

Statistics and Applications

Outline

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

Mean

Data

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

Given the data

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

$$N = 6$$

Formula

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

$$\begin{aligned} \mu &= \frac{1}{N} \sum_{i=1}^N X_i = \frac{1}{6} (2 + 8 + 5 + 4 + 1 + 8) \\ &= \frac{18}{6} = 3 \end{aligned}$$

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



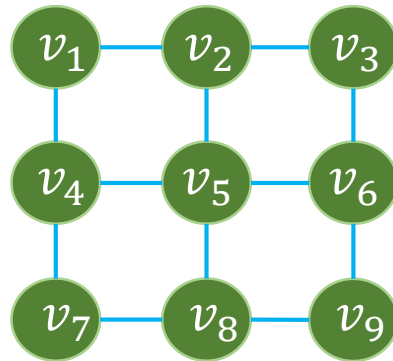
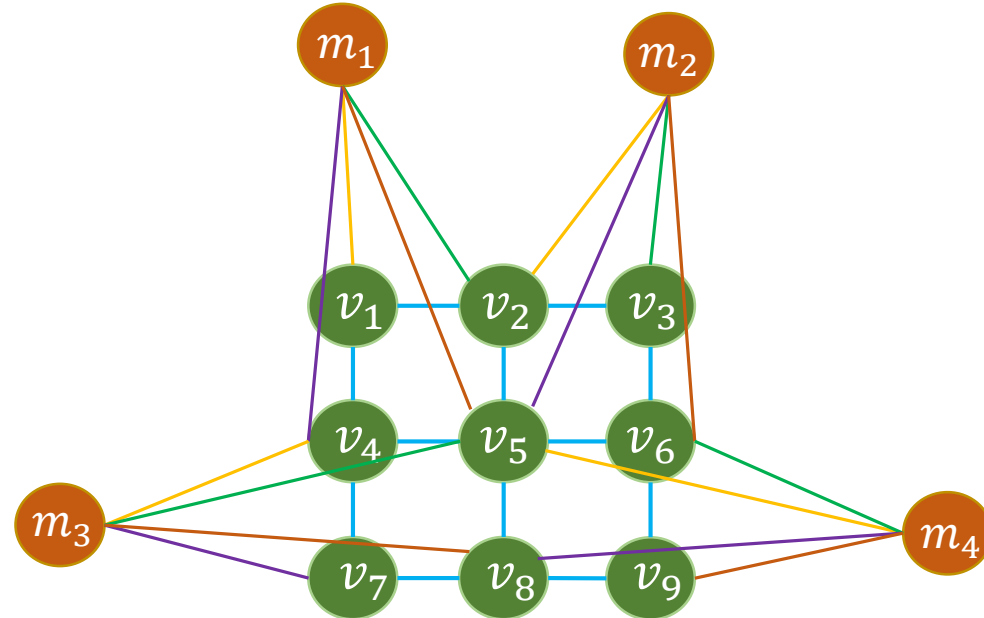
Mean

❖ Correlation (~convolution)



Kernel

$$m_1 = v_1w_1 + v_2w_2 + v_4w_3 + v_5w_4$$

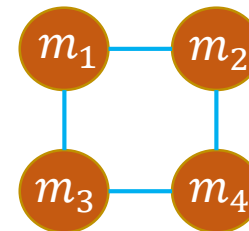


Image

*



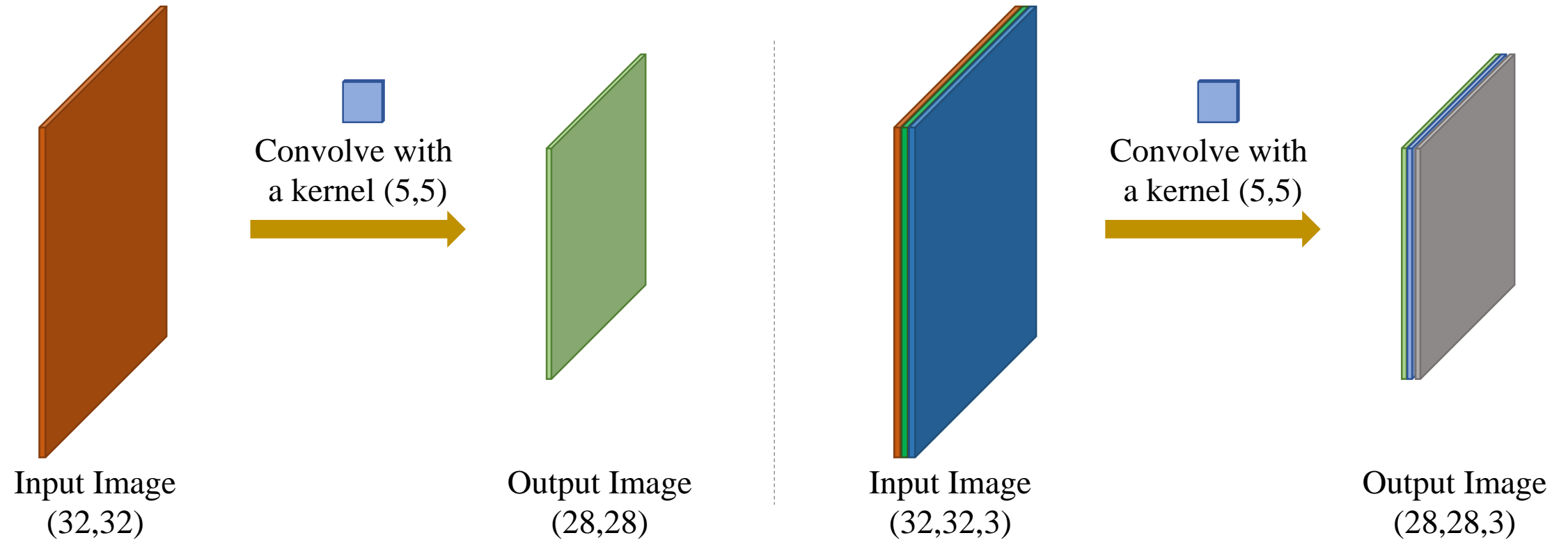
Kernel



Output

Mean

❖ Correlation (~convolution)



Mean

❖ Correlation (~convolution)

Kernels for computing mean

Numpy	<code>np.einsum()</code>
Scipy	<code>scipy.signal.convolve2d()</code>
OpenCV	<code>cv2.filter2D()</code>

1	1	1
1	1	1
1	1	1

(3x3) kernel

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

$\frac{1}{25}$

(5x5) kernel

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


Mean

❖ Correlation (~convolution)

```
1  # load image and blurring
2
3  import numpy as np
4  import cv2
5
6  # load image in grayscale mode
7  image = cv2.imread('mrbean.jpg', 0)
8
9  # create kernel
10 kernel = np.ones((5,5), np.float32) / 25.0
11
12 # compute mean for each pixel
13 dst = cv2.filter2D(image, cv2.CV_8U, kernel)
14
15 # show images
16 cv2.imshow('image', image)
17 cv2.imshow('dst', dst)
18
19 # waiting for any keys pressed and close windows
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
```

image



dst



Mean

❖ Numpy review

```
1  # numpy review
2  import numpy as np
3
4  arr = np.ones((5,5))
5  print(arr)
6
7  roi = arr[1:4, 1:4]
8  roi = roi + 1
9  print(roi)
10
11 arr[1:4, 1:4] = roi
12 print(arr)
```

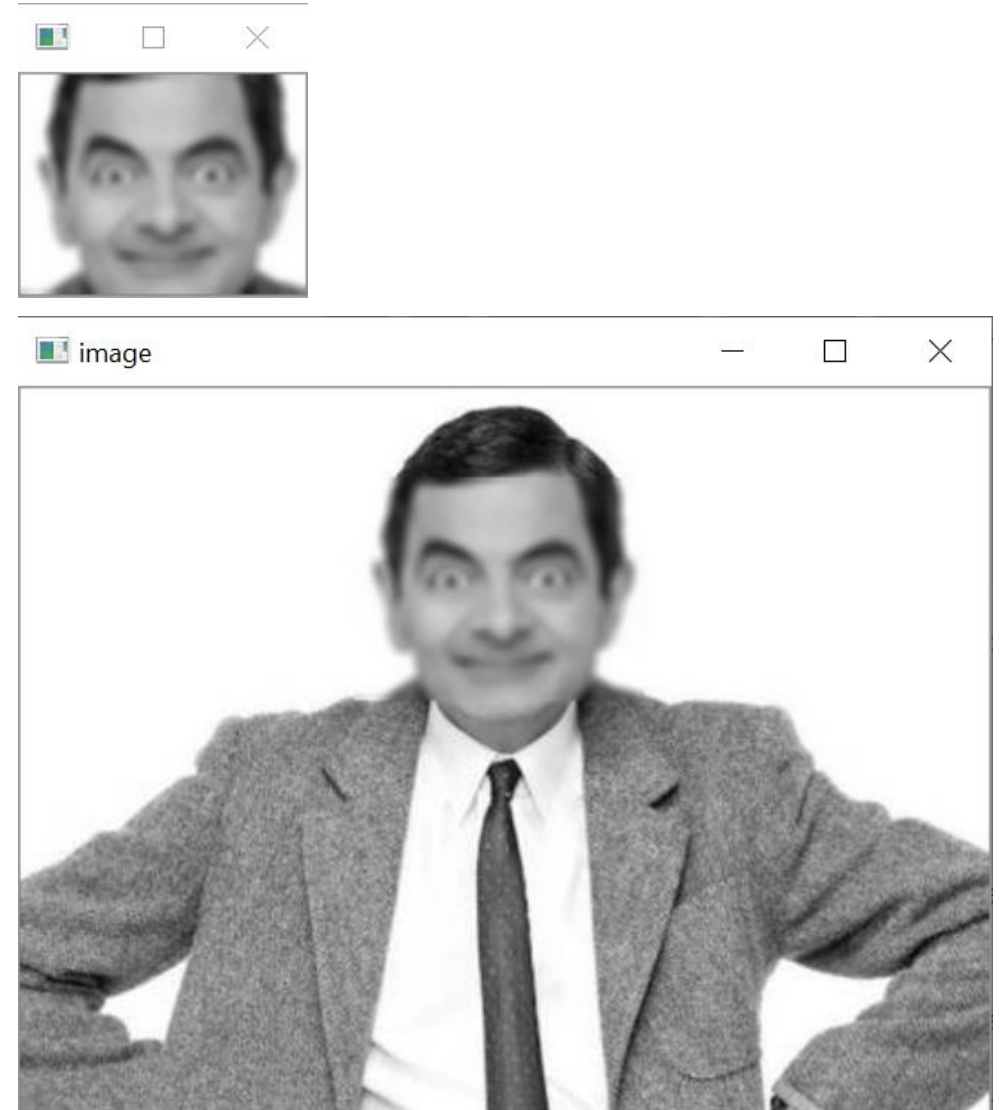
```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
```

```
[[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.]
```

