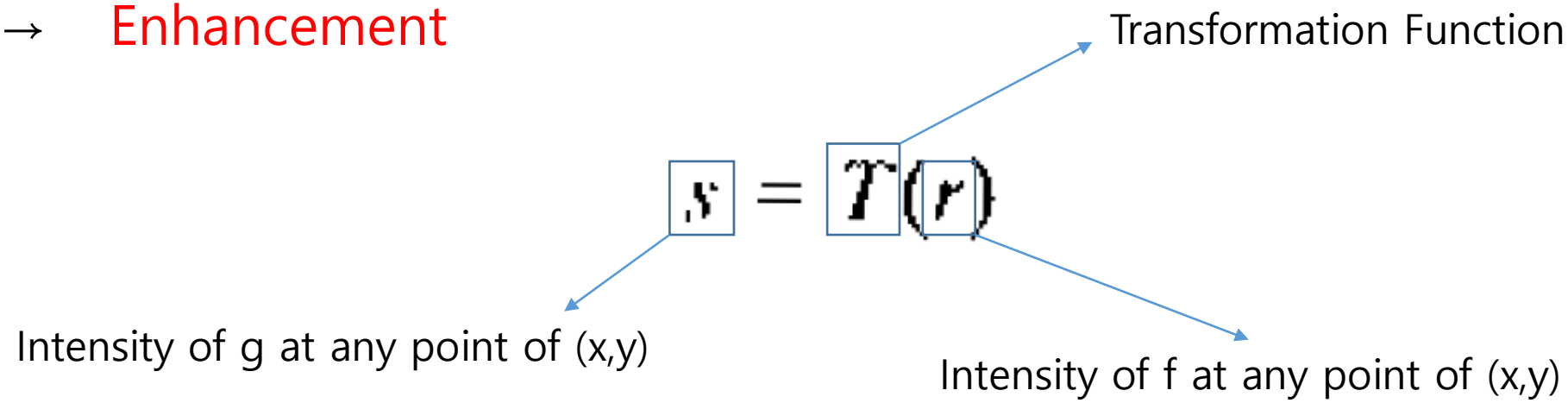
$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0, N-1) \\ f(1,0) & f(1,1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1, N-1) \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}$$

Purpose of Intensity transformation

"Process of manipulating an image so that the result is more suitable than the original for a specific application"

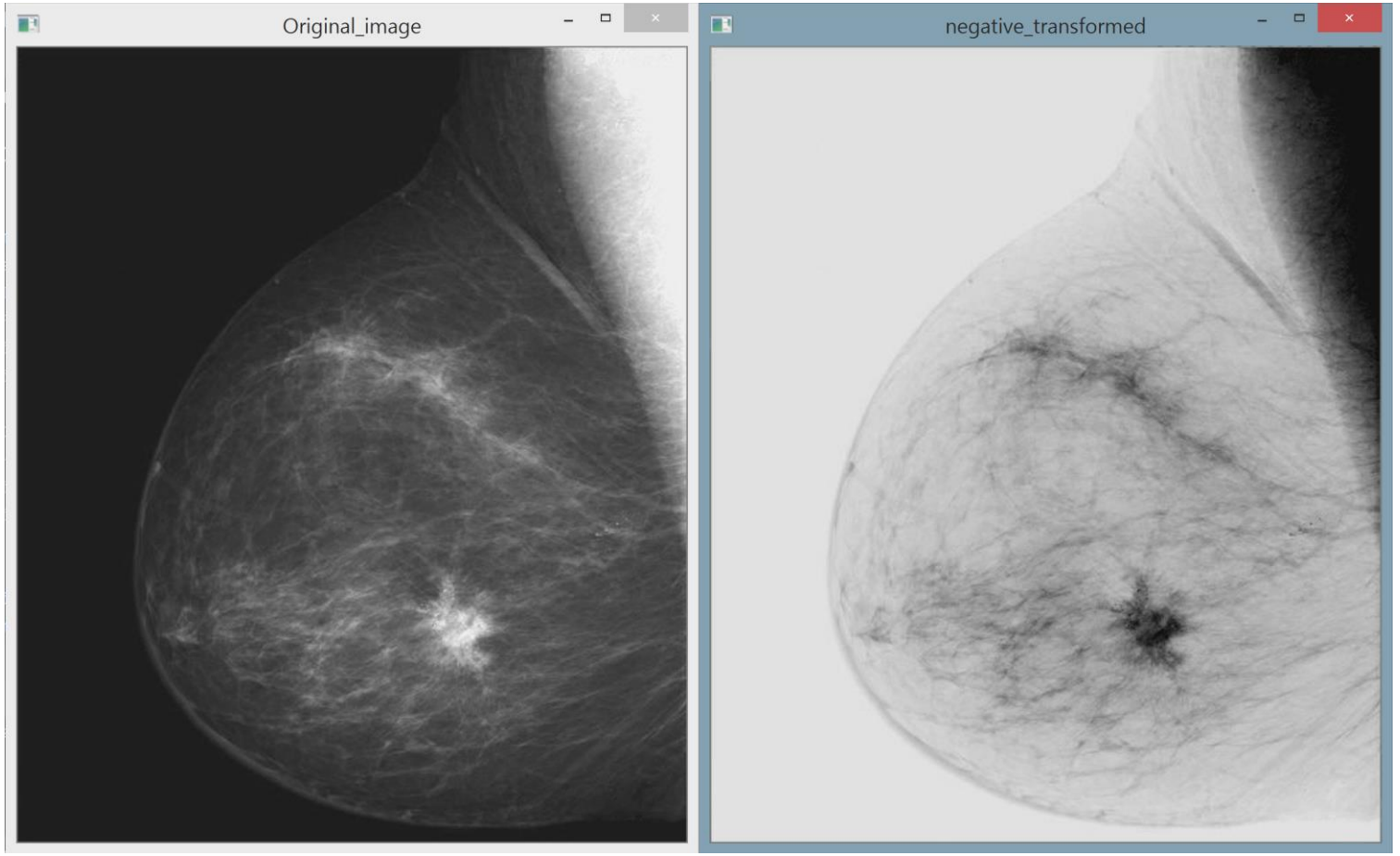
→ Enhancement



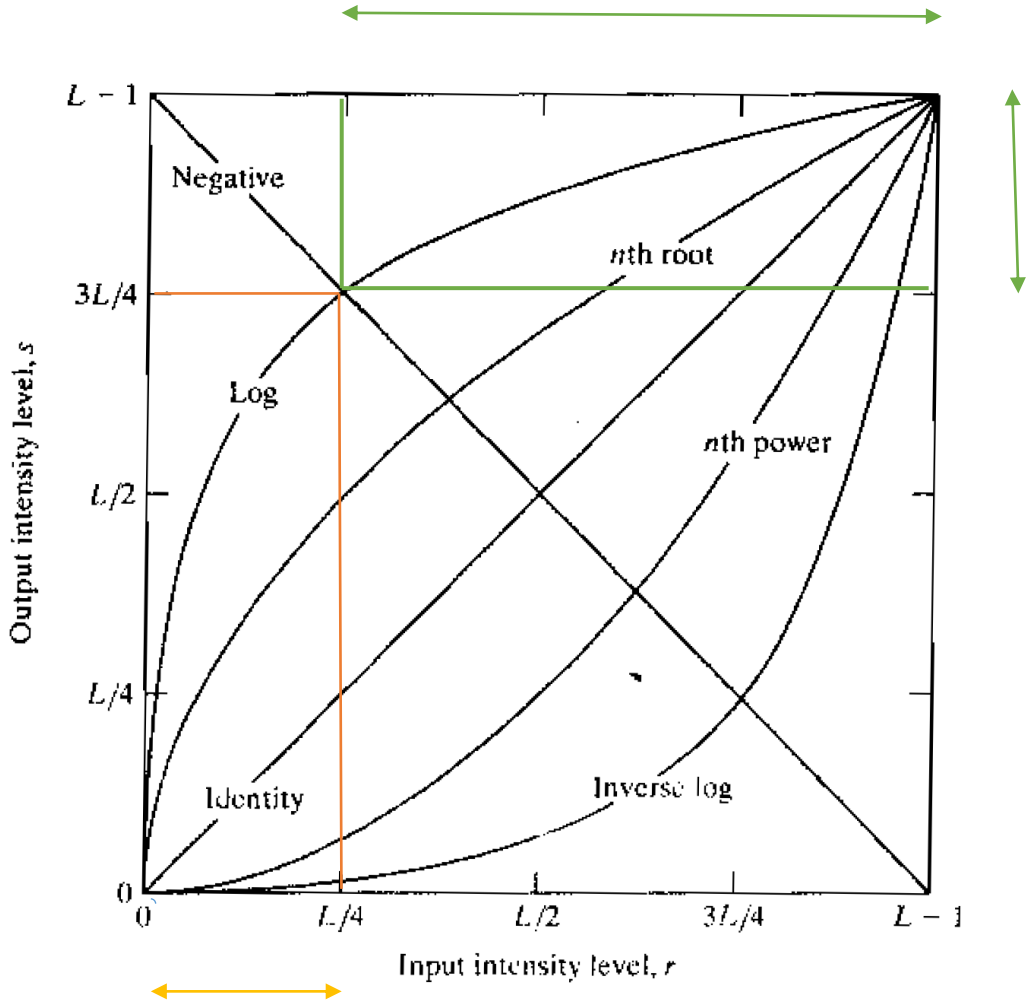
Intensity Transform – Negative(Grayscale)

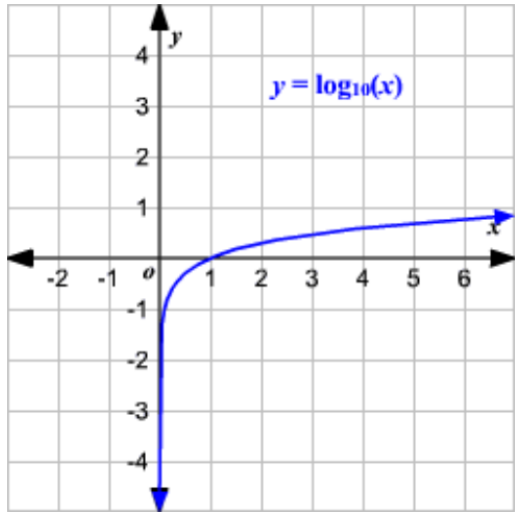
$$s = L - 1 - r$$

For Image Intensity range[0,L-1]
Max intensity



Intensity Transform- Log transformation





$s = clog(1 + r)$

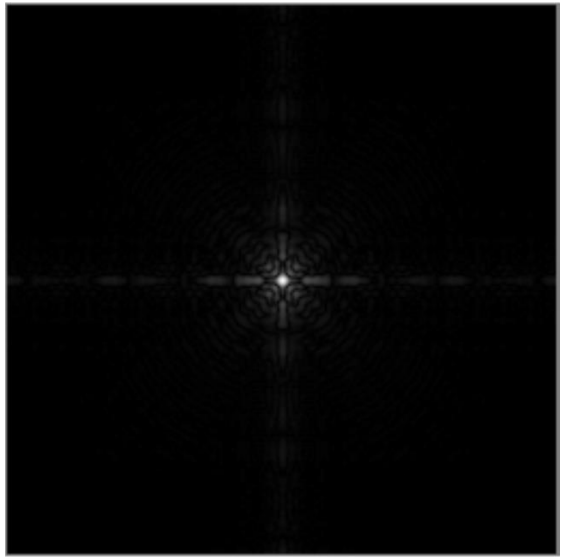
Meaning?

