AI VIETNAM
Free AI Course 2020

Statistics and Applications

Quang-Vinh Dinh
Hong-Phuc Nguyen

Year 2020

Outline

- Mean and Median
- Range and Histogram
- Variance
- Correlation Coefficient

Mean

Data

$$X = \{X_1, \dots, X_N\}$$

Formula

$$\mu = \frac{1}{N}\sum_{i-1}^{N} X_i$$

Given the data

$$X = \{2, 8, 5, 4, 1, 8\}$$

$$N = 6$$

$$\mu = \frac{1}{N}\sum_{i-1}^{N} X_i = \frac{1}{6}(2 + 8 + 5 + 4 + 1 + 8)$$

$$= \frac{18}{6} = 3$$

Mean

❖ Code

```
1.  def calculate_mean(numbers): #1
2.      s = sum(numbers)         #2
3.      N = len(numbers)         #3
4.      mean = s/N               #4
5.      return mean              #5
6.
7.  # Tạo mảng donations đại diện cho số tiền quyên góp trong 12 ngày
8.  donations = [100, 60, 70, 900, 100, 200, 500, 500, 503, 600, 1000,1200]
9.
10. mean_value = calculate_mean(donations)
11. print('Trung bình số tiền quyên góp là: ', mean_value)
```

#1. Đặt tên là calculate_mean(), hàm này sẽ nhận đối số numbers, là chuỗi các số cần tính trung bình.

#2. Sử dụng hàm sum() để tính tổng dãy số cho trước.

#3. Sử dụng hàm len() để tính chiều dài của dãy số cần tính.

#4. Tính trung bình của dãy số trên bằng cách lấy tổng chia cho chiều dài.

#5. Cuối cùng ta cho hàm trả về giá trị mean tính được.

Mean

❖ Application

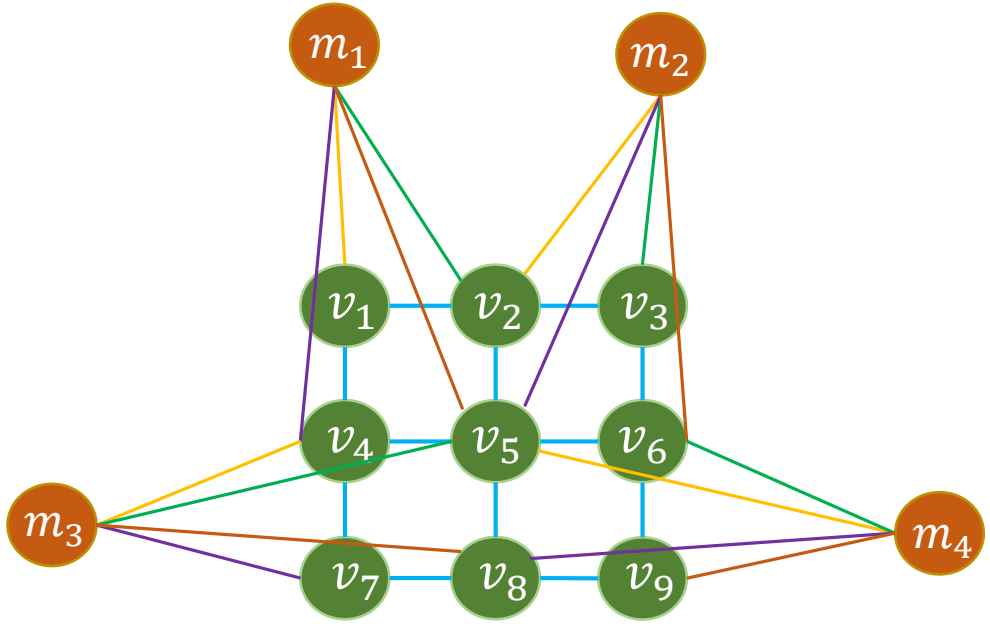
Làm mờ ảnh
dựa vào mean

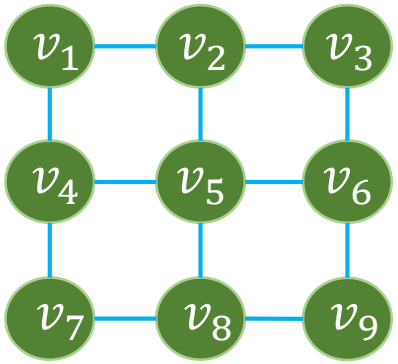
Mean

❖ Correlation (~convolution)

Kernel

$$m_1 = v_1 w_1 + v_2 w_2 + v_4 w_3 + v_5 w_4$$



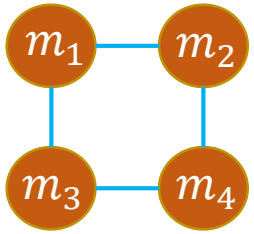


Image

$*$

Kernel

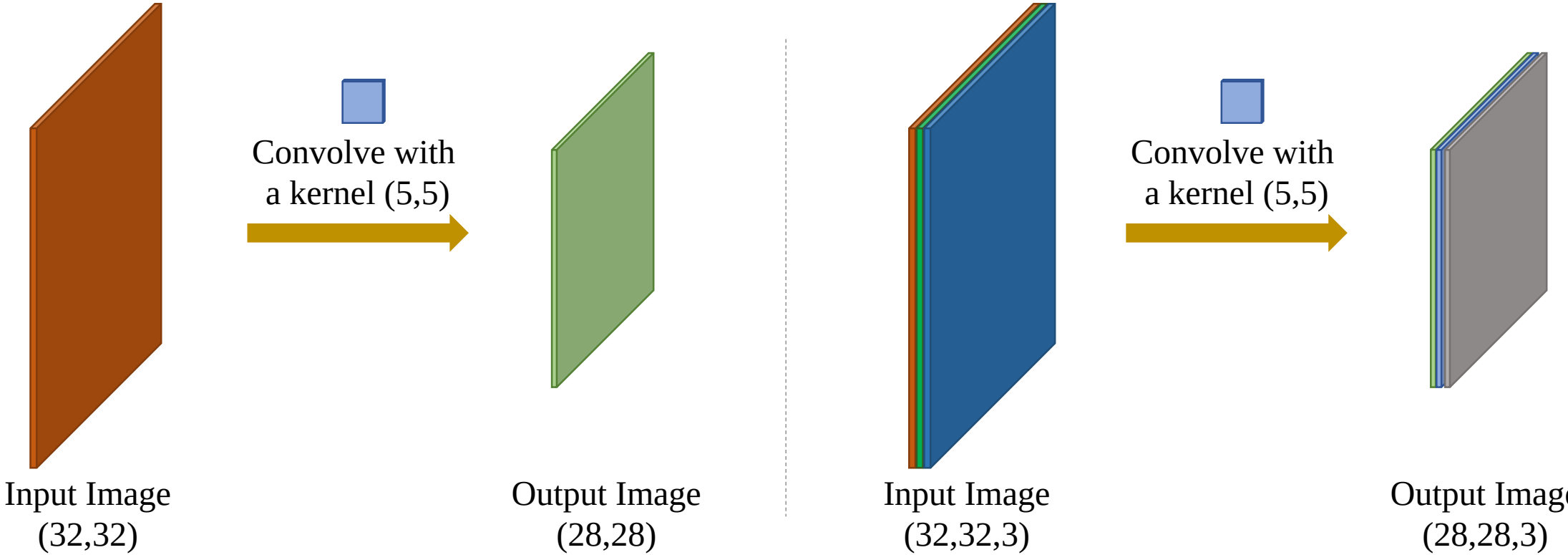
➡



Output

Mean

❖ Correlation (~convolution)



Mean

❖ Correlation (~convolution)

Numpy	np.einsum()
Scipy	scipy.signal.convolve2d()
OpenCV	cv2.filter2D()

Kernels for computing mean

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

(3x3) kernel

$$\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

(5x5) kernel

```
output_image = cv2.filter2D(input_image, cv2.CV_8U, kernel)
```

AI VIETNAM
Free AI Course

Mean

❖ Correlation (~convolution)

```python
# load image and blurring

import numpy as np
import cv2

# load image in grayscale mode
image = cv2.imread('mrbean.jpg', 0)

# create kernel
kernel = np.ones((5,5), np.float32) / 25.0

# compute mean for each pixel
dst = cv2.filter2D(image, cv2.CV_8U, kernel)

# show images
cv2.imshow('image', image)
cv2.imshow('dst', dst)

# waiting for any keys pressed and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Mean

❖ Numpy review

```python
# numpy review
import numpy as np

arr = np.ones((5,5))
print(arr)

roi = arr[1:4, 1:4]
roi = roi + 1
print(roi)

arr[1:4, 1:4] = roi
print(arr)
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

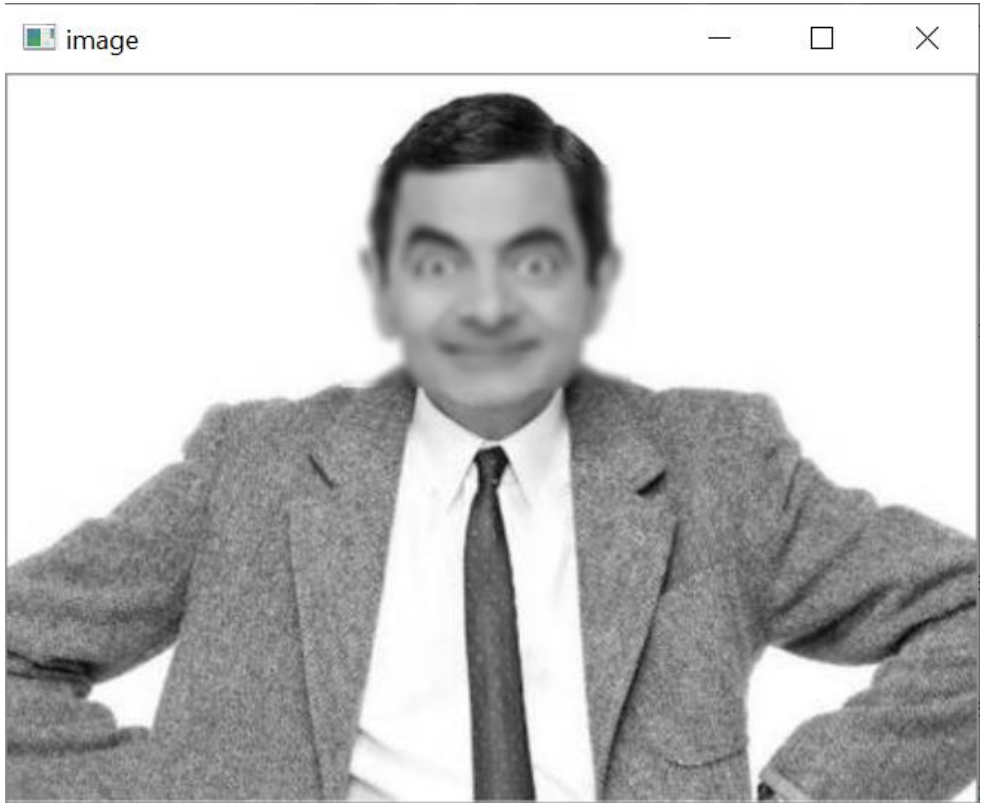
```
[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]
```

```
[[1. 1. 1. 1. 1.]
 [1. 2. 2. 2. 1.]
 [1. 2. 2. 2. 1.]
 [1. 2. 2. 2. 1.]
 [1. 1. 1. 1. 1.]]
```