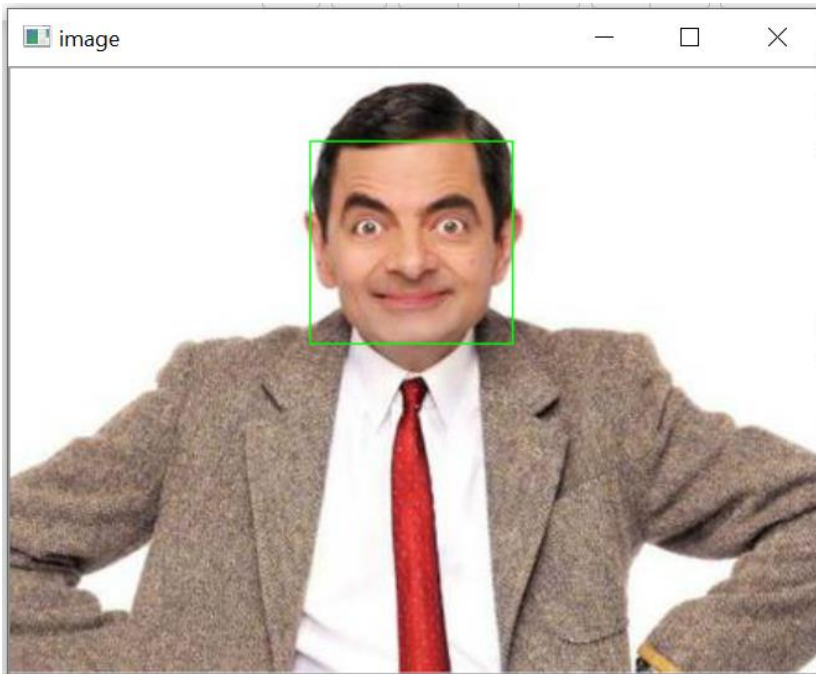
❖ Image Blurring

```
1  # load image and blurring using mask-simple
2
3  import numpy as np
4  import cv2
5
6  # load image in grayscale mode
7  image = cv2.imread('mrbean.jpg', 0)
8
9  # create kernel
10 kernel = np.ones((5,5), np.float32) / 25.0
11
12 # Select ROI (top_y,top_x,height, width)
13 roi = image[40:140,150:280]
14
15 # compute mean for each pixel
16 roi = cv2.filter2D(roi, cv2.CV_8U, kernel)
17
18 image[40:140,150:280] = roi
19
20 # show images
21 cv2.imshow('roi', roi)
22 cv2.imshow('image', image)
23
24 # waiting for any keys pressed and close windows
25 cv2.waitKey(0)
26 cv2.destroyAllWindows()
```



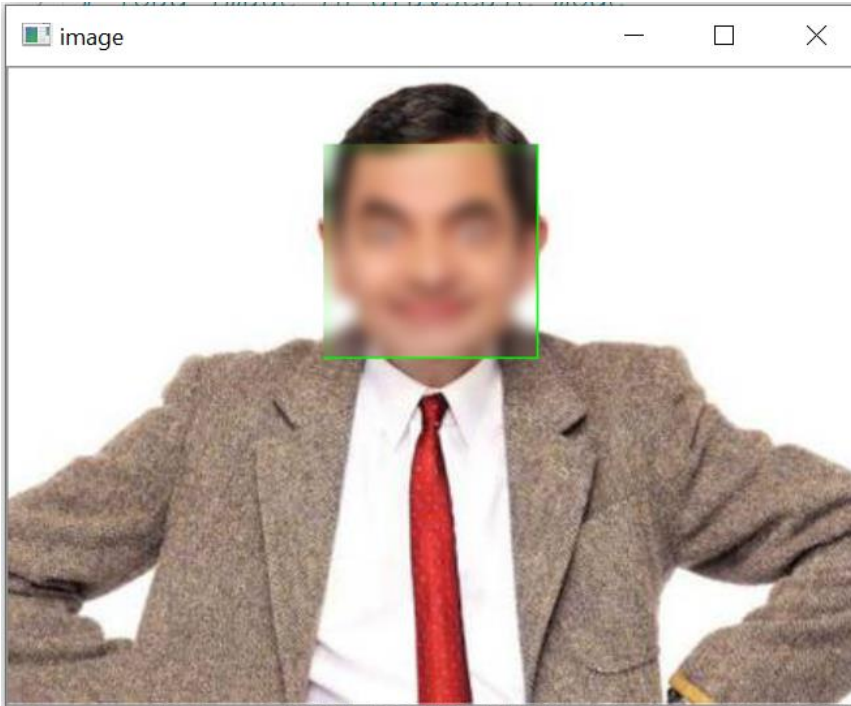
Mean

❖ Image Blurring



```
1  # load image and blurring using face detection
2
3  import numpy as np
4  import cv2
5
6  # face detection setup
7  face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
8
9  # load image in grayscale mode
10 image = cv2.imread('mrbean.jpg', 1)
11
12 # Convert to grayscale
13 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
14
15 # face detection
16 faces = face_cascade.detectMultiScale(gray, 1.1, 4)
17
18 # Draw the rectangle around each face
19 for (x, y, w, h) in faces:
20     cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 1)
21
22 # show images
23 cv2.imshow('image', image)
24
25 # waiting for any keys pressed and close windows
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
```

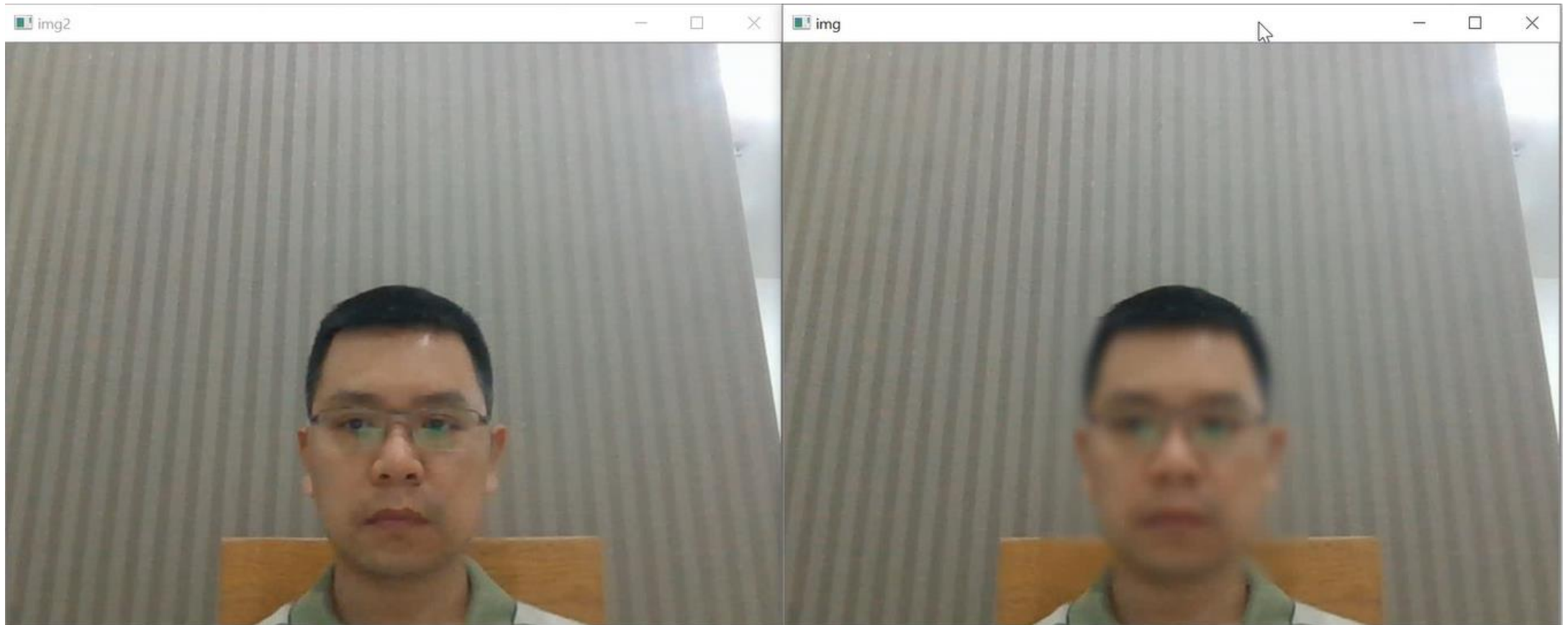
❖ Image Blurring



```
1 # load image and blurring using face detection
2
3 import numpy as np
4 import cv2
5
6 # face detection setup
7 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
8
9 # load image in grayscale mode
10 image = cv2.imread('mrbean.jpg', 1)
11
12 # Convert to grayscale
13 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
14
15 # face detection
16 faces = face_cascade.detectMultiScale(gray, 1.1, 4)
17
18 # create kernel
19 kernel = np.ones((7,7), np.float32) / 49.0
20
21 # Draw the rectangle around each face
22 for (x, y, w, h) in faces:
23     cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 1)
24     roi = image[y:y+h,x:x+w]
25
26     # compute mean for each pixel
27     roi = cv2.filter2D(roi, cv2.CV_8U, kernel)
28     roi = cv2.filter2D(roi, cv2.CV_8U, kernel)
29     roi = cv2.filter2D(roi, cv2.CV_8U, kernel)
30
31     # update
32     image[y:y+h,x:x+w] = roi
33
34 # show images
35 cv2.imshow('image', image)
36
37 # waiting for any keys pressed and close windows
38 cv2.waitKey(0)
39 cv2.destroyAllWindows()
```

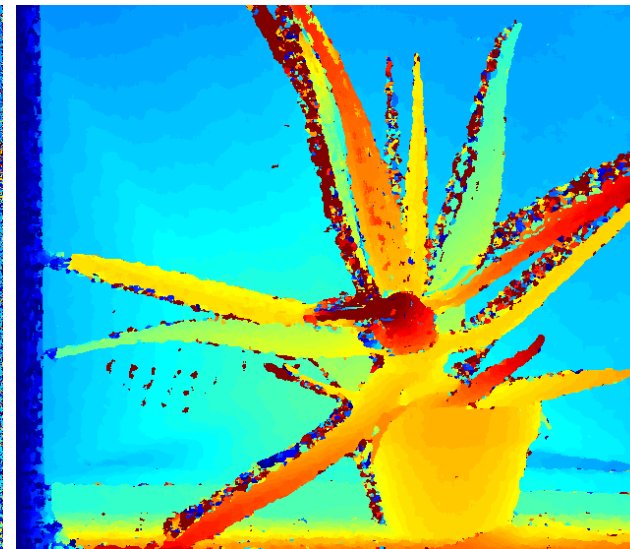
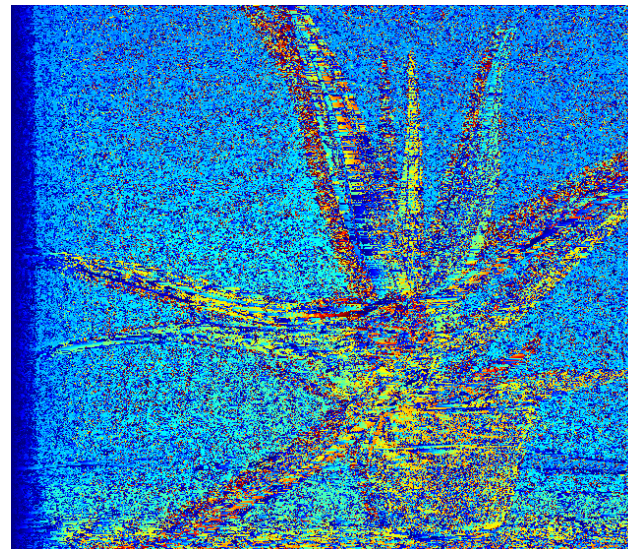

