BLG 453E

Week 2
Pointwise Image Processing

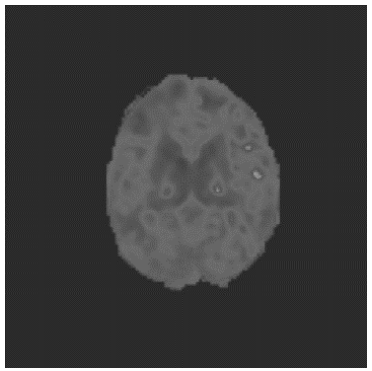
Dr. Yusuf H. Sahin

Image

- An image is a multi-dimensional representation that captures a measurement of a physical quantity.
  - 2D Images

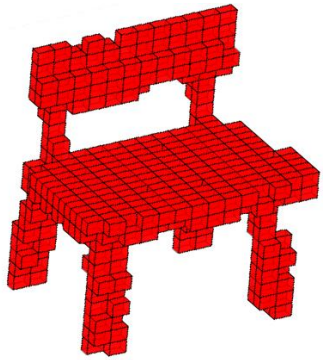
Digital Photo

Thermal Photo



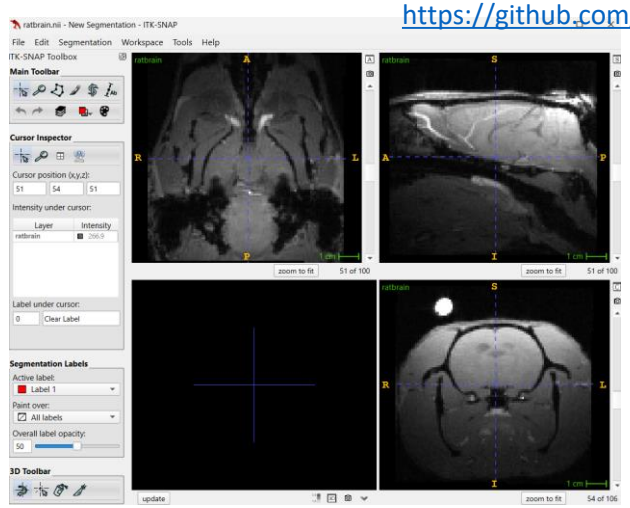
Medical Image Slices

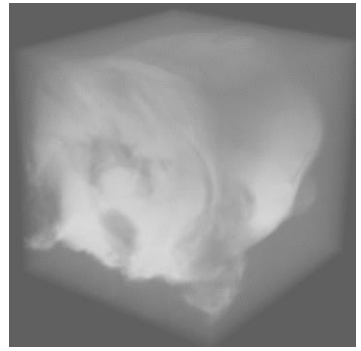
  - 3D Images



Voxel Images

https://github.com/xinan-nancy-chen/urOMT





3D Medical Scans

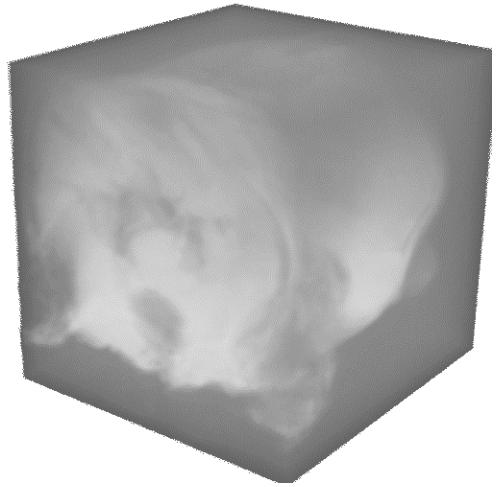
  - Video

Digital Image

- Mathematically, we can represent an image as a multi-dimensional function.

Grayscale Image
$I: \mathbb{R}^2 \rightarrow \mathbb{R}$
$I(x, y) = v$

RGB Image
$I: \mathbb{R}^2 \rightarrow \mathbb{R}^3$
$I(x, y) = [v_1, v_2, v_3]^T$



3D Medical Scan
$I: \mathbb{R}^3 \rightarrow \mathbb{R}$
$I(x, y, z) = v$

- A digital image is a collection of pixels on a 2D or 3D grid.
  The term resolution infers the size of the image (width, height).

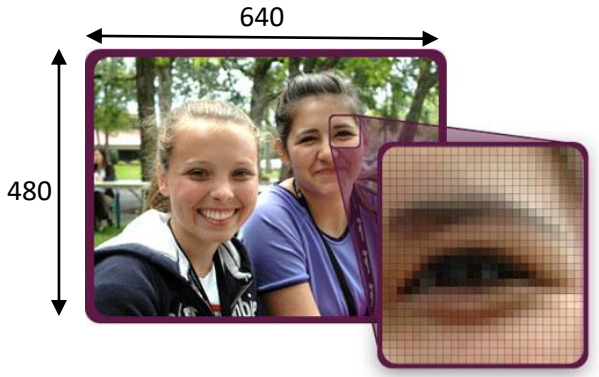


Image Operations in Python

```
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('bear.jpg')
# Convert color from BGR (used by OpenCV) to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


flipped_image = cv2.flip(image_rgb, 1)
face_image = image_rgb[200:500,120:400,:]


plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.imshow(image_rgb)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(flipped_image)
plt.title('Horizontally Flipped Image')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(face_image)
plt.title('Cropped Image')
plt.axis('off')

plt.tight_layout()
plt.show()
```

- OpenCV is an open source library for reading, writing and manipulating images.

```
>>print(image.shape)
(640, 586, 3)
```

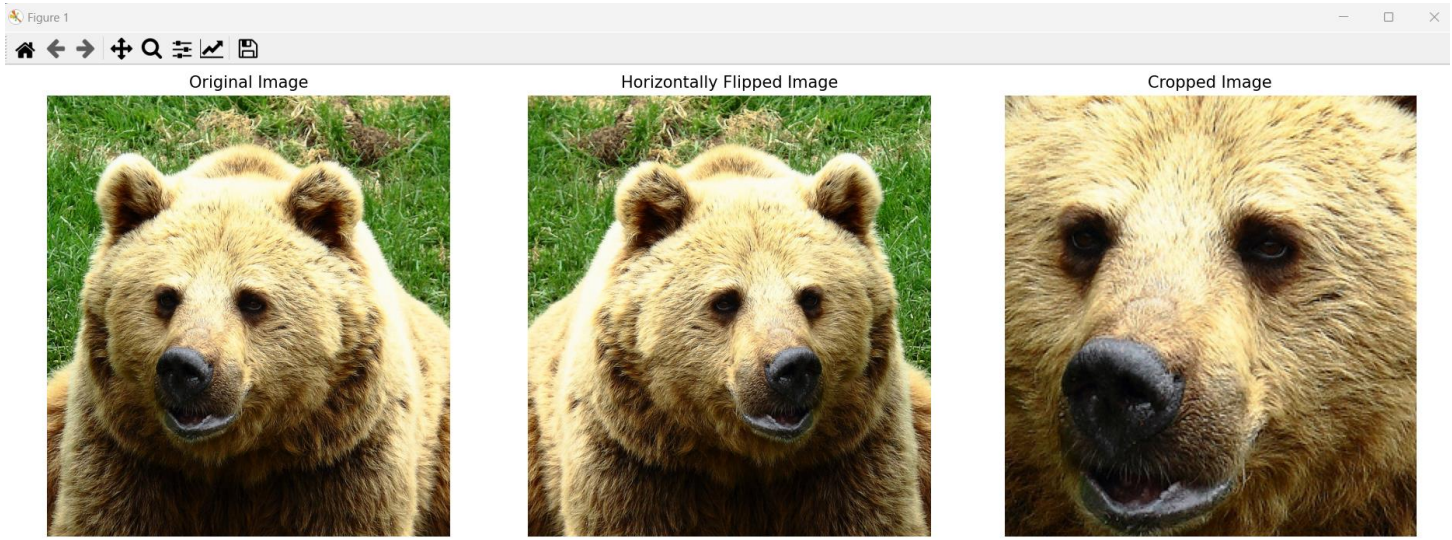


Image Operations in Python

```
from moviepy.editor import VideoFileClip, ImageSequenceClip
import numpy as np

input_video = "vid.mp4"

clip = VideoFileClip(input_video)

processed_frames = []

for frame in clip.iter_frames():
    new_frame = frame.copy()
    new_frame[:, :, 0] = 0
    processed_frames.append(new_frame)

modified_clip = ImageSequenceClip(processed_frames, fps=30)
modified_clip.write_videofile("vid_c.mp4", codec='libx264')
```

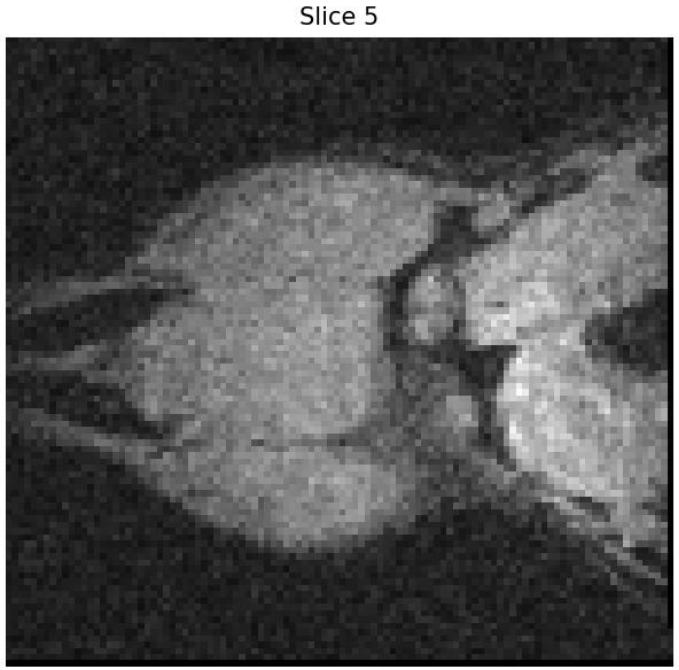
- Moviepy library could be used to do basic operations on video frames.

