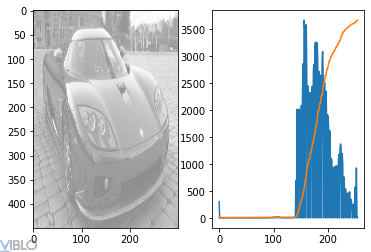
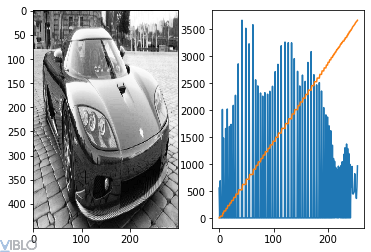
**Lý thuyết**

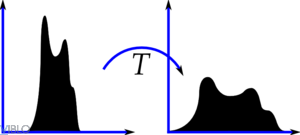
**1. Khái niệm.**

Histogram là biểu đồ tần xuất thống kê số lần xuất hiện các mức sáng trong ảnh. Dưới đây là ảnh minh họa. Nhìn vào biểu đồ (chưa cần quan tâm tới đường màu đỏ), dựa vào các cột gía trị có thể dễ dàng thấy được rằng: có khoảng 2000 điểm ảnh mang giá trị 150, giá trị pixel chủ yếu trong khoảng [150, 200] nên độ tương phản không cao, không rõ nét.





**Cân bằng histogram** (histogram equalization) là sự điều chỉnh histogram về trạng thái cân bằng, giá trị các điểm ảnh không bị co cụm tại một khoảng nhỏ mà được "kéo dãn" ra. Cân bằng histogram là một phương pháp tiền/hậu xử lí ảnh rất mạnh mẽ. Đặc biệt trong nhiều bài toán mình từng làm trong lĩnh vực compute vision, phương pháp tiền xử lí ảnh này cho chất lượng dữ liệu rất cao, cải thiện chất lượng model deep learning rất nhiều.



**2. Công thức.**

Mình nhận ra các bài viết về histogram equalization thường sơ sài chỉ có code, lại có những bài hơi nặng về công thức toán, gây khó hiểu cho những bạn mới. Trong bài này mình sẽ diễn giải rõ hơn về thuật toán. Mình sẽ làm việc với ảnh H1.

Gọi hàm biến đổi ta cần xác định là K(i)*K*(*i*) với i \epsilon [0,255]*iϵ*[0,255] . Với hình H1, ta cần xác định K sao cho K(150) \approx 0*K*(150)≈0, tức pixel có giá trị 150 được thay thế bằng giá trị 0.

**B1**: Thống kê số lượng pixel cho từng mức sáng, ta được histogram H(i)*H*(*i*)

**B2**: Tính "hàm tích lũy" H'*H*′ cho từng mức sáng theo công thứcA picture containing text

Description automatically generated

Trong đó H(i)*H*(*i*) chính là tổng số pixel có giá trị \leqslant i⩽*i* . Trên hình H1, đường màu đỏ chính là đường minh họa H'*H*′ (hàm đồng biến). (Note: Giá trị H'(i) trên hình đã được thay đổi để dễ dàng show trên cùng 1 biểu đồ).

Giả sử H'(140) = 100, H'(150) = 200, H'(160) = 5000. Thấy H(150) - H(140)*H*(150)−*H*(140) = 100, trong khi H(150) - H(140)*H*(150)−*H*(140) = 4800 (tức có tận 4800 pixel nằm trong khoảng [150,160][150,160] ). Ta cần 1 hàm biến đổi sao cho K(150) - K(140)*K*(150)−*K*(140) << K(160) - K(150)*K*(160)−*K*(150) ... (<< là nhỏ hơn nhiều).

**B3**: Hàm biến đổi K tại một mức sáng i được tính như sau:

K(i) = \dfrac{H'(i) - min(H')}{max(H') - min(H')}\* 255*K*(*i*)=*max*(*H*′)−*min*(*H*′)*H*′(*i*)−*min*(*H*′)​∗255

A picture containing diagram

Description automatically generated

Với hàm biến đổi K, với H'(i) - H'(j) nhỏ thì K(i) - K(j) nhỏ, và ngược lại. Nói cách khác, các khoảng sáng có nhiều pixel thì được giãn ra, những khoảng sáng có ít pixel thì được co lại.

import numpy

import cv2

import matplotlib.pyplot as plt

def compute\_hist(img):

hist = np.zeros((256,), np.uint8)

h, w = img.shape[:2]

for i in range(h):

for j in range(w):

hist[img[i][j]] += 1

return hist

def equal\_hist(hist):

cumulator = np.zeros\_like(hist, np.float64)

for i in range(len(cumulator)):

cumulator[i] = hist[:i].sum()

print(cumulator)

new\_hist = (cumulator - cumulator.min())/(cumulator.max() - cumulator.min()) \* 255

new\_hist = np.uint8(new\_hist)

return new\_hist

img = cv2.imread("img.png", 0)

hist = compute\_hist(img).ravel()

new\_hist = equal\_hist(hist)

h, w = img.shape[:2]

for i in range(h):

for j in range(w):

img[i,j] = new\_hist[img[i,j]]

fig = plt.figure()

ax = plt.subplot(121)

plt.imshow(img, cmap='gray')

plt.subplot(122)

plt.plot(new\_hist)

plt.show()