Mean

❖ Image Blurring



Mean

❖ Stereo matching



Median

Data

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

Formula

Step 1: Sort $X \rightarrow S$

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 $X \rightarrow S$

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; $N = 6$

$$\begin{aligned} m &= \frac{S_3 + S_4}{2} \\ &= \frac{4 + 5}{2} = 4.5 \end{aligned}$$

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

```
1 import numpy as np
2 import cv2
3
4 img1 = cv2.imread('mrbean_noise.jpg')
5 img2 = cv2.medianBlur(img1, 3)
6
7 # show images
8 cv2.imshow('img1', img1)
9 cv2.imshow('img2', img2)
10
11 # waiting for any keys pressed and close windows
12 cv2.waitKey(0)
13 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 $X \rightarrow S$

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**
- **Variance**
- **Correlation Coefficient**
- **Range and Histogram**

Variance

Formula:

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

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

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

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

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

$$\begin{aligned} \text{var}(X) &= \frac{1}{5} [(5 - 5)^2 + (3 - 5)^2 + (6 - 5)^2 + \\ &\quad (7 - 5)^2 + (4 - 5)^2] \\ &= \frac{1}{5} (0 + 4 + 1 + 4 + 1) = 2 \end{aligned}$$

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

Variance

❖ Code

```
1  # variance
2  def calculate_mean(numbers):          #1
3      s = sum(numbers)
4      N = len(numbers)
5      mean = s/N
6      return mean
7
8  def caculate_variance(numbers):       #2
9      mean = calculate_mean(numbers)    #3
10
11     diff = []                         #4
12     for num in numbers:
13         diff.append(num-mean)
14
15     squared_diff = []                 #5
16     for d in diff:
17         squared_diff.append(d**2)
18
19     sum_squared_diff = sum(squared_diff)
20     variance = sum_squared_diff/len(numbers)
21
22     return variance
```


Variance

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

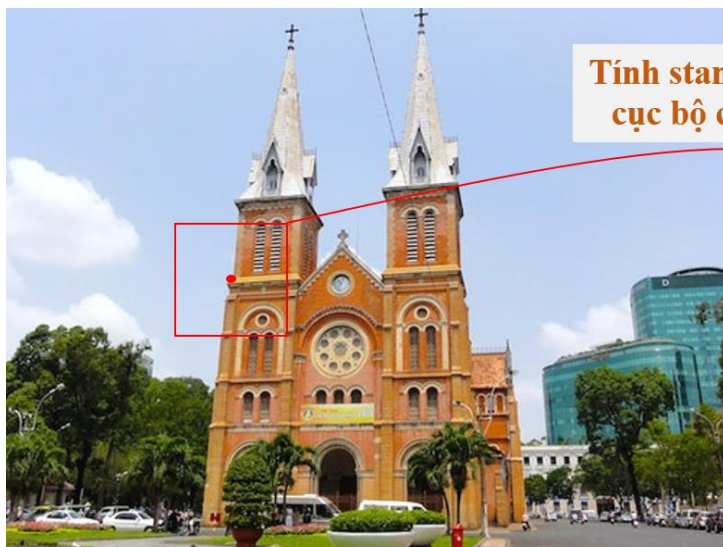


Ảnh gốc

Tính standard deviation
cục bộ cho từng pixel.



Ảnh thông tin texture



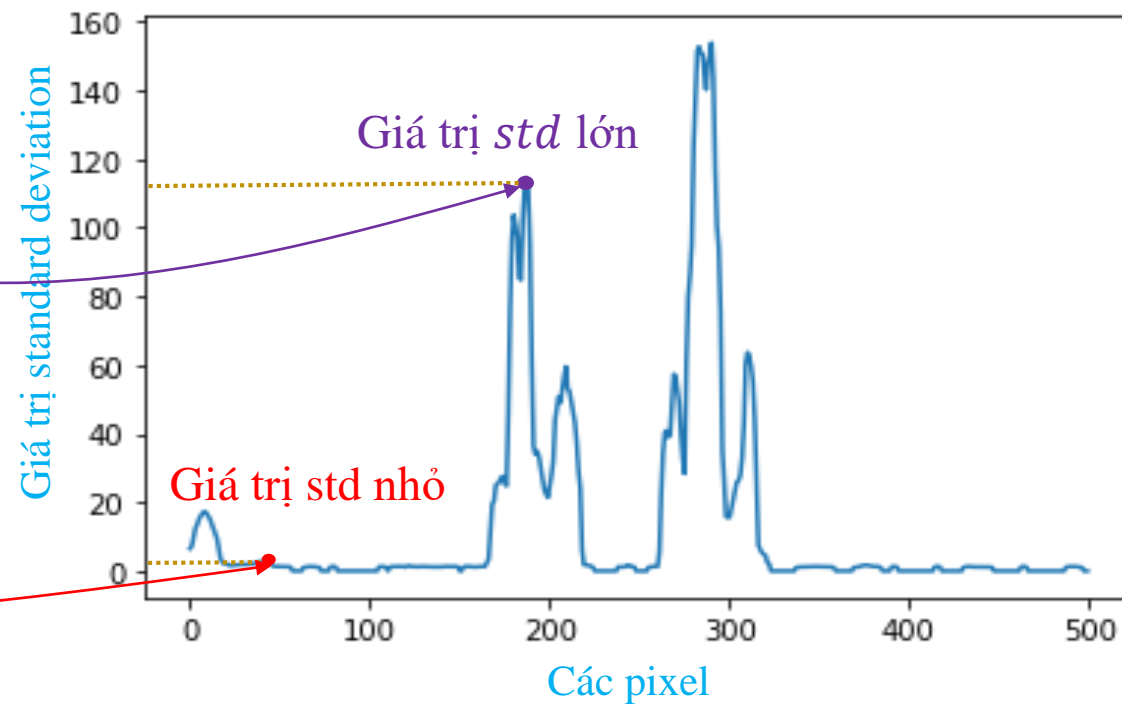
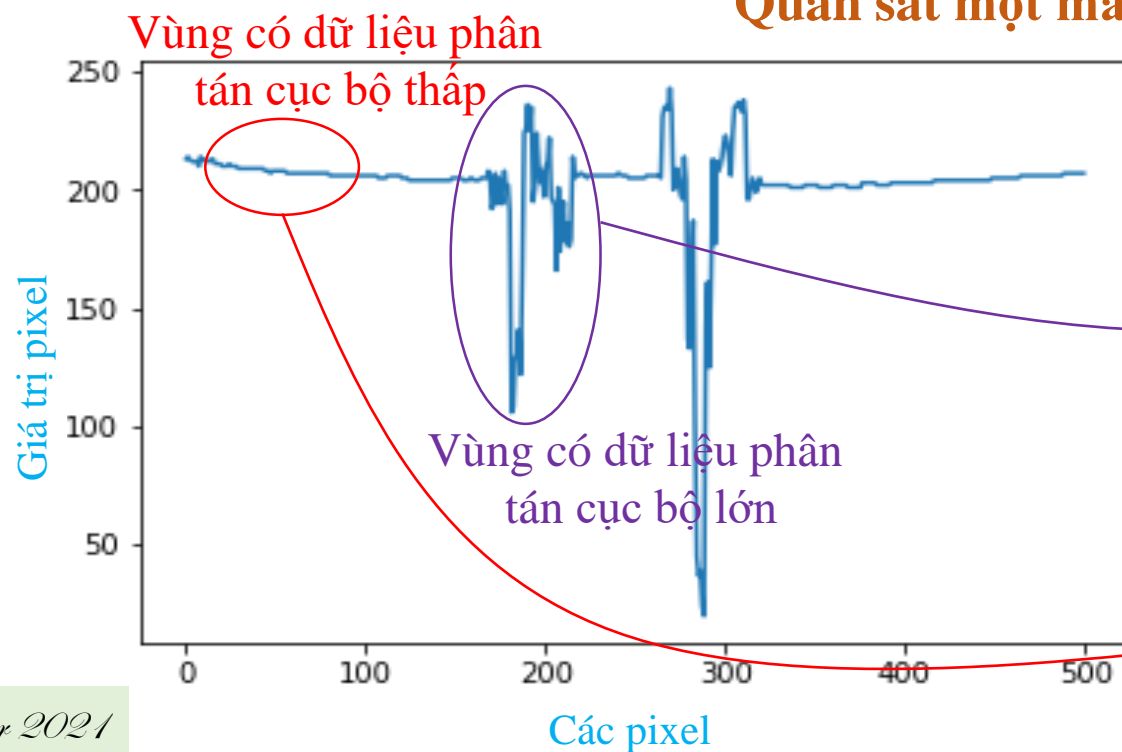
Ảnh gốc

Tính standard deviation
cục bộ cho từng pixel.



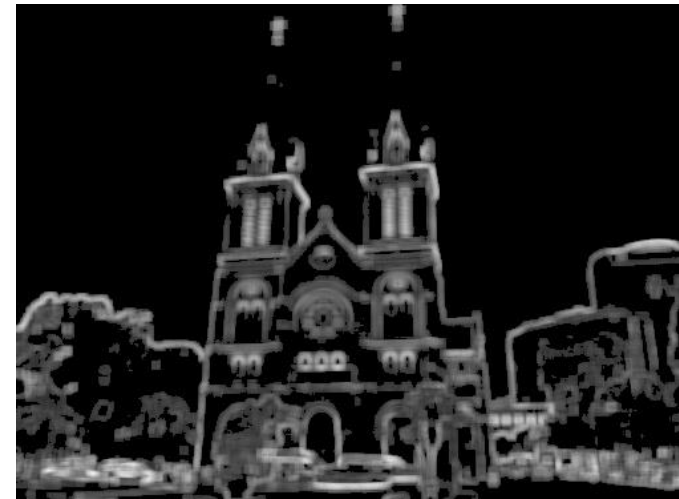
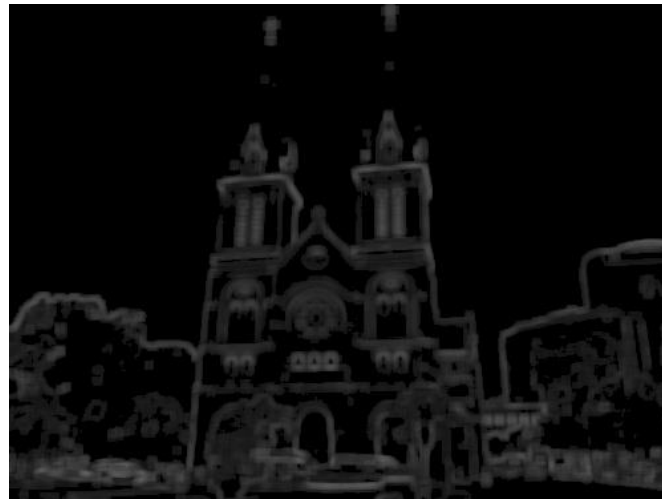
Ảnh thông tin texture

Quan sát một mảng các giá trị pixel.



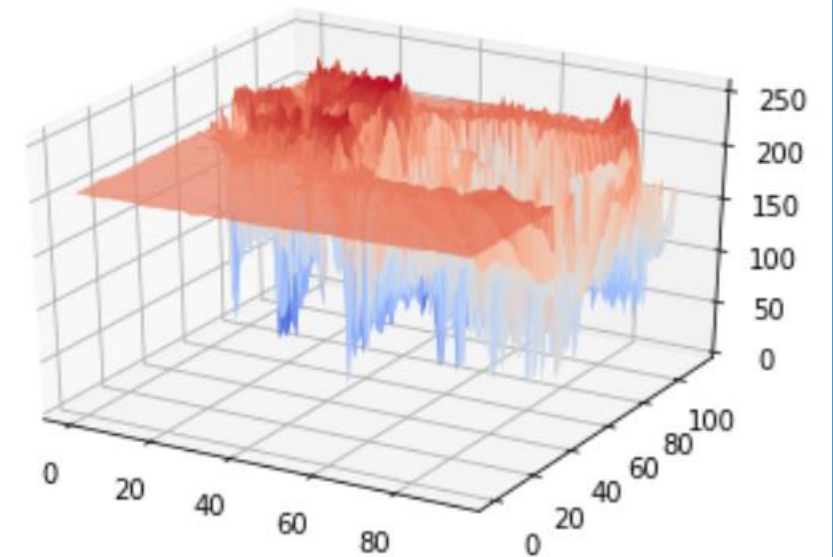
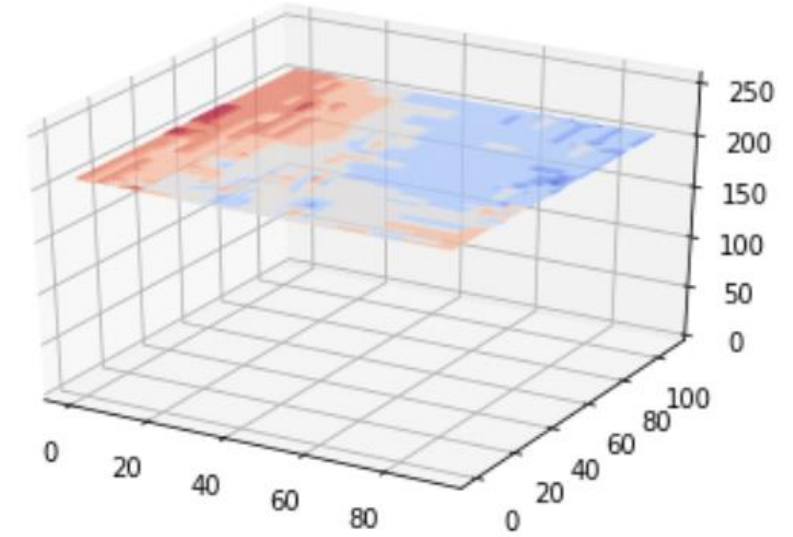
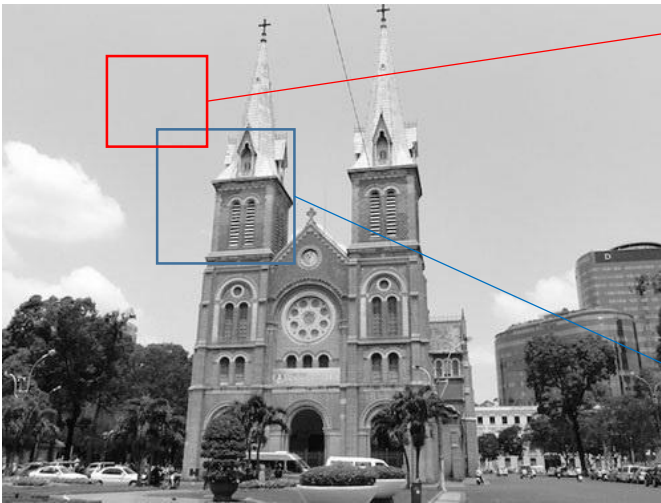
Variance

❖ Implementation



Variance

❖ Implementation



Variance

❖ Implementation

