AI VIETNAM
All-In-One Course

Basic Statistics

Quang-Vinh Dinh
PhD in Computer Science

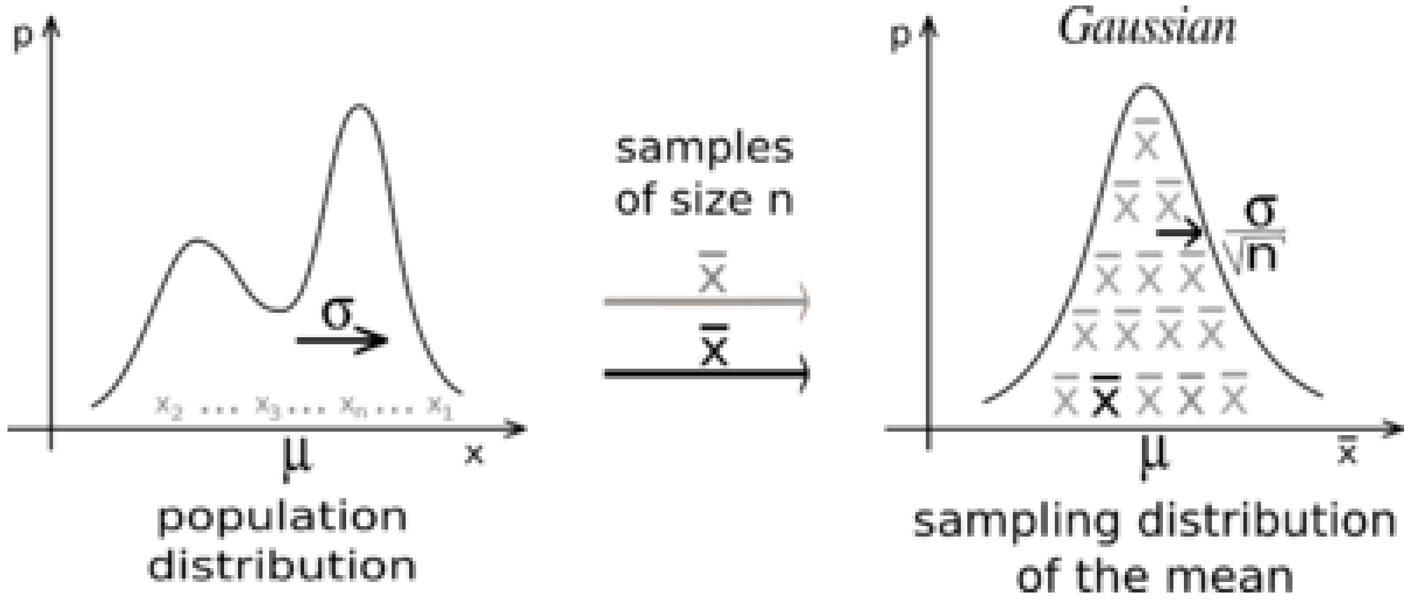
Outline

- Central limit theorem
- Mean and its Applications
- Median and its Applications
- Variance and its Applications
- Quick Introduction to Correlation

Central limit theorem

❖ Example

When independent random variables are summed up, their properly normalized sum tends toward a normal distribution even if the original variables themselves are not normally distributed.