: Mapping a narrow range of low intensity values of input into a wider range of output values

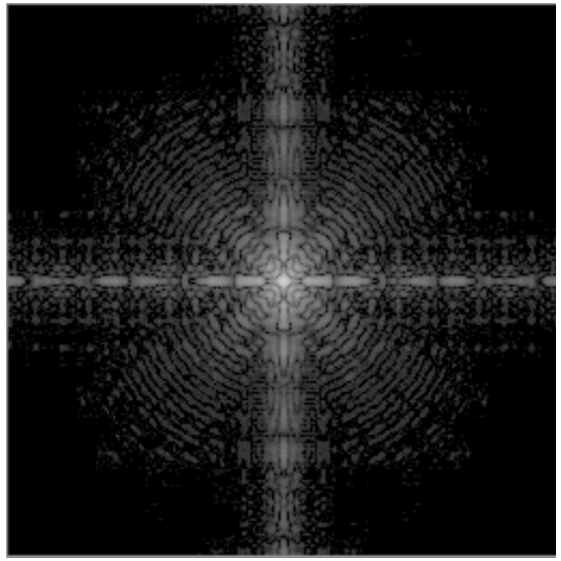
Mapping a wider range of high intensity values of input into a narrow range of output values

$c = (L - 1)/1og(L)$

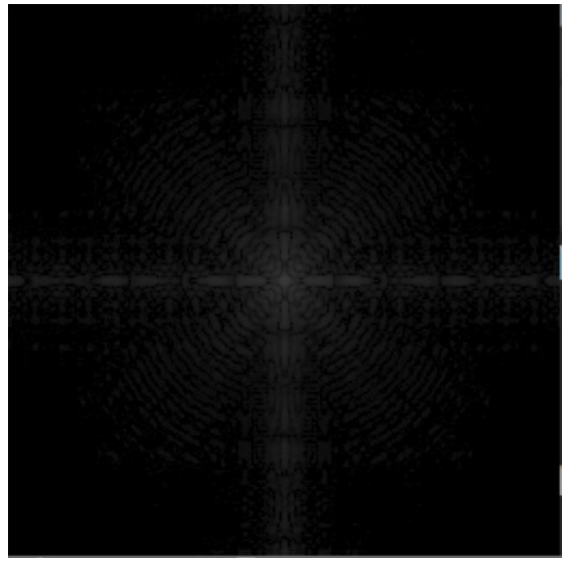
Intensity Transform –Log Transformation



Original



Log base=3

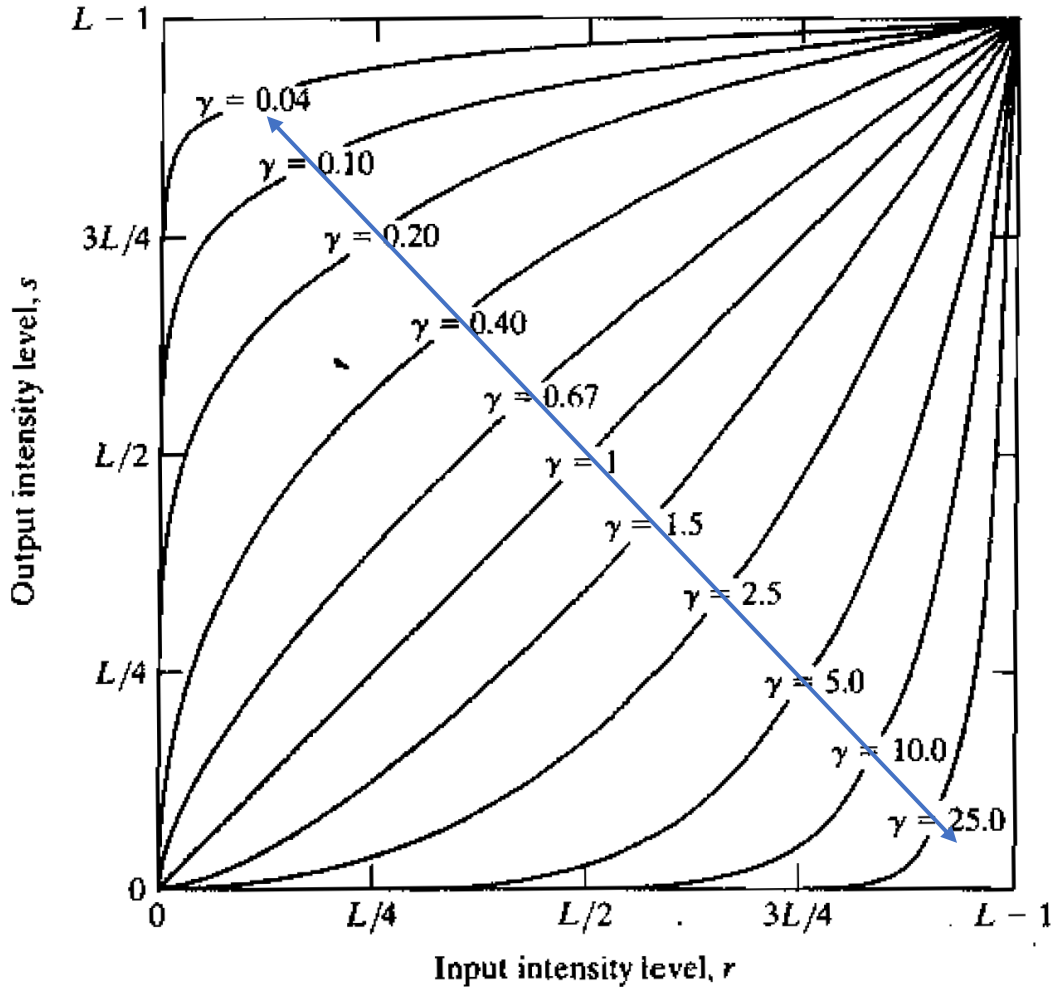


Log base=10

Why useful?

- Can show more Details
- Compress the dynamic range of images with large variation in pixel value

Intensity Transform – Gamma Transformation

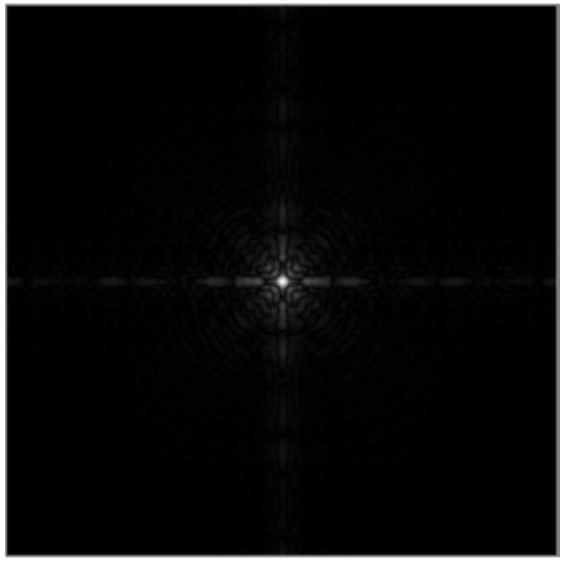


$$s = cr^{\Upsilon}(c = 1)$$

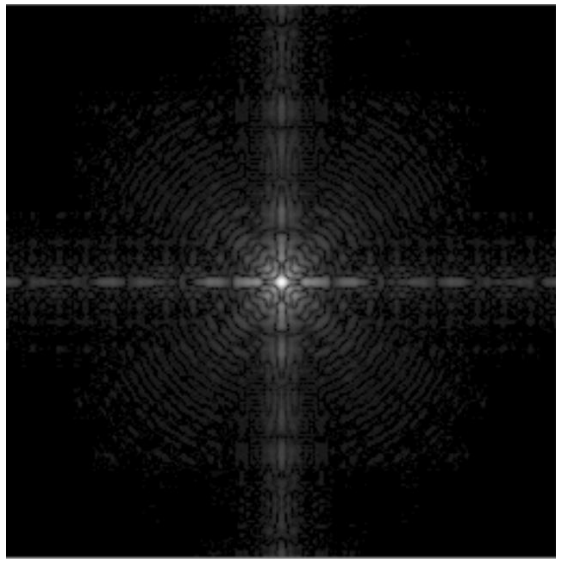
Meaning?

- same characteristic with Log transformation
- But unlike log transformation by just changing Υ value we can make more efficient characteristic of log and inverse log transformation.

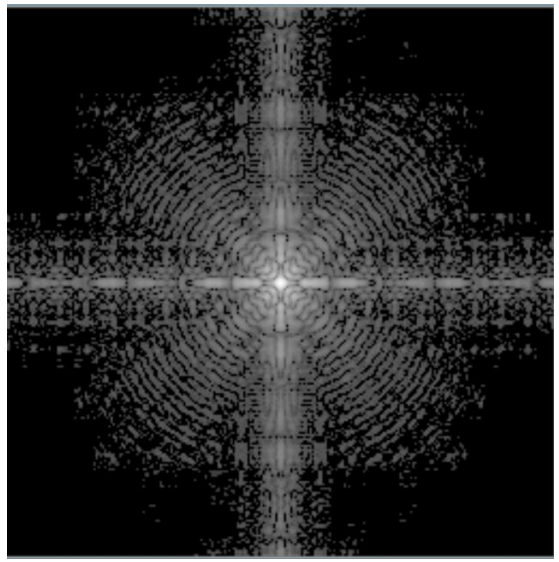
Intensity Transform –Gamma Transformation



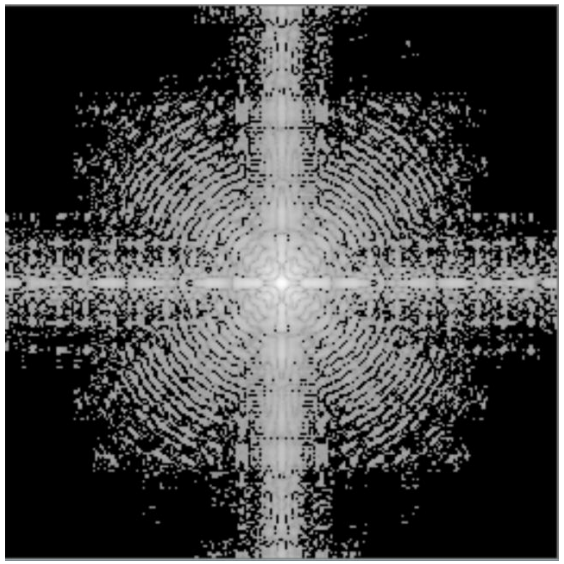
$c = 1, \Upsilon = 1$ (Original)