```
1 import numpy as np
2 import cv2
3 import math
4 from scipy.ndimage.filters import generic_filter
5
6 img = cv2.imread('img.jpg')
7 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
8 cv2.imwrite('edge_s1.jpg', gray)
9
10 x = gray.astype('float')
11 x_filt = generic_filter(x, np.std, size=7)
12 cv2.imwrite('edge_s2.jpg', x_filt)
13
14 x_filt[x_filt < 20] = 0
15 cv2.imwrite('edge_s3.jpg', x_filt)
16
17 maxv = np.max(x_filt)
18 print(maxv)
19
20 x_filt = x_filt*2.5
21 cv2.imwrite('edge_s4.jpg', x_filt)
```



Outline

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

Hệ số tương quan (correlation coefficient)

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

$$\begin{aligned}\rho_{xy} &= \frac{E[(x - \mu_x)(y - \mu_y)]}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} \\ &= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n \sum_i x_i^2 - (\sum_i x_i)^2} \sqrt{n \sum_i y_i^2 - (\sum_i y_i)^2}}\end{aligned}$$

Tính chất 1

$$\begin{array}{ccc} -1 & \leq & \rho_{xy} \leq 1 \\ \longleftarrow & & \longrightarrow \\ \text{Tương quan} & & \text{Tương quan} \\ \text{nghịch} & & \text{thuận} \end{array}$$

Tính chất 2

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

trong đó

$$\begin{aligned}u &= ax + b \\ v &= cy + d\end{aligned}$$

Ví dụ 1

$$\begin{aligned}x &= [7, 18, 29, 2, 10, 9, 9] \\ y &= [1, 6, 12, 8, 6, 21, 10]\end{aligned}$$

$$\begin{aligned}\rho_{xy} &= \frac{E[(x - \mu_x)(y - \mu_y)]}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} \\ &= \frac{n * 818 - 84 * 64}{\sqrt{n * 1480 - 7056} \sqrt{n * 822 - 4096}} = 0.149\end{aligned}$$

Ví dụ 2

$$\begin{aligned}u &= 2 * x - 14 = [0, 22, 44, -10, 6, 4, 4] \\ v &= y + 2 = [3, 8, 14, 10, 8, 23, 12]\end{aligned}$$

$$\begin{aligned}\rho_{uv} &= \frac{E[(u - \mu_u)(v - \mu_v)]}{\sqrt{\text{var}(u)}\sqrt{\text{var}(v)}} \\ &= \frac{n * 880 - 70 * 78}{\sqrt{n * 2588 - 4900} \sqrt{n * 1106 - 6084}} = 0.149\end{aligned}$$

Hệ số tương quan (correlation coefficient)

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

$$\begin{aligned}\rho_{xy} &= \frac{E[(x - \mu_x)(y - \mu_y)]}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} \\ &= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n \sum_i x_i^2 - (\sum_i x_i)^2} \sqrt{n \sum_i y_i^2 - (\sum_i y_i)^2}}\end{aligned}$$

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

$$\begin{aligned}\rho_{xy}(ax, by) &= \frac{E[(ax - \mu_{ax})(by - \mu_{by})]}{\sqrt{\text{var}(ax)}\sqrt{\text{var}(by)}} \\ &= \frac{n(\sum_i ax_i by_i) - (\sum_i ax_i)(\sum_i by_i)}{\sqrt{n \sum_i a^2 x_i^2 - (\sum_i ax_i)^2} \sqrt{n \sum_i b^2 y_i^2 - (\sum_i by_i)^2}} \\ &= \frac{n(ab \sum_i x_i y_i) - (a \sum_i x_i)(b \sum_i y_i)}{\sqrt{na^2 \sum_i x_i^2 - a^2 (\sum_i x_i)^2} \sqrt{nb^2 \sum_i y_i^2 - b^2 (\sum_i y_i)^2}} \\ &= \frac{ab[n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)]}{a \sqrt{n \sum_i x_i^2 - (\sum_i x_i)^2} b \sqrt{n \sum_i y_i^2 - (\sum_i y_i)^2}} \\ &= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n \sum_i x_i^2 - (\sum_i x_i)^2} \sqrt{n \sum_i y_i^2 - (\sum_i y_i)^2}} = \rho_{xy}\end{aligned}$$

Hệ số tương quan (correlation coefficient)

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

$$\begin{aligned}\rho_{xy} &= \frac{E[(x - \mu_x)(y - \mu_y)]}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} \\ &= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n \sum_i x_i^2 - (\sum_i x_i)^2} \sqrt{n \sum_i y_i^2 - (\sum_i y_i)^2}}\end{aligned}$$

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

$$\begin{aligned}\rho_{xy}(x + c, y + d) &= \frac{E[((x + c) - \mu_{(x+c)})((y + d) - \mu_{(y+d)})]}{\sqrt{\text{var}(x + c)}\sqrt{\text{var}(y + d)}} \\ &= \frac{E[((x + c) - (\mu_x + c))((y + d) - (\mu_y + d))]}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} \\ &= \frac{E[(x - \mu_x)(y - \mu_y)]}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} = \rho_{xy}\end{aligned}$$

Correlation Coefficient

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

$$\begin{aligned}\rho_{xy} &= \frac{E[(x - \mu_x)(y - \mu_y)]}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} \\ &= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n \sum_i x_i^2 - (\sum_i x_i)^2} \sqrt{n \sum_i y_i^2 - (\sum_i y_i)^2}}\end{aligned}$$

```
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

$$\begin{aligned}\rho_{xy} &= \frac{E[(x - \mu_x)(y - \mu_y)]}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}} \\ &= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n \sum_i x_i^2 - (\sum_i x_i)^2} \sqrt{n \sum_i y_i^2 - (\sum_i y_i)^2}}\end{aligned}$$

```
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

$$\rho_{P_1 P_2} = 0.55$$

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

$$\rho_{P_1 P_4} = 0.30$$



P_1



$P_2 = P_1 + 50$



$P_3 = 1.2P_1 + 10$

$$\rho_{P_1 P_2} = 0.9970$$

$$\rho_{P_1 P_3} = 0.9979$$

$\rightarrow \rho$ hoạt động tốt dưới sự thay đổi tuyến tính