https://davischallenge.org/davis2016/code.html

Image Operations in Python

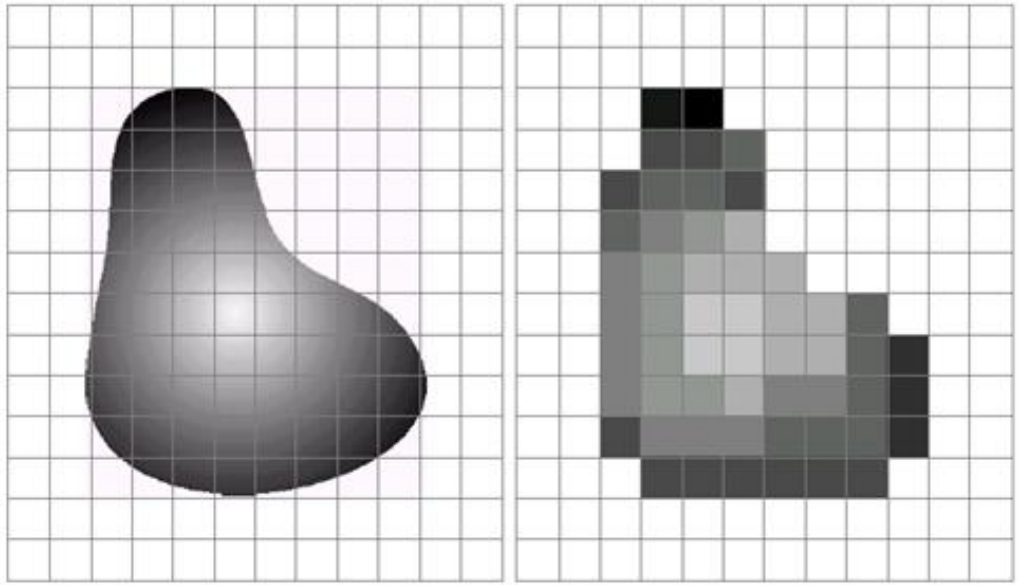
- 3D Medical images could be stored using different data formats:
  - NifTi, MetaImage, VTI etc.

```
import nibabel as nib
import matplotlib.pyplot as plt


img = nib.load("ratbrain.nii")
img_data = img.get_fdata()

plt.figure(figsize=(6, 6))
plt.imshow(img_data[:, :, 5], cmap='gray')
plt.title('Slice 5')
plt.axis('off')
plt.show()
```



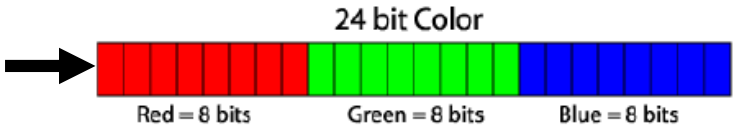
Sampling

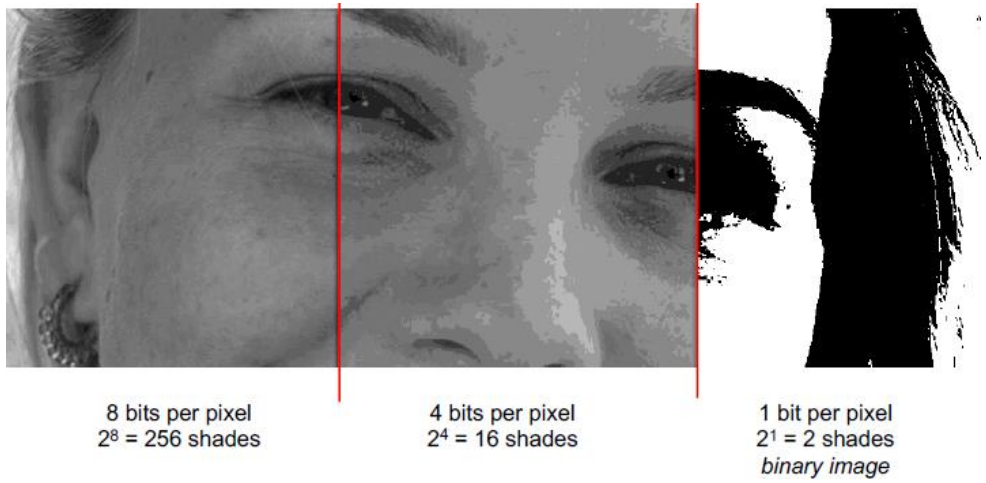
- Sampling transforms a continuous signal into a discrete one.
- The light hitting the image plane is continuous in both x and y dimensions. In a digital camera, this signal is sampled at regular intervals on a grid. The resolution is determined by the sampling rate.



Quantisation

- Quantisation limits the values a pixel can have to a finite set.
- Grayscale images or RGB color images are represented using 8 pixel per channel: [0-255].

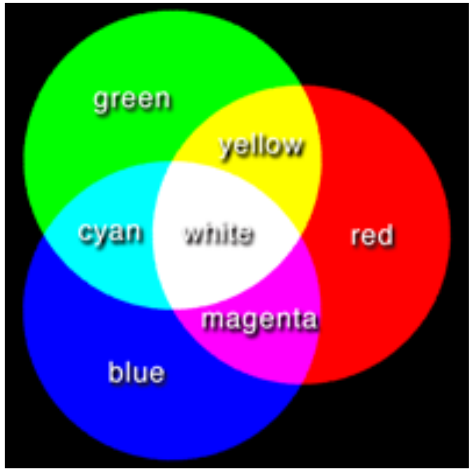


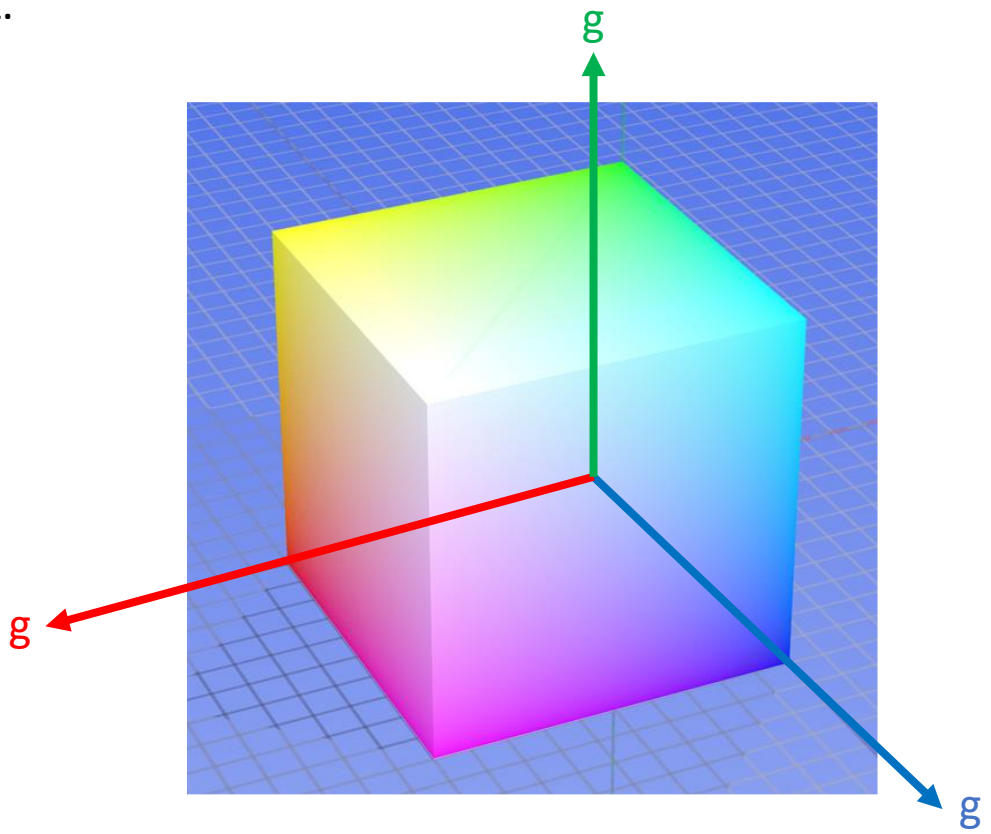


RGB Color Space

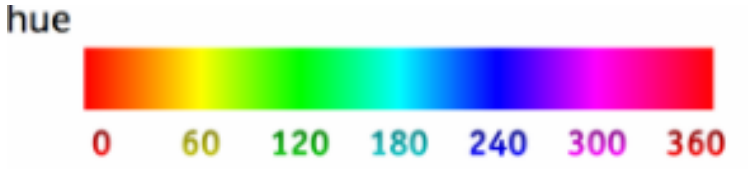
- The pixel color is found by combining red, green and blue light.
  - Additive color mixing

R	G	B	Color
0	0	0	Black
0	0	255	Blue
0	255	0	Green
0	255	255	Cyan
255	0	0	Red
255	0	255	Magenta
255	255	0	Yellow
255	255	255	White





HSV Color Space



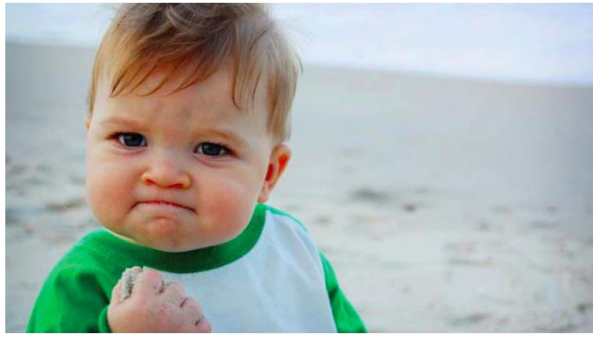
- Represents hue, saturation and value for each pixel.
  - **Hue:** Dominant color family or wavelength
  - **Saturation:** Purity of the color
  - **Value:** Lightness or darkness of a color