https://en.wikipedia.org/wiki/Central_limit_theorem

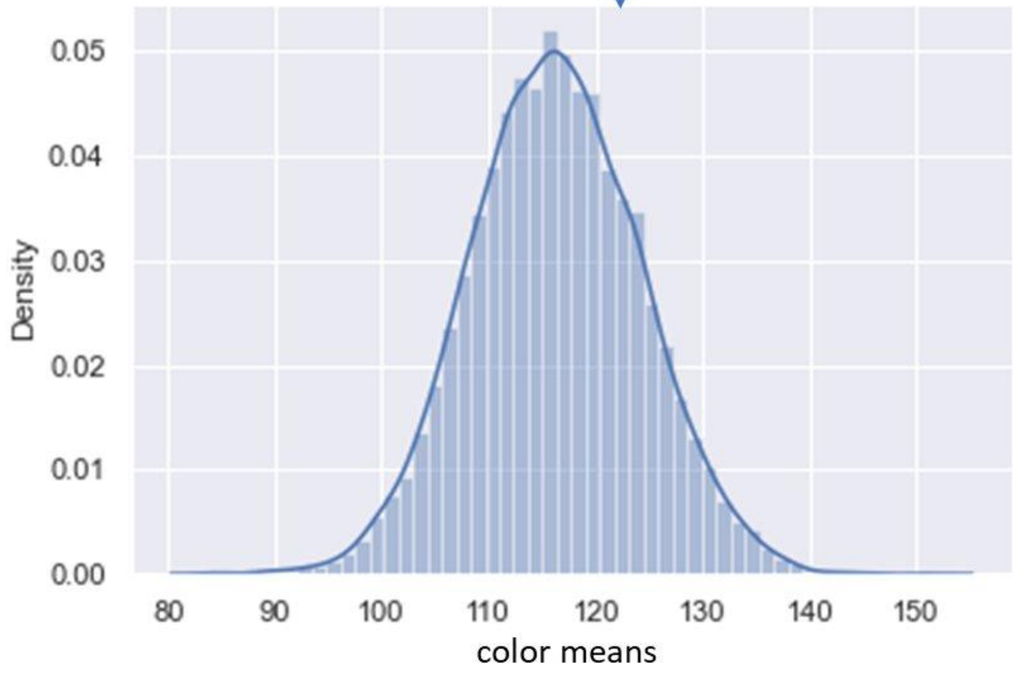
Central Limit Theorem

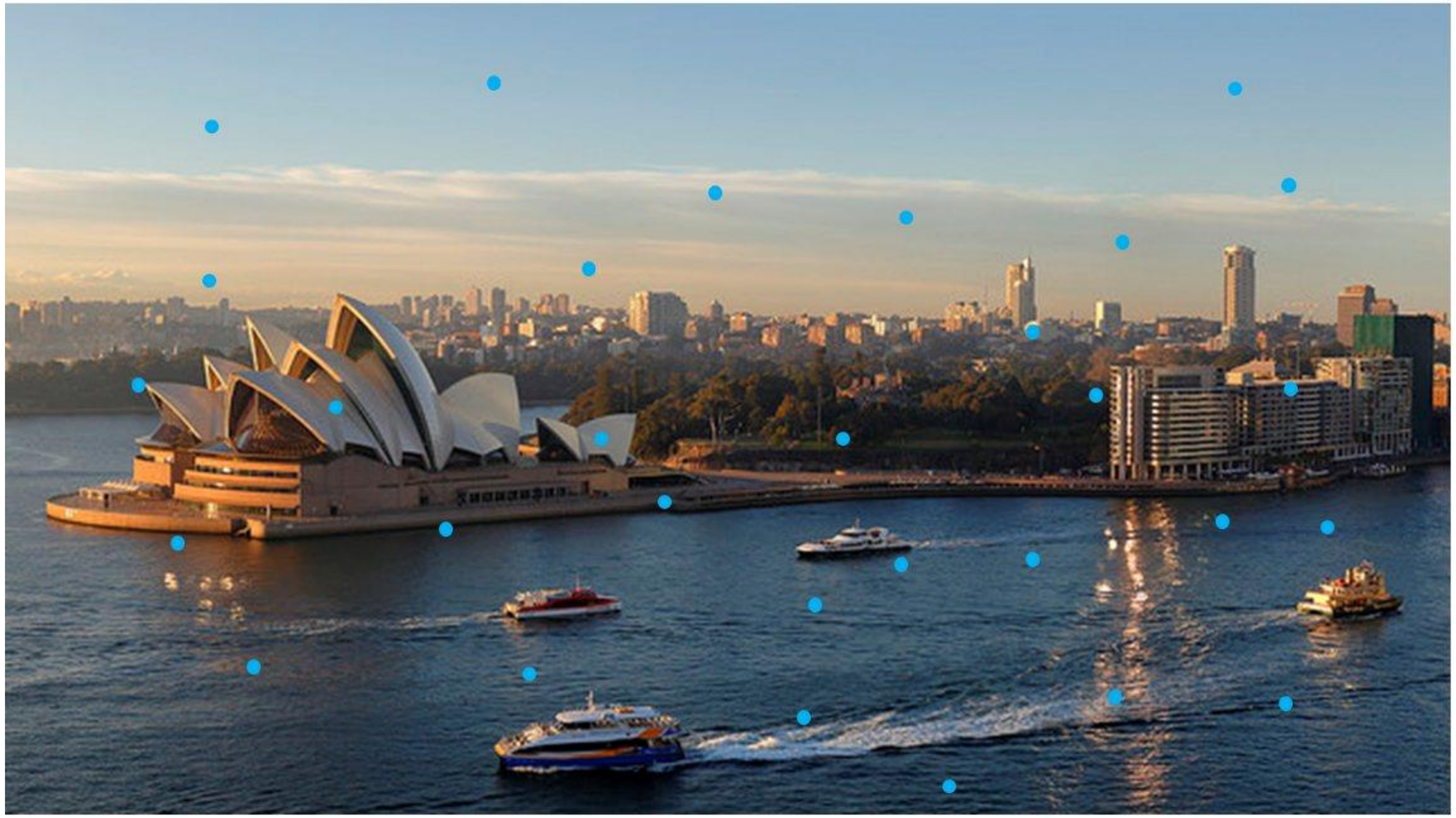
Randomly selected 30 pixels

Compute color mean for the 30 pixels

Repeat 10000 times

Histogram of the 10000 color means





Population: pixel colors of the entire image

Example

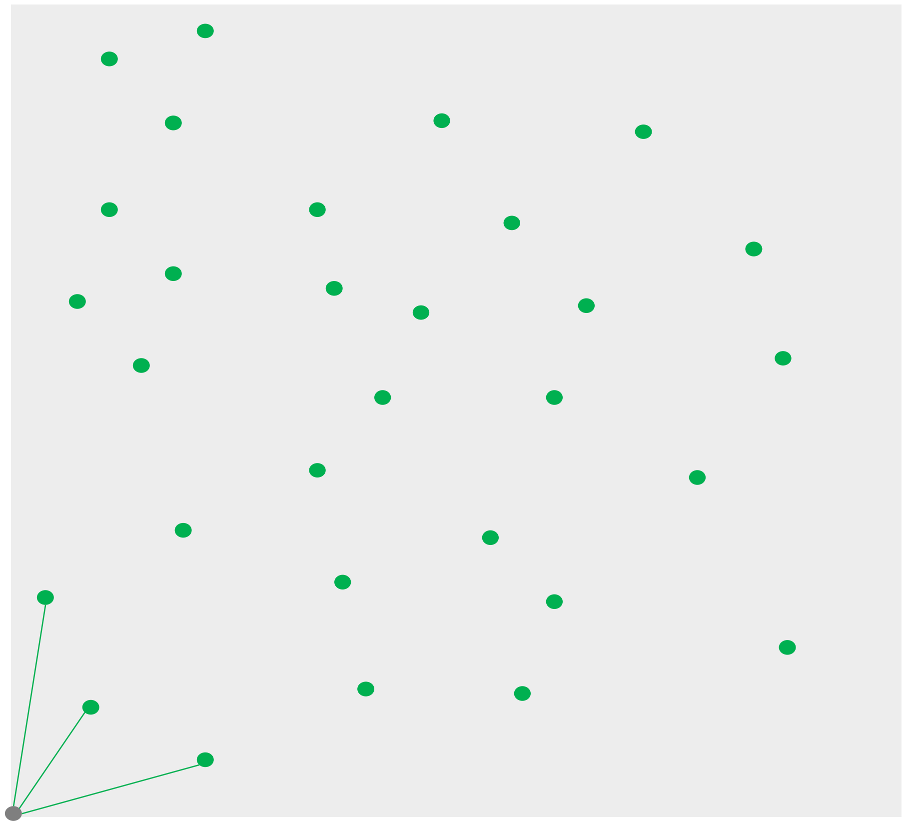
Randomly selected 30 pixels

Compute distance mean

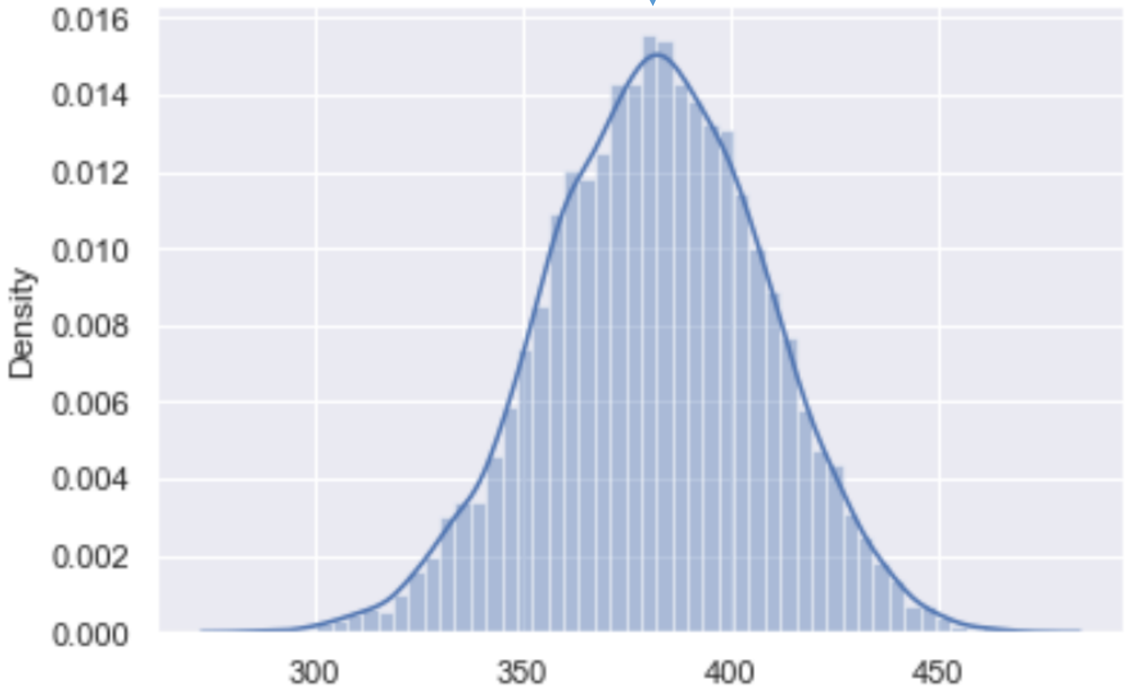
Repeat 10000 times

Histogram of 1000 distance means

500



500



Outline

- Central limit theorem
- Mean and its Applications
- Median and its Applications
- Variance and its Applications
- Quick Introduction to Correlation

Mean

❖ Definition

Data

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

Formula

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

Given the data

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

$$N = 6$$

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

$$= \frac{24}{6} = 4$$

Mean

Data

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