```
1. # aivietnam.ai
2.
3. import numpy as np
4. from PIL import Image
5.
6. # load ảnh và chuyển về kiểu list
7. image1 = Image.open('images/img1.png')
8. image2 = Image.open('images/img2.png')
9. image3 = Image.open('images/img3.png')
10. image4 = Image.open('images/img4.png')
11.
12. image1_list = np.asarray(image1).flatten().tolist()
13. image2_list = np.asarray(image2).flatten().tolist()
14. image3_list = np.asarray(image3).flatten().tolist()
15. image4_list = np.asarray(image4).flatten().tolist()
16.
17.
18. # tính correlation coefficient
19. corr_1_2 = find_corr_x_y(image1_list, image2_list)
20. corr_1_3 = find_corr_x_y(image1_list, image3_list)
21. corr_1_4 = find_corr_x_y(image1_list, image4_list)
22.
23. print('corr_1_2:', corr_1_2)
24. print('corr_1_3:', corr_1_3)
25. print('corr_1_4:', corr_1_4)
```

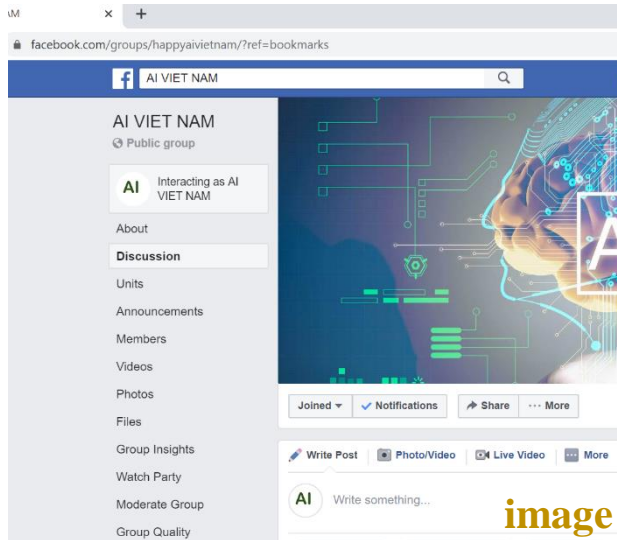
Correlation Coefficient

❖ Application to template matching

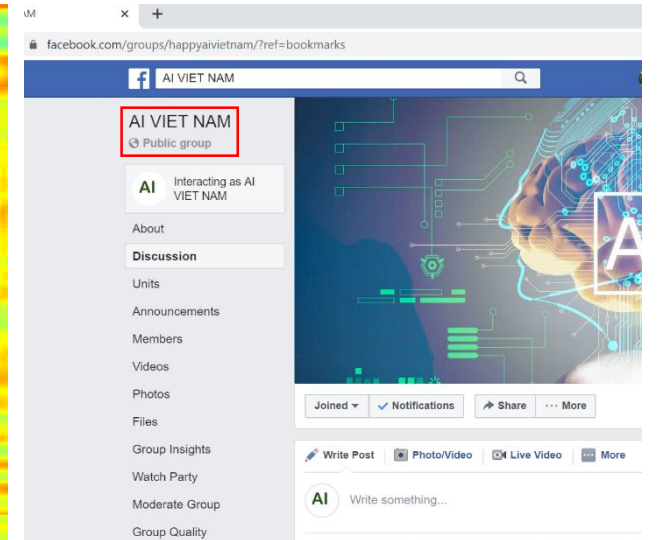
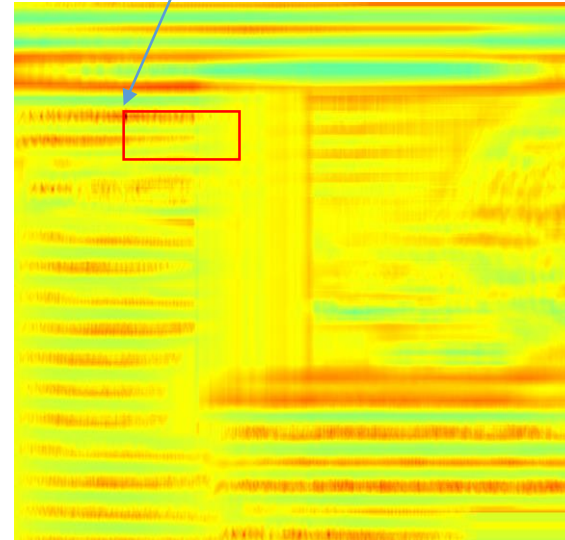
AI VIET NAM
Public group

template

Tìm template có
trong hình image



max

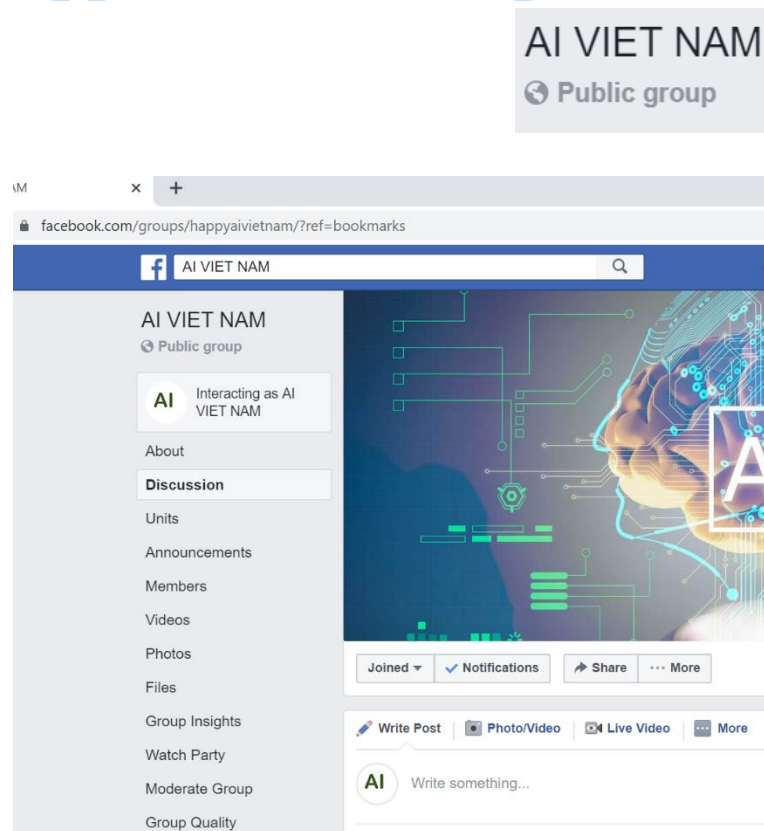


Kết quả

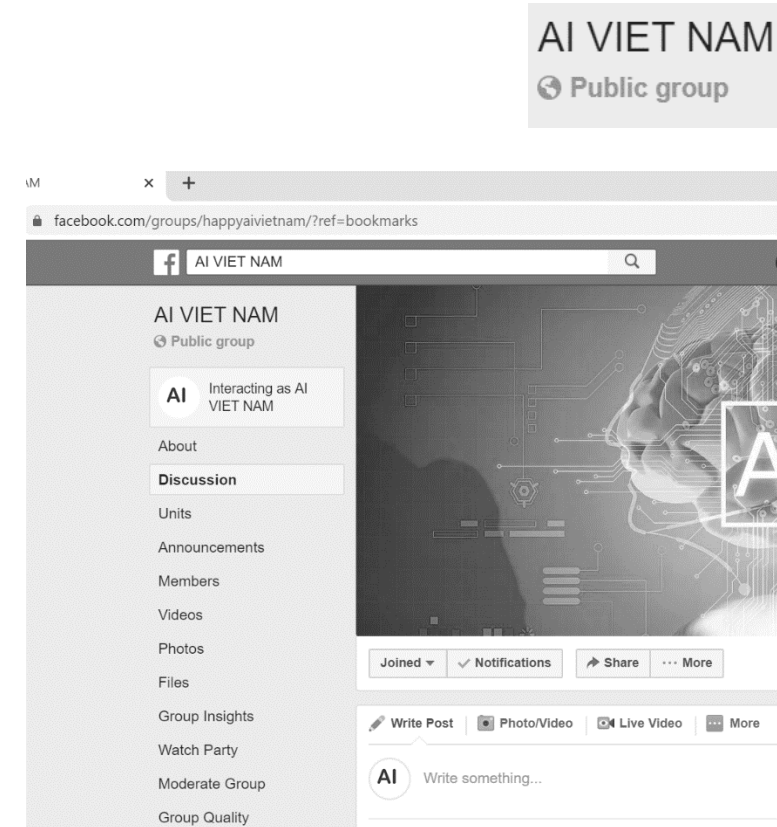
Correlation Coefficient

❖ 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

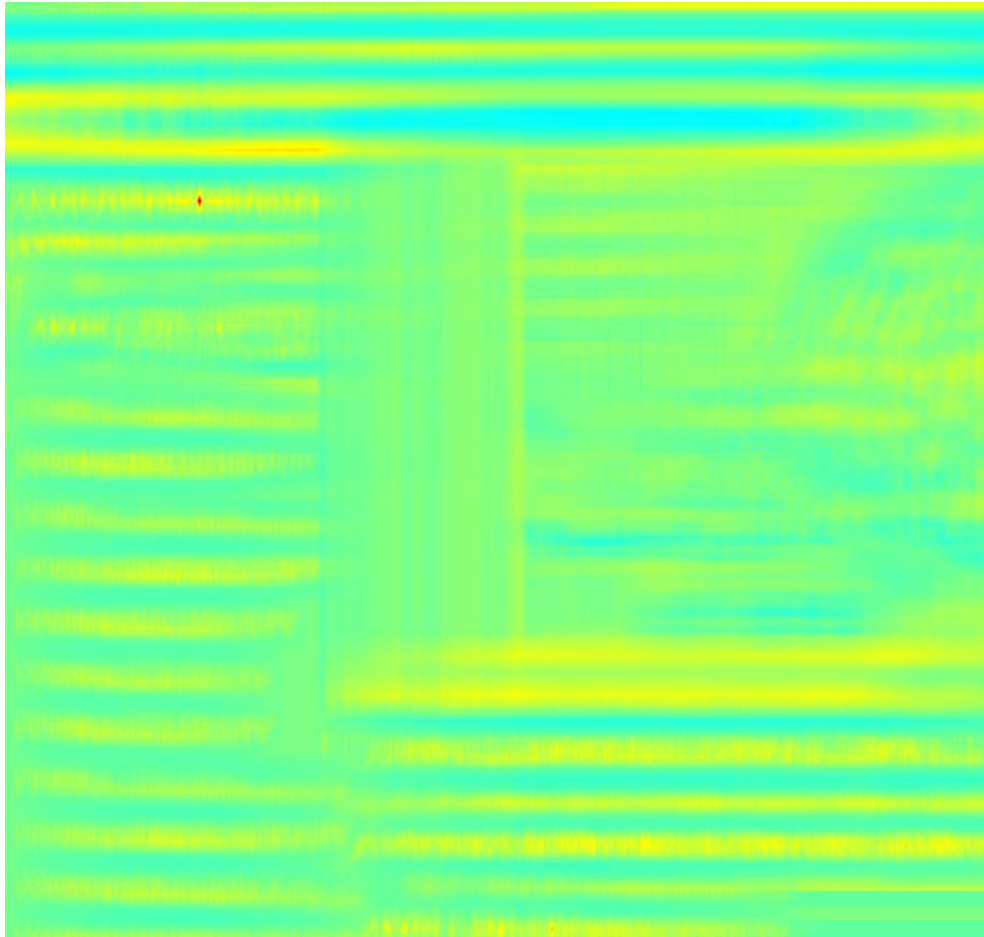
```
1  # template matching
2
3  import cv2
4  import numpy as np
5  from matplotlib import pyplot as plt
6
7  # Load image and convert to grayscale
8  image = cv.imread('image.png',1)
9  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
10
11 print(type(gray[0][0]))
12
13 template = cv.imread('template.png',0)
14 w, h = template.shape[::-1]
15
16 # Apply template Matching
17 corr_map = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF_NORMED)
18
19 # save correlation map
20 corr_map = (corr_map+1.0)*127.5
21 corr_map = corr_map.astype('uint8')
22 cv.imwrite('corr_map_grayscale.png', corr_map)
```



Correlation Map

Correlation Coefficient

❖ Application to template matching