$c = 1, \Upsilon = 0.5$



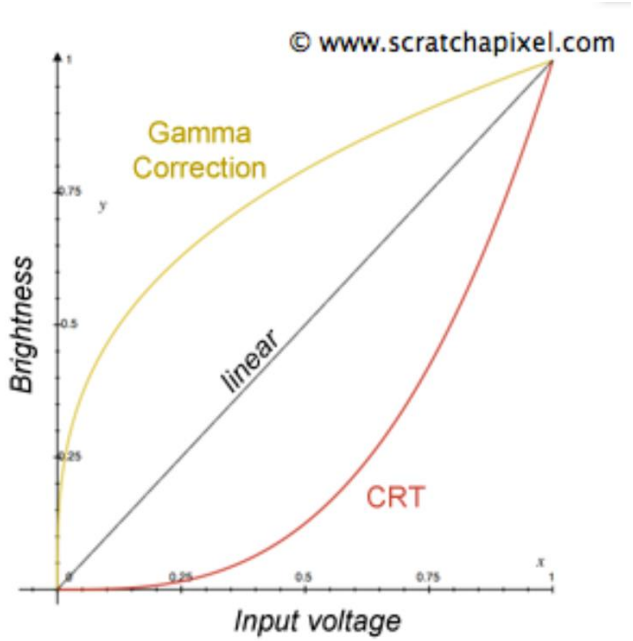
$c = 1, \Upsilon = 0.25$



$c = 1, \Upsilon = 0.1$

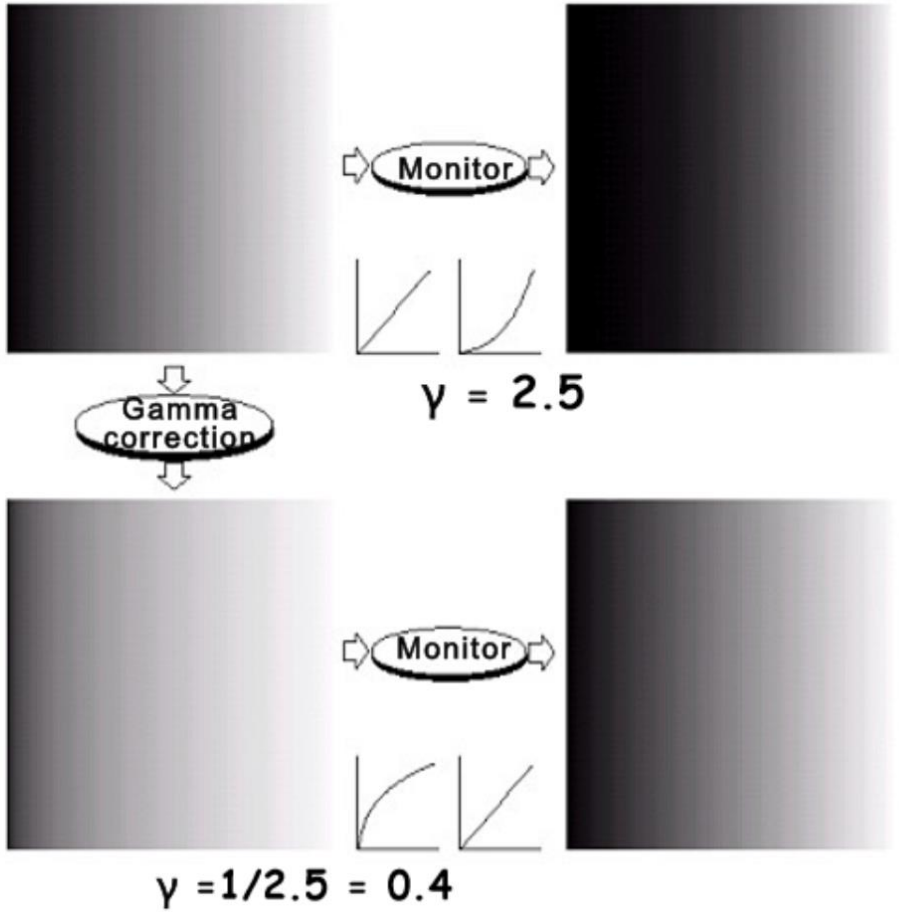
Intensity Transform –Gamma correction

Eg) CRT monitor



$$brightness = image^{\frac{1}{\gamma}} \cdot voltage^{\gamma} \rightarrow \text{linear curve}$$

Gamma Correction



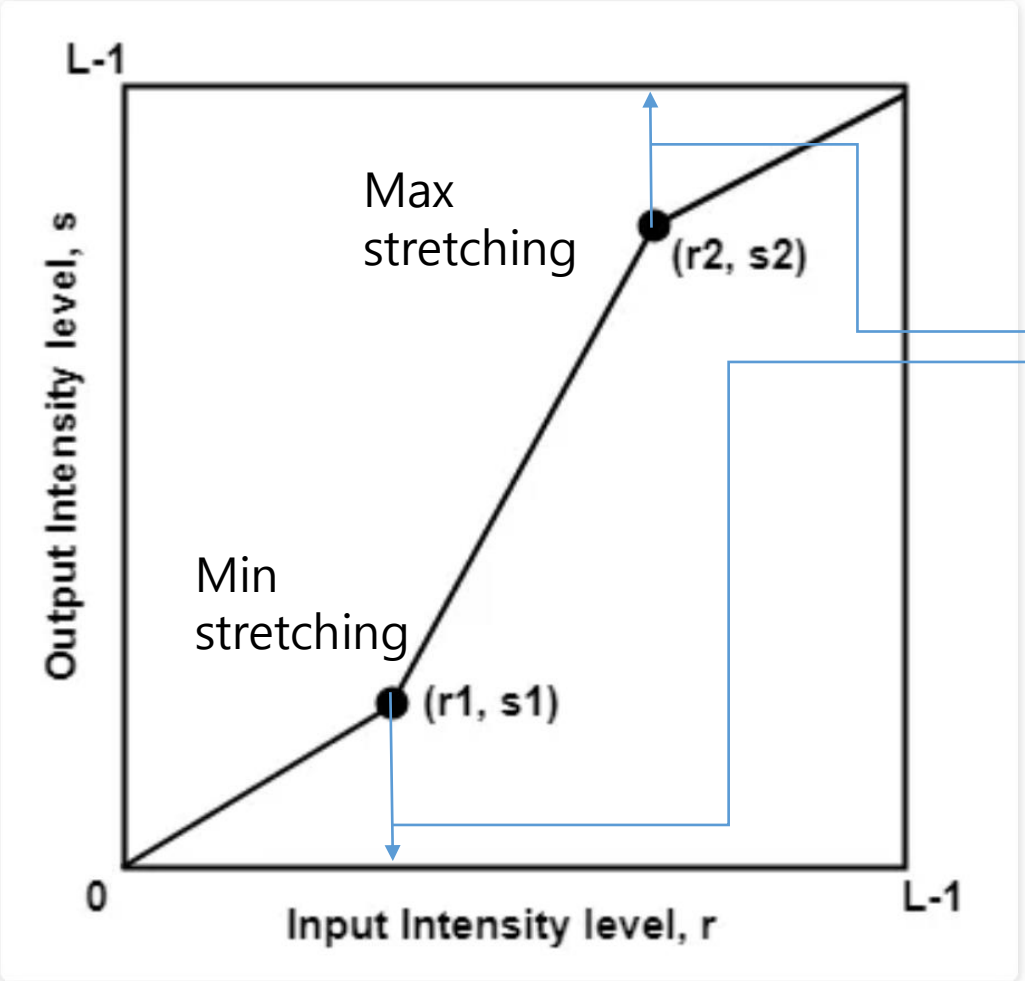
Intensity Transform –Gamma Transformation

Eg) MRI,X-ray

$c = 1, \Upsilon = 1$ (Original)

$c = 1, \Upsilon = 5$

Intensity Transform –Contrast Stretching



When r1 =s1 and r2=s2,
transformation becomes a **Linear function**.

When (r1, s1) = (r_{min}, 0) and (r2, s2) = (r_{max}, L-1),
this is known as **Min-Max Stretching.**

Using numpy.linspace (Piecewise Linear Function)

numpy.**linspace**(*start*, *stop*, *num=50*, *endpoint=True*, *retstep=False*, *dtype=None*, *axis=0*)

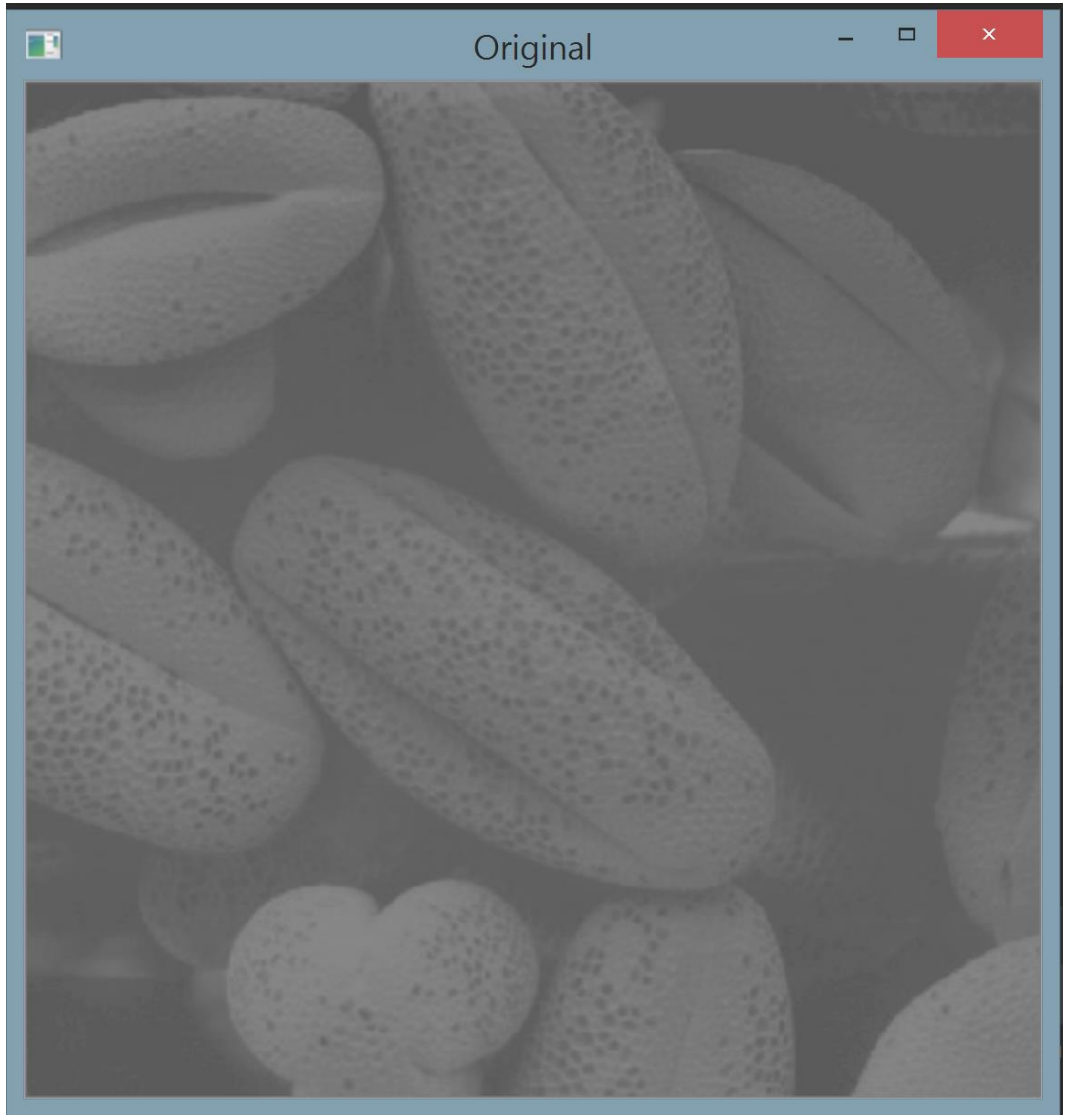
Return evenly spaced numbers over a specified interval.

Returns *num* evenly spaced samples, calculated over the interval [*start*, *stop*].