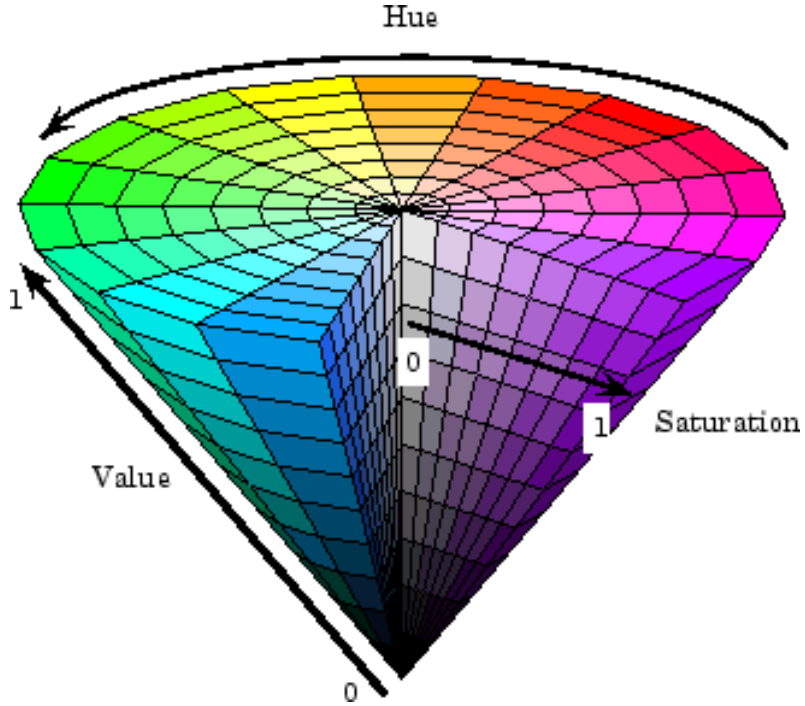
$$V = \max(R, G, B)$$

$$S = \frac{max - min}{max}$$

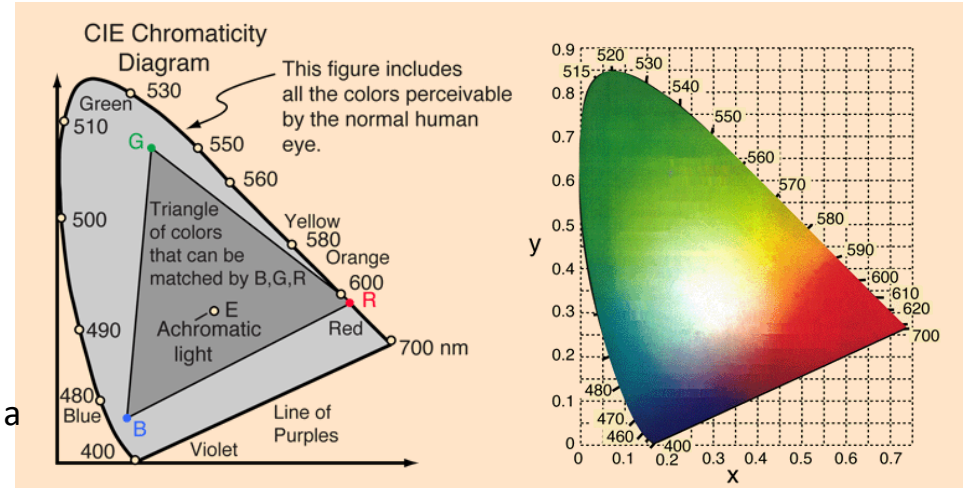
$$(max = 0 \rightarrow S = 0)$$

$$H = \begin{cases} 60 * (G - B)/(max - min) \\ 120 + 60 * (B - R)/(max - min) \\ 240 + 60 * (R - G)/(max - min) \end{cases}$$





Other Color Spaces



- CIE-XYZ color space (Commission Internationale de l'éclairage)
  - Standard reference model
  - Unlike the RGB color space, which varies depending on the device (like different cameras, screens, or printers), the XYZ color space is designed to represent color in a way that is not dependent on any particular hardware.
  - The CIE XYZ color space includes all the color perceptions that are visible to a person with normal vision.

http://hyperphysics.phy-astr.gsu.edu/hbase/vision/cie.html

If R_S, G_S and B_S are normalized RGB values (in the range [0, 1])

$$R = \begin{cases} \frac{R_s}{12.92}, & \text{if } R_s \leq 0.04045 \\ \left(\frac{R_s+0.055}{1.055}\right)^{2.4}, & \text{if } R_s > 0.04045 \end{cases} \quad G = \begin{cases} \frac{G_s}{12.92}, & \text{if } G_s \leq 0.04045 \\ \left(\frac{G_s+0.055}{1.055}\right)^{2.4}, & \text{if } G_s > 0.04045 \end{cases} \quad B = \begin{cases} \frac{B_s}{12.92}, & \text{if } B_s \leq 0.04045 \\ \left(\frac{B_s+0.055}{1.055}\right)^{2.4}, & \text{if } B_s > 0.04045 \end{cases}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Chong color space
  - A linear transformation over the XYZ color space where the illumination change effects are decreased.

$$A = \begin{bmatrix} 2.707439 \times 10^{1} & -2.280783 \times 10^{1} & -1.806681 \\ -5.646736 & -7.722125 & 1.286503 \times 10^{1} \\ -4.163133 & -4.579428 & -4.576049 \end{bmatrix}$$

$$B = \begin{bmatrix} 9.465229 \times 10^{-1} & 2.946927 \times 10^{-1} & -1.313419 \times 10^{-1} \\ -1.179179 \times 10^{-1} & 9.929960 \times 10^{-1} & 7.371554 \times 10^{-3} \\ 9.230461 \times 10^{-2} & -4.645794 \times 10^{-2} & 9.946464 \times 10^{-1} \end{bmatrix}$$

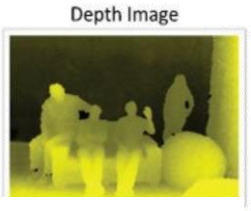
$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = A * \log\left(B * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right)$$

(a) Source image 1. (b) Source image 2.

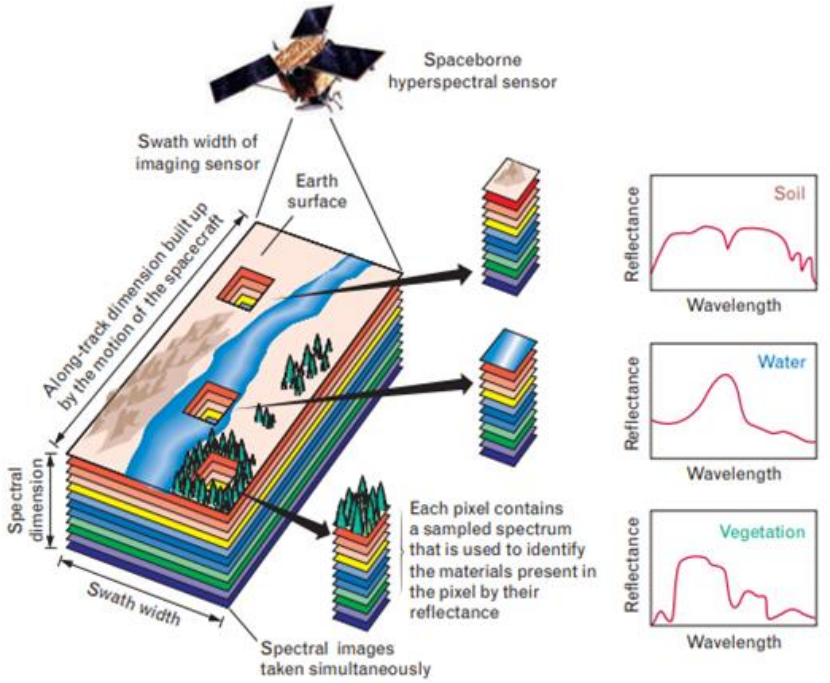
https://www.cs.harvard.edu/~sjg/papers/cspace.pdf

Other Common Image Types

- RGBD
  - The CIE XYZ color space includes all the color perceptions that are visible to a person with normal vision.



- Hyperspectral images
  - Multispectral imaging uses spectroscopy and optics to capture an object's unique spectral fingerprint for light.
  - Spectral cameras and sensors gather this data from the Earth's surface.



Point Processing of Images


- gamma

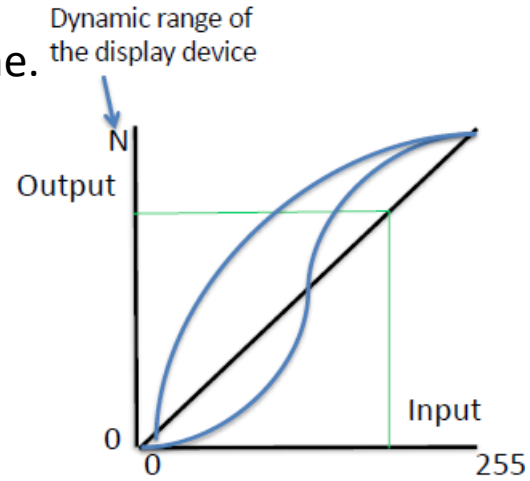

-brightness


original


+brightness

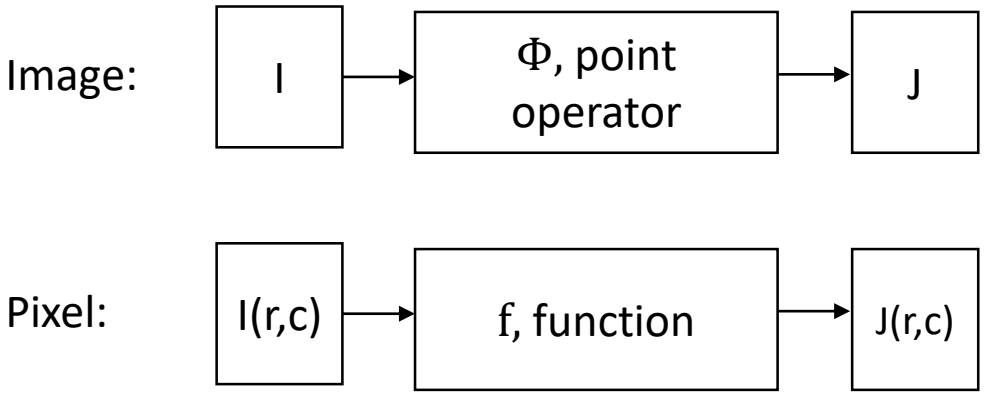

+gamma

- Point processing transforms a pixel's intensity value as function of its value alone.
- The result do not depend on the values of the pixel's neighbors.