```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4
5 # Load image and convert to grayscale
6 image = cv.imread('image.png',1)
7 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8
9 template = cv.imread('template.png',0)
10 w, h = template.shape[::-1]
11
12 # Apply template Matching
13 corr_map = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF_NORMED)
14 min_val, max_val, min_loc, max_loc = cv.minMaxLoc(corr_map)
15
16 # take minimum
17 top_left = max_loc
18 bottom_right = (top_left[0] + w, top_left[1] + h)
19
20 # draw
21 cv.rectangle(image, top_left, bottom_right, (0, 255, 0), 2)
22
23 # save corr map in grayscale
24 corr_map = (corr_map+1.0)*127.5
25 corr_map = corr_map.astype('uint8')
26 cv.imwrite('corr_map_grayscale.png', corr_map)
27
28 # applyColorMap
29 corr_map = cv2.applyColorMap(corr_map, cv2.COLORMAP_JET)
30
31 # save results
32 cv.imwrite('corr_map_color.png', corr_map)
33 cv.imwrite('result.png', image)
```


Correlation Coefficient



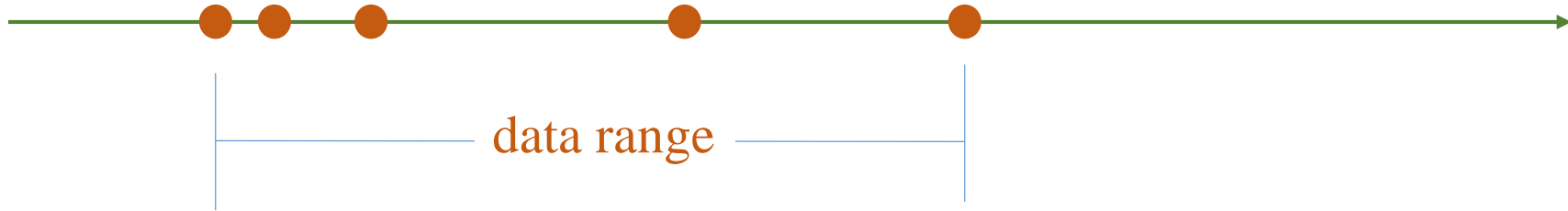
```
17
18     # Display
19     cv2.imshow('img', img)
20
21     # Stop if escape key is pressed
22     if cv2.waitKey(33) == ord('a'):
23         break
24
25 template = gray[300:400, 200:400]
26 w, h = template.shape[:2]
27 cv2.imshow('template', template)
28 cv2.waitKey(0)
29
30 while True:
31     # Read the frame and Convert to grayscale
32     _, img = cap.read()
33     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
34
35     corr_map = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF_NORMED)
36     min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(corr_map)
37
38     # take minimum
39     top_left = max_loc
40     bottom_right = (top_left[0] + w, top_left[1] + h)
41
42     # draw
43     cv2.rectangle(img, top_left, bottom_right, (0, 255, 0), 2)
44
45
46     # Display
47     cv2.imshow('img', img)
48
49     # Stop if escape key is pressed
50     if cv2.waitKey(33) == ord('a'):
51         break
52
53 # Release the VideoCapture object
54 cap.release()
55 cv2.destroyAllWindows()
```

Outline

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

Range

❖ Procedure

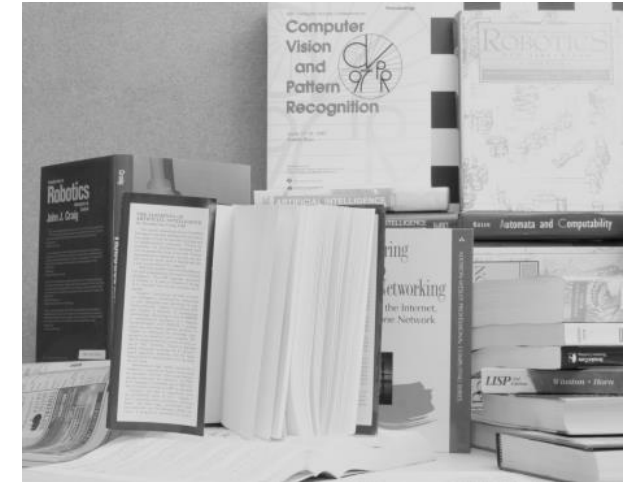
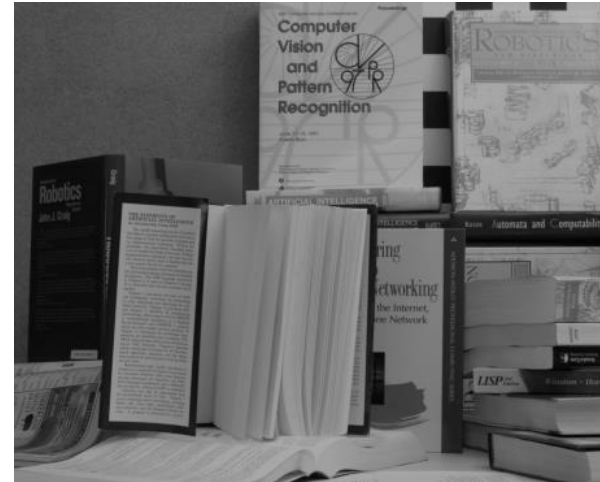
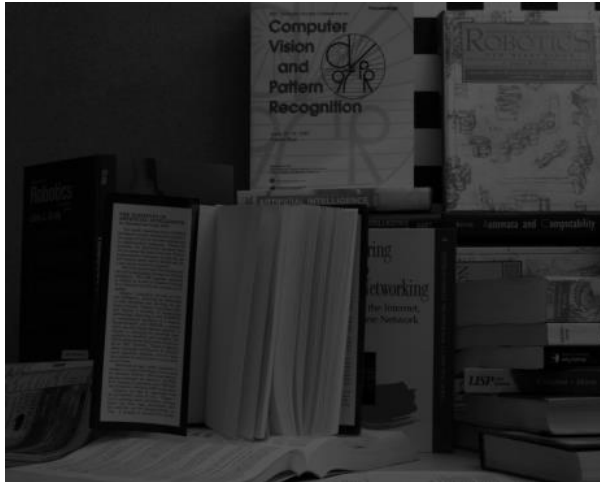


```
1. def find_range(numbers):           #1
2.     lowest = min(numbers)          #2
3.     highest = max(numbers)         #3
4.     r = highest-lowest              #4
5.     print('Lowest: {0}\tHighest: {1}\tRange: {2}'.format(lowest, highest, r))
6.
7. # data
8. points = [7, 8, 9, 2, 10, 9, 9, 9,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

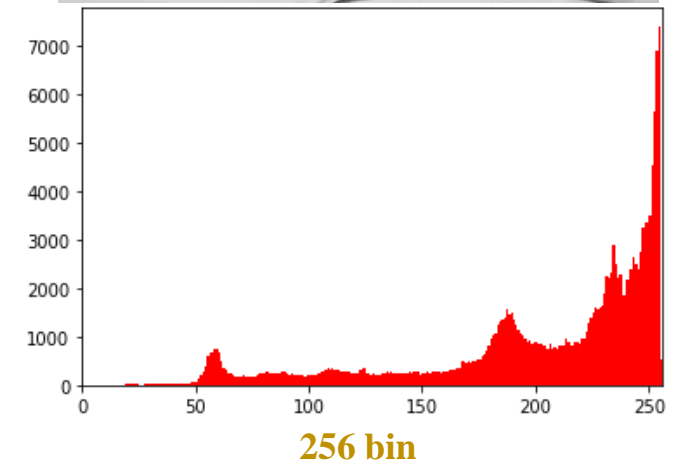
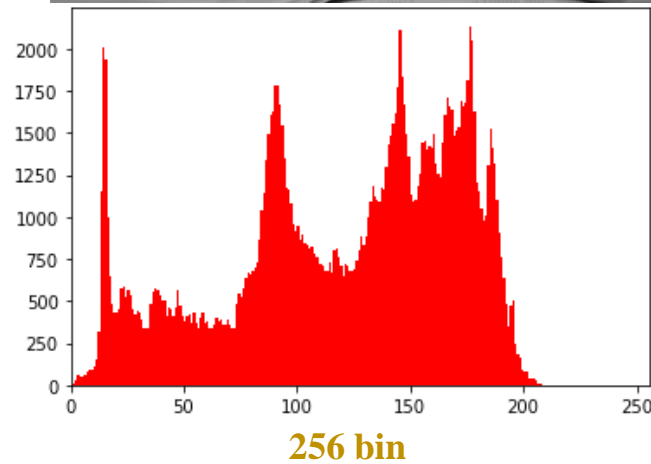
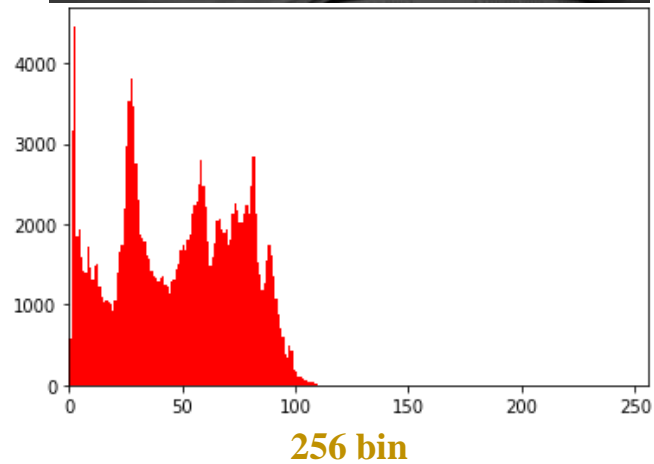
Histogram cho ảnh grayscale

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.

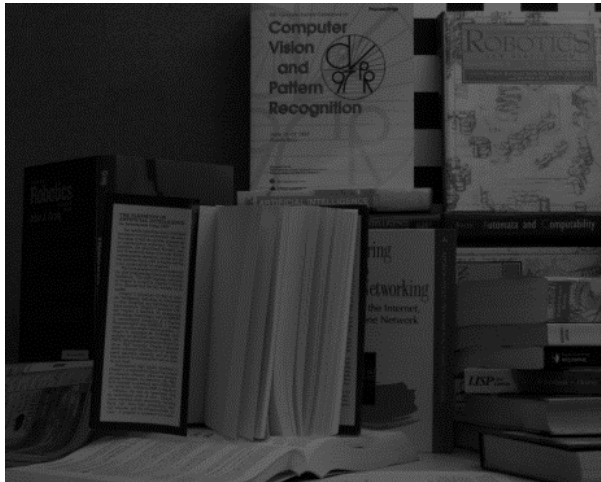
Ảnh



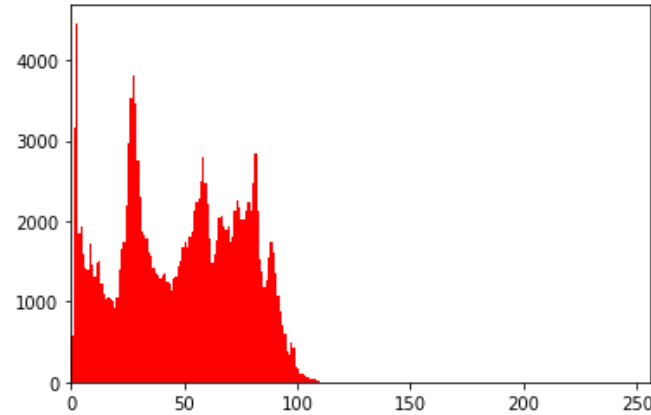
Histogram



Histogram cho ảnh grayscale



Ảnh



Histogram

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

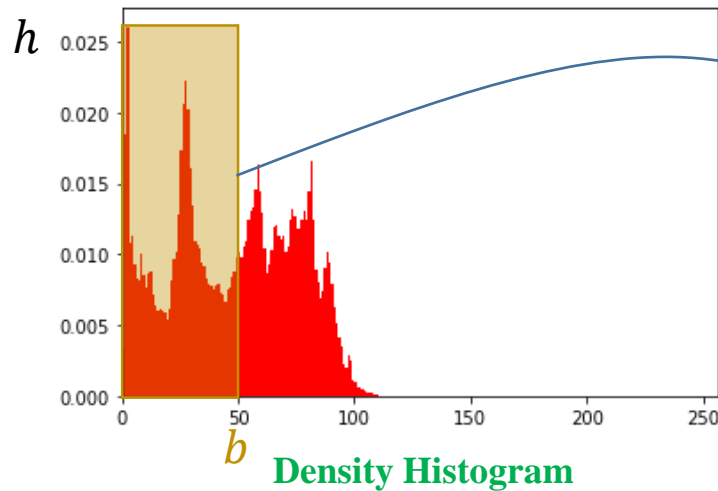
Giá trị density histogram ở bin thứ b được tính như sau:

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

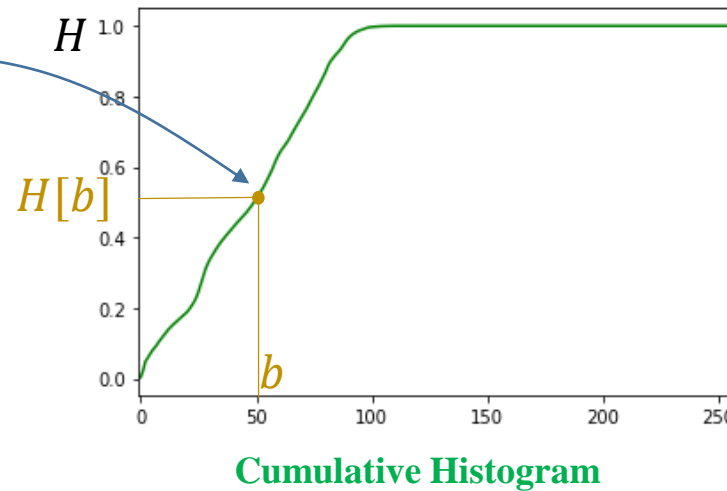
Giá trị cumulative histogram ở bin thứ b được tính như sau:

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

Cộng dồn giá trị $h[k]$ từ 0 đến b



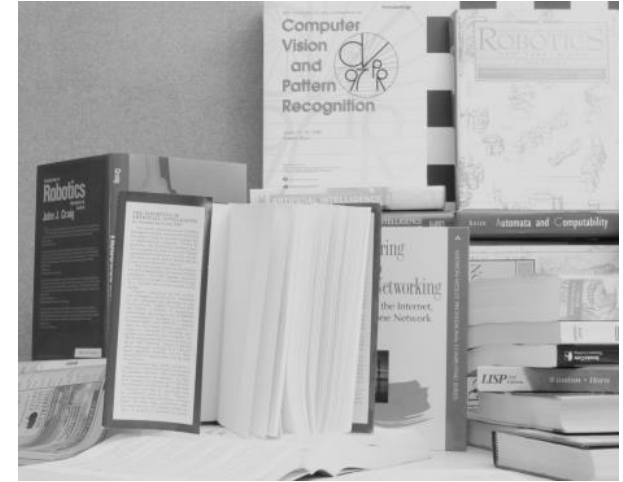
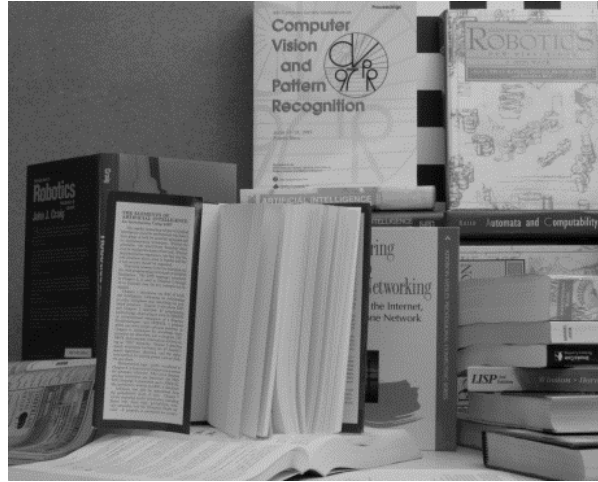
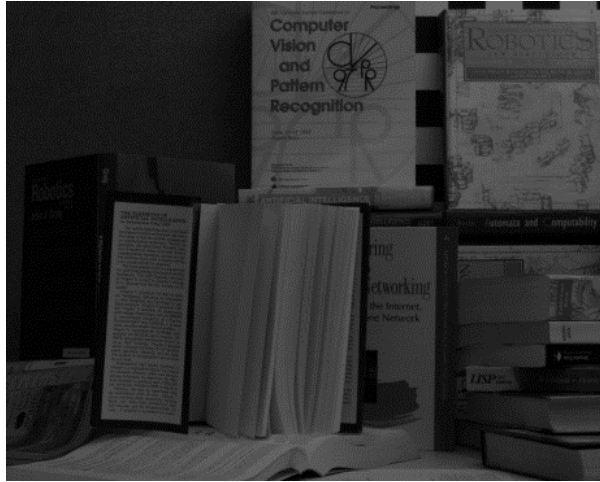
Density Histogram



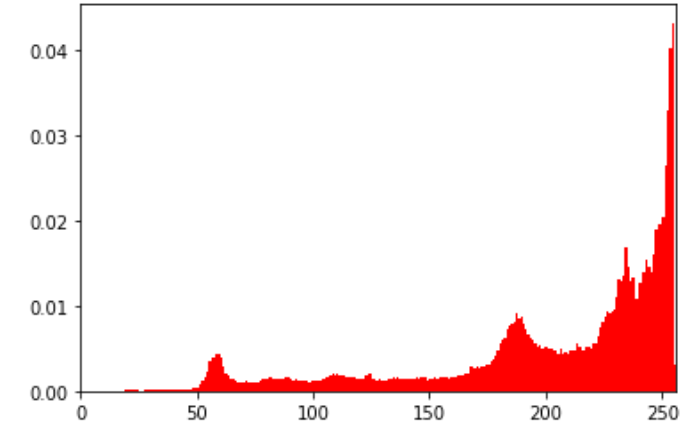
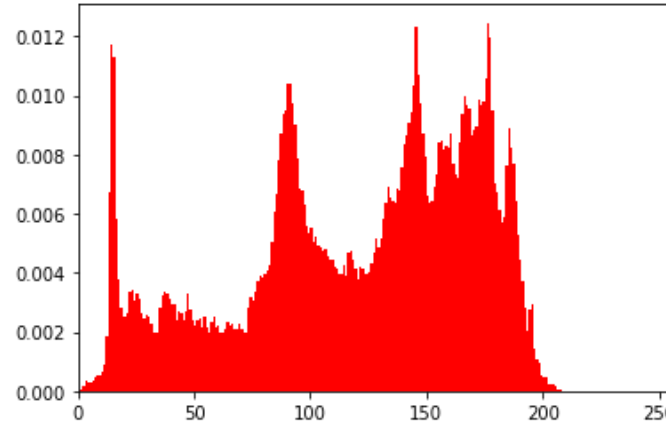
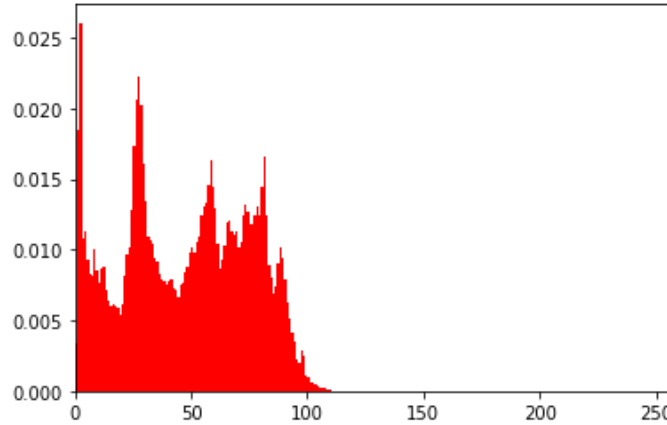
Cumulative Histogram

Histogram cho ảnh grayscale

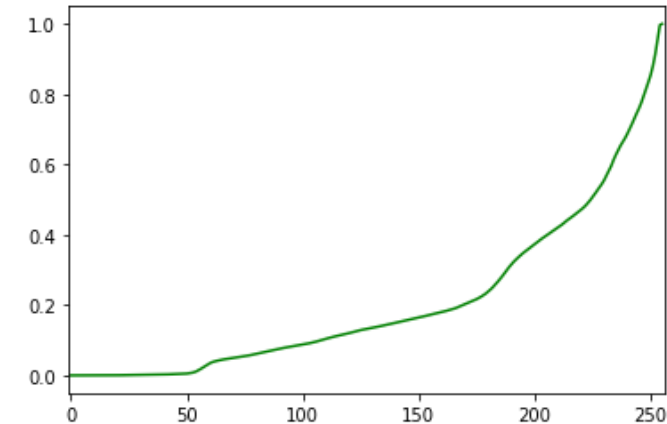
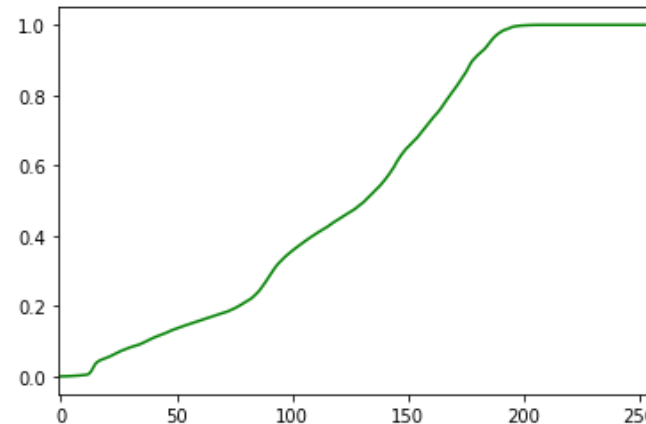
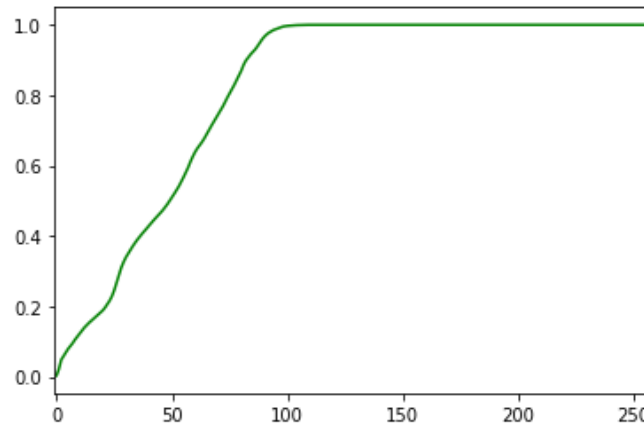
Ảnh