Mean

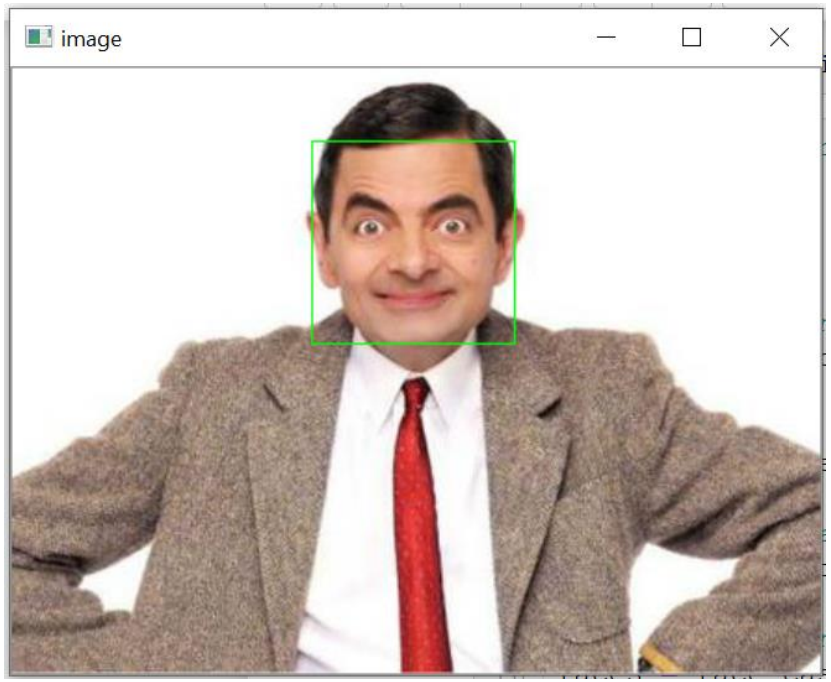
❖ Image Blurring

```python
# load image and blurring using mask-simple

import numpy as np
import cv2

# load image in grayscale mode
image = cv2.imread('mrbean.jpg', 0)

# create kernel
kernel = np.ones((5,5), np.float32) / 25.0

# Select ROI (top_y,top_x,height, width)
roi = image[40:140,150:280]

# compute mean for each pixel
roi = cv2.filter2D(roi, cv2.CV_8U, kernel)

image[40:140,150:280] = roi

# show images
cv2.imshow('roi', roi)
cv2.imshow('image', image)

# waiting for any keys pressed and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Mean

❖ Image Blurring

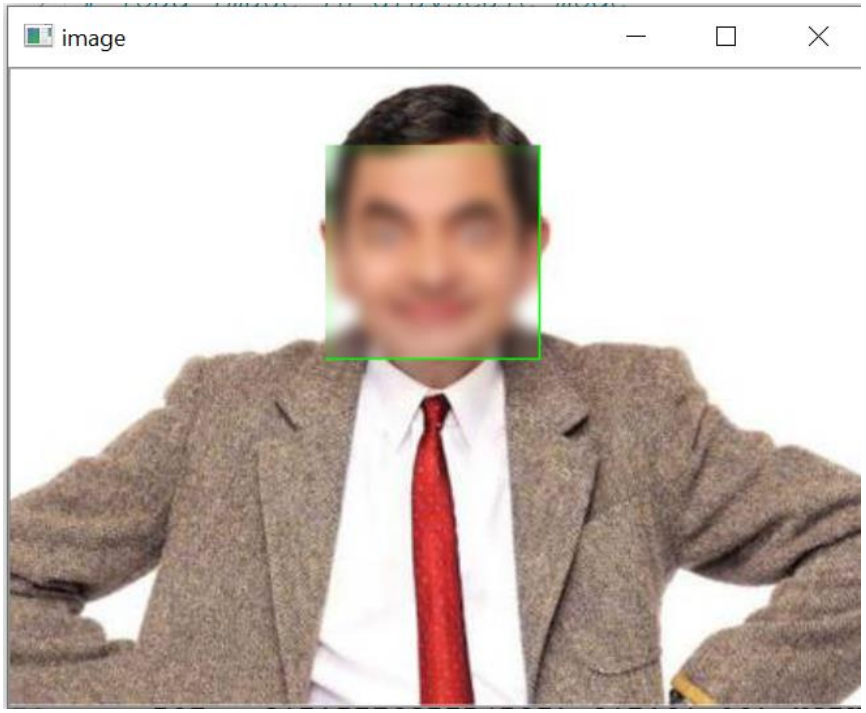



```python
# load image and blurring using face detection

import numpy as np
import cv2

# face detection setup
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# load image in grayscale mode
image = cv2.imread('mrbean.jpg', 1)

# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# face detection
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

# Draw the rectangle around each face
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 1)

# show images
cv2.imshow('image', image)

# waiting for any keys pressed and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```


https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81

Mean

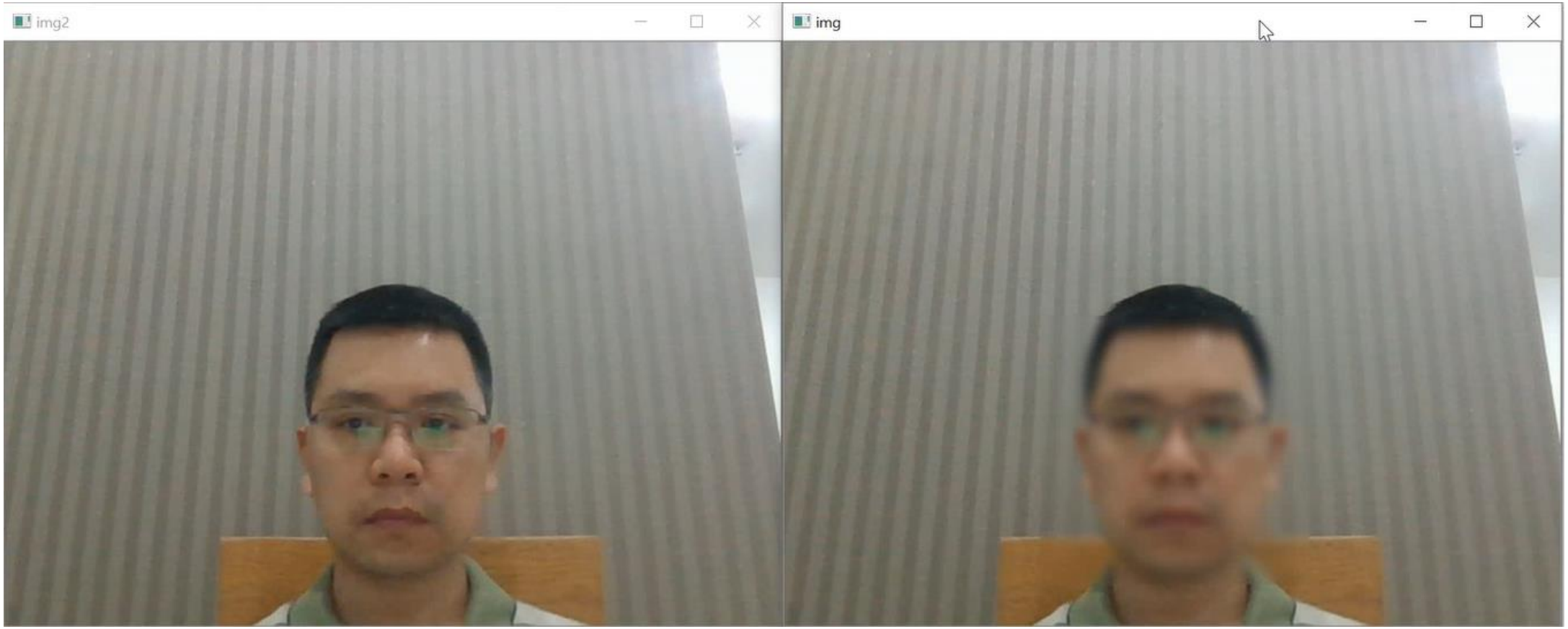
❖ Image Blurring




```python
# load image and blurring using face detection

import numpy as np
import cv2

# face detection setup
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# load image in grayscale mode
image = cv2.imread('mrbean.jpg', 1)

# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# face detection
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

# create kernel
kernel = np.ones((7,7), np.float32) / 49.0

# Draw the rectangle around each face
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 1)
    roi = image[y:y+h,x:x+w]

    # compute mean for each pixel
    roi = cv2.filter2D(roi, cv2.CV_8U, kernel)
    roi = cv2.filter2D(roi, cv2.CV_8U, kernel)
    roi = cv2.filter2D(roi, cv2.CV_8U, kernel)

    # update
    image[y:y+h,x:x+w] = roi

# show images
cv2.imshow('image', image)

# waiting for any keys pressed and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Mean

❖ Image Blurring

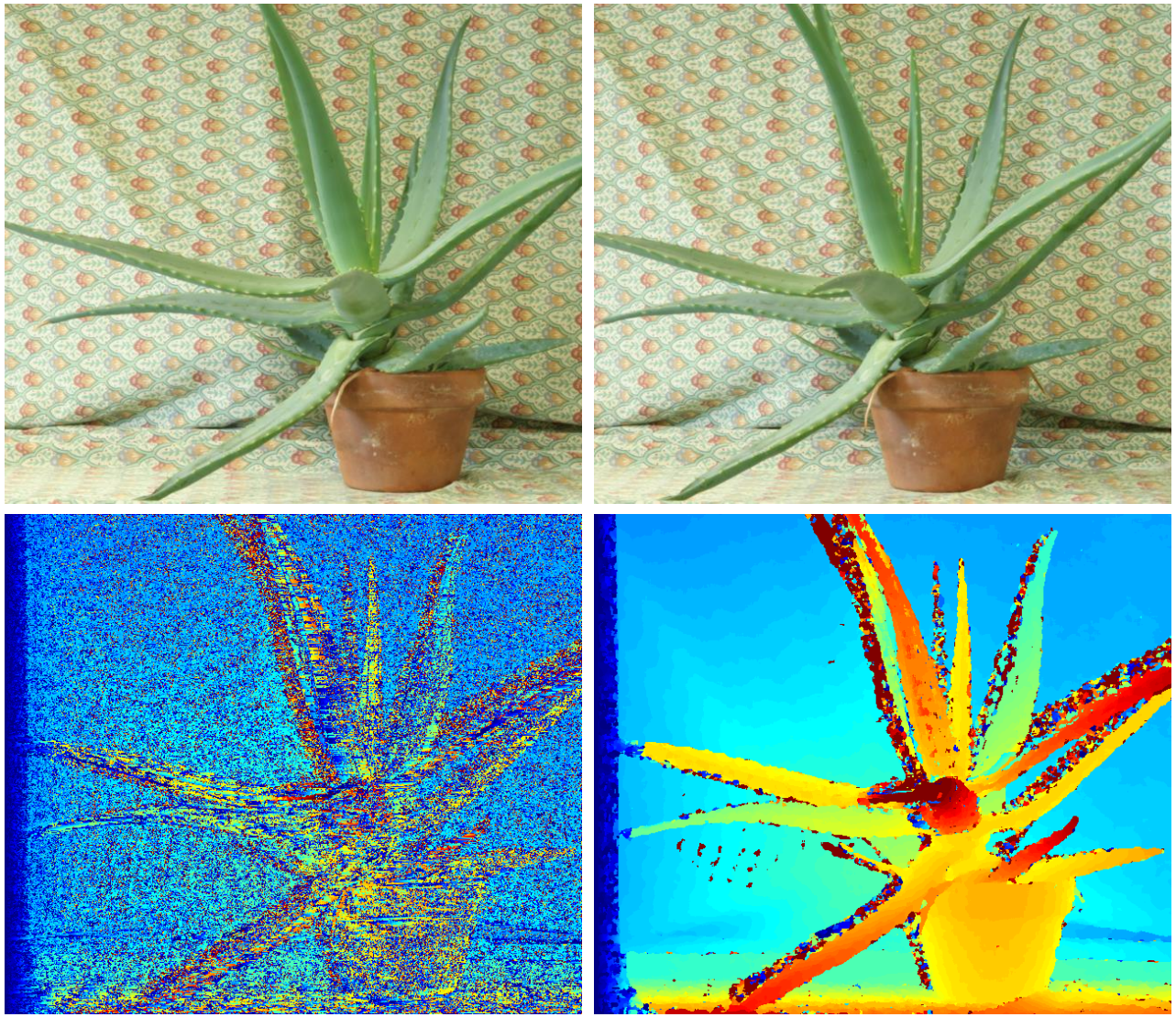


https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81

Video demo

Mean

- **Stereo matching**



Median

Data

$$X = \{X_1, \dots, X_N\}$$

Formula

Step 1: Sort X → S

Step 2

If N is odd, then $m = S_{\left(\frac{N+1}{2}\right)}$

If N is even, then $m = \left(S_{\left(\frac{N}{2}\right)} + S_{\left(\frac{N}{2}+1\right)}\right)/2$

Given the data

$$X = \{2, 8, 5, 4, 1\}$$

$$N = 5$$

Step 1

$$S = \{1, 2, 4, 5, 8\}$$

1 2 3 4 5

Step 2; $N = 5$

$$k = \frac{N+1}{2} = 3$$

$$m = S_k = 4$$

Median

Data

$$X = \{X_1, \dots, X_N\}$$

Formula

Step 1: Sort X → S

Step 2

If N is odd, then $m = S_{\left(\frac{N+1}{2}\right)}$

If N is even, then $m = \left(S_{\left(\frac{N}{2}\right)} + S_{\left(\frac{N}{2}+1\right)}\right)/2$

Given the data

$$X = \{2, 8, 5, 4, 1, 8\}$$

$$N = 6$$

Step 1

$$S = \{1, 2, 4, 5, 8, 8\}$$

1 2 3 4 5 6

Step 2; $N = 6$

$$m = \frac{S_3 + S_4}{2}$$

$$= \frac{4 + 5}{2} = 4.5$$

Median

❖ Code