Point Processing of Images

- Brightness and contrast adjustment
- Gamma correction
- Histogram equalization
- Histogram matching



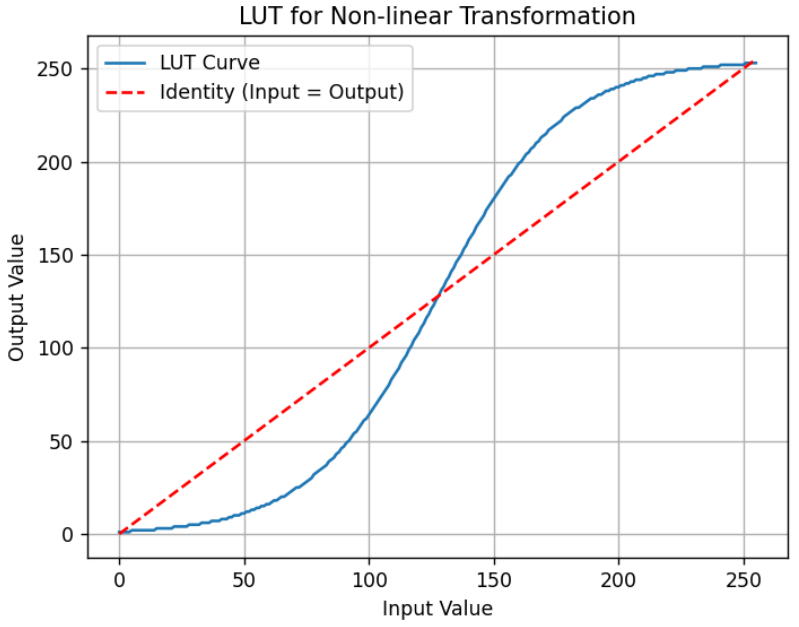
The function Φ transforms every pixel in the image according to the mapping.

$$I(r,c) = g, f(g) = k \rightarrow J(r,c) = k$$

Point Operations Using Look-up Tables

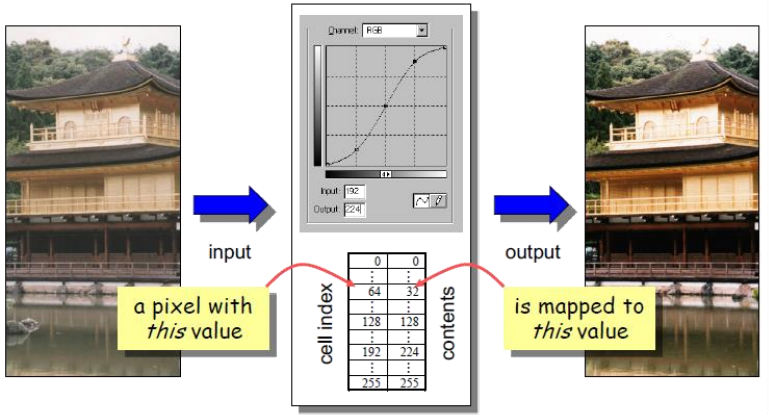
- A Look-up Table (LUT) contains a functional mapping.

```
import matplotlib.pyplot as plt
import numpy as np
import cv2

def create_lut():
    input_values = np.arange(256)

    normalized_input = input_values / 255.0
    lut_values = 255 * (1 / (1 + np.exp(-10 * (normalized_input - 0.5))))
    lut_values = np.clip(lut_values, 0, 255).astype(np.uint8)

    return lut_values


image = cv2.imread('bear.jpg')
image_grayscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

print(image_grayscale.shape)

image_converted = np.zeros_like(image_grayscale)

for i in range(image_grayscale.shape[0]):
    for j in range(image_grayscale.shape[1]):
            image_converted[i,j] = lut[int(image_grayscale[i,j])]
```



Point Operations Using Look-up Tables

- If the image I has 3-channels,
  - **Option 1:** Same LUT could be used for every channel.
  - **Option 2:** Each channel is mapped using different LUTs.



Increase Brightness

$$J_k(r,c) = \begin{cases} I_k(r,c) + g, & I_k(r,c) + g < 256 \\ 255, & otherwise \end{cases}$$

where $g \geq 0,\ k \in \{1,2,3\}$ indicates channel ID

Decrease Brightness

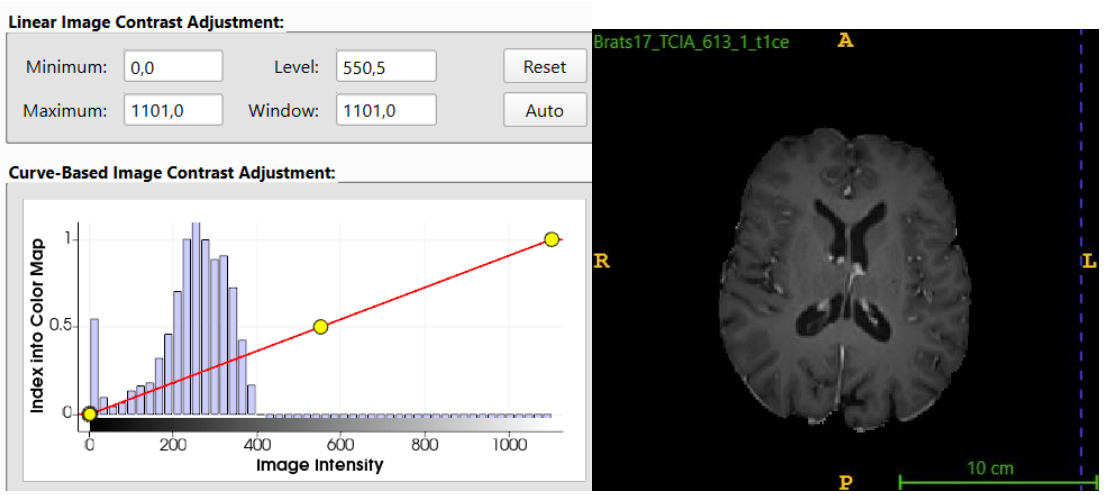
$$J_k(r,c) = \begin{cases} I_k(r,c) - g, & I_k(r,c) - g > 0 \\ 0, & otherwise \end{cases}$$

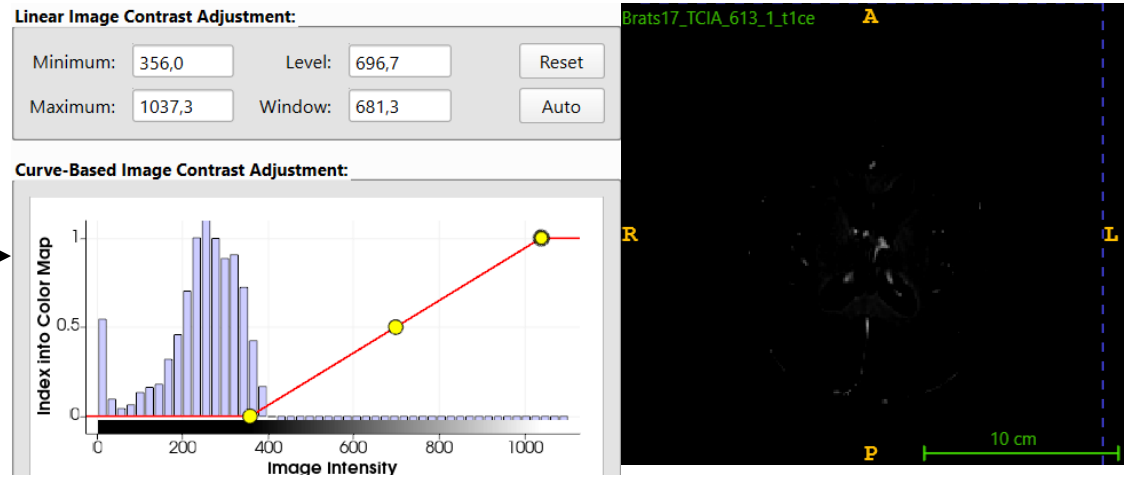
where $g \geq 0,\ k \in \{1,2,3\}$ indicates channel ID

Window-Level Transformation

- Window-Level Transformation is a technique used in medical imaging, particularly in modalities like CT (Computed Tomography) and MRI (Magnetic Resonance Imaging).
- The main focus is to enable the visualization of different structures.

$$J(r,c) = \begin{cases} 0, & I(r,c) \le WL - \frac{WW}{2} \\ 255, & I(r,c) \ge WL + \frac{WW}{2} \\ \frac{I(r,c) - \left(WL - \frac{WW}{2}\right) * 225}{WW}, & otherwise \end{cases}$$