Formula

$$E(X) = \sum_{i=1}^{N} X_i P_X(X_i)$$

Given the data

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

$$N = 6$$

$$P_X(X = 2) = \frac{1}{6} \qquad P_X(X = 4) = \frac{2}{6}$$

$$P_X(X = 8) = \frac{1}{6} \qquad P_X(X = 1) = \frac{1}{6}$$

$$P_X(X = 5) = \frac{1}{6}$$

$$E(X) = 2 \times \frac{1}{6} + 8 \times \frac{1}{6} + 5 \times \frac{1}{6} + 4 \times \frac{2}{6} + 1 \times \frac{1}{6}$$

$$= \frac{2}{6} + \frac{8}{6} + \frac{5}{6} + \frac{8}{6} + \frac{1}{6} = 4$$

Mean

❖ Application

```python
def calculate_mean(numbers): #1
    s = sum(numbers)         #2
    N = len(numbers)         #3
    mean = s/N               #4
    return mean              #5

# Tạo mảng donations đại diện cho số tiền quyên góp trong 12 ngày
donations = [100, 60, 70, 900, 100, 200, 500, 500, 503, 600, 1000,1200]

mean_value = calculate_mean(donations)
print('Trung bình số tiền quyên góp là: ', mean_value)
```

Làm mờ ảnh dựa vào mean

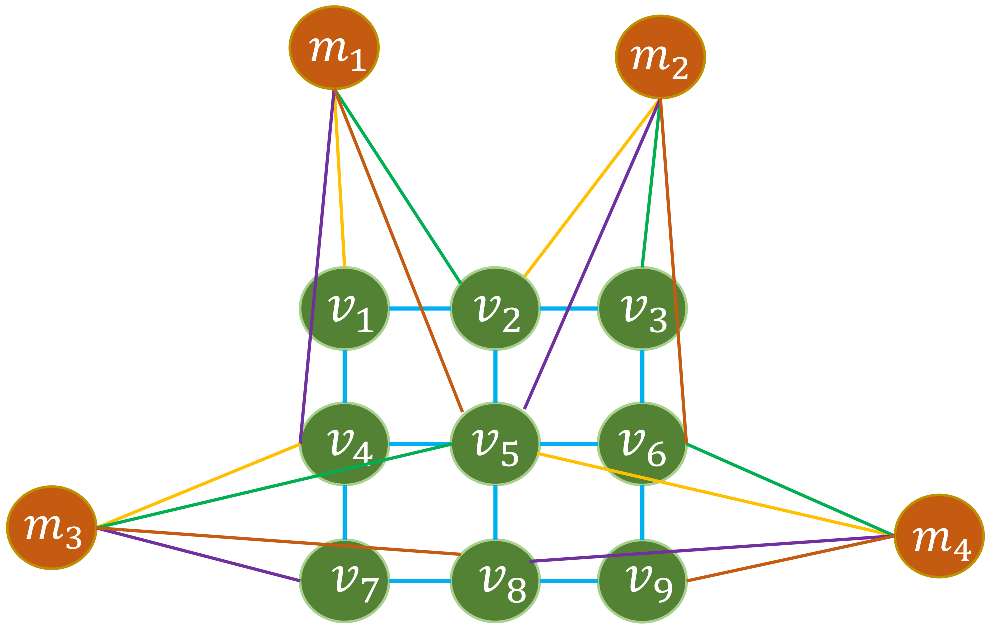
Mean Applications

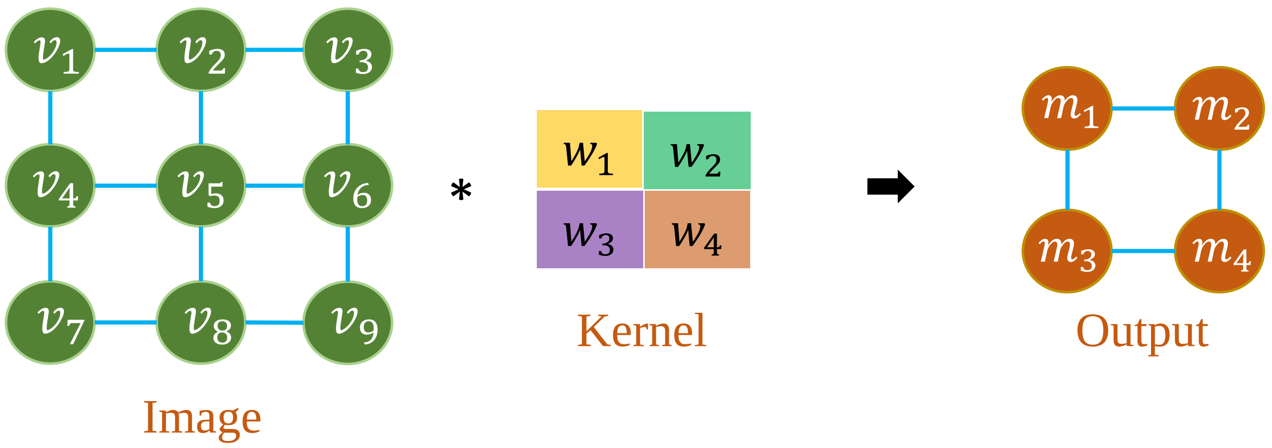
❖ Correlation (~convolution)