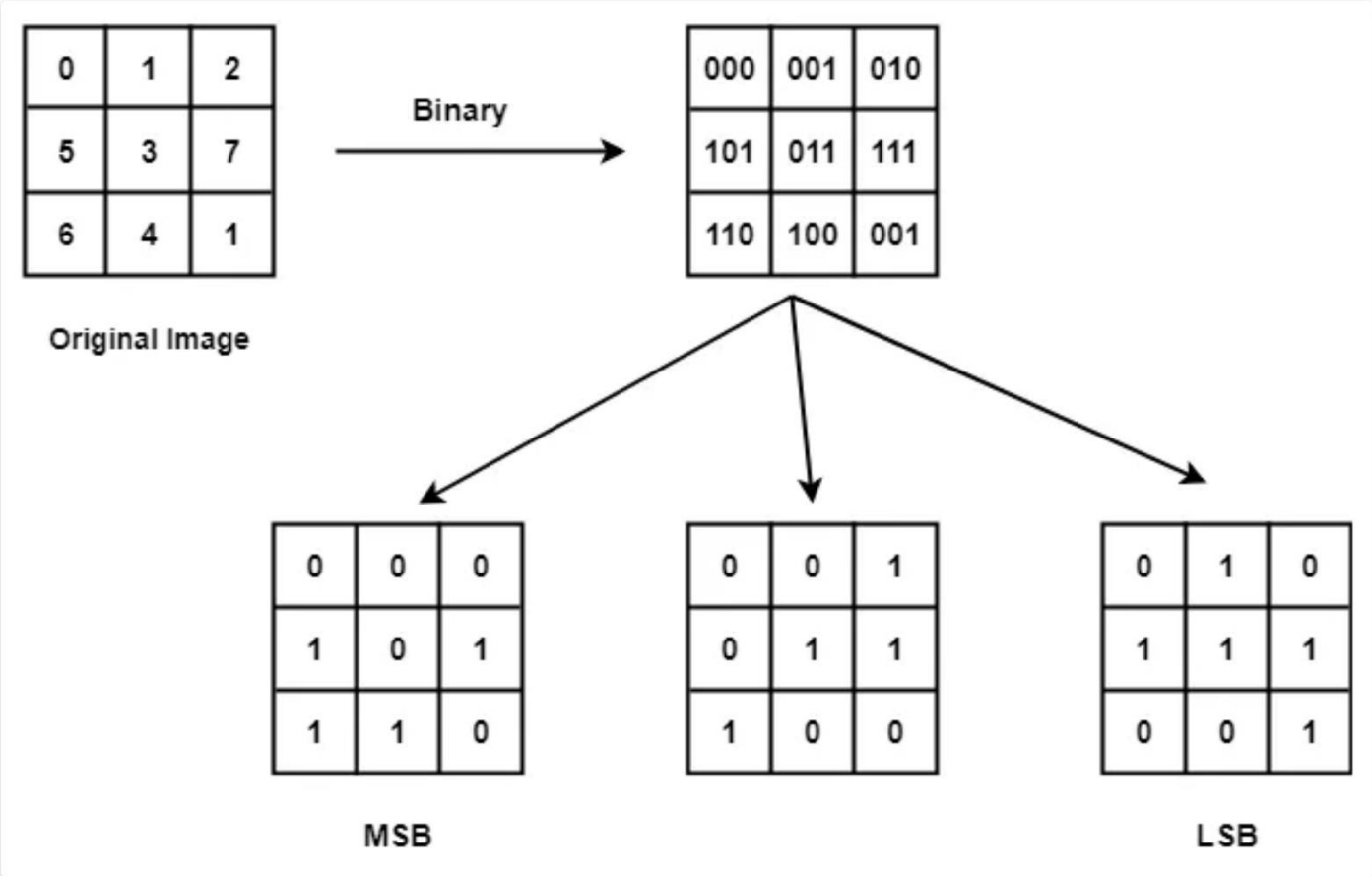
The endpoint of the interval can optionally be excluded.

```
def ContrastStretching(img):
    original = np.array(img)
    min = np.min(original)
    max = np.max(original)
    PiecewiseLinear = np.zeros(256, dtype=np.uint8)
    PiecewiseLinear[min:max+1] = np.linspace(0, 255, max-min+1,True,dtype=np.uint8)
    result = np.array(PiecewiseLinear[original],dtype=np.uint8)
    return result
```

Intensity Transform –Contrast Stretching

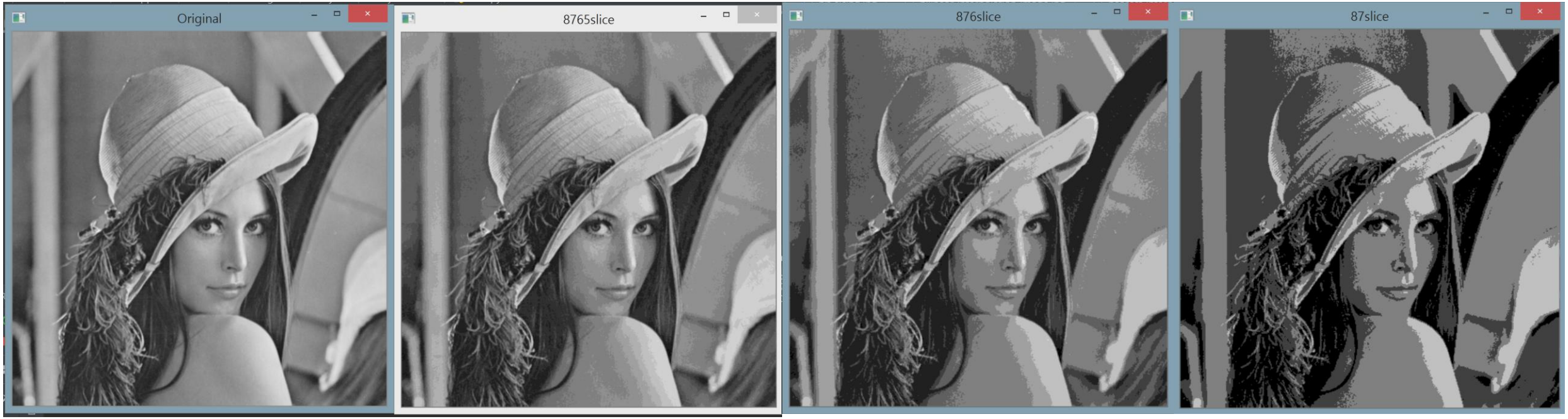


Intensity Transform – Bit Slicing




```
eight_bit_img = (np.array([int(i[0]) for i in lst], dtype=np.uint8) * 128).reshape(img.shape[0], img.shape[1])
seven_bit_img = (np.array([int(i[1]) for i in lst], dtype=np.uint8) * 64).reshape(img.shape[0], img.shape[1])
six_bit_img = (np.array([int(i[2]) for i in lst], dtype=np.uint8) * 32).reshape(img.shape[0], img.shape[1])
five_bit_img = (np.array([int(i[3]) for i in lst], dtype=np.uint8) * 16).reshape(img.shape[0], img.shape[1])
```

Intensity Transform – Bit Slicing



Intensity Transform – Bit Slicing

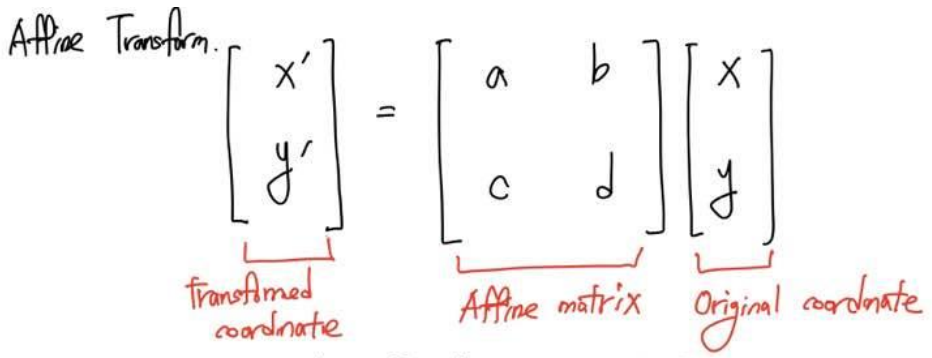
Benefit?

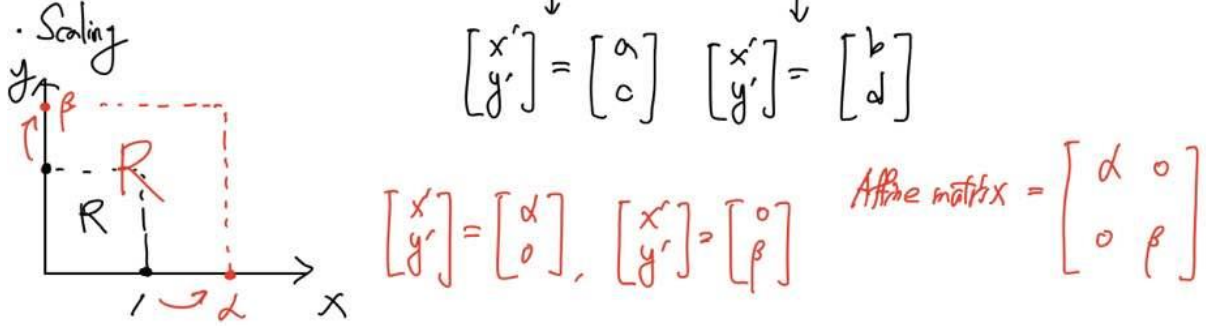
1.Image Compression (Constructing nearly the original image using less number of bits).

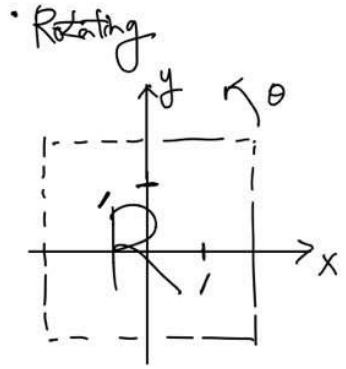
2.Through this, we can analyze the relative importance of each bit in the image that will help in determining the number of bits used to quantize the image.

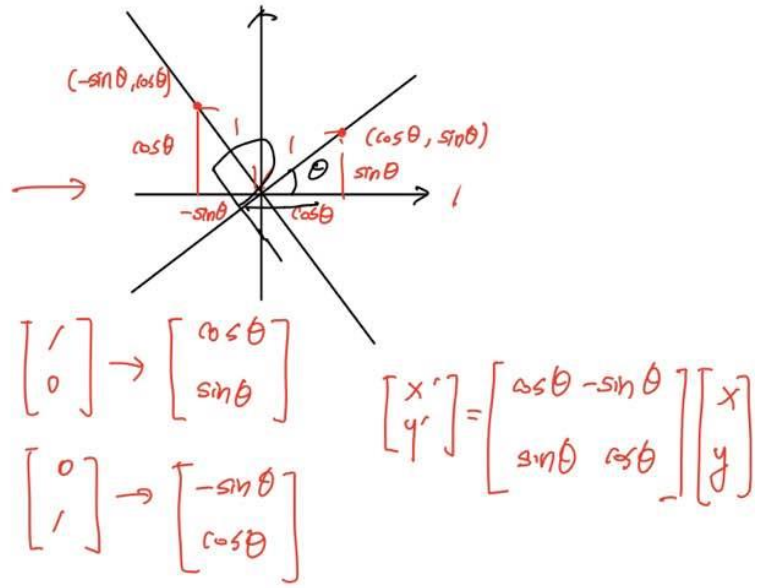
Affine Transform

Meaning: linear mapping method that preserves points, straight lines, and planes (in spatial domain)





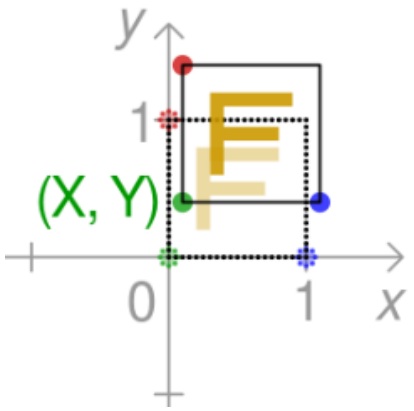




Affine Matrix

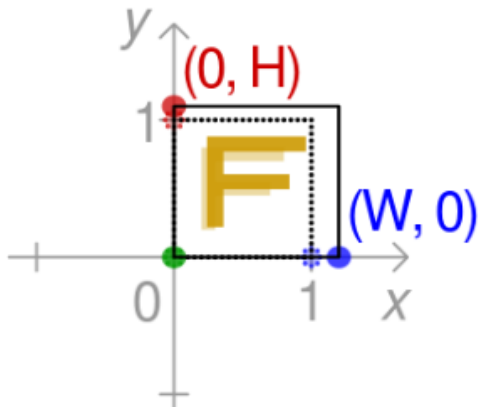
Translate

$$\begin{bmatrix} 1 & 0 & X \\ 0 & 1 & Y \\ 0 & 0 & 1 \end{bmatrix}$$



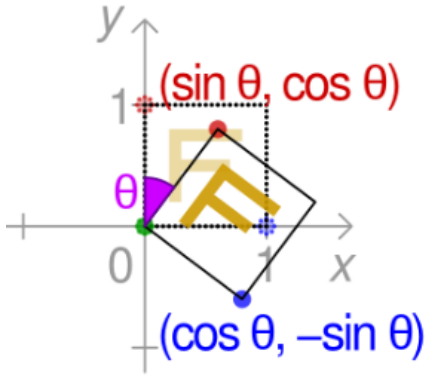
Scale about origin

$$\begin{bmatrix} W & 0 & 0 \\ 0 & H & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