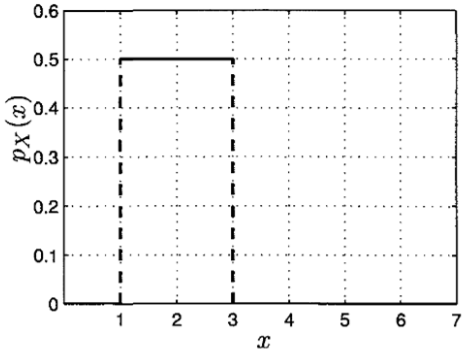


https://github.com/pyushkevich/itksnap

Probability Density Function

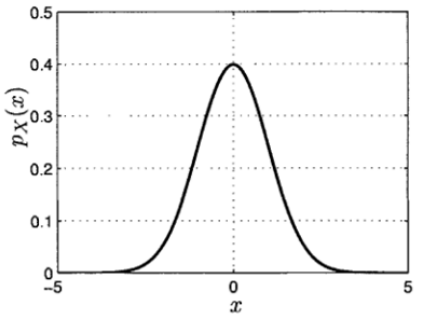
- A **Probability Density Function** $p(x)$ shows how likely different values of a continuous random variable are.
- $P[a \leq X \leq b] = \int_a^b p(x)dx$
- $p(x) \geq 0, \quad -\infty < x < \infty$
- $\int_{-\infty}^{\infty} p(x)dx = 1$

Uniform Distribution



$$p(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0, & otherwise \end{cases}$$

Gaussian Distribution



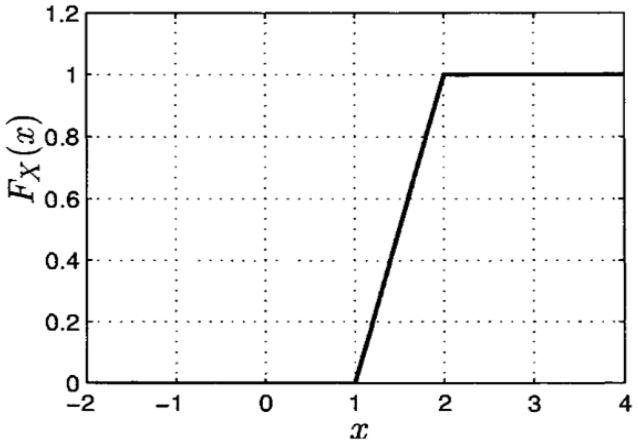
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2\right], -\infty < x < \infty$$

Kay, S. (2006). *Intuitive probability and random processes using MATLAB®*. Springer Science & Business Media.

Cumulative Density Function

- The cumulative density function $F(x)$ shows the total probability to obtain a value smaller than a given value.
- $F(x) = P[X < x], -\infty < x < \infty$
- $F(x) = \int_{-\infty}^{x} p(t)dt$

Uniform Distribution

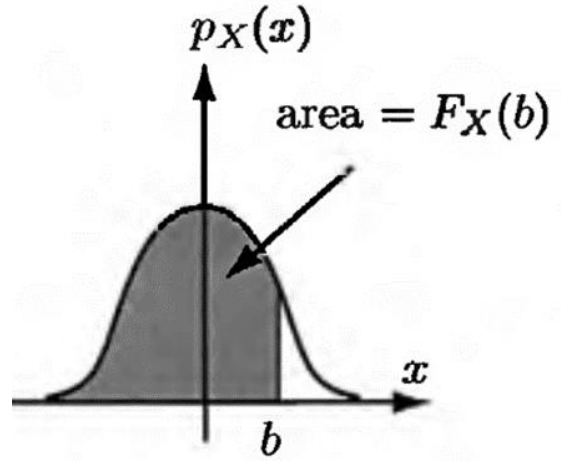


CDF for uniform random variable over interval $(1, 2)$

$$\mathrm{F}(x) = \begin{cases} 0, & x \le a \\ \int_a^x \frac{1}{b-a} dt, & a < x < b \\ 1, & x \ge b \end{cases}$$

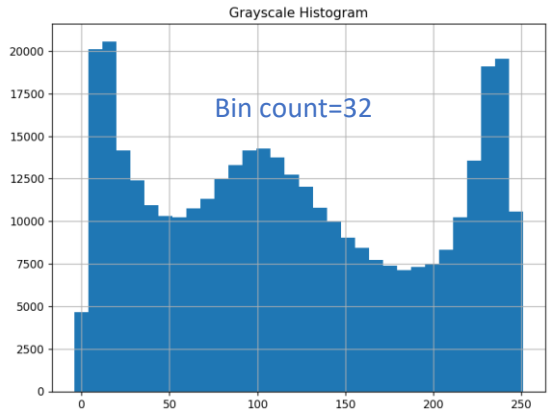
$$= \begin{cases} 0, & x \le a \\ \frac{1}{b-a}(x-a), & a < x < b \\ 1, & x \ge b \end{cases}$$

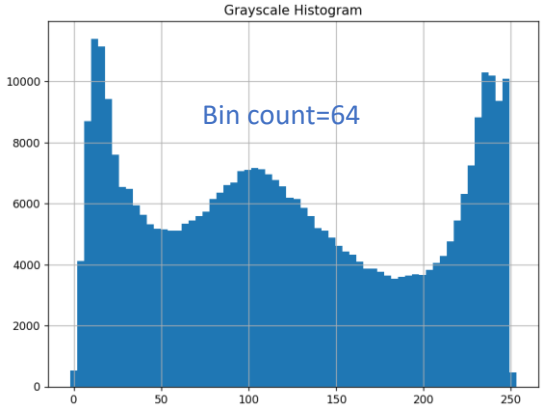
Gaussian Distribution

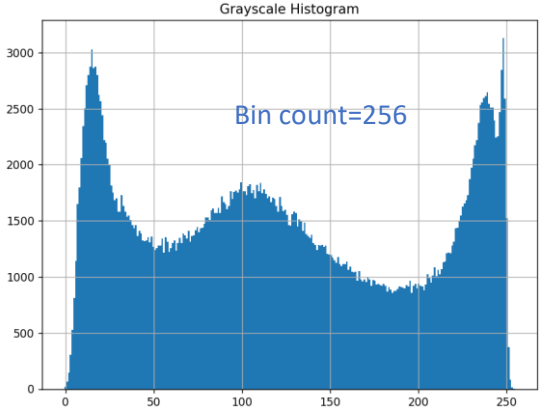


Histogram

- A histogram is a graphical representation that shows the distribution of data points within specified ranges or "bins."
  - The x-axis of the histogram represents the pixel intensity values (ranging from 0 to 255 for an 8-bit image).
  - The y-axis represents the frequency (or count) of pixels that have a particular intensity value.
  - Bins are used to group the pixel intensities according to a predefined range.

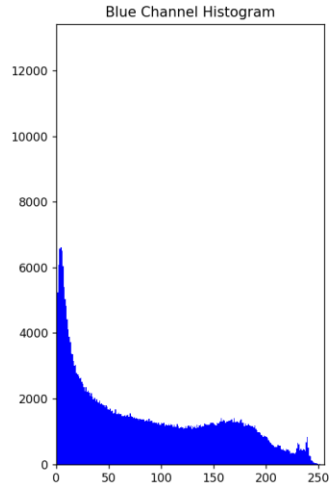


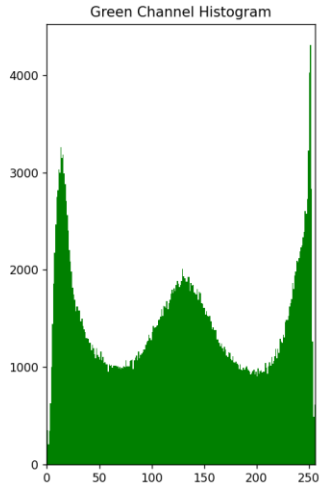


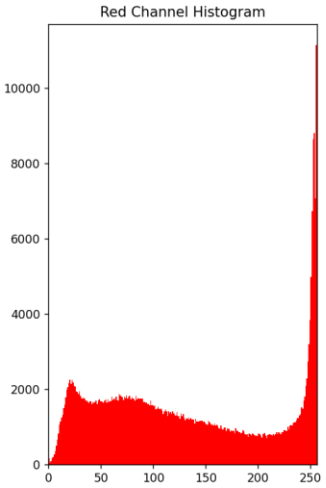


Histogram of a Color Image

- For color images, different histograms are calculated for each channel.

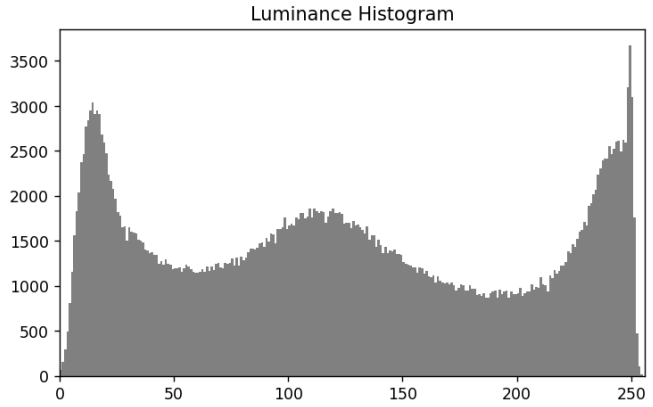






- A histogram about luminance could also be calculated considering the perception of red green and blue channels.

$$L(r,c) = 0.299\,R(r,c) + 0.587\,G(r,c) + 0.114\,B(r,c)$$



Probability Density Function of An Image

- For an image containing a total of N pixels, the normalized histogram p_I is obtained as

$$p_I(g) = \frac{1}{N} h(g)$$

- Sum of all $p_I(g)$ over all g is 1.
- $p_I(g)$ represents the probability that a randomly chosen pixel from the image has an intensity value of g.