w_1	w_2
w_3	w_4

Kernel

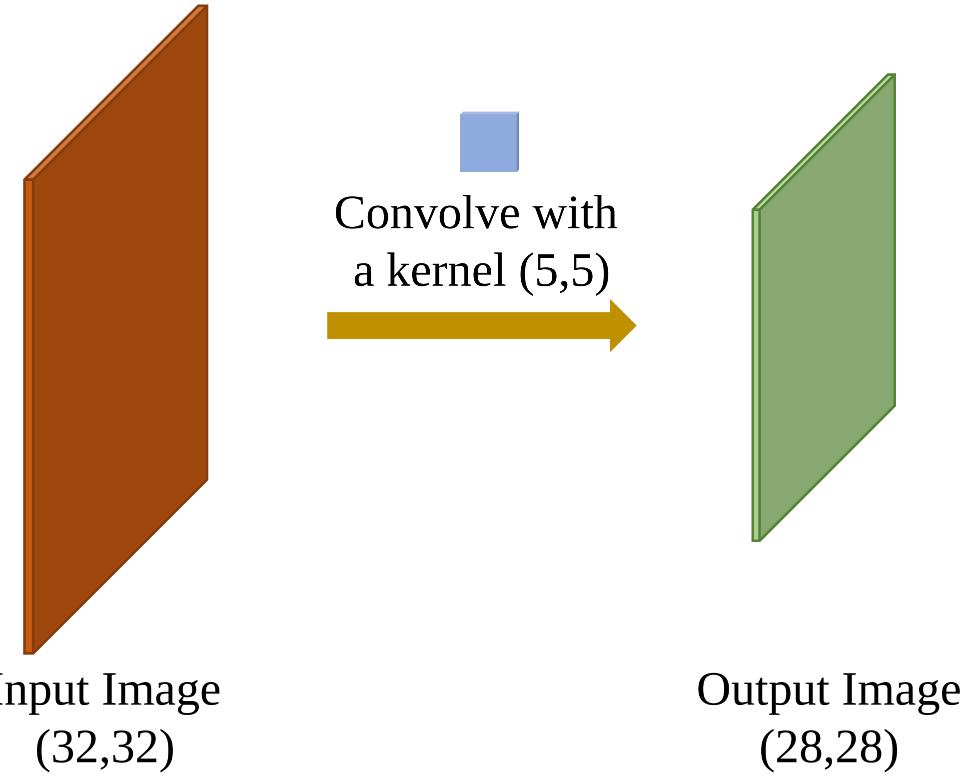
$$m_1 = v_1w_1 + v_2w_2 + v_4w_3 + v_5w_4$$

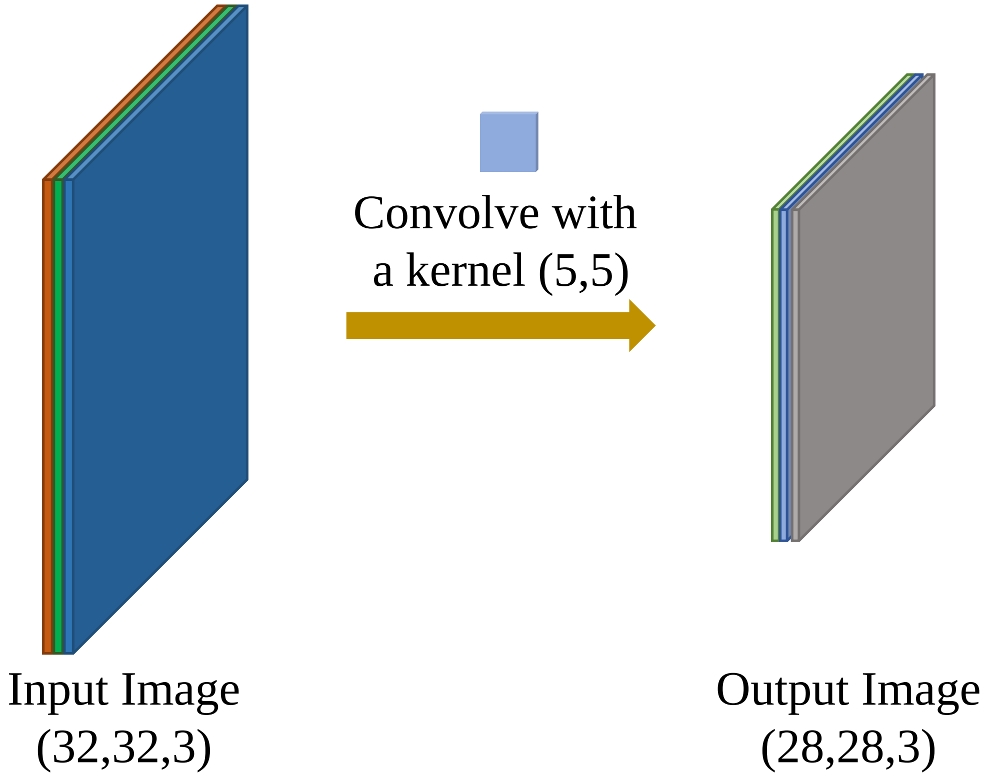




Mean Applications

❖ Correlation (~convolution)





Mean Applications

❖ Correlation (~convolution)

Kernels for computing mean

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

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

(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

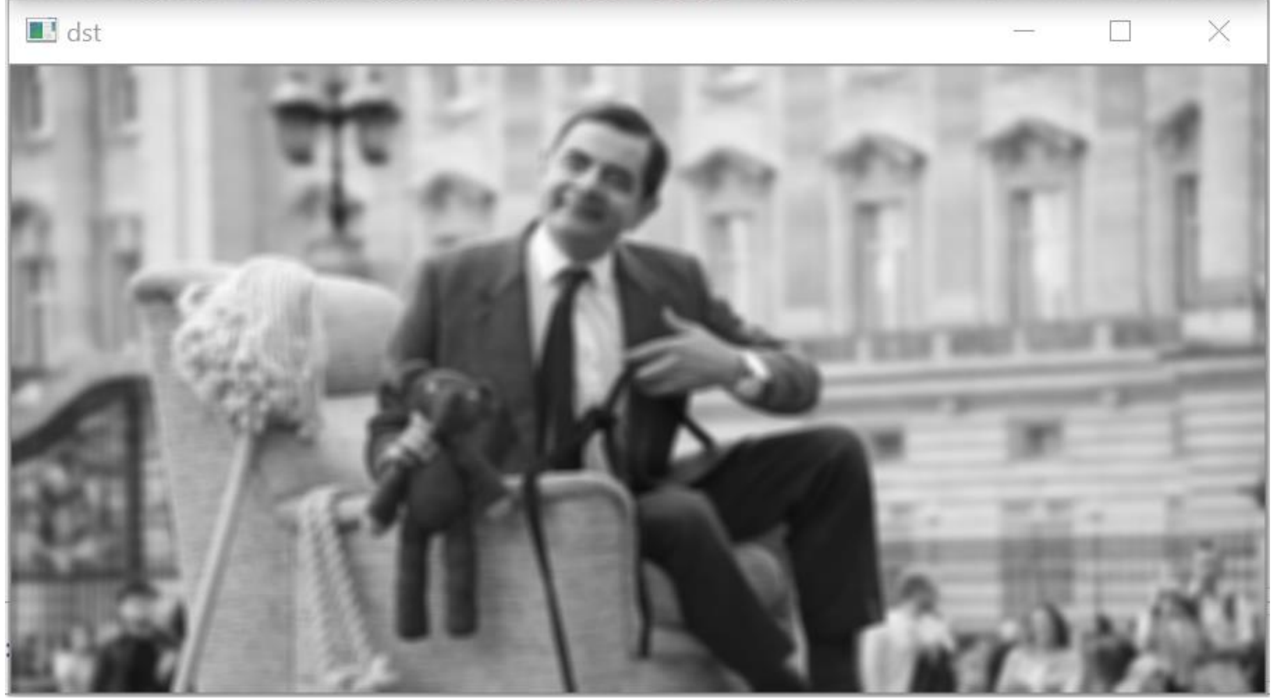
```
output_image = cv2.filter2D(input_image, cv2.CV_8U, kernel)
```