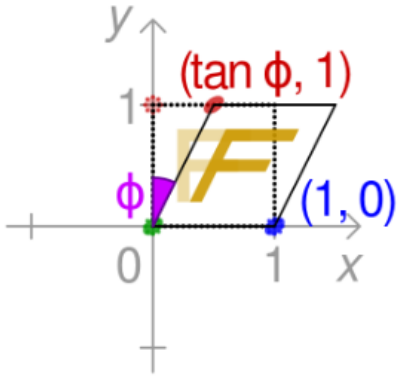
Rotate about origin

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



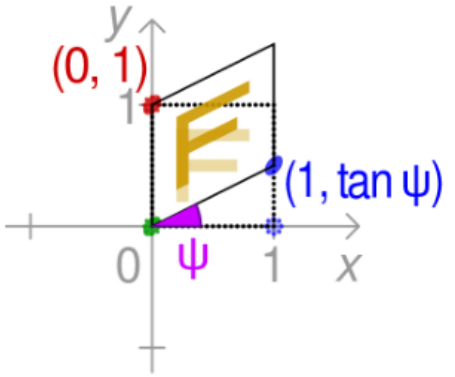
Shear in x direction

$$\begin{bmatrix} 1 & \tan\phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

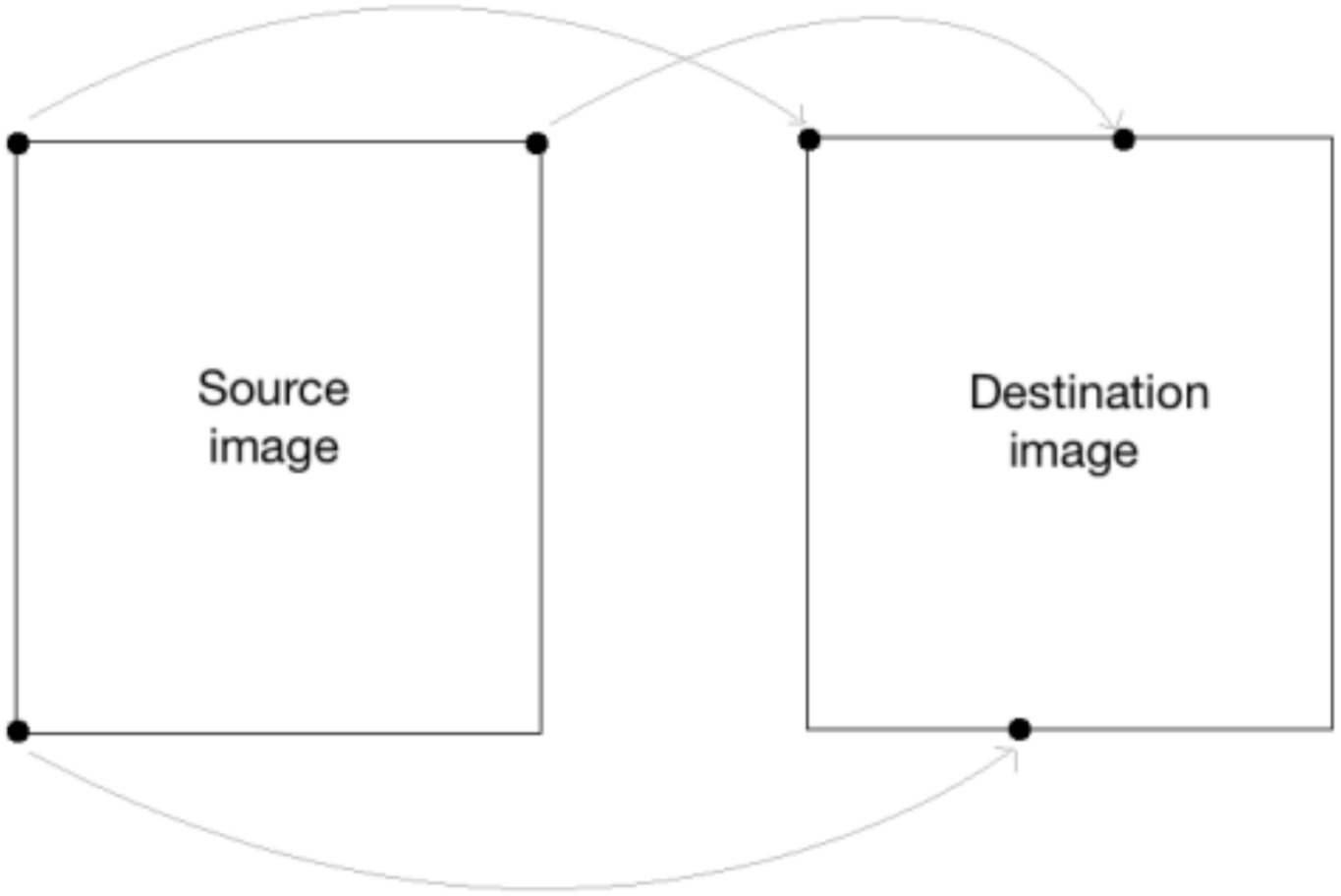


Shear in y direction

$$\begin{bmatrix} 1 & 0 & 0 \\ \tan\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



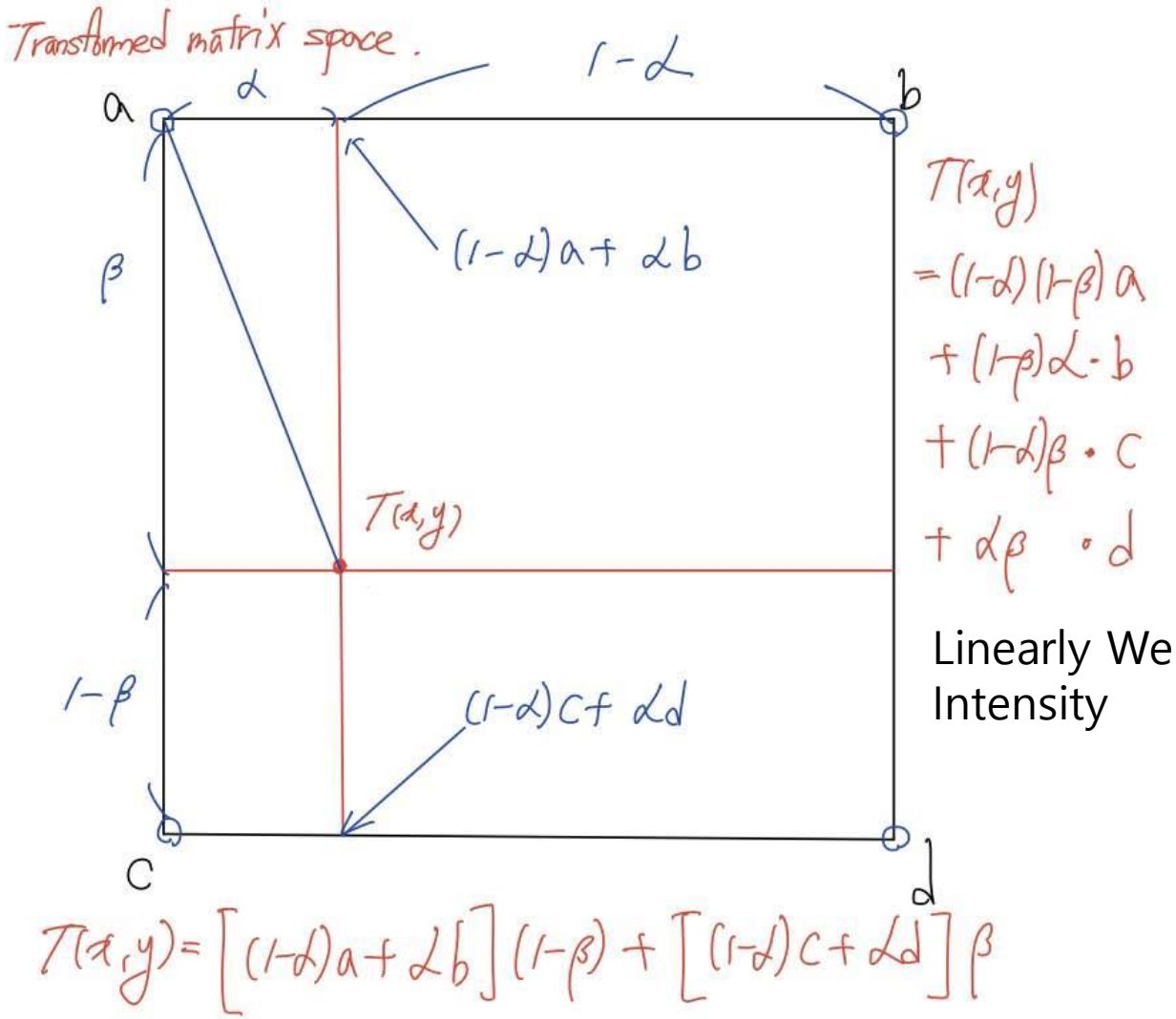
Problem by Discrete Coordination of Matrix



①If not integer
= No matching coordination

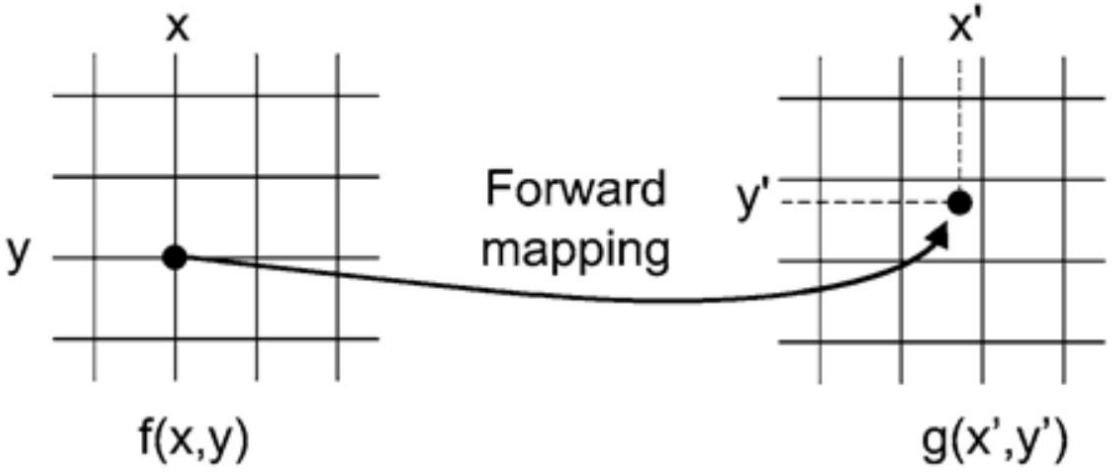
②If there is no one-to-one
=correspondence coordination with Source and Destination matrix.

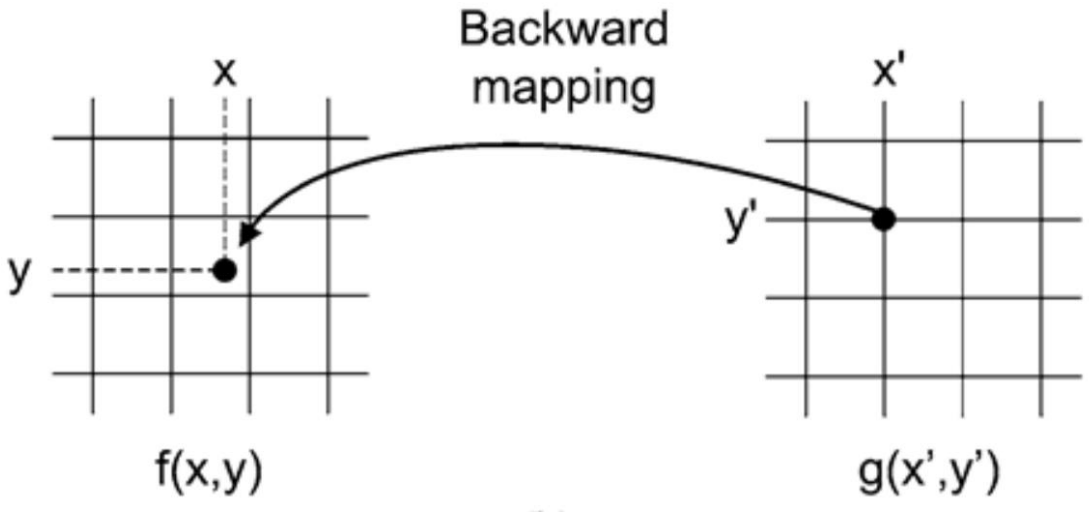
Bilinear Interpolation



Linearly Weighted Sum of 4 nearest pixel Intensity

Mapping direction





Depending on specific spatial transform function, there is Two main problem of Forward mapping

-Gaps: Some output pixels that did not receive any input image pixels.