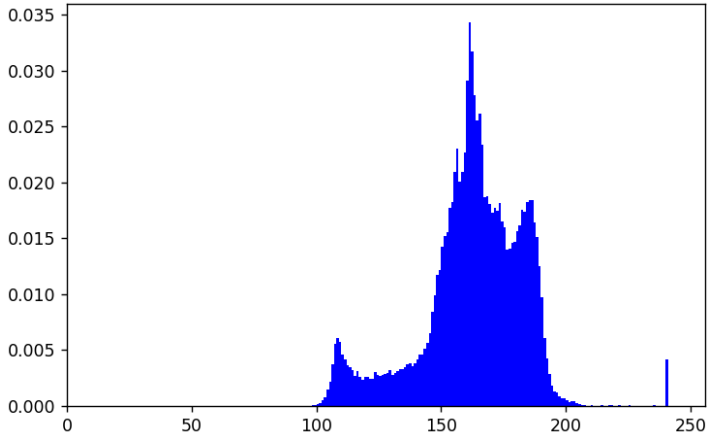
- For an image containing a total of N pixels, cumulative density function P_I could be obtained as

$$P_I(g) = \frac{1}{N} \sum_{i=0}^{g} p_I(g)$$

$$P_I(0) = p_I(0)$$
$$P_I(255) = 1$$

$P_I(g)$ is a nondecreasing function.

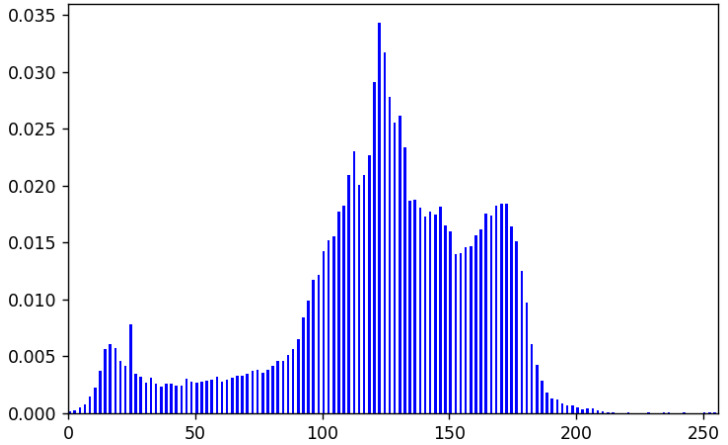
Histogram as a pdf



- The image seems washed out, since there are nearly no pixels found for darker and lighter values.
- We can do «contrast stretching» to change the distribution of intensities.

$$J = \alpha I + \beta$$

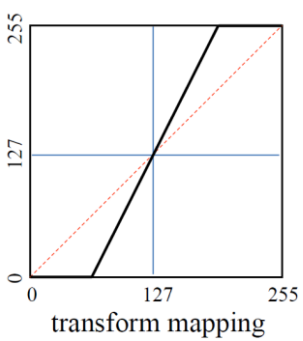
For $\alpha = 2, \beta = -200$:



Contrast Operations

Original Image:

Increase Contrast

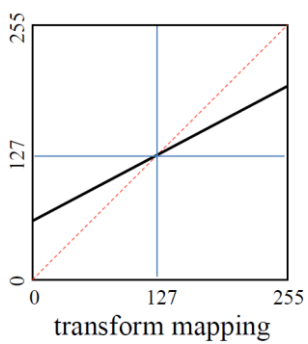


$$T_k(r,c) = \alpha[I_k(r,c) - 127] + 127, \alpha > 1.0$$

$$J_k(r,c) = \begin{cases} 0, & T_k(r,c) < 0 \\ T_k(r,c), & 0 \le T_k(r,c) \le 255 \\ 255, & T_k(r,c) > 255 \end{cases}$$

$k \in \{0,1,2\}$

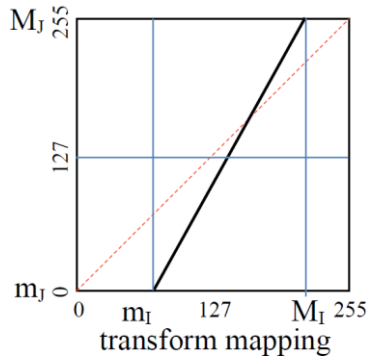
Decrease Contrast



$$T_k(r,c) = \alpha[I_k(r,c) - 127] + 127, 1.0 > \alpha \ge 0$$

$k \in \{0,1,2\}$

Contrast Stretching



Let $m_I = \min[I(r,c)], M_I = max[I(r,c)],$
$m_J = \min[J(r,c)], M_J = max[J(r,c)],$
Then,

$$J(r,c) = (M_J - M_I)\frac{I(r,c) - m_I}{M_I - m_I} + m_J$$

Gamma Correction

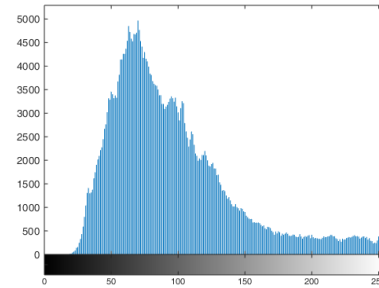
- Gamma correction applies a non-linear adjustment to enhance sensitivity to variations in darker tones more than in lighter ones.

$$J(r,c) = 255 * \left[\frac{I(r,c)}{255}\right]^{\frac{1}{\gamma}}, \text{ for } \gamma > 1.0$$

For $\gamma = 2$:

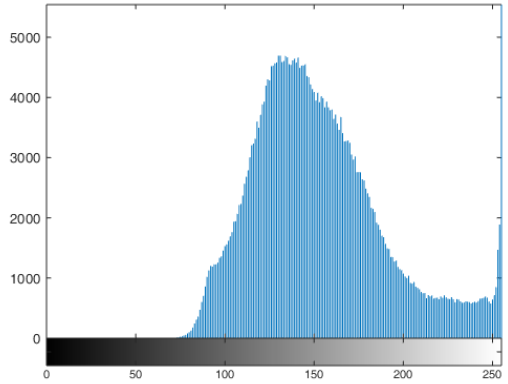
Gamma Operations

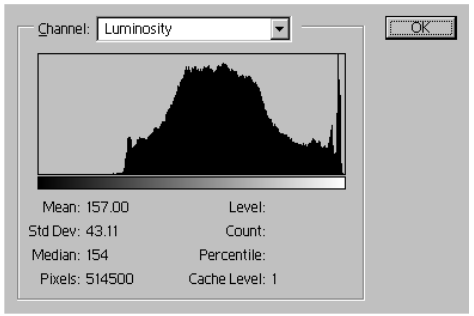
Original Image:

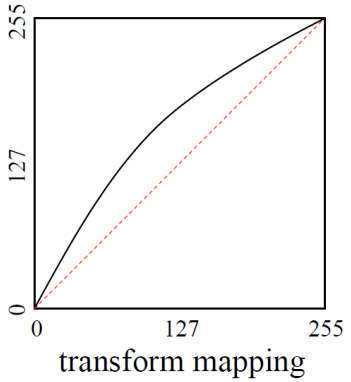


Increasing Gamma

$\gamma = 2$

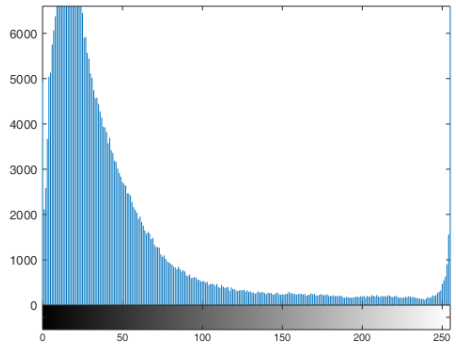


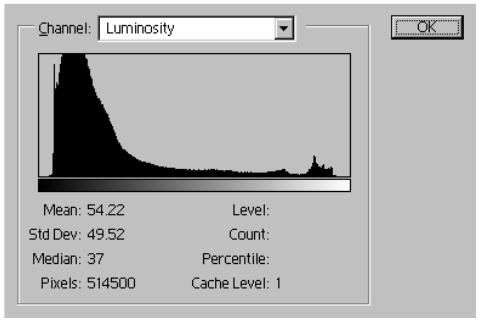


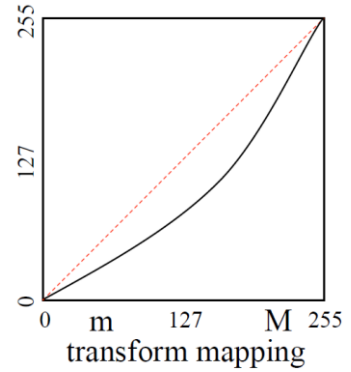


Decreasing Gamma

$\gamma = \frac{1}{2}$

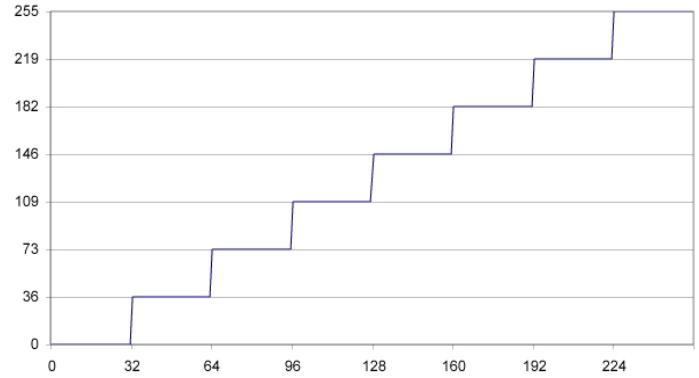






Quantisation as a Pointwise Operation

- The transformation mapping could be designed to divide the pixel space into levels.