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()
```

Given the face region



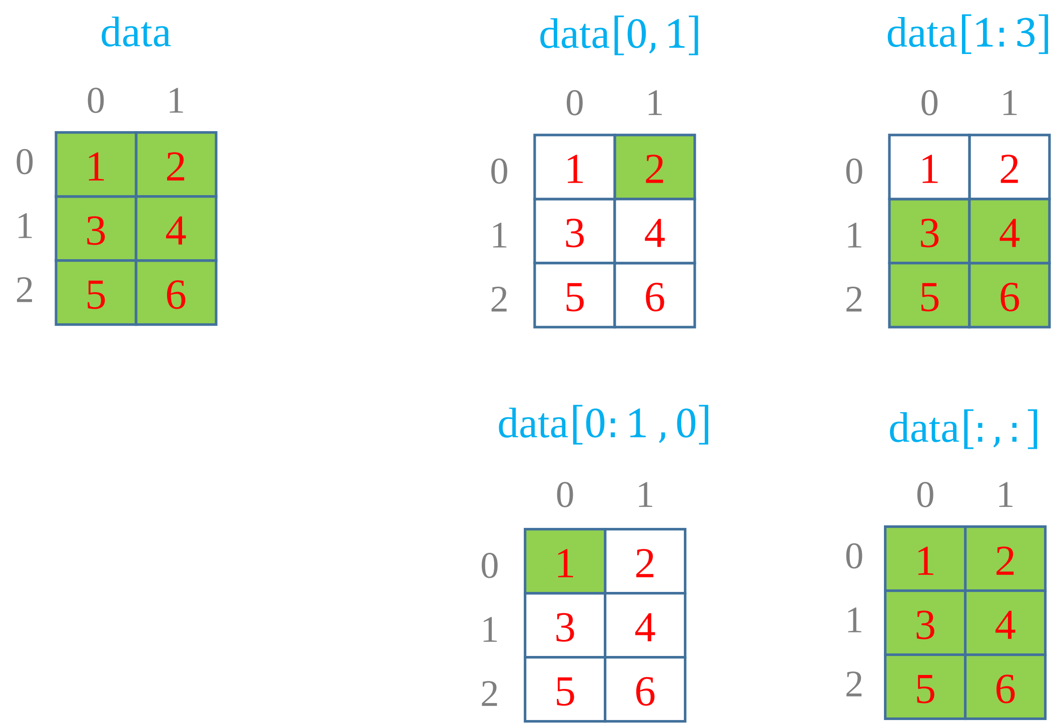
Numpy Review

❖ Slicing

arr[for_axis_0, for_axis_1, …]

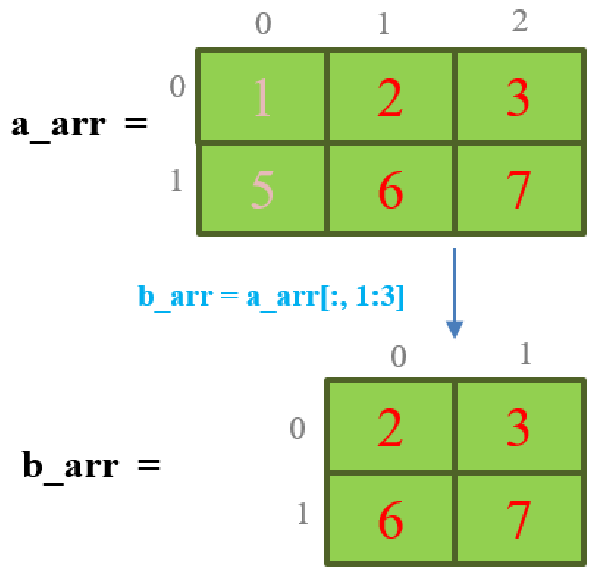
‘:’: get all the elements

‘a:b’: get the elements from a^{th} to $(b^{th}-1)$