-Overlaps: Some output pixels can received more than one input image pixel

-Interpolation By using Original pixel's information

-Every destination pixel can get own value.

Interpolation Exception (Code)

```
a=result[1].is_integer()

if a:
    result[2] = Original[y1,x1]
    transformed[y,x] = result[2]

else :
    result[2] = (Original[y2,x1]-Original[y1,x1])*(result[1]-y1)+Original[y1,x1]
    transformed[y,x] = result[2]
```

Exception for Edge(corner) integer coordination

Scaling

Considered in code

1. Corner to Corner correspondence

- Image Scaling k times
-> Coordination Scaling $\frac{kN-1}{N-1}$ times

2. Interpolation Exception (for boundary)

3. Extracting Coordination by using enumerate(np.array)

```
def enumerate2(np_array):

    for y, row in enumerate(np_array):
        for x, element in enumerate(row):
            yield (x, y, element)
```

4. Scaling magnitude integer casting

```
shape=(int(c*(Original.shape[0])),int((Original.shape[1])))
```

Interpolation (Nearest, Bilinear)

Rotating

Considered in code

1. Deciding destination matrix scale

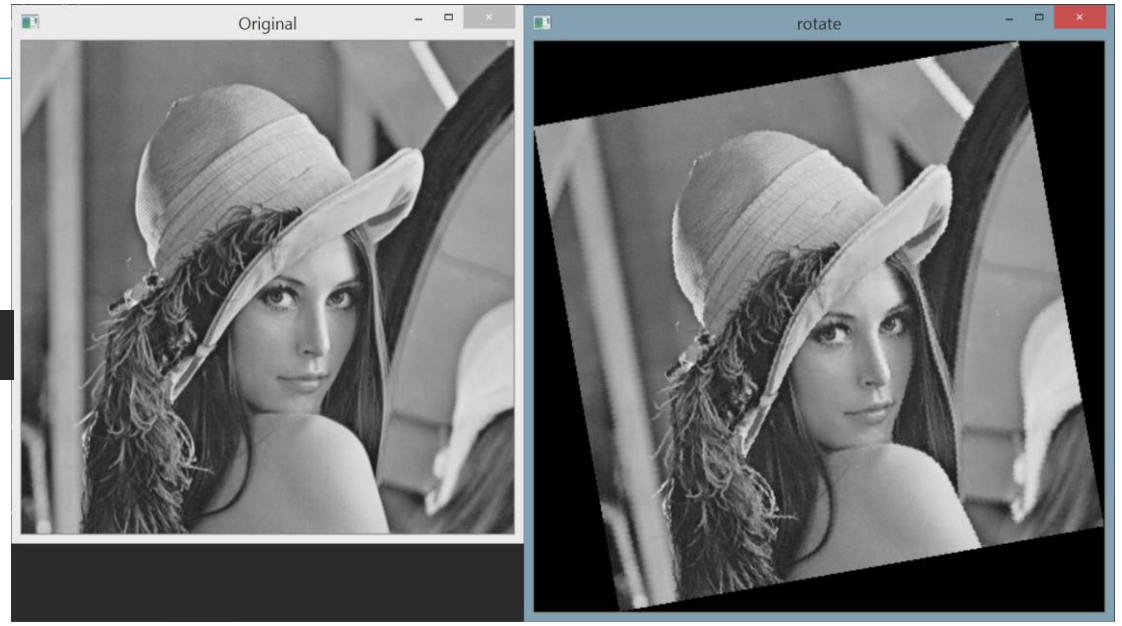
```
= (int(np.ceil(c.max()-c.min())) for c in (corner_x, corner_y))
```

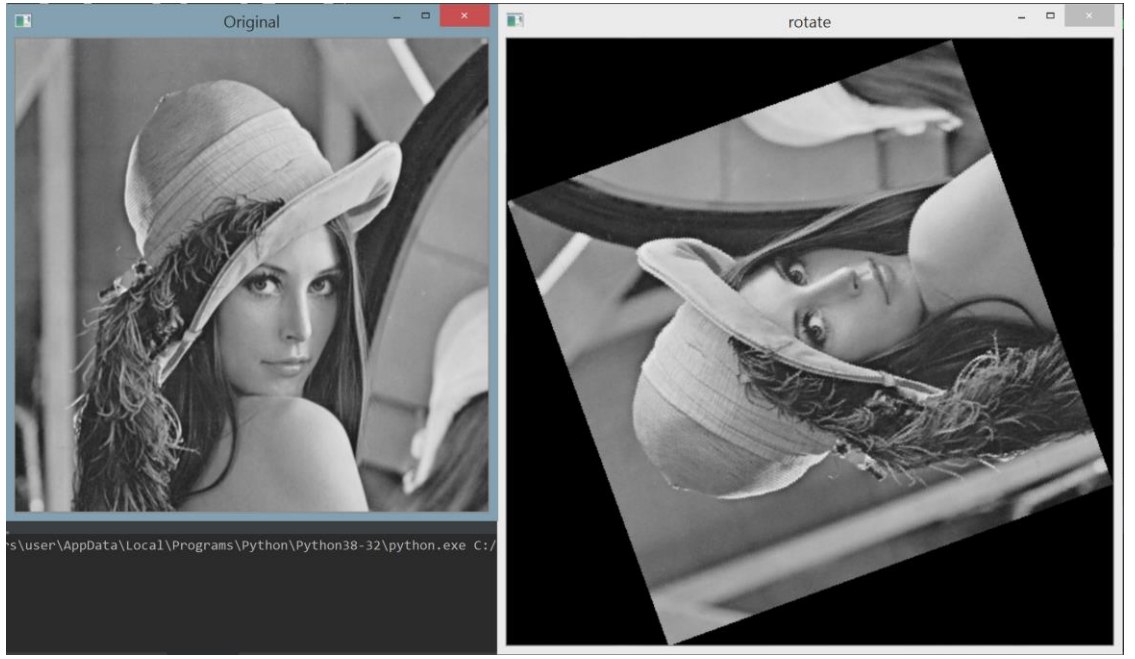
2. Backward pixel selection by casting

```
backward_x.round().astype(int), backward_y.round().astype(int)
```

3. Zero padding with nonvalid elements

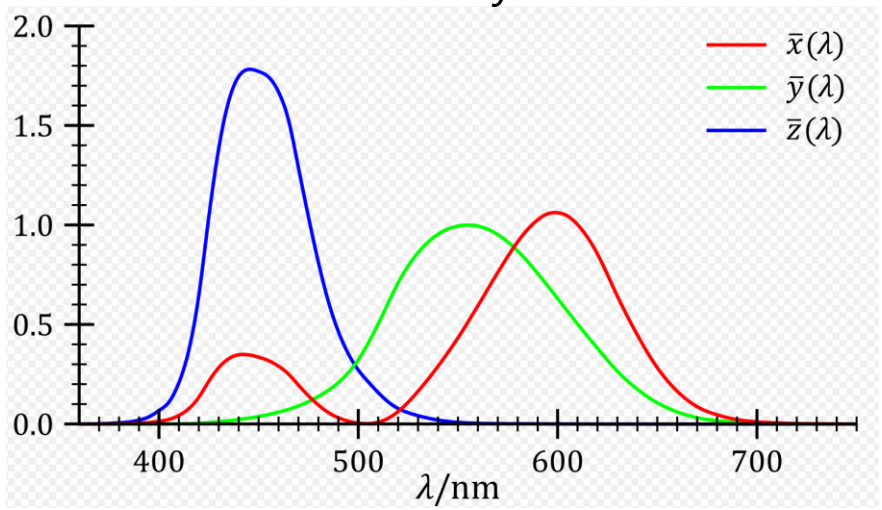
```
transformed[destination_y[~valid], destination_x[~valid]]=0
```





Light and color

Tristimulus values of eye



Light spectrum

$$X = \int_\lambda L_{\mathrm{e},\Omega,\lambda}(\lambda)\,\bar{x}(\lambda)\,d\lambda,$$

$$Y = \int_\lambda L_{\mathrm{e},\Omega,\lambda}(\lambda)\,\bar{y}(\lambda)\,d\lambda,$$

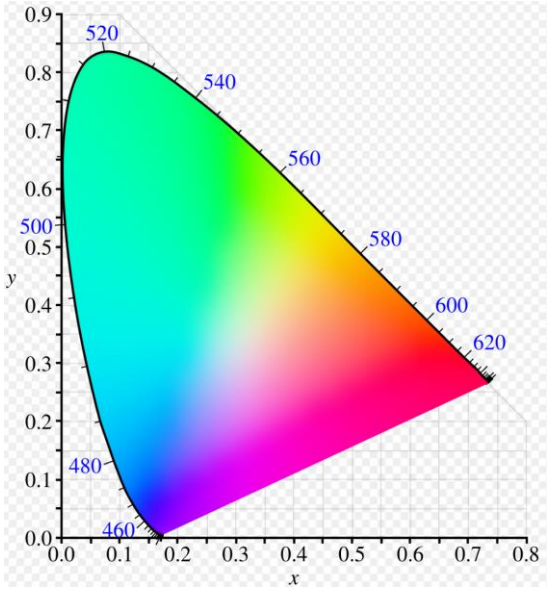
$$Z = \int_\lambda L_{\mathrm{e},\Omega,\lambda}(\lambda)\,\bar{z}(\lambda)\,d\lambda.$$

CIE 1931 xy Color Space

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

$$z = \frac{Z}{X+Y+Z} = 1 - x - y$$



Color model

Every color which human can see
Can be respresented in X,Y,Z color space.

→ X,Y,Z color space is independent with Characteristic of device.