Density
Histogram



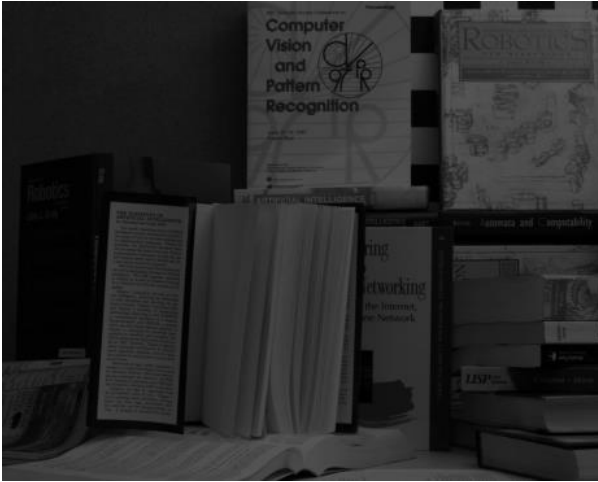
Cumulative
Histogram



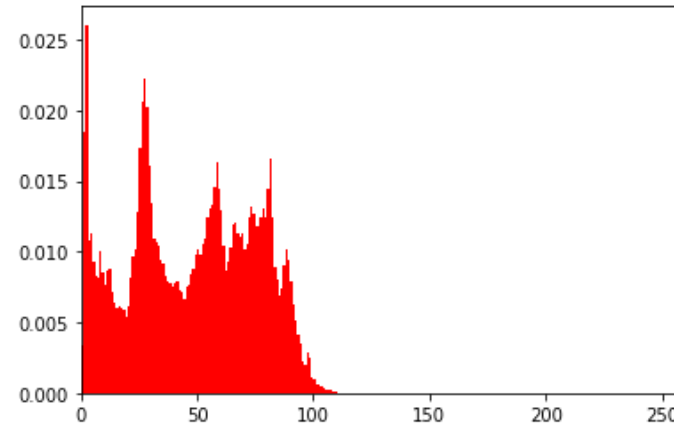
Histogram equalization

Được dùng để tăng độ tương phản của ảnh

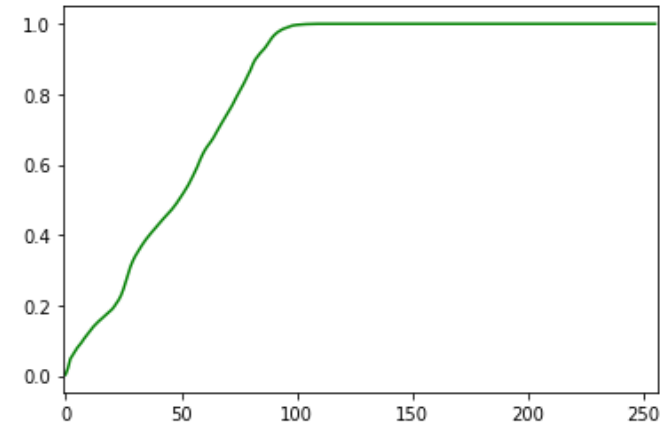
Idea: Kéo phân bố của density histogram sao cho xấp xỉ với uniform distribution.



Ảnh



Density Histogram

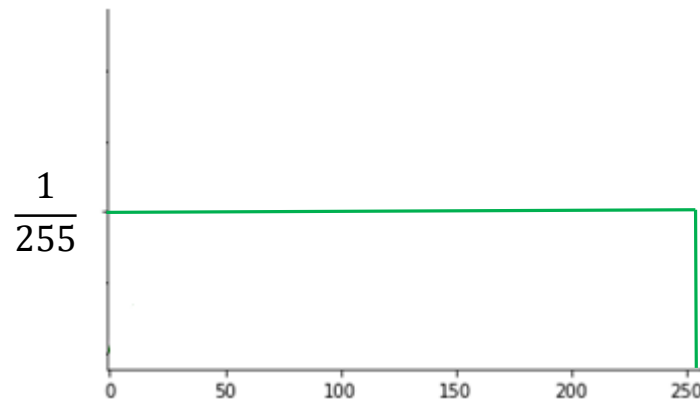


Cumulative Histogram

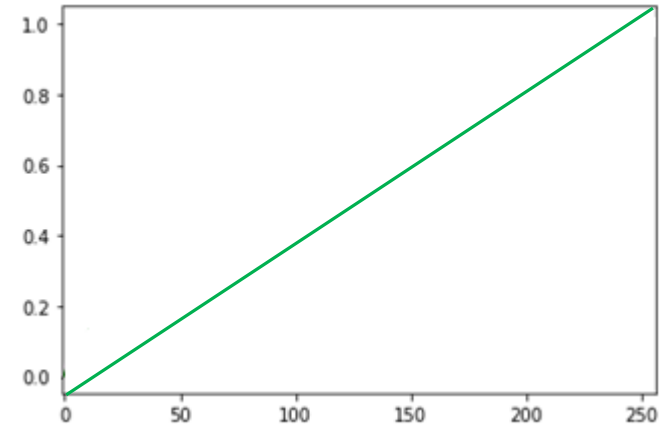
Công thức

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

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



Uniform density function
trong khoảng $[0, 255]$

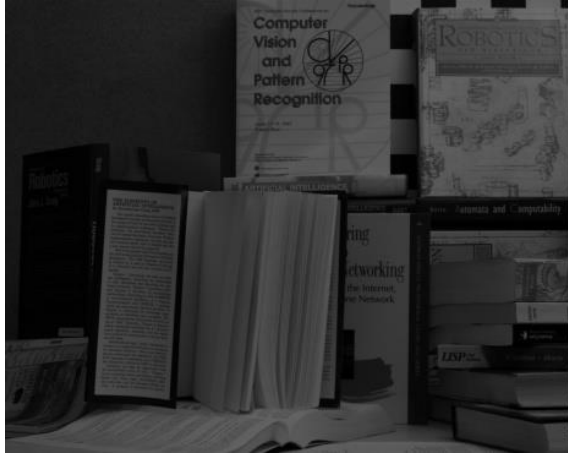


Cumulative density function
cho uniform distribution

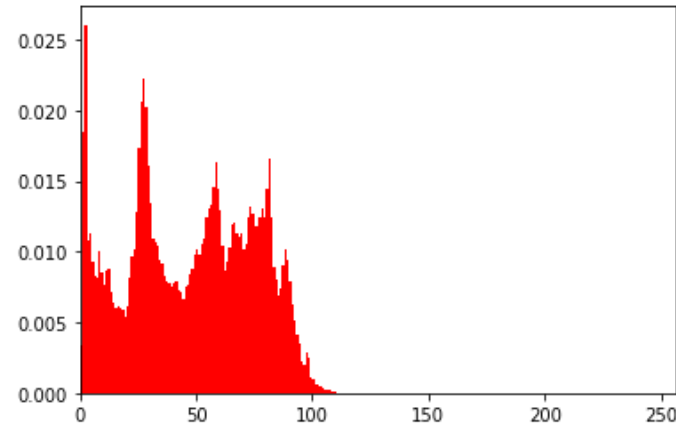
Exercise: Why?

Histogram equalization để tăng độ tương phản

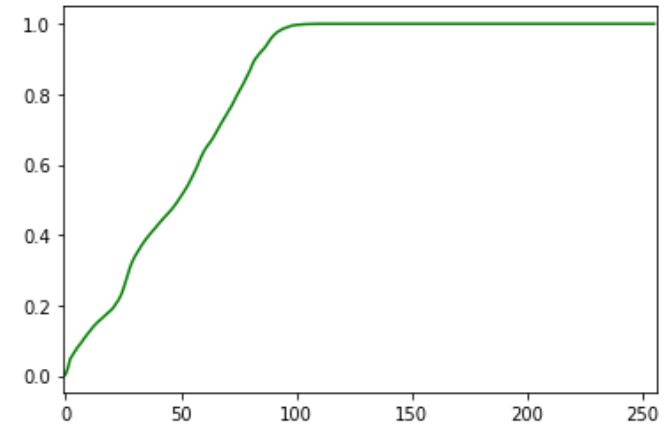
Cho ảnh grayscale



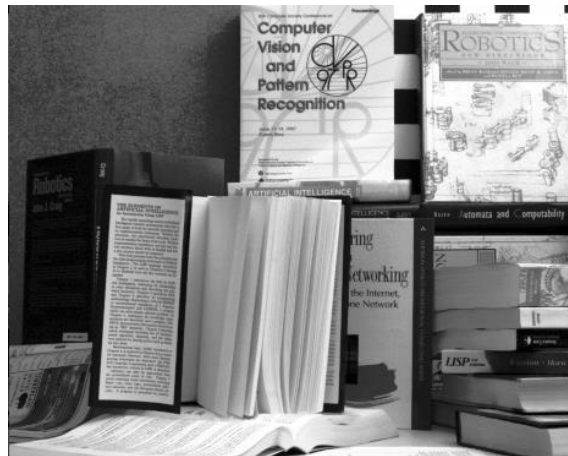
Ảnh gốc



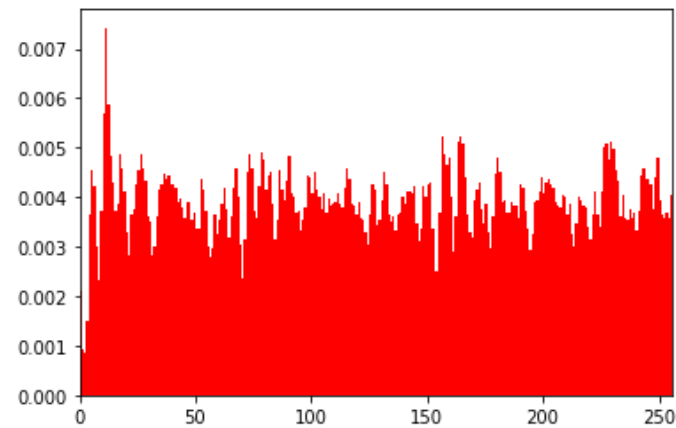
Density Histogram



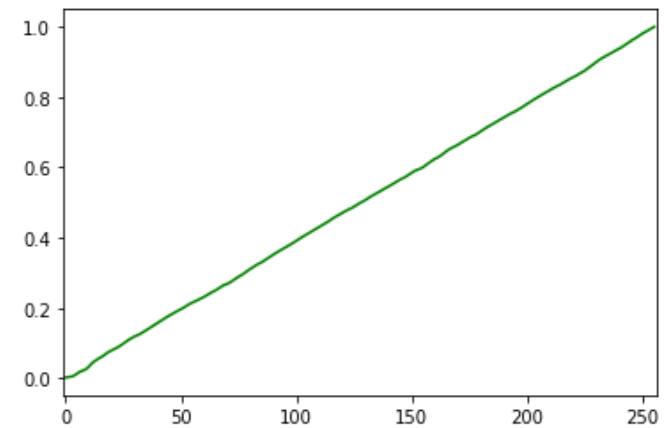
Cumulative Histogram



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

Summary

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