```python
# aivietnam.ai
import numpy as np

# Khởi tạo numpy array a_arr
a_arr = np.array([[1,2,3],
                  [5,6,7]])

# Sử dụng slicing để tạo mảng b_arr
# bằng cách lấy tất cả các dòng và cột 1,2
b_arr = a_arr[:, 1:3]

print(a_arr)
print(b_arr)
```

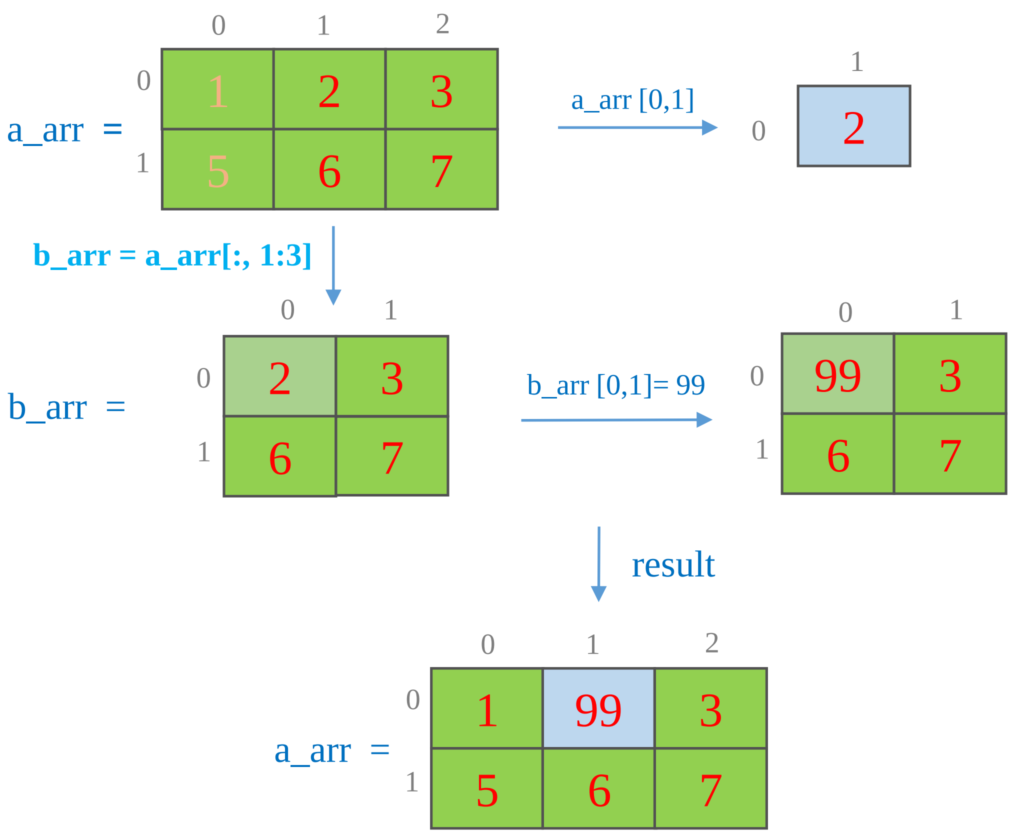
```
[[1 2 3]
 [5 6 7]]
[[2 3]
 [6 7]]
```



Numpy Review

❖ Slicing

❖ Mutable




```python
# aivietnam.ai
import numpy as np

# Khởi tạo numpy array a_arr
a_arr = np.array([[1,2,3],
                  [5,6,7]])
print('a_arr \n', a_arr)

# Sử dụng slicing để tạo mảng b_arr
b_arr = a_arr[:, 1:3]
print('b_arr \n', b_arr)

print('before changing \n', a_arr[0, 1])
b_arr[0, 0] = 99
print('after changing \n', a_arr[0, 1])
```

```
a_arr
 [[1 2 3]
 [5 6 7]]
b_arr
 [[2 3]
 [6 7]]
before changing
 2
after changing
 99
```

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.]]
```

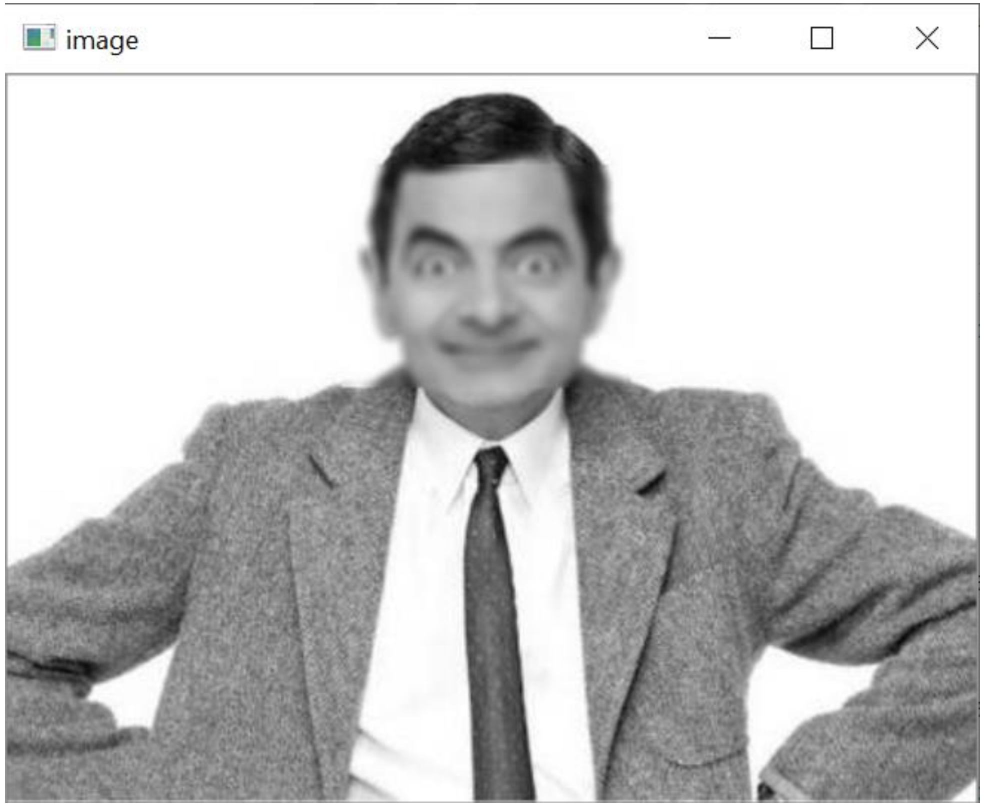
```
[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]
```

```
[[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

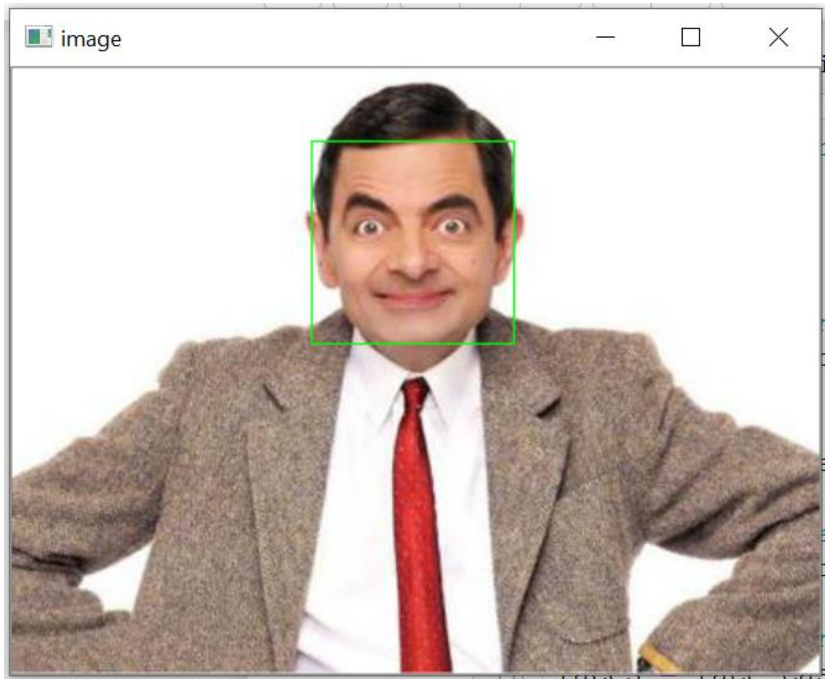
❖ Image Blurring