Original

2 Levels

3 Levels

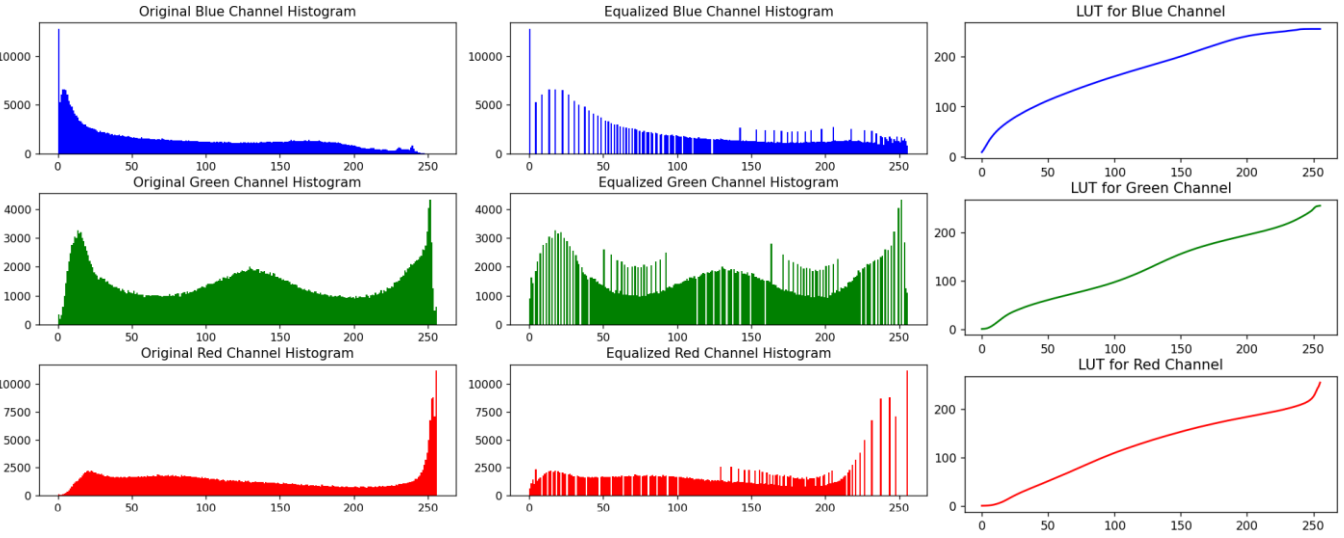
8 Levels

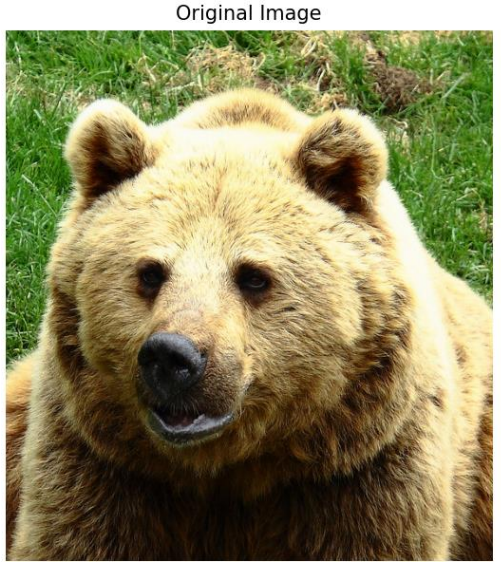
Histogram Equalization

- Remapping the image I so that its histogram is as close to uniform distribution as possible.
- All channels are processed independently.

$$J(r,c) = 255 * P_I(\mathrm{I(r,c)})$$

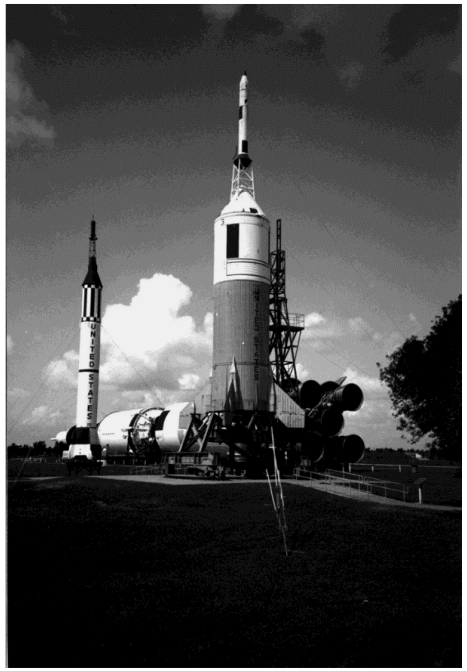
```
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * hist.max()/ cdf.max()
plt.plot(cdf_normalized, color = 'b')
```



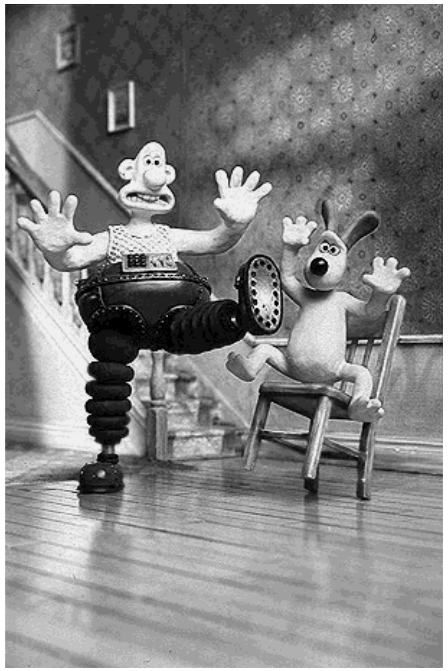


Histogram Matching

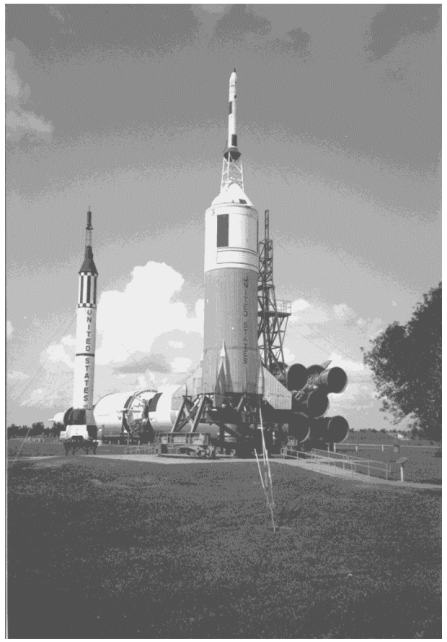
- Remapping the source image I so that it has, as closely as possible, the same histogram as the target image J.
- Because the images are digital it is not, in general, possible to make $h_I \equiv h_J$. Therefore, $p_I \not\equiv p_J$.
- Thus, matching the percentiles may be a solution.



Source



Target

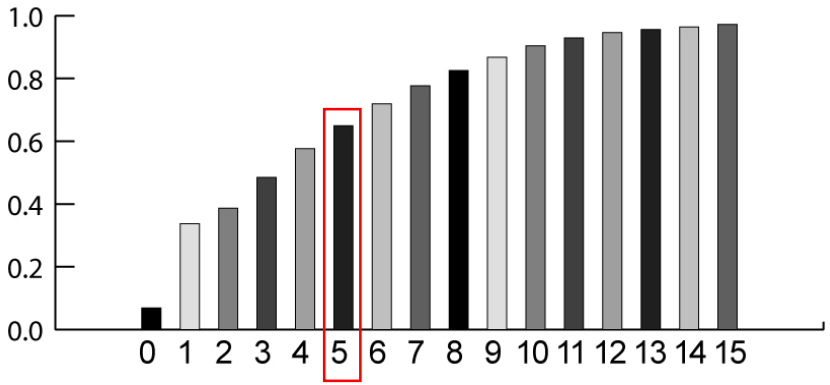


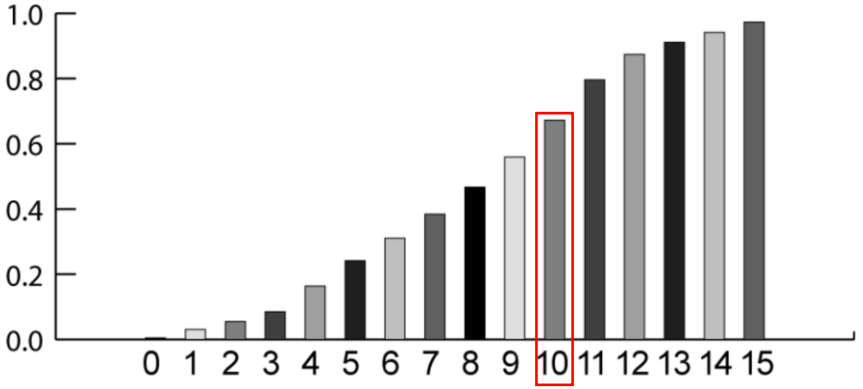
Remapped

Matching Percentiles

- Recall that, for an image I,
  - $P_I(g_I) = c$ means that c is the fraction of pixels in I that have a value less than or equal to g_I.
  - $100c$ is the percentile of pixels in I, that are less than or equal to g_I.
- To align percentiles, replace every instance of the value g_I in image I with the value g_J from image J, where g_J's percentile in J closely corresponds to the percentile of g_I in I.
- To decide on the pixel $I(r,c) = g_I$'s corresponding value $K(r,c) = g_J$

$$P_J(g_J - I) < P_I(g_I) \leq P_J(g_J)$$





Example:

$I(r,c) = 5$

$P_I(5) = 0.65$

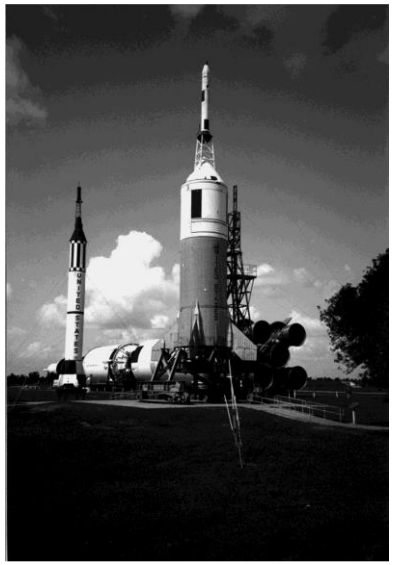
$P_j(9) = 0.56$

$P_j(10) = 0.67$

$K(r,c) = 10$

Histogram Matching Algorithm

For images with 16 intensity values:



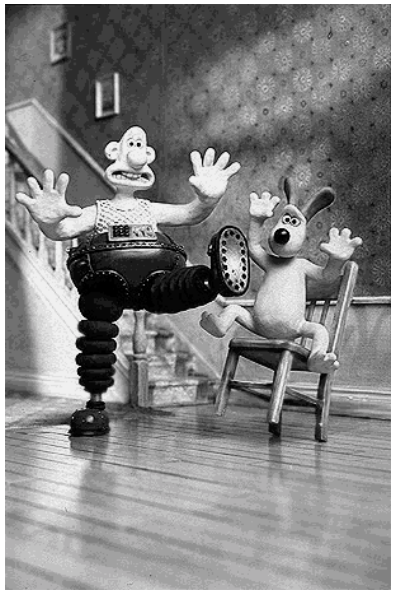


Image PDF

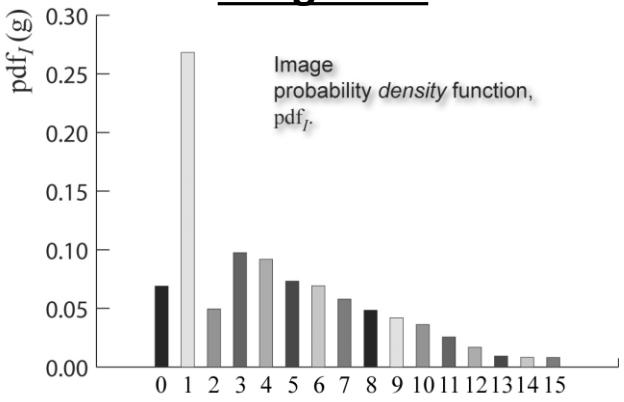


Image CDF

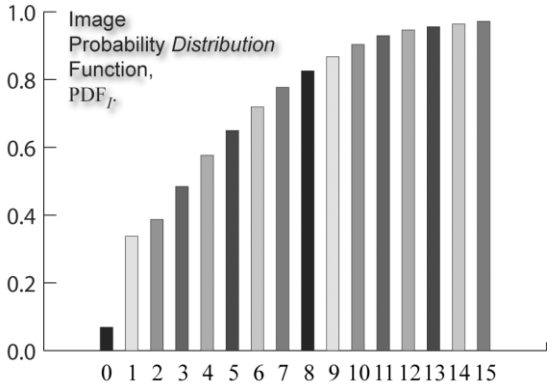


Image PDF

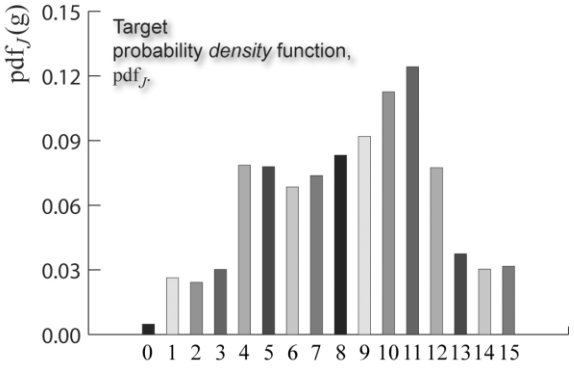
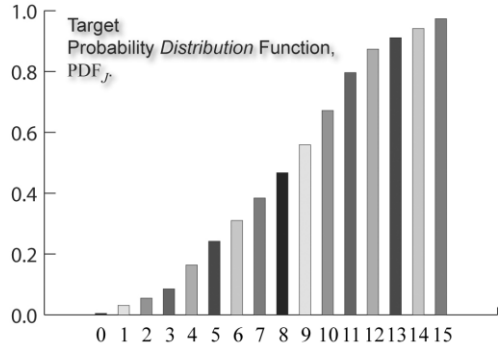


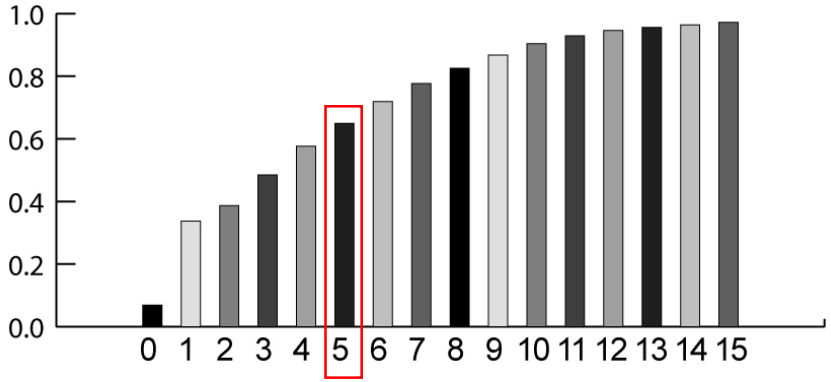
Image CDF



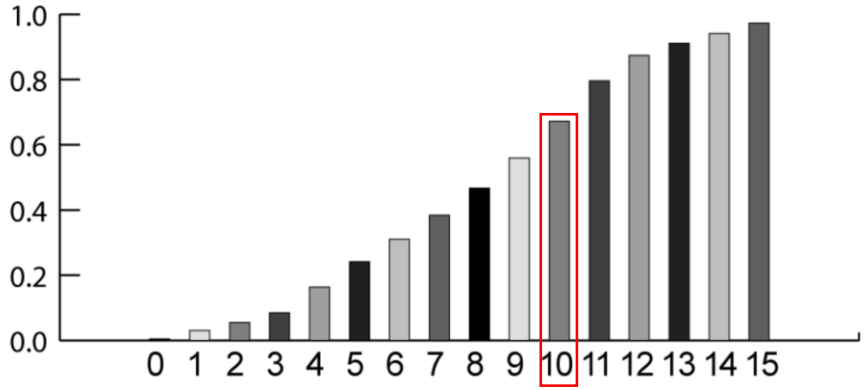
LUT Creation for Histogram Matching

```
LUT = np.zeros((256, 1))
g_J = 0
for g_I in range(255):
    while P_J[g_J] < P_I[g_I] AND g_J < 255:
        g_J = g_J + 1;
    end
    LUT[g_I] = g_J;
end
```

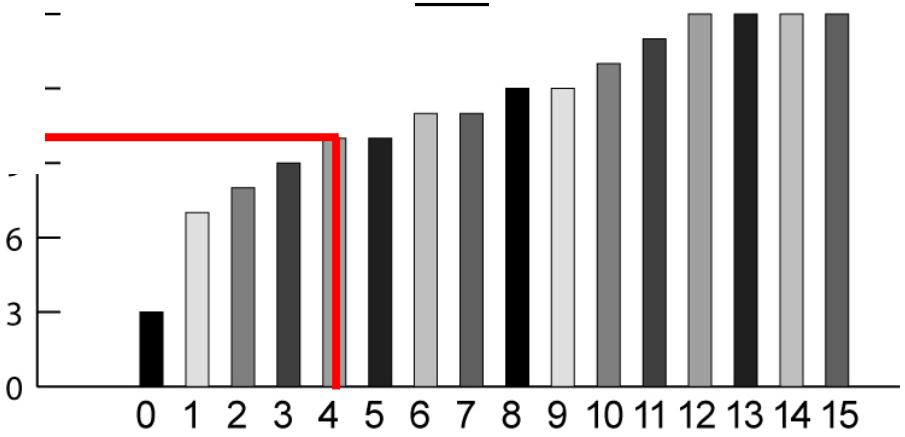
Image CDF



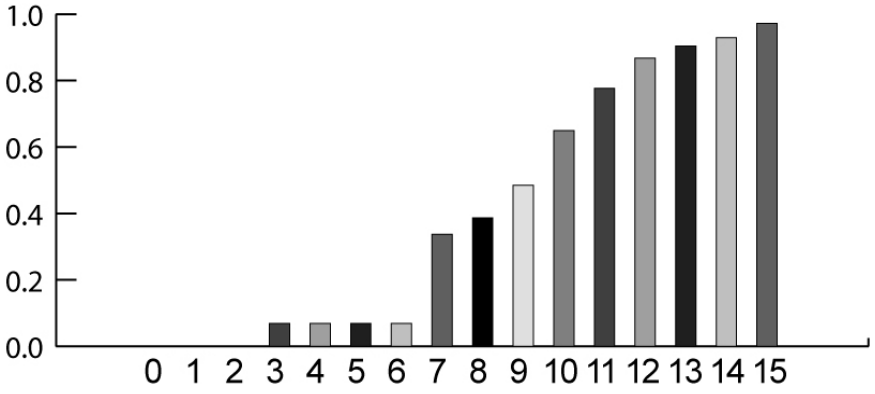
Target CDF



LUT



Resulting CDF



Remapping a Color Image

Original

G+B ← R

R+B ← G

R+G ← B

Image Augmentation

- For many machine learning models, augmented version of the train images are also fed to the algorithm to increase robustness of the model.

- Augementations could be geometric operations, pointwise operations and some filters.

- **Invertion:** Inverts all values in images from v to 255-v.
- **Solarization:** Inverting all values above a threshold.
- **Multiply Hue:** Multiply only the Hue channel with some value.
- **Color Quantization:** Quantize the values according to levels.
- **Grayscale:** $(R + G + B)/3$

Original

Inverse

Solarized

Hue Multip.

Quantized

Grayscale

https://github.com/aleju/imgaug