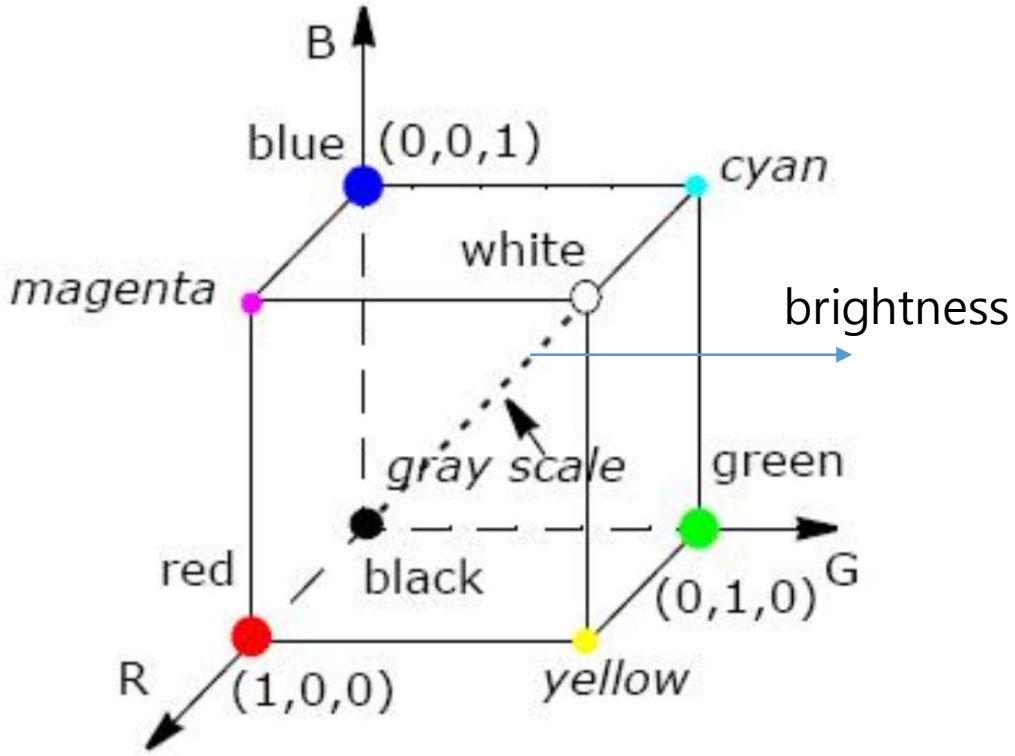
↓

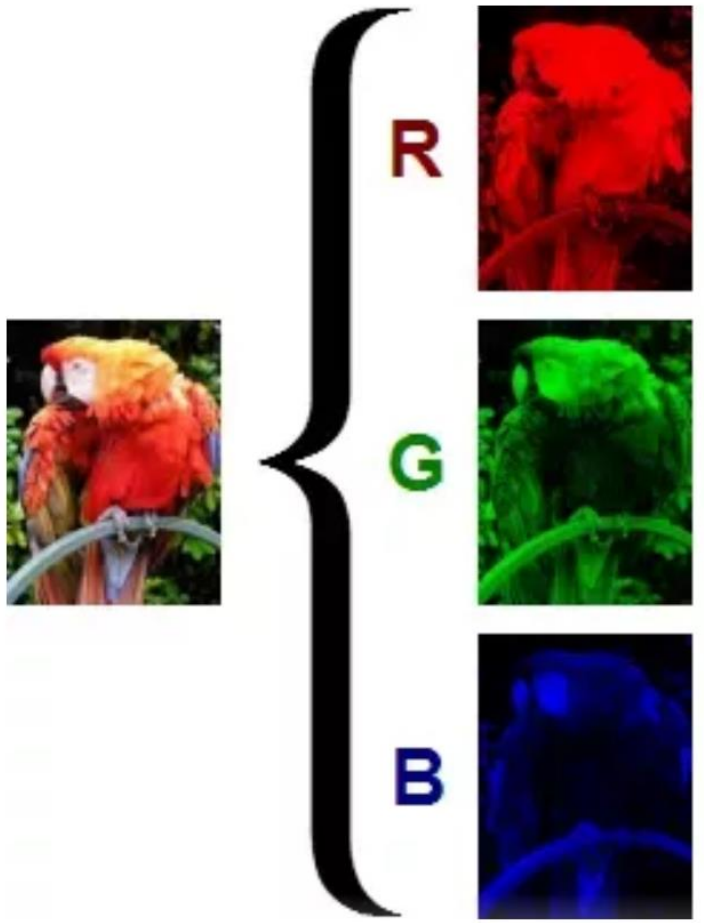
RGB color Space

↓

Various color space (HIS, CMY, Ycbcr …) in Various purpose
But there is no absolute color model
(we can't make mathematical function of brain)

RGB color model

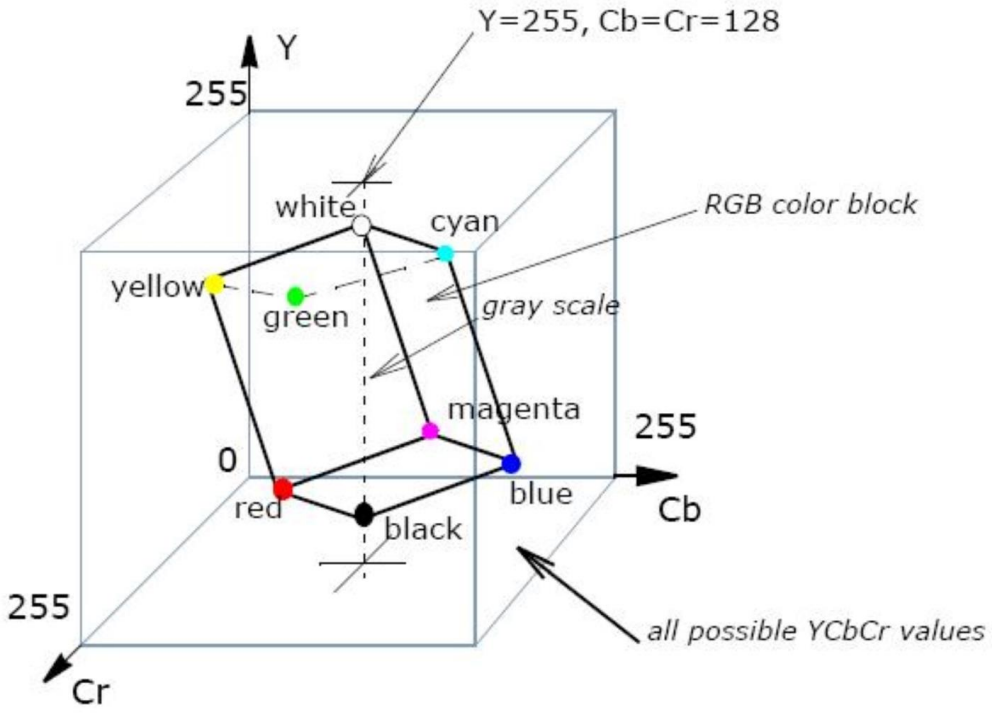


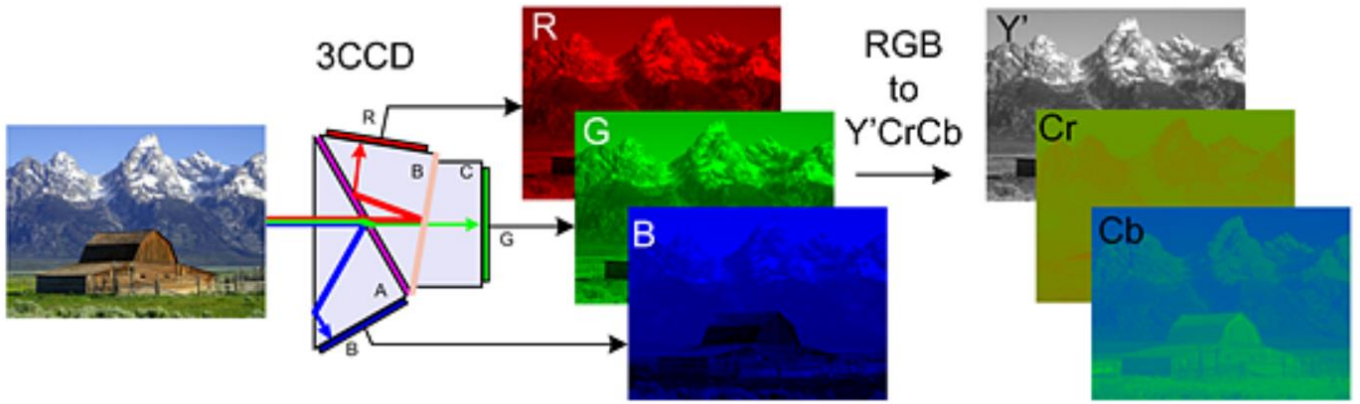


Ycbcr color model

Y= luminance
Cb , Cr = color Difference

Image compressing





$$Y = 0.299*R + 0.587*G + 0.114*B$$

$$Cb = -0.16874*R - 0.33126*G + 0.5*B + 128$$

$$Cr = 0.5*R - 0.41869*G - 0.08131*B + 128$$

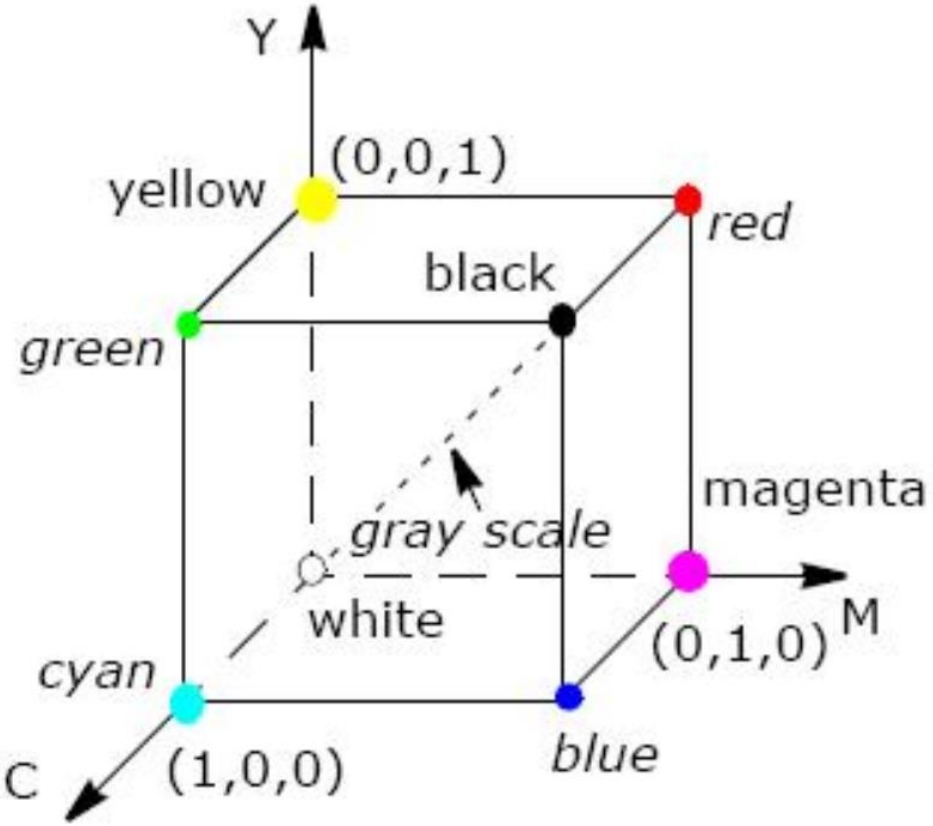
$$R = Y + 1.402*Cr - 179,456$$

$$G = Y - 0.34414*Cb - 0.71414*Cr + 135.45984$$

$$B = Y + 1.772*Cb - 226.816$$

CMY(K) color model

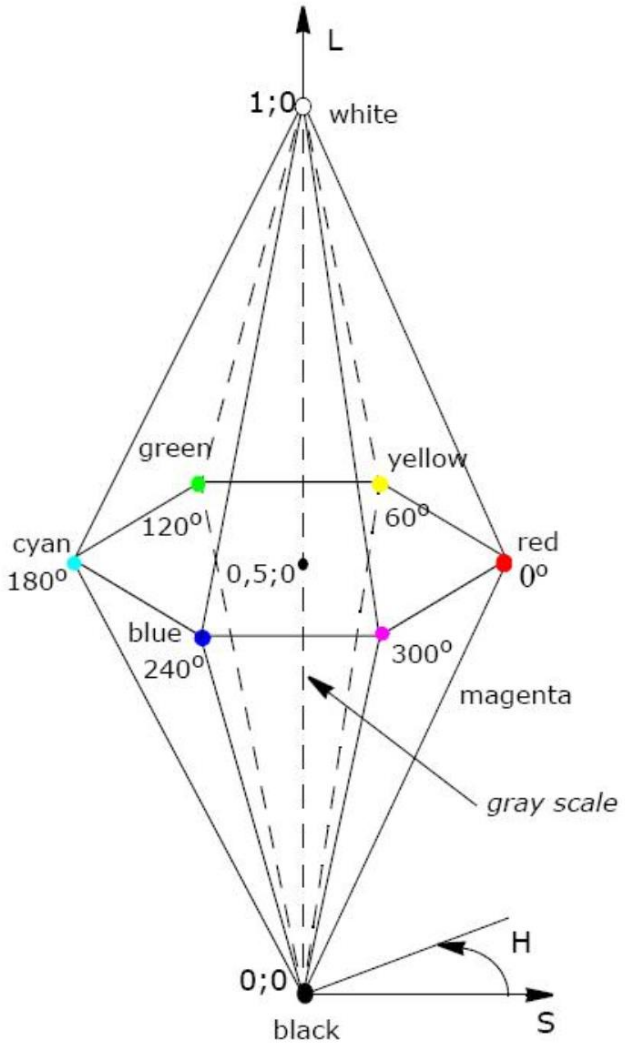
Color printer, copy machine



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

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

HSI color model



more "intuitive" in manipulating with color and were designed to approximate the way humans perceive and interpret color