```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

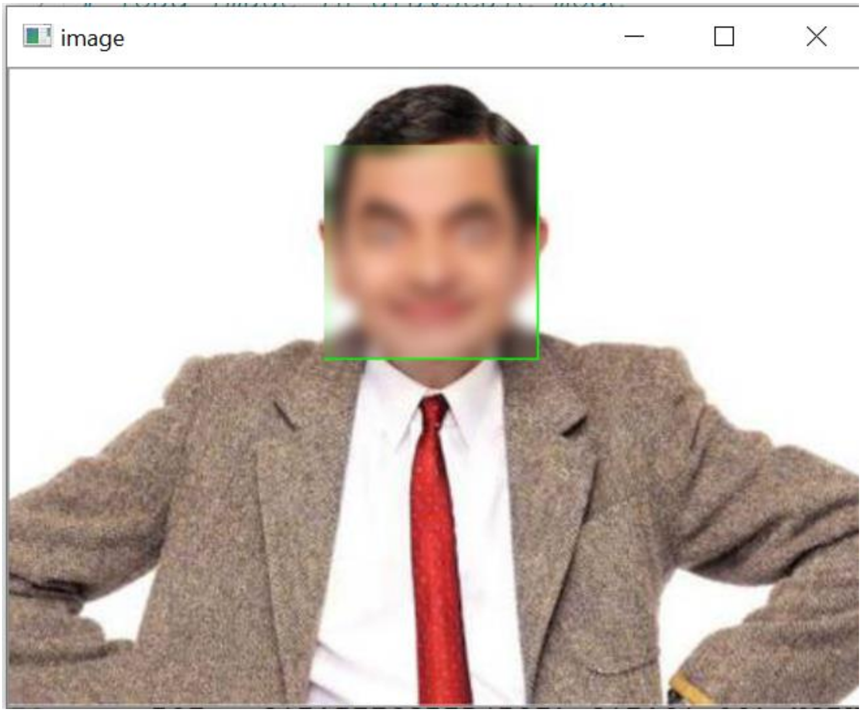



```python
# load image and blurring using face detection

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()
```

Run Code

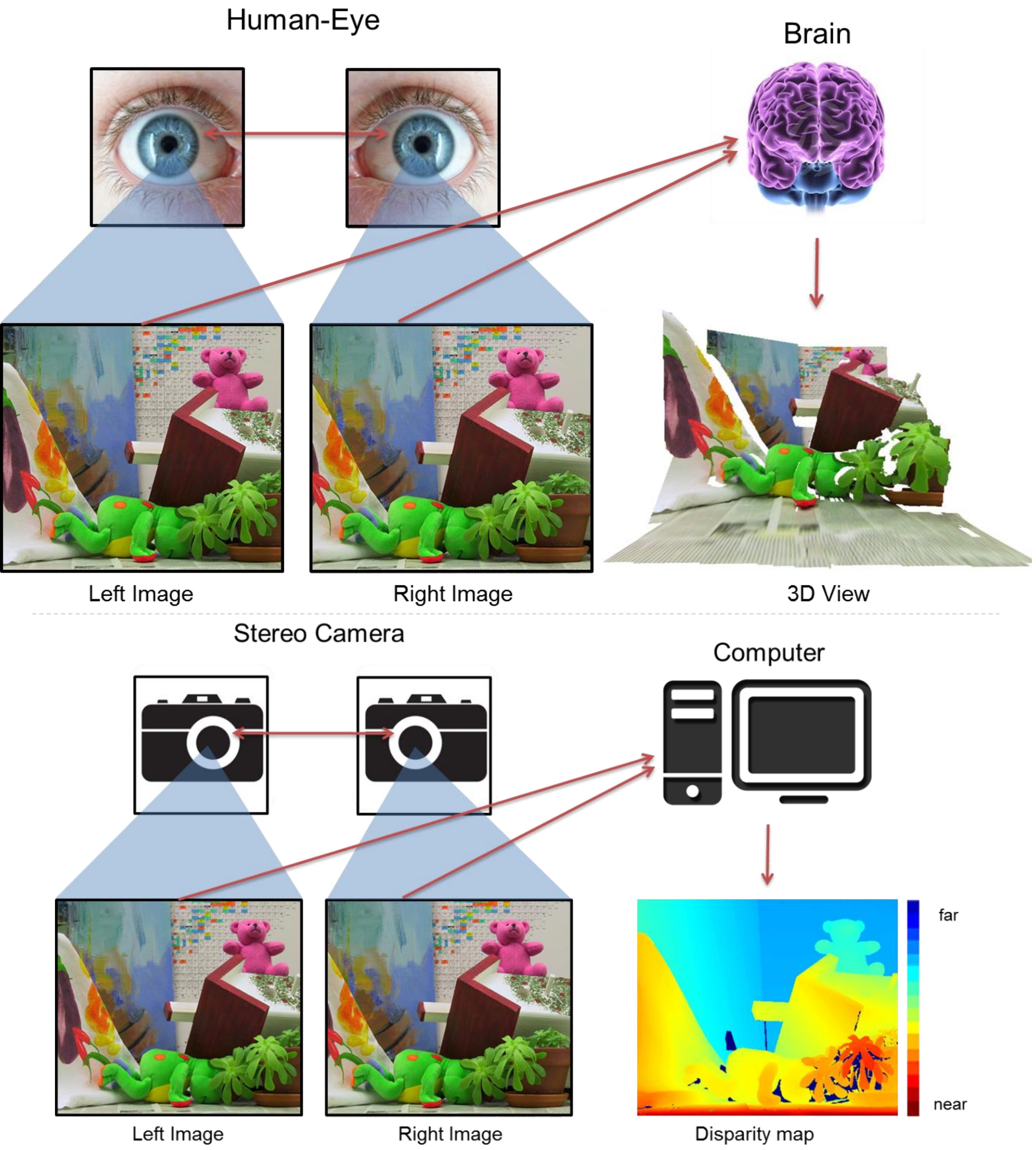
```
        # update
        img[y-pad:y+h+pad,x-pad:x+w+pad] = roi  # boundary

    # Display
    cv2.imshow('img', img)
    cv2.imshow('img2', img2)

    # Stop if escape key is pressed
    key = cv2.waitKey(30) & 0xff
    if key==27:
        break

# Release the VideoCapture object
cap.release()
cv2.destroyAllWindows()
```

In [ ]:

```

```

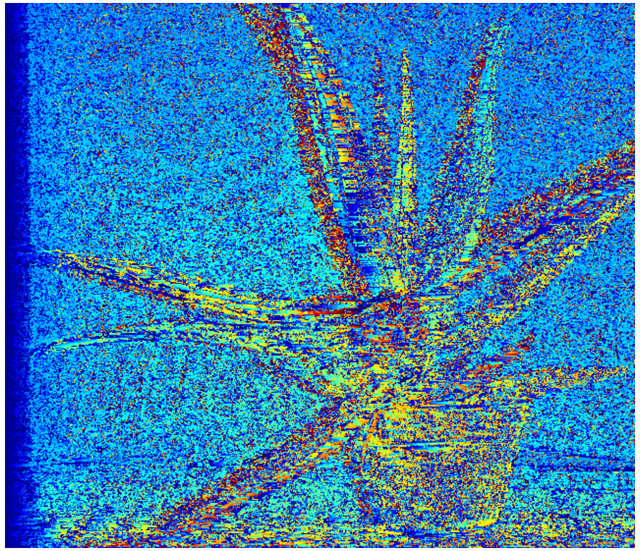
In [18]:

```
import cv2
import numpy as np

# To capture video from webcam.
cap = cv2.VideoCapture(0)

# select region
while True:
    # Read the frame and Convert to grayscale
    _, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    top_left = (200,300)
```



Mean

❖ Stereo matching



Stereo Matching

❖ Simple Method

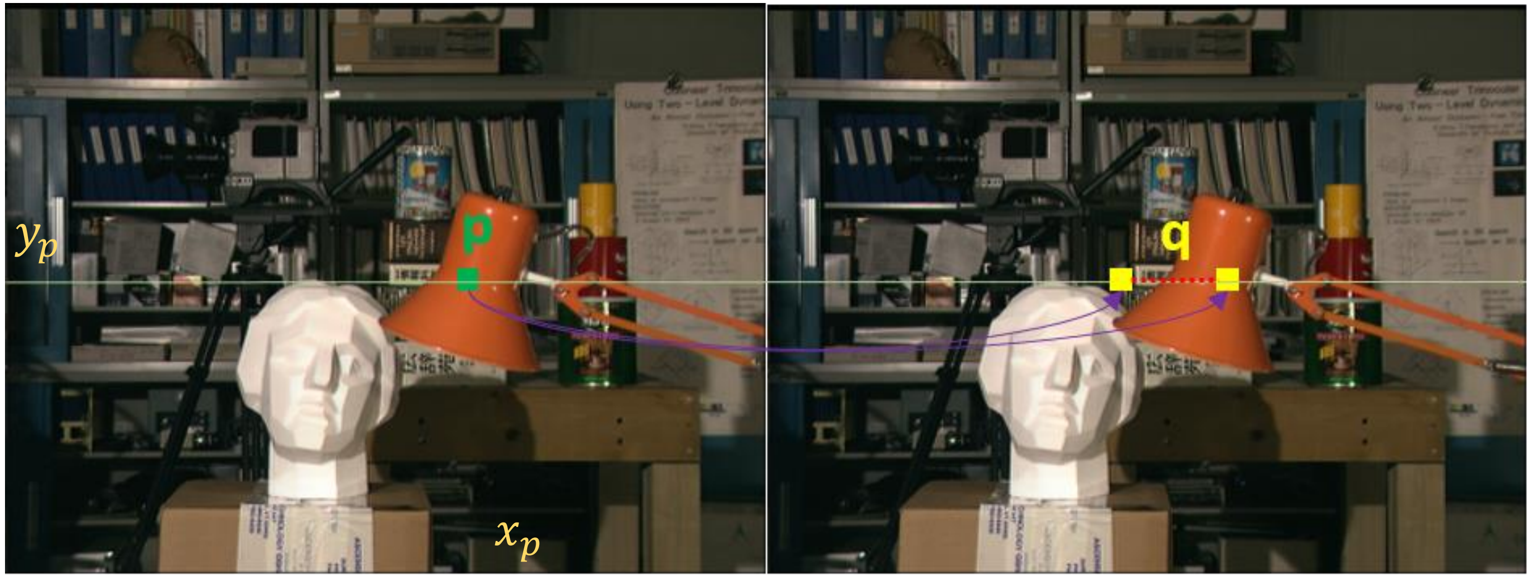
L is the left image

R is the right image

$L(\boldsymbol{p})$ is the (vector) value of **p**

$$\boldsymbol{p} = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} 234 \\ 140 \end{bmatrix}$$

$$D = 16$$



Left Image   Right Image

$\boldsymbol{q}$

$$\begin{bmatrix} x_p - 0 \\ y_p \end{bmatrix} = \begin{bmatrix} 234 \\ 140 \end{bmatrix} \quad \begin{bmatrix} x_p - 8 \\ y_p \end{bmatrix} = \begin{bmatrix} 226 \\ 140 \end{bmatrix}$$

$$\begin{bmatrix} x_p - 1 \\ y_p \end{bmatrix} = \begin{bmatrix} 233 \\ 140 \end{bmatrix} \quad \begin{bmatrix} x_p - 9 \\ y_p \end{bmatrix} = \begin{bmatrix} 225 \\ 140 \end{bmatrix}$$

$$\begin{bmatrix} x_p - 2 \\ y_p \end{bmatrix} = \begin{bmatrix} 232 \\ 140 \end{bmatrix} \quad \begin{bmatrix} x_p - 10 \\ y_p \end{bmatrix} = \begin{bmatrix} 224 \\ 140 \end{bmatrix}$$

$$\begin{bmatrix} x_p - 3 \\ y_p \end{bmatrix} = \begin{bmatrix} 231 \\ 140 \end{bmatrix} \quad \begin{bmatrix} x_p - 11 \\ y_p \end{bmatrix} = \begin{bmatrix} 223 \\ 140 \end{bmatrix}$$

$$\begin{bmatrix} x_p - 4 \\ y_p \end{bmatrix} = \begin{bmatrix} 230 \\ 140 \end{bmatrix} \quad \begin{bmatrix} x_p - 12 \\ y_p \end{bmatrix} = \begin{bmatrix} 222 \\ 140 \end{bmatrix}$$

$$\begin{bmatrix} x_p - 5 \\ y_p \end{bmatrix} = \begin{bmatrix} 229 \\ 140 \end{bmatrix} \quad \begin{bmatrix} x_p - 13 \\ y_p \end{bmatrix} = \begin{bmatrix} 221 \\ 140 \end{bmatrix}$$

$$\begin{bmatrix} x_p - 6 \\ y_p \end{bmatrix} = \begin{bmatrix} 228 \\ 140 \end{bmatrix} \quad \begin{bmatrix} x_p - 14 \\ y_p \end{bmatrix} = \begin{bmatrix} 220 \\ 140 \end{bmatrix}$$

$$\begin{bmatrix} x_p - 7 \\ y_p \end{bmatrix} = \begin{bmatrix} 227 \\ 140 \end{bmatrix} \quad \begin{bmatrix} x_p - 15 \\ y_p \end{bmatrix} = \begin{bmatrix} 219 \\ 140 \end{bmatrix}$$

Stereo Matching

❖ Simple Method