```
1.  def calculate_median(numbers):                    #1
2.      N = len(numbers)                              #2
3.      numbers.sort()                                #3
4.      if N%2 == 0:                                  #4
5.          m1 = N/2
6.          m2 = (N/2) + 1
7.          m1 = int(m1)-1
8.          m2 = int(m2)-1
9.          median = (numbers[m1] + numbers[m2])/2
10.     else:                                         #5
11.         m = (N+1)/2
12.         m = int(m)-1
13.         median = numbers[m]
14.     return median                                 #6
```

Median

❖ Image Denoising

Input Image

(3x3) kernel

(5x5) kernel

Median

❖ Image Denoising

```python
import numpy as np
import cv2

img1 = cv2.imread('mrbean_noise.jpg')
img2 = cv2.medianBlur(img1, 3)

# show images
cv2.imshow('img1', img1)
cv2.imshow('img2', img2)

# waiting for any keys pressed and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Mean and Median

❖ Comparison

Data

$$X = \{X_1, \dots, X_N\}$$

Formula

$$\mu = \frac{1}{N}\sum_{i-1}^{N} X_i$$

Formula

Step 1: Sort X → S

Step 2

If N is odd, then $m = S_{\left(\frac{N+1}{2}\right)}$

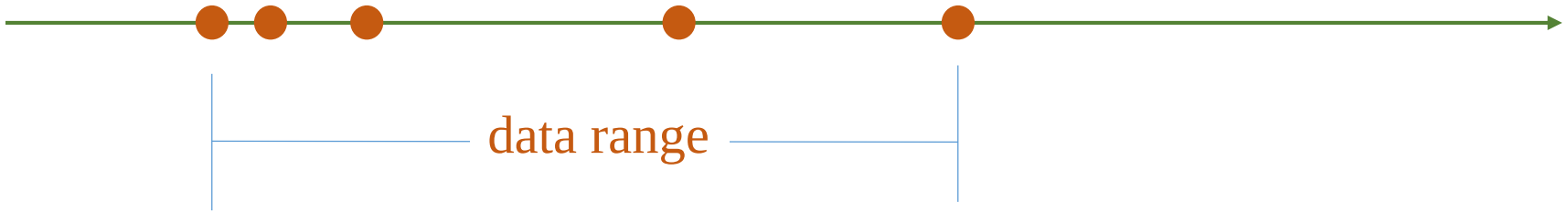
If N is even, then $m = \left(S_{\left(\frac{N}{2}\right)} + S_{\left(\frac{N}{2}+1\right)}\right)/2$

Outline

- Mean and Median
- Range and Histogram
- Variance
- Correlation Coefficient

Range

❖ Procedure

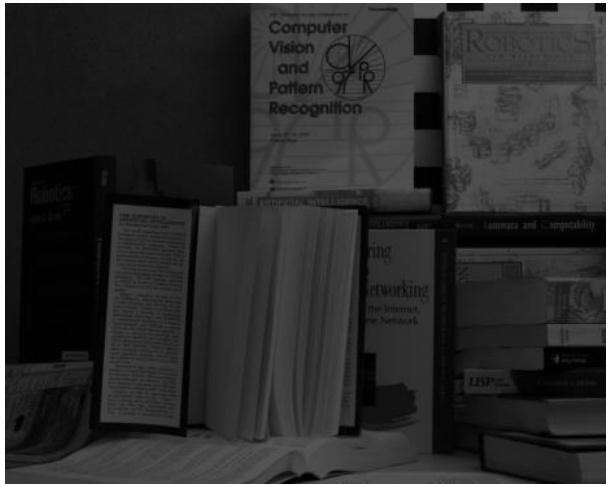


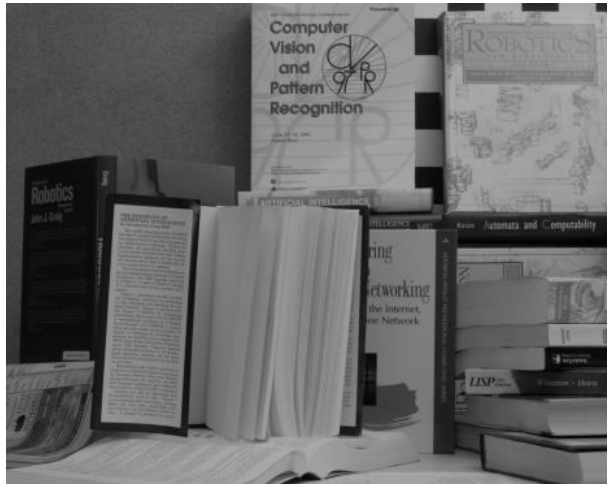

```
1.  def find_range(numbers):          #1
2.      lowest  = min(numbers)        #2
3.      highest = max(numbers)        #3
4.      r = highest-lowest            #4
5.      print('Lowest: {0}\tHighest: {1}\tRange: {2}'.format(lowest, highest, r))
6.
7.  # data
8.  points = [7, 8, 9, 2, 10, 9, 9, 9, 9, 4, 5, 6, 1, 5, 6, 7, 8, 6, 1, 10, 6, 6]
9.  find_range(points)
```

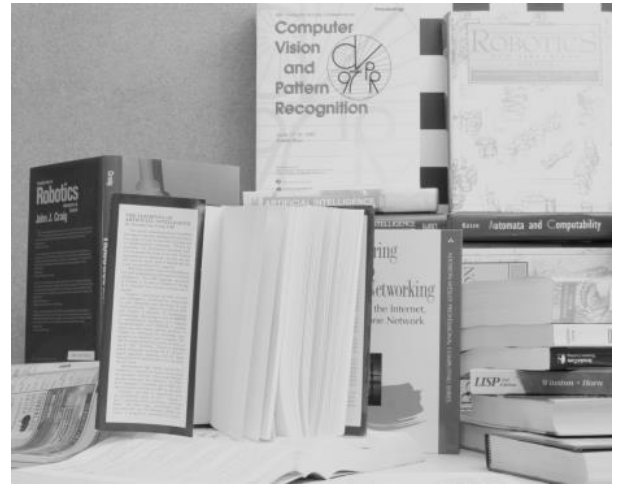
Histogram cho ảnh grayscale

Pixel có 256 giá trị khác nhau (từ 0 đến 255) → chia thành 256 bin. Mỗi bin chứa (đếm) số lượng pixel có cùng giá trị.

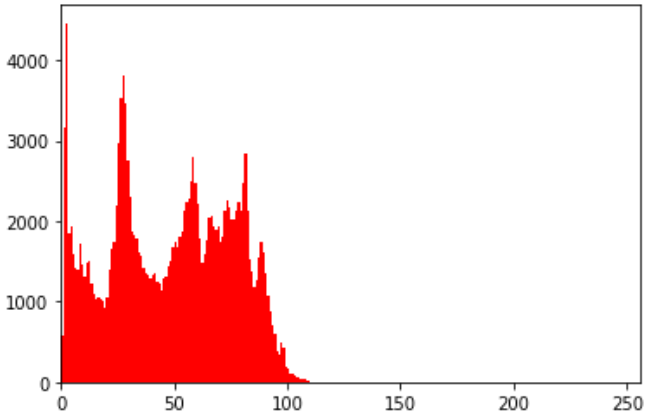
Ảnh



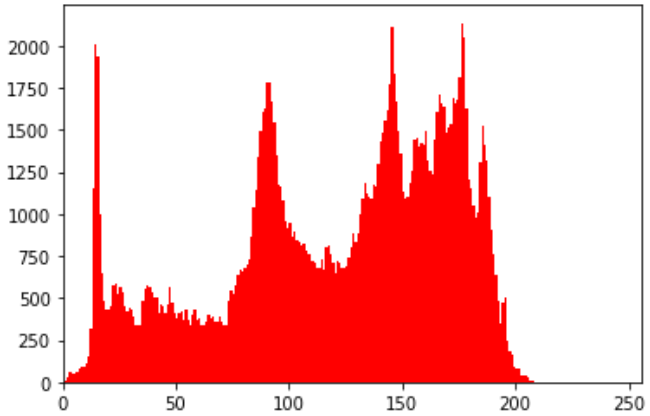




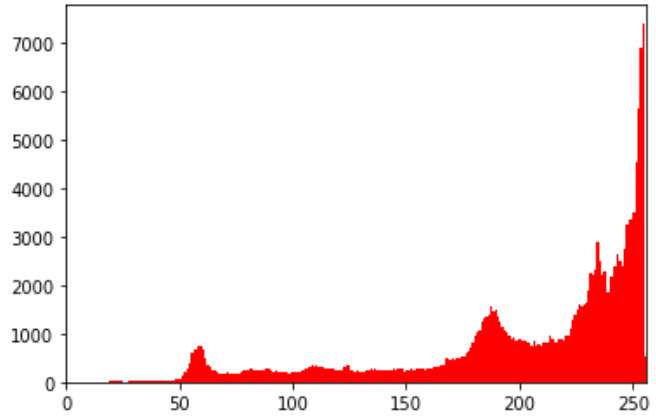
Histogram



256 bin

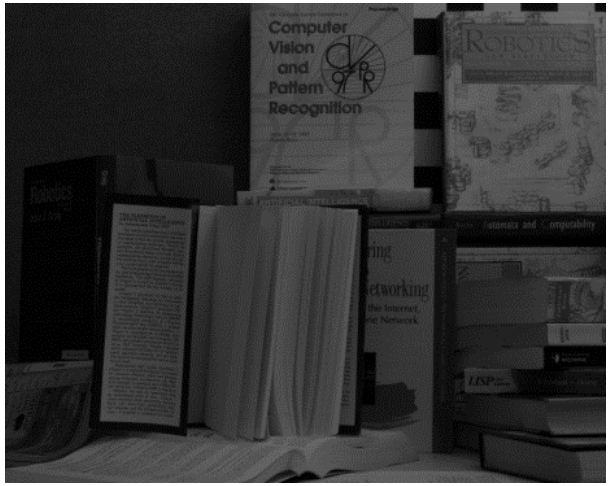


256 bin

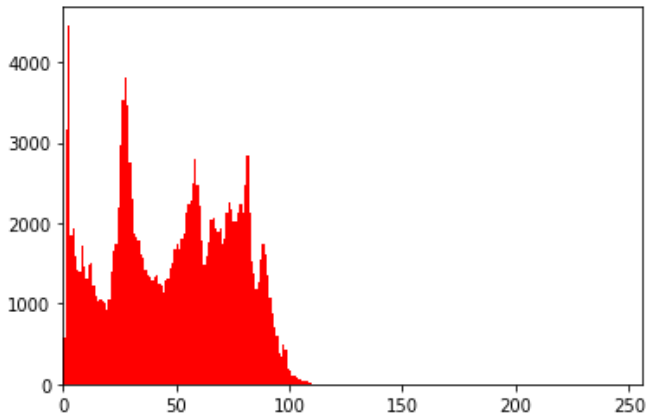


256 bin

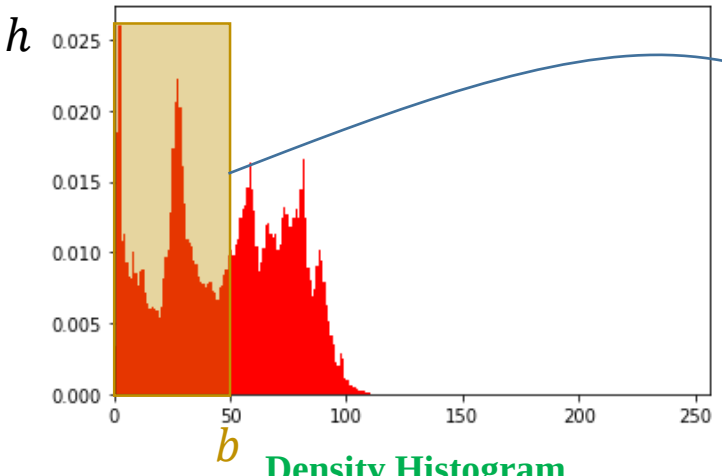
Histogram cho ảnh grayscale



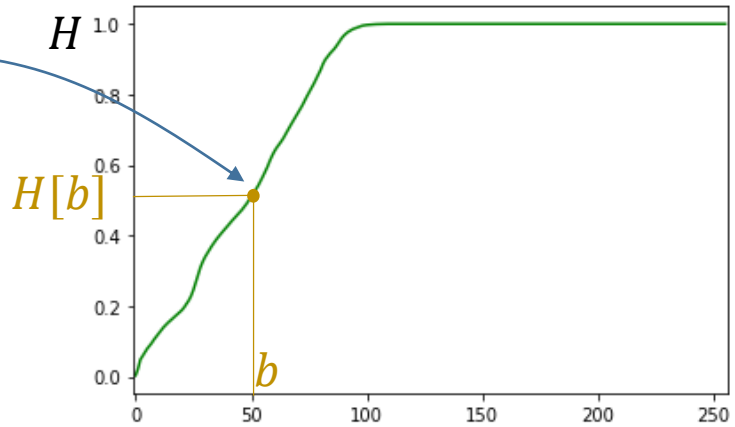
Ảnh



Histogram



Density Histogram



Cumulative Histogram

Gọi N là tổng số pixel của ảnh và n_b là số lượng pixel ở bin thứ b.

Giá trị density histogram ở bin thứ b được tính như sau:

$$h[b] = \frac{n_b}{N}$$

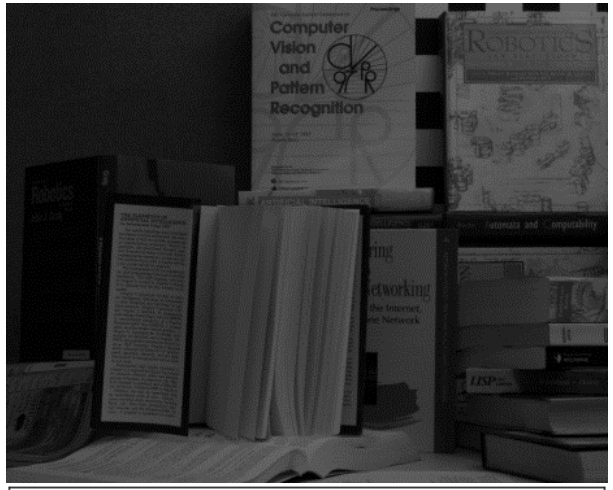
Giá trị cumulative histogram ở bin thứ b được tính như sau:

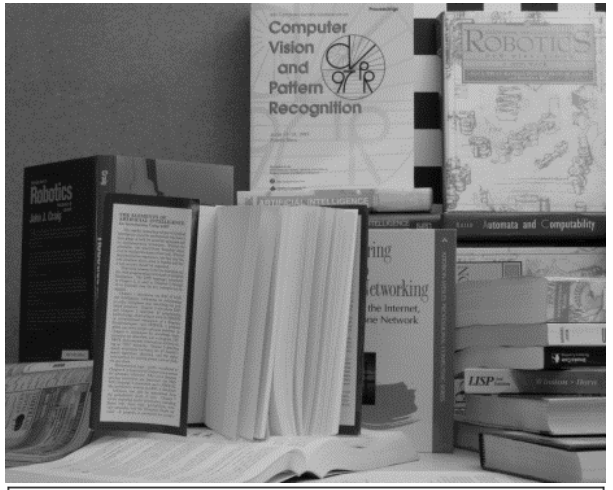
$$H[b] = \sum_{k=0}^{b} h[k]$$

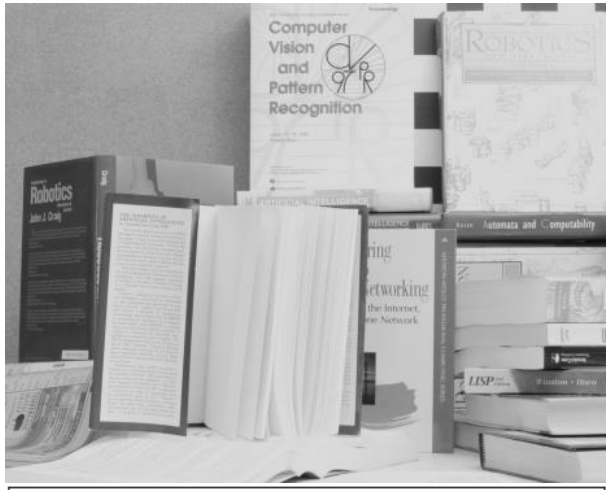
Cộng dồn giá trị $h[k]$ từ 0 đến b

Histogram cho ảnh grayscale

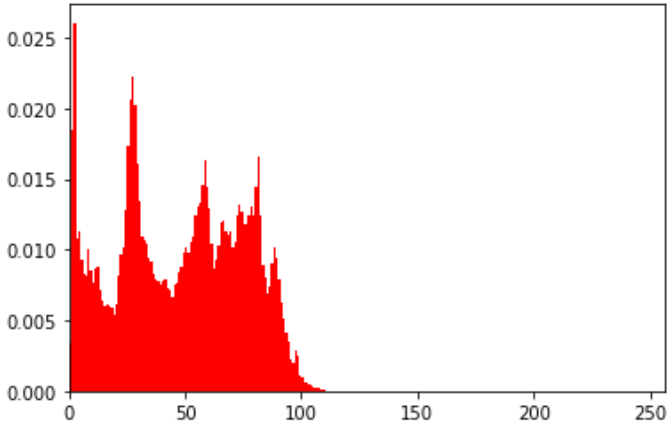
Ảnh

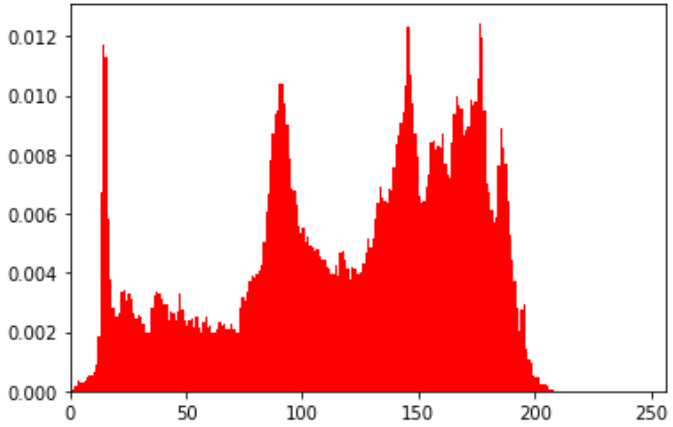


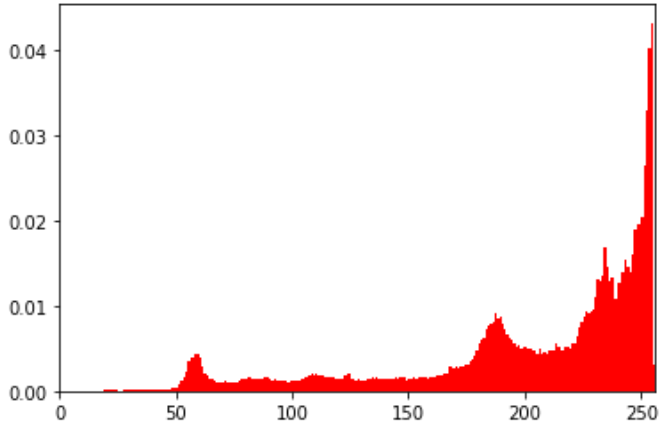




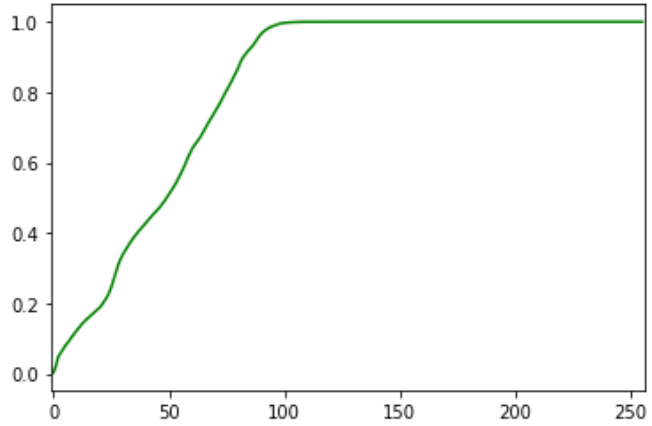
Density Histogram

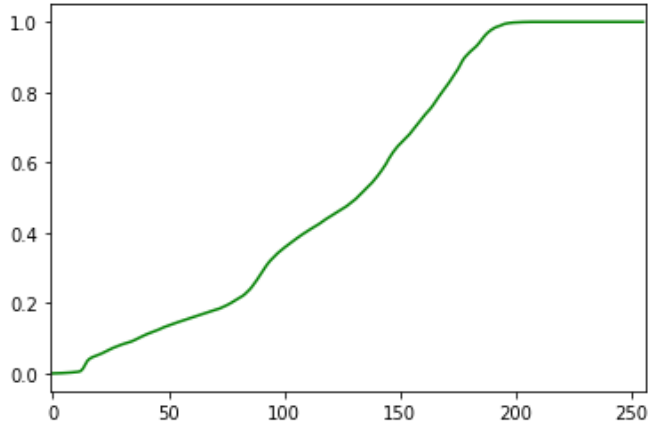


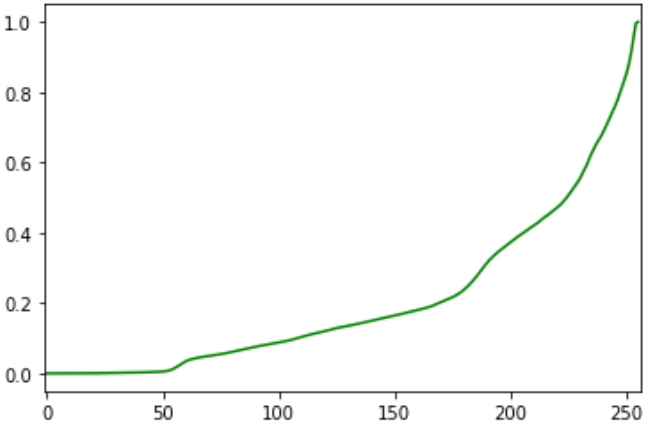




Cumulative Histogram



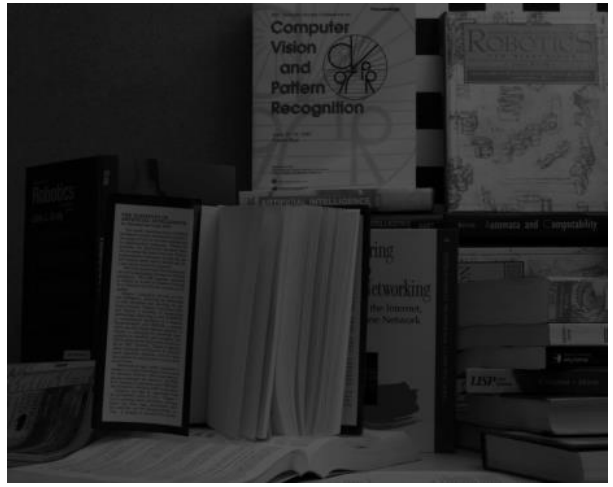




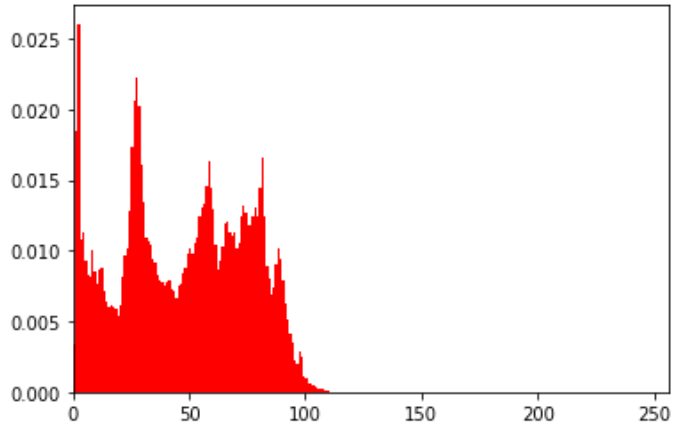
Histogram equalization

Được dùng để tăng độ tương phản của ảnh

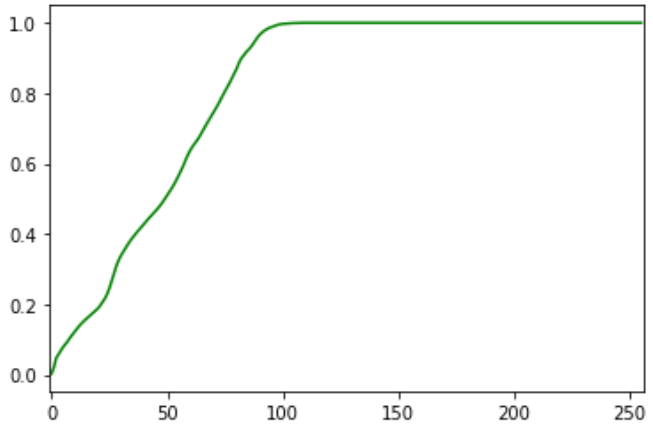
Idea: Kéo phân bố của density histogram sao cho xấp xỉ với uniform distribution.



Ảnh



Density Histogram

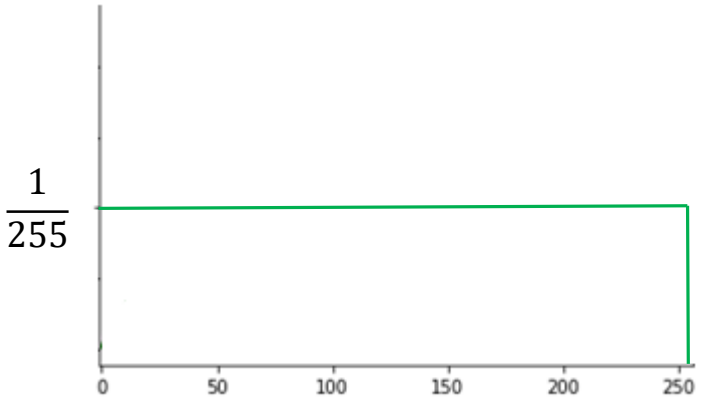


Cumulative Histogram

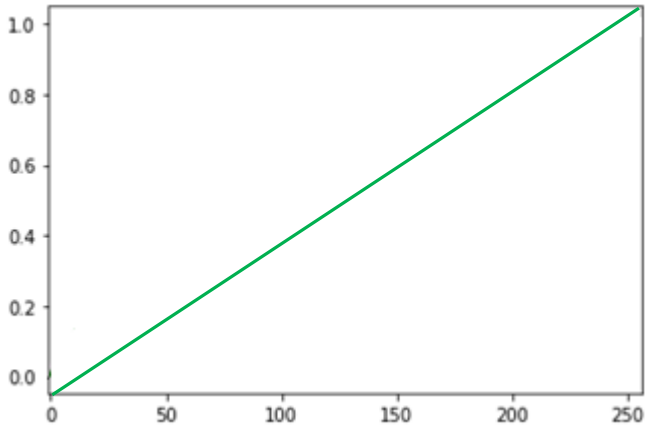
Công thức

$$b_{new} = [255 * H[b] + 0.5]$$

Các bin (giá trị pixel) được thay đổi theo $H[b]$



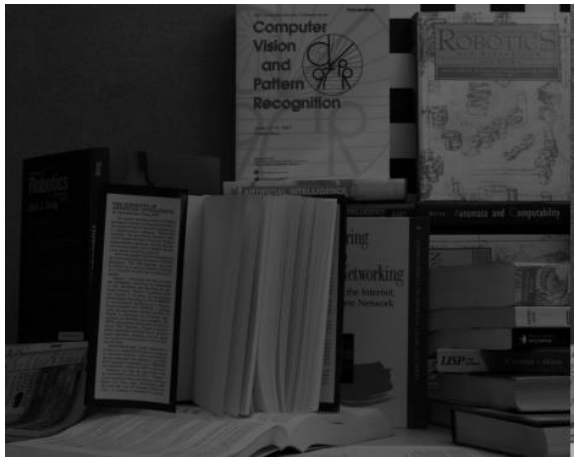
Uniform density function trong khoảng [0,255]



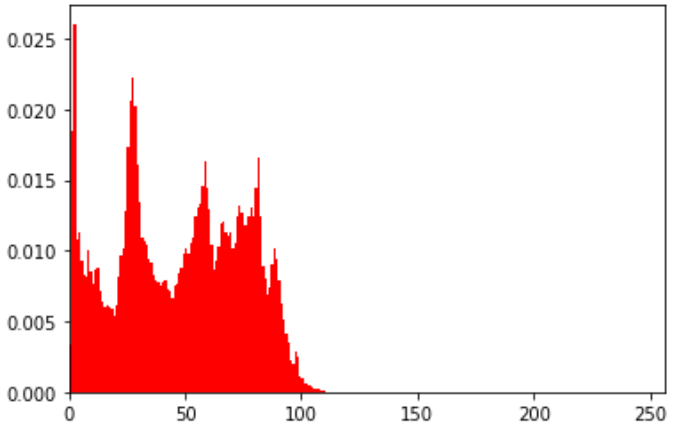
Cumulative density function cho uniform distribution

Histogram equalization để tăng độ tương phản

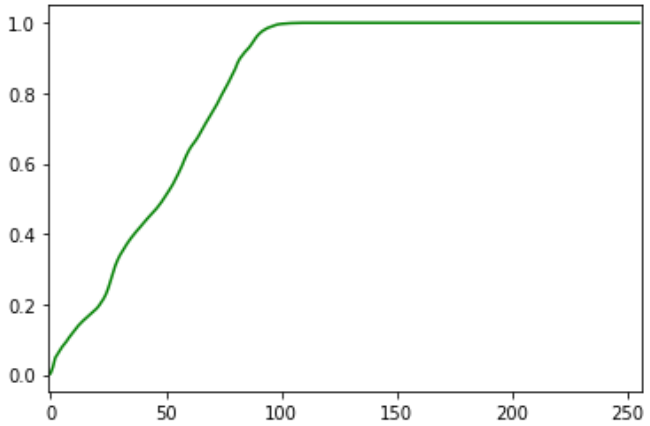
Cho ảnh grayscale



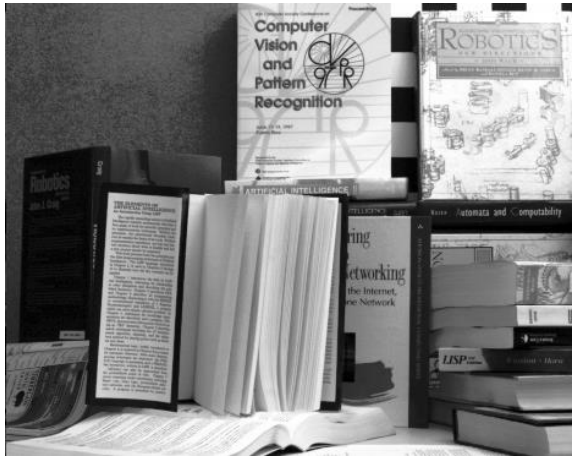
Ảnh gốc



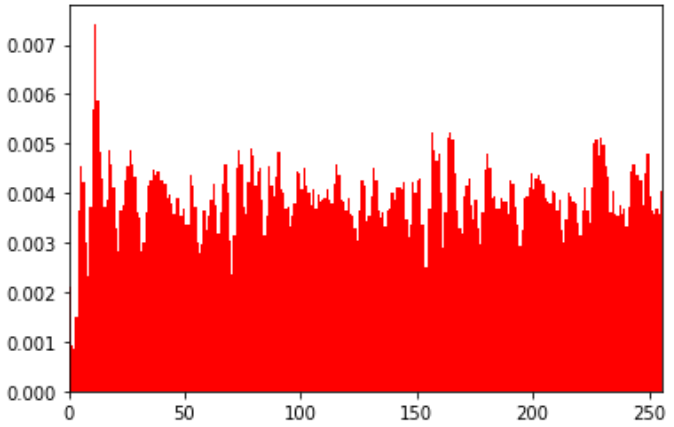
Density Histogram



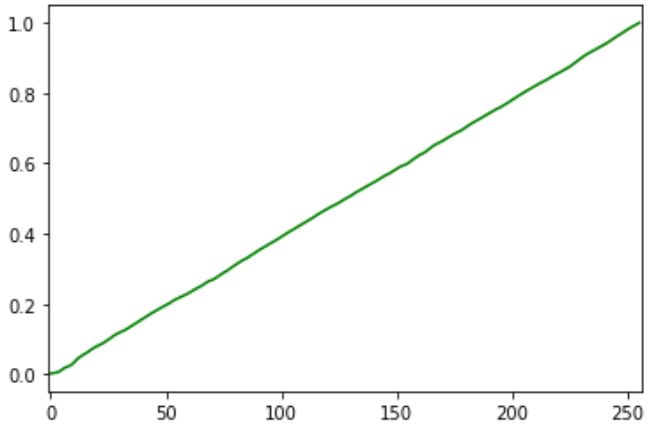
Cumulative Histogram



Ảnh kết quả



Density Histogram



Cumulative Histogram

Histogram equalization để tăng độ tương phản

Cho ảnh grayscale

Ảnh gốc

Ảnh kết quả

Cho ảnh màu

Ảnh gốc

Ảnh kết quả

Outline

- Mean and Median
- Range and Histogram
- Variance
- Correlation Coefficient

Variance

Formula:

mean $\mu = \frac{1}{n}\sum_{k=1}^{n} x_i$

variance $var(X) = \frac{1}{n}\sum_{k=1}^{n} (x_i - \mu)^2$

Standard deviation $\sigma = \sqrt{var(X)}$

Example: $X = \{5, 3\ 6, 7, 4\}$

$$\mu = \frac{1}{5}\sum_{k=1}^{n} (5+3+6+7+4) = \frac{25}{5} = 5$$

$$var(X) = \frac{1}{5}[(5-5)^2 + (3-5)^2 + (6-5)^2 + (7-5)^2 + (4-5)^2]$$

$$= \frac{1}{5}(0+4+1+4+1) = 2$$

$$\sigma = \sqrt{var(X)} = 1.41$$

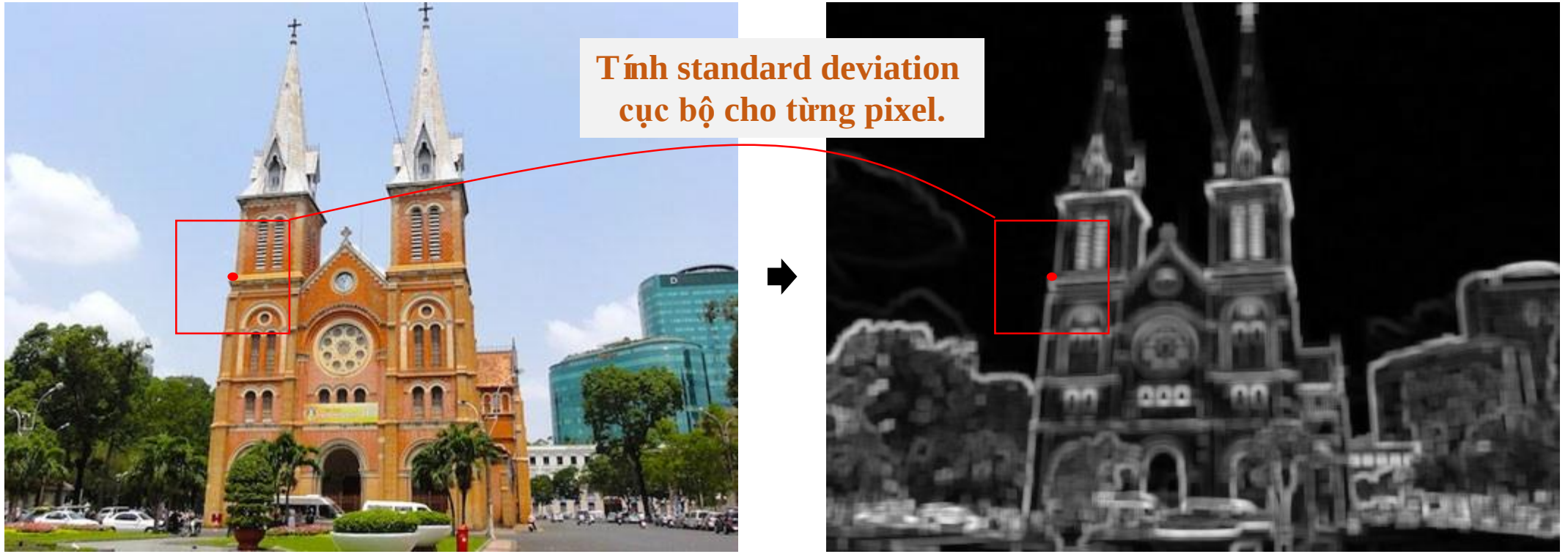
Variance

❖ Code

```python
# variance
def calculate_mean(numbers):                    #1
    s = sum(numbers)
    N = len(numbers)
    mean = s/N
    return mean

def caculate_variance(numbers):                 #2
    mean = calculate_mean(numbers)              #3

    diff = []                                   #4
    for num in numbers:
        diff.append(num-mean)

    squared_diff = []                           #5
    for d in diff:
        squared_diff.append(d**2)

    sum_squared_diff = sum(squared_diff)
    variance = sum_squared_diff/len(numbers)

    return variance
```

Variance

Ứng dụng tính chất của variance (~standard deviation) để tìm texture cho một hình



Ảnh gốc

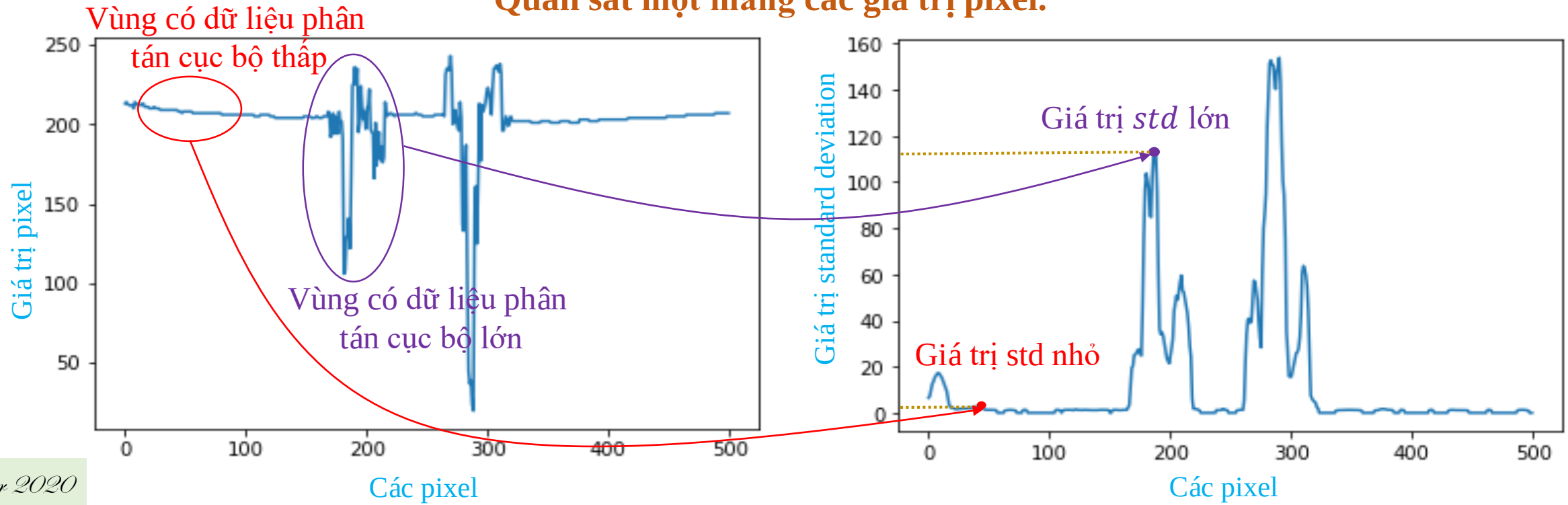
Ảnh thông tin texture

Tính standard deviation cục bộ cho từng pixel.

Ảnh gốc

Ảnh thông tin texture

Quan sát một mảng các giá trị pixel.

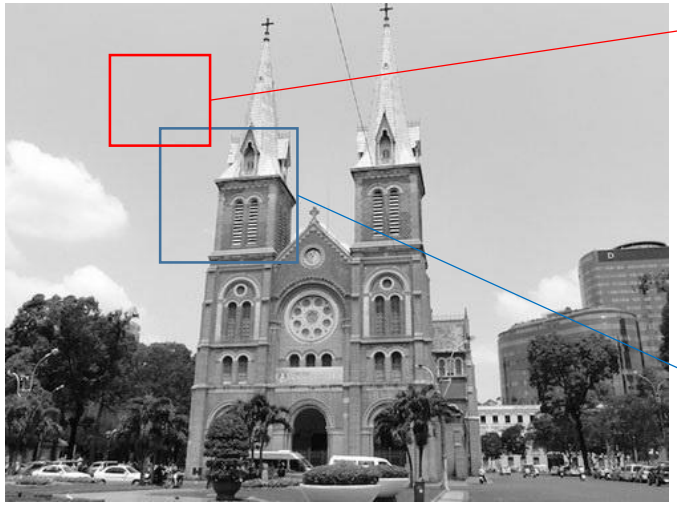


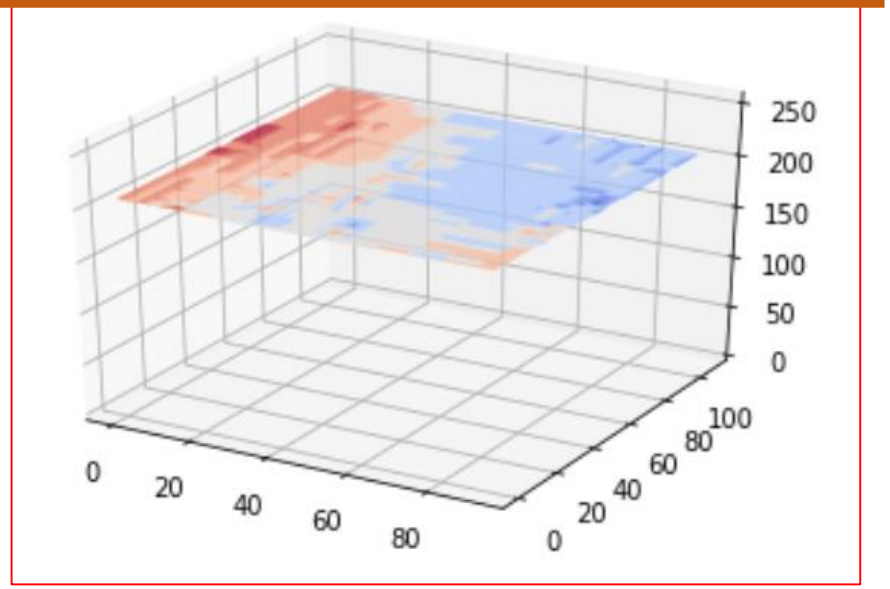
Variance

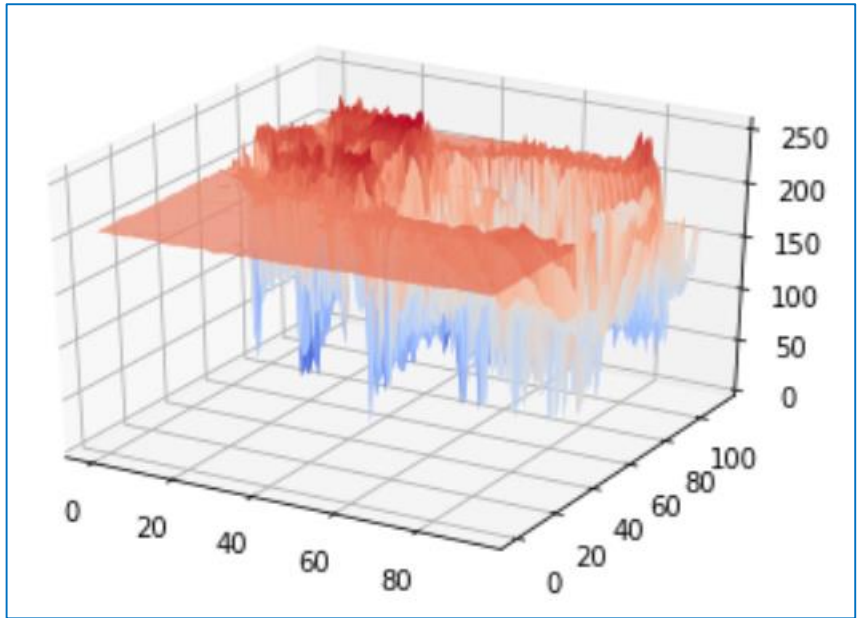
❖ Implementation

Variance

❖ Implementation







Variance

❖ Implementation

```python
import numpy as np
import cv2
import math
from scipy.ndimage.filters import generic_filter

img  = cv2.imread('img.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imwrite('edge_s1.jpg', gray)

x = gray.astype('float')
x_filt = generic_filter(x, np.std, size=7)
cv2.imwrite('edge_s2.jpg', x_filt)

x_filt[x_filt < 20] = 0
cv2.imwrite('edge_s3.jpg', x_filt)

maxv = np.max(x_filt)
print(maxv)

x_filt = x_filt*2.5
cv2.imwrite('edge_s4.jpg', x_filt)
```

Outline

- Mean and Median
- Range and Histogram
- Variance
- Correlation Coefficient

Hệ số tương quan (correlation coefficient)

Công thức: Gọi x,y là hai biến ngẫu nhiên

$$\rho_{xy} = \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}$$

Tính chất 1

$$-1 \leq \rho_{xy} \leq 1$$

Tương quan nghịch ← | → Tương quan thuận

Tính chất 2

$$\rho_{xy} = \rho_{uv}$$

trong đó

$$u = ax + b$$

$$v = cy + d$$

Ví dụ 1

x = [7, 18, 29, 2, 10, 9, 9]

y = [1, 6, 12, 8, 6, 21, 10]

$$\rho_{xy} = \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n*818 - 84*64}{\sqrt{n*1480 - 7056}\sqrt{n*822 - 4096}} = 0.149$$

Ví dụ 2

u=2*x-14 = [0, 22, 44, -10, 6, 4, 4]

v=y+2 = [3, 8, 14, 10, 8, 23, 12]

$$\rho_{uv} = \frac{E[(u-\mu_u)[(v-\mu_v)]}{\sqrt{var(u)}\sqrt{var(v)}}$$

$$= \frac{n*880 - 70*78}{\sqrt{n*2588 - 4900}\sqrt{n*1106 - 6084}} = 0.149$$

Hệ số tương quan (correlation coefficient)

Công thức: Gọi x,y là hai biến ngẫu nhiên

$$\rho_{xy} = \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}$$

Tính chất: $\rho_{xy}(x,y) = \rho_{xy}(ax,by)$

$$\rho_{xy}(ax,by) = \frac{E[(ax-\mu_{ax})[(by-\mu_{by})]}{\sqrt{var(ax)}\sqrt{var(by)}}$$

$$= \frac{n(\sum_i ax_i by_i) - (\sum_i ax_i)(\sum_i by_i)}{\sqrt{n\sum_i a^2x_i^2 - (\sum_i ax_i)^2}\sqrt{n\sum_i b^2y_i^2 - (\sum_i by_i)^2}}$$

$$= \frac{n(ab\sum_i x_i y_i) - (a\sum_i x_i)(b\sum_i y_i)}{\sqrt{na^2\sum_i x_i^2 - a^2(\sum_i x_i)^2}\sqrt{nb^2\sum_i y_i^2 - b^2(\sum_i y_i)^2}}$$

$$= \frac{ab[n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)]}{a\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\, b\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}$$

$$= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}} = \rho_{xy}$$

Hệ số tương quan (correlation coefficient)

Công thức: Gọi x,y là hai biến ngẫu nhiên

$$\rho_{xy} = \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}$$

Tính chất: $\rho_{xy}(x,y) = \rho_{xy}(x+c, y+d)$

$$\rho_{xy}(x+c, y+d) = \frac{E[((x+c)-\mu_{(x+c)})[((y+d)-\mu_{(y+d)})]}{\sqrt{var(x+c)}\sqrt{var(y+d)}}$$

$$= \frac{E[((x+c)-(\mu_x+c))[((y+d)-(\mu_y+d))]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}} = \rho_{xy}$$

Correlation Coefficient

Công thức: Gọi x,y là hai biến ngẫu nhiên

$$\rho_{xy} = \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}$$

```
def find_corr_x_y(x,y):                                    #1
    n = len(x)                                             #2
    prod = []
    for xi,yi in zip(x,y):                                 #3
        prod.append(xi*yi)

    sum_prod_x_y = sum(prod)                               #4

    sum_x = sum(x)
    sum_y = sum(y)

    squared_sum_x = sum_x**2
    squared_sum_y = sum_y**2

    x_square = []
    for xi in x:
        x_square.append(xi**2)
    x_square_sum = sum(x_square)

    y_square=[]
    for yi in y:
        y_square.append(yi**2)
    y_square_sum = sum(y_square)

    # Use formula to calculate correlation                 #5
    numerator = n*sum_prod_x_y - sum_x*sum_y
    denominator_term1 = n*x_square_sum - squared_sum_x
    denominator_term2 = n*y_square_sum - squared_sum_y
    denominator = (denominator_term1*denominator_term2)**0.5
    correlation = numerator/denominator

    return correlation
```

Correlation Coefficient

Công thức: Gọi x,y là hai biến ngẫu nhiên

$$\rho_{xy} = \frac{E[(x-\mu_x)[(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}$$

```
def find_corr_x_y(x,y):                                    #1
    n = len(x)                                             #2
    prod = []
    for xi,yi in zip(x,y):                                 #3
        prod.append(xi*yi)

    sum_prod_x_y = sum(prod)                               #4

    sum_x = sum(x)
    sum_y = sum(y)

    squared_sum_x = sum_x**2
    squared_sum_y = sum_y**2

    x_square = []
    for xi in x:
        x_square.append(xi**2)
    x_square_sum = sum(x_square)

    y_square=[]
    for yi in y:
        y_square.append(yi**2)
    y_square_sum = sum(y_square)

    # Use formula to calculate correlation                 #5
    numerator = n*sum_prod_x_y - sum_x*sum_y
    denominator_term1 = n*x_square_sum - squared_sum_x
    denominator_term2 = n*y_square_sum - squared_sum_y
    denominator = (denominator_term1*denominator_term2)**0.5
    correlation = numerator/denominator

    return correlation
```

Ứng dụng cho patch matching

P_1 P_2 P_3 P_4

$\rho_{P_1P_2} = 0.55$

$\rho_{P_1P_3} = 0.23$ ➧ Ảnh P_2 giống với ảnh P_1 hơn so với P_3 và P_4

$\rho_{P_1P_4} = 0.30$

P_1 $P_2 = P_1+50$ $P_3 = 1.2P_1+10$

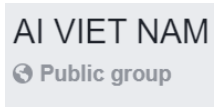
$\rho_{P_1P_2} = 0.9970$

$\rho_{P_1P_3} = 0.9979$ ➧ ρ hoạt động tốt dưới sự thay đổi tuyến tính

```python
# aivietnam.ai

import numpy as np
from PIL import Image

# load ảnh và chuyển về kiểu list
image1 = Image.open('images/img1.png')
image2 = Image.open('images/img2.png')
image3 = Image.open('images/img3.png')
image4 = Image.open('images/img4.png')

image1_list = np.asarray(image1).flatten().tolist()
image2_list = np.asarray(image2).flatten().tolist()
image3_list = np.asarray(image3).flatten().tolist()
image4_list = np.asarray(image4).flatten().tolist()


# tính correlation coefficient
corr_1_2 = find_corr_x_y(image1_list, image2_list)
corr_1_3 = find_corr_x_y(image1_list, image3_list)
corr_1_4 = find_corr_x_y(image1_list, image4_list)

print('corr_1_2:', corr_1_2)
print('corr_1_3:', corr_1_3)
print('corr_1_4:', corr_1_4)
```

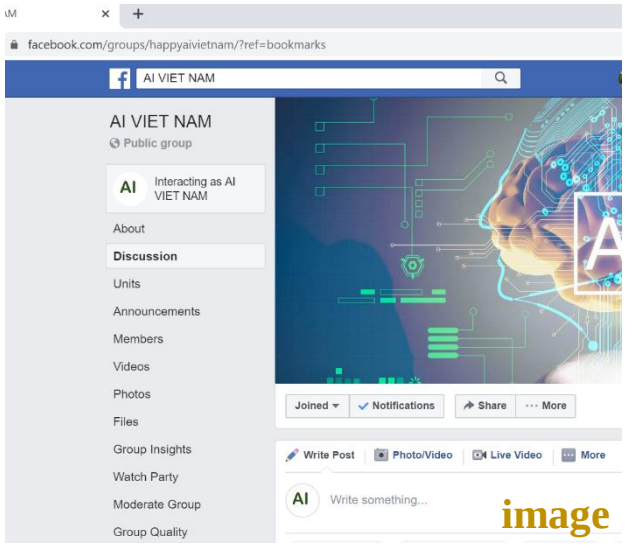
Correlation Coefficient

❖ Application to template matching



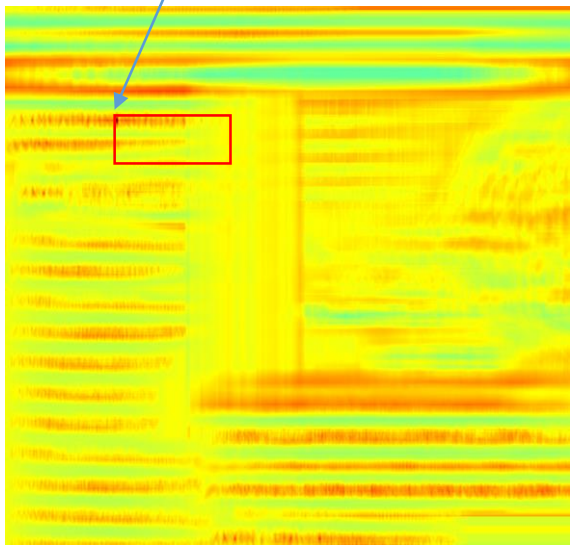
template

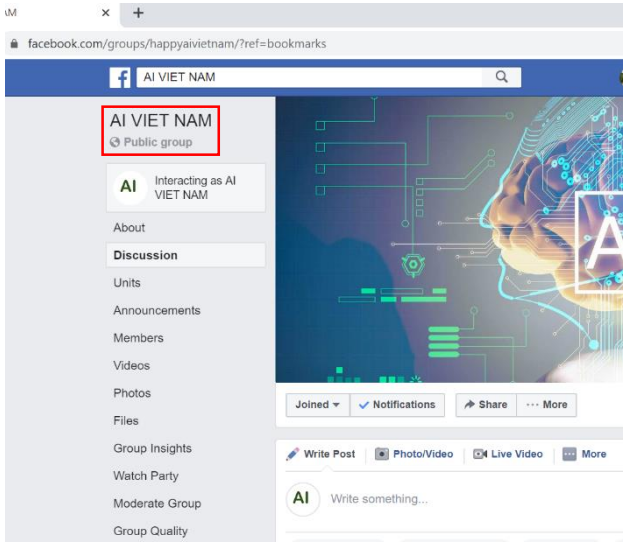
Tìm template có trong hình image



image

max

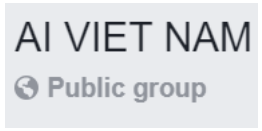




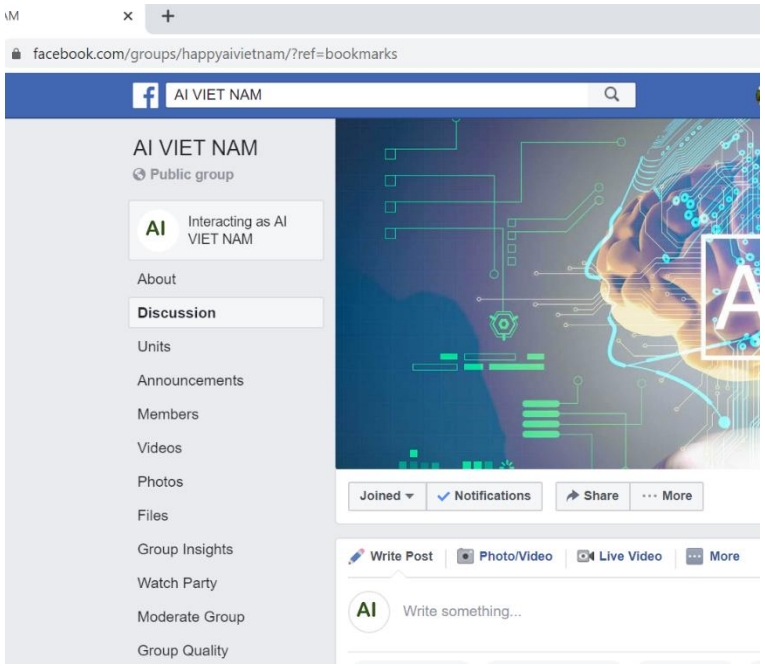
Kết quả

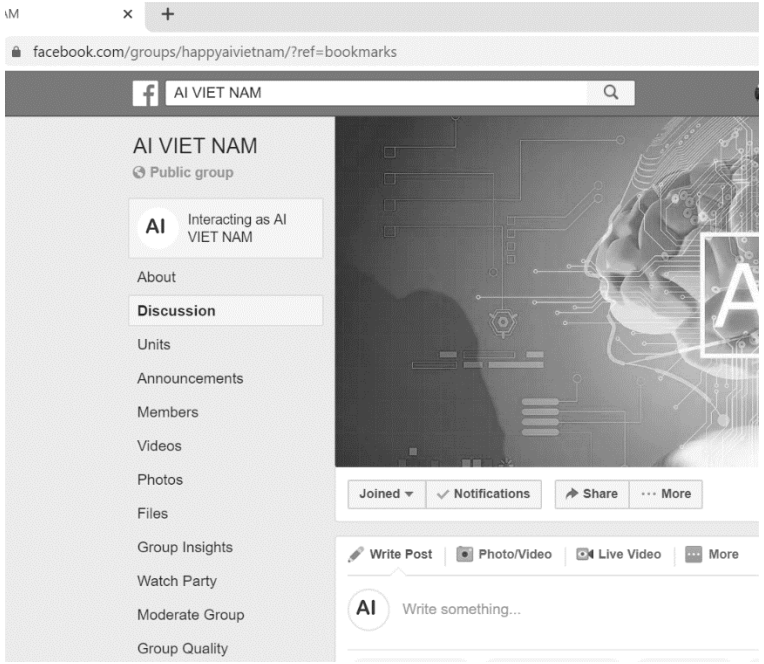
Correlation Coefficient

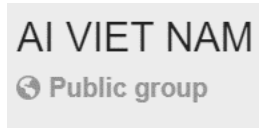
❖ Application to template matching



Color images







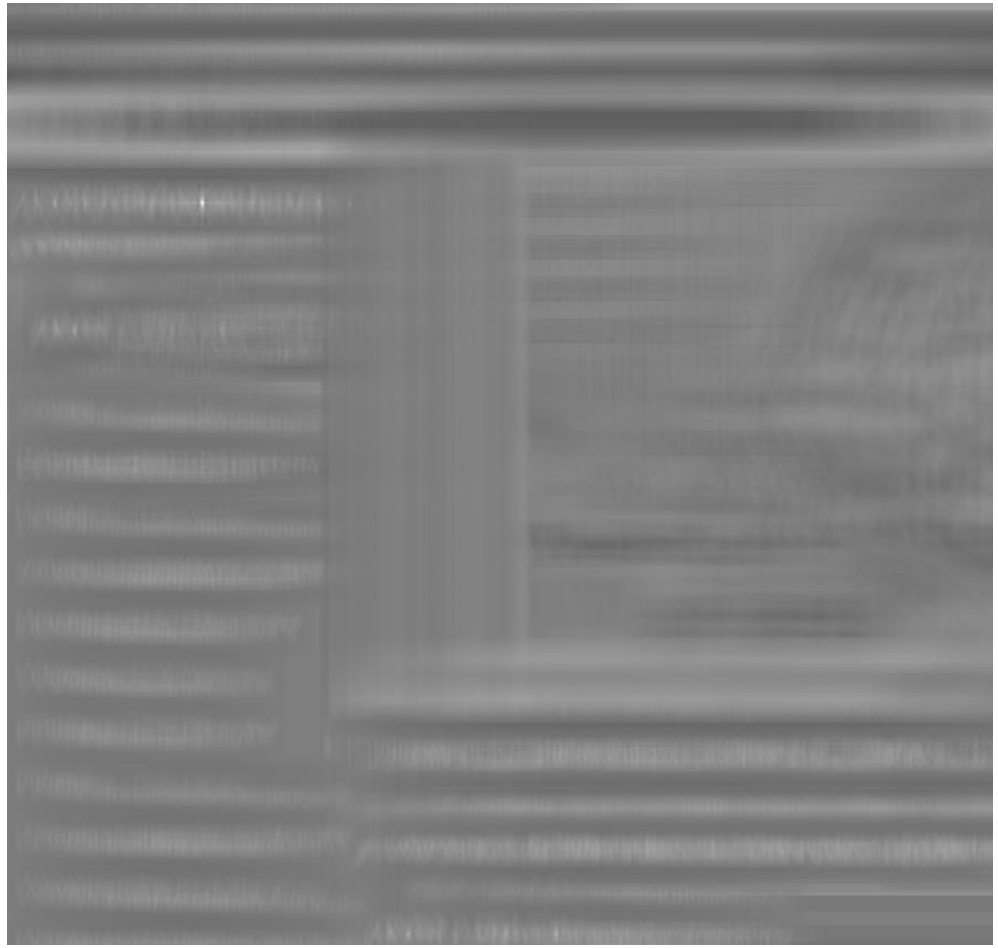
Grayscale images

```
grayscale_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)
output = cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)
```

Correlation Coefficient

❖ Application to template matching

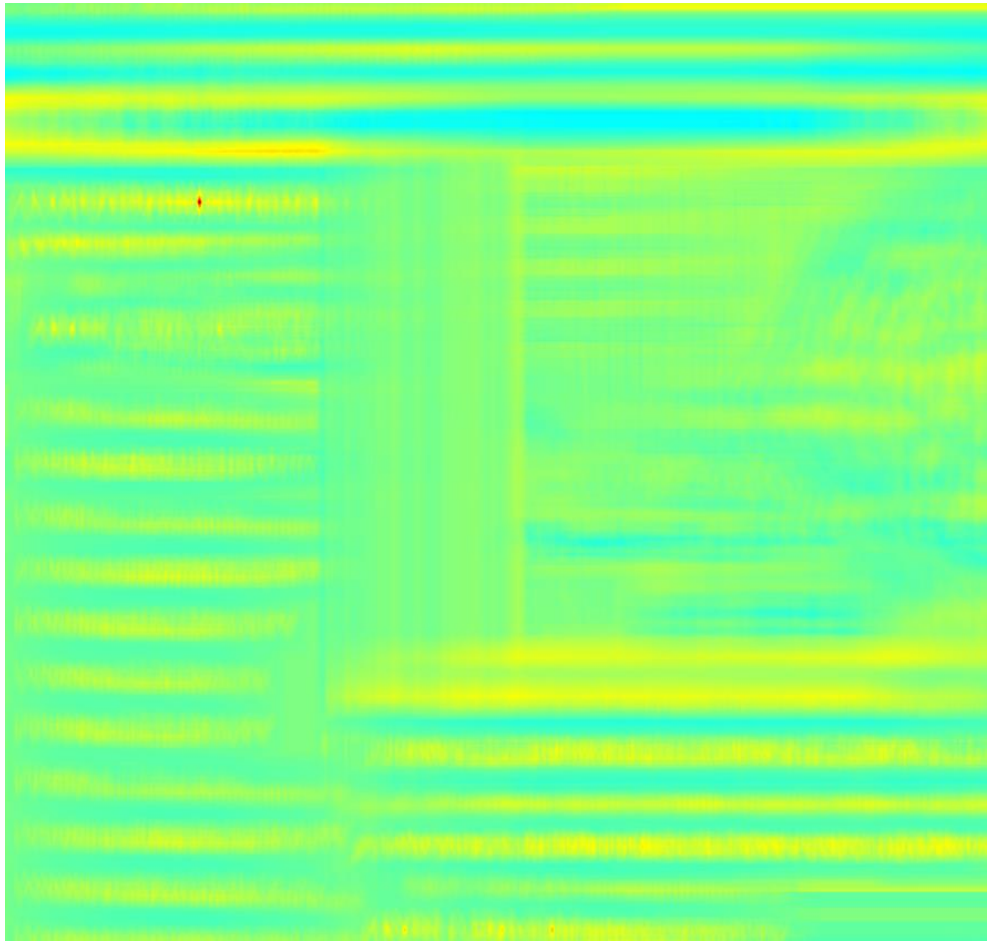
```python
# template matching

import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load image and convert to grayscale
image = cv.imread('image.png',1)
gray  = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

print(type(gray[0][0]))

template = cv.imread('template.png',0)
w, h = template.shape[::-1]

# Apply template Matching
corr_map = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF_NORMED)

# save correlation map
corr_map = (corr_map+1.0)*127.5
corr_map = corr_map.astype('uint8')
cv.imwrite('corr_map_grayscale.png', corr_map)
```



Correlation Map

Correlation Coefficient

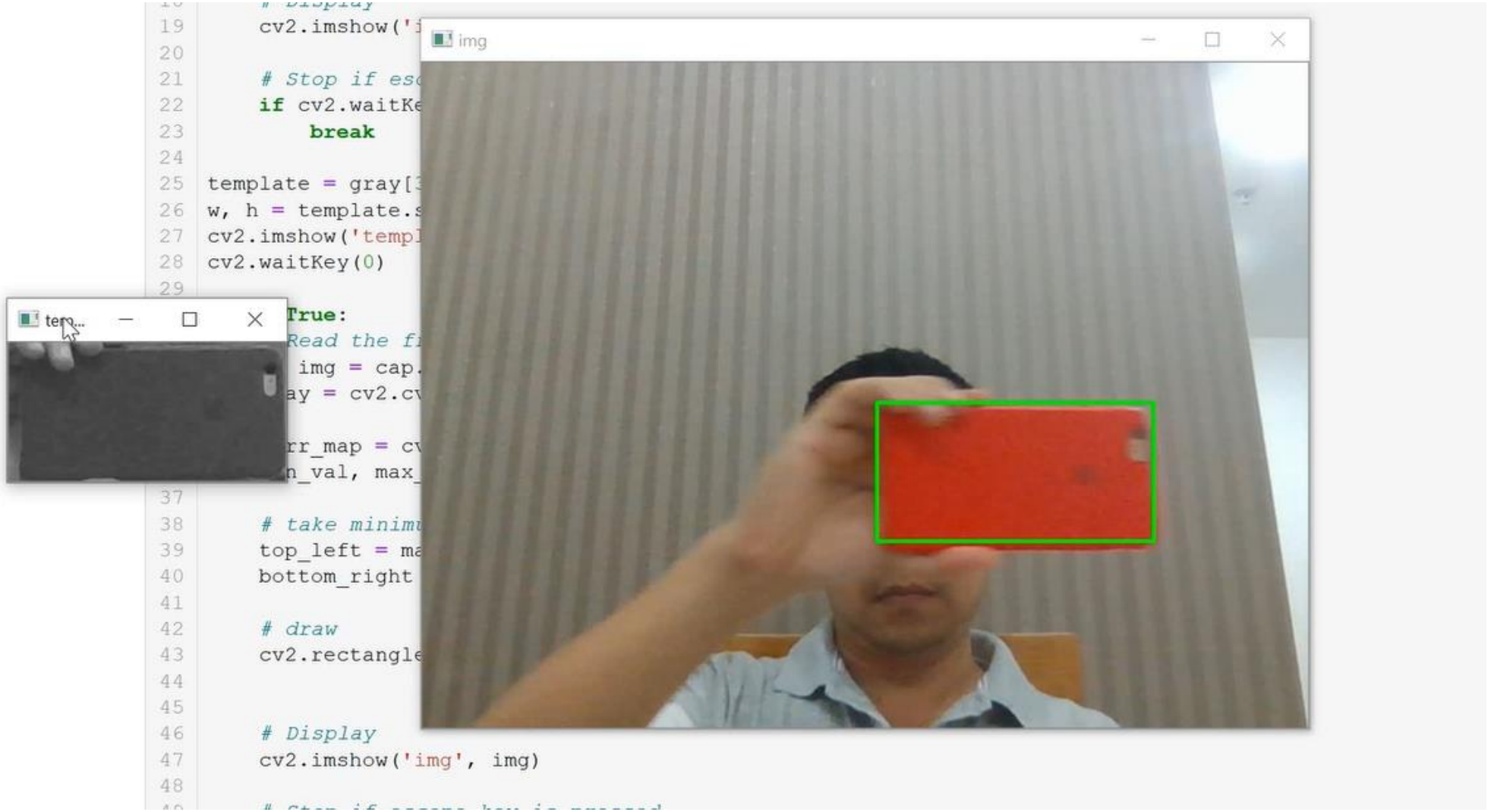
- **Application to template matching**



```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load image and convert to grayscale
image = cv.imread('image.png',1)
gray  = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

template = cv.imread('template.png',0)
w, h = template.shape[::-1]

# Apply template Matching
corr_map = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv.minMaxLoc(corr_map)

# take minimum
top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)

# draw
cv.rectangle(image, top_left, bottom_right, (0, 255, 0), 2)

# save corr map in grayscale
corr_map = (corr_map+1.0)*127.5
corr_map = corr_map.astype('uint8')
cv.imwrite('corr_map_grayscale.png', corr_map)

# applyColorMap
corr_map = cv2.applyColorMap(corr_map, cv2.COLORMAP_JET)

# save results
cv.imwrite('corr_map_color.png', corr_map)
cv.imwrite('result.png', image)
```

Correlation Coefficient

❖ Demo



Summary

- Mean and Median
- Range and Histogram
- Variance
- Correlation Coefficient


Thank you!