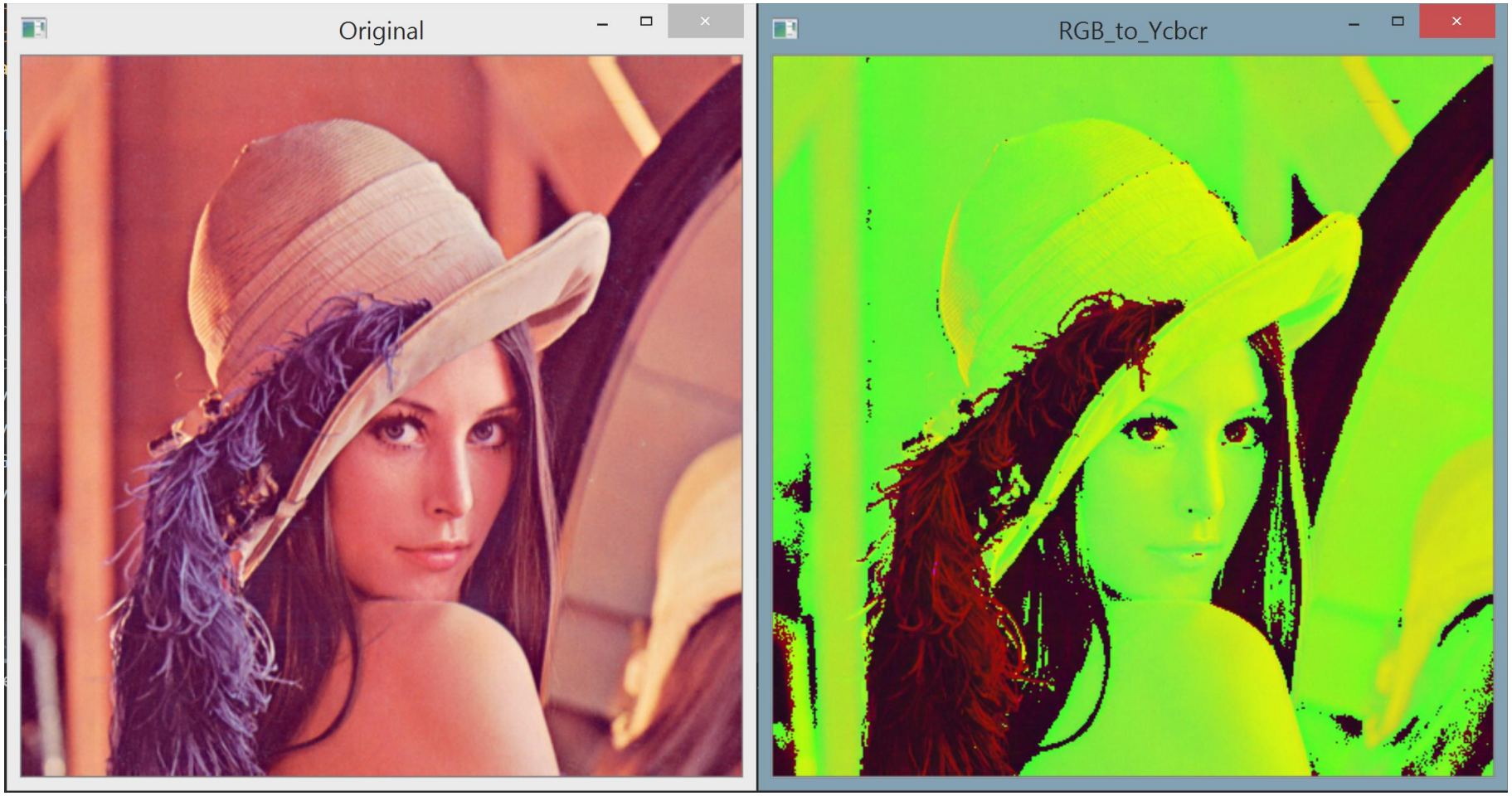
H: Hue S : Saturation I : Intensity

$$H = \cos^{-1}\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right)$$

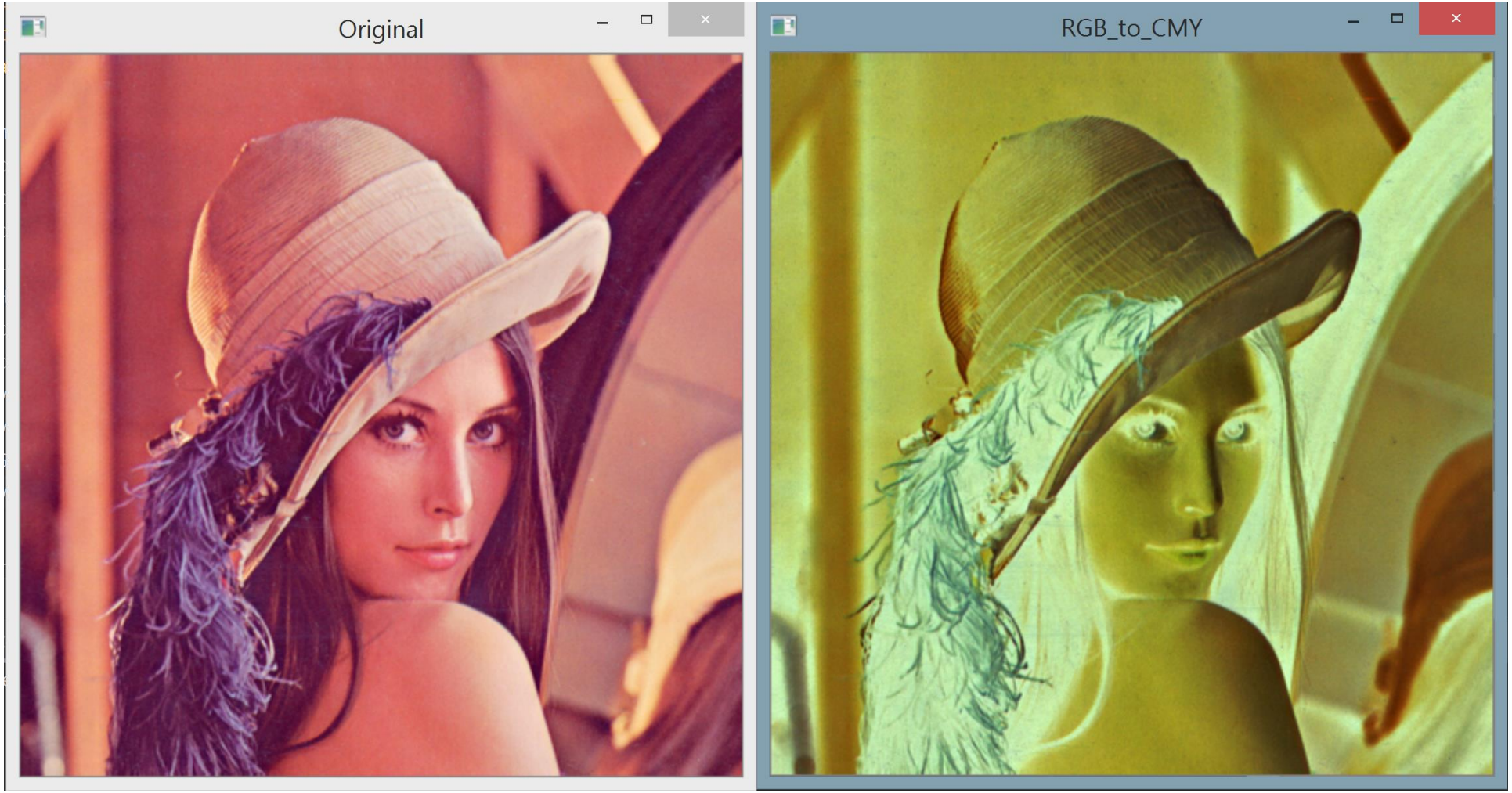
$$S = 1 - 3\ \min(R, G\,B)/I$$

$$I = (R+G+B)/3$$

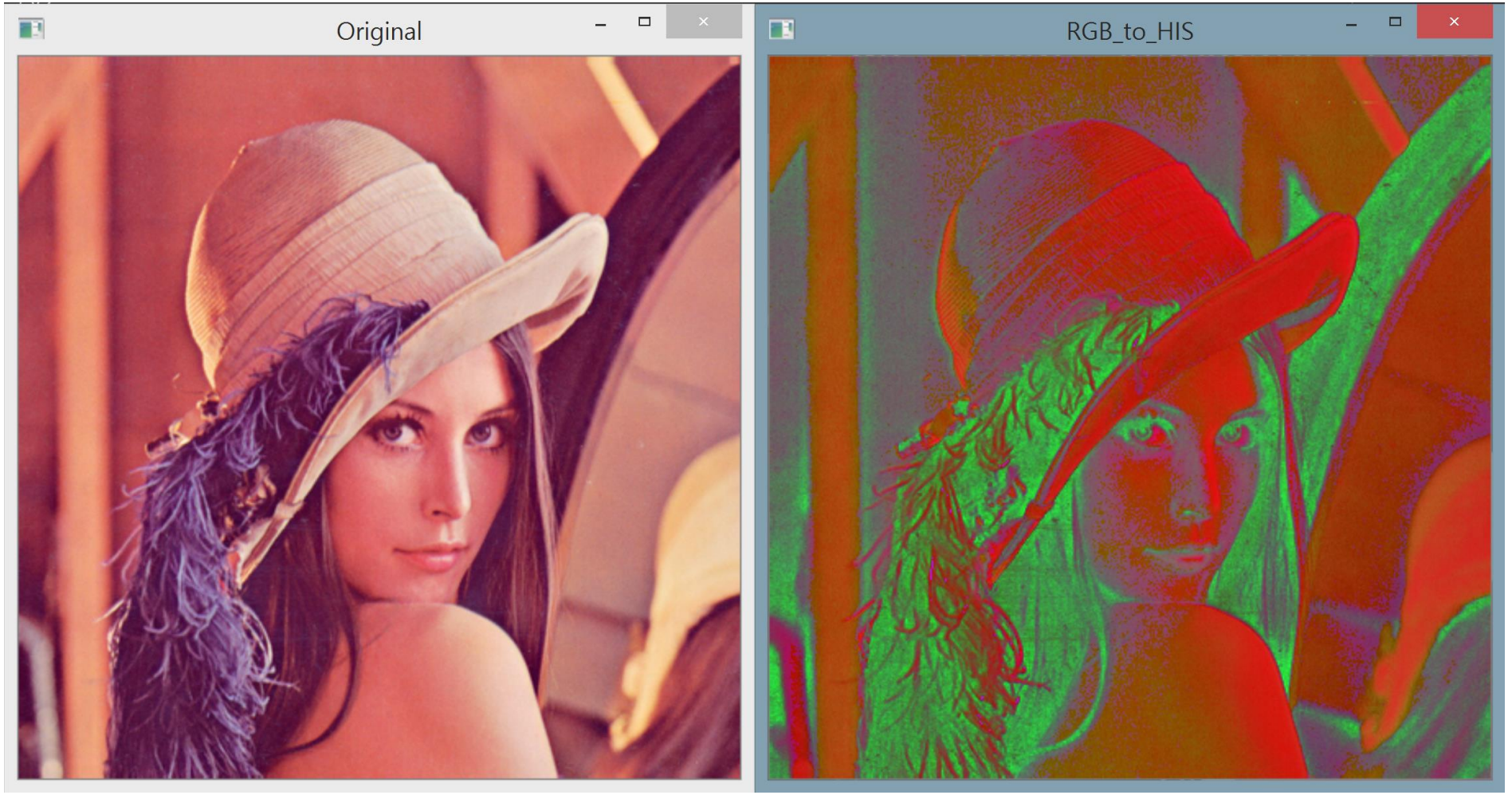
Color Transformation RGB to Ycbcr



Color Transformation RGB to CMY



Color Transformation RGB to HSI



THANK YOU
for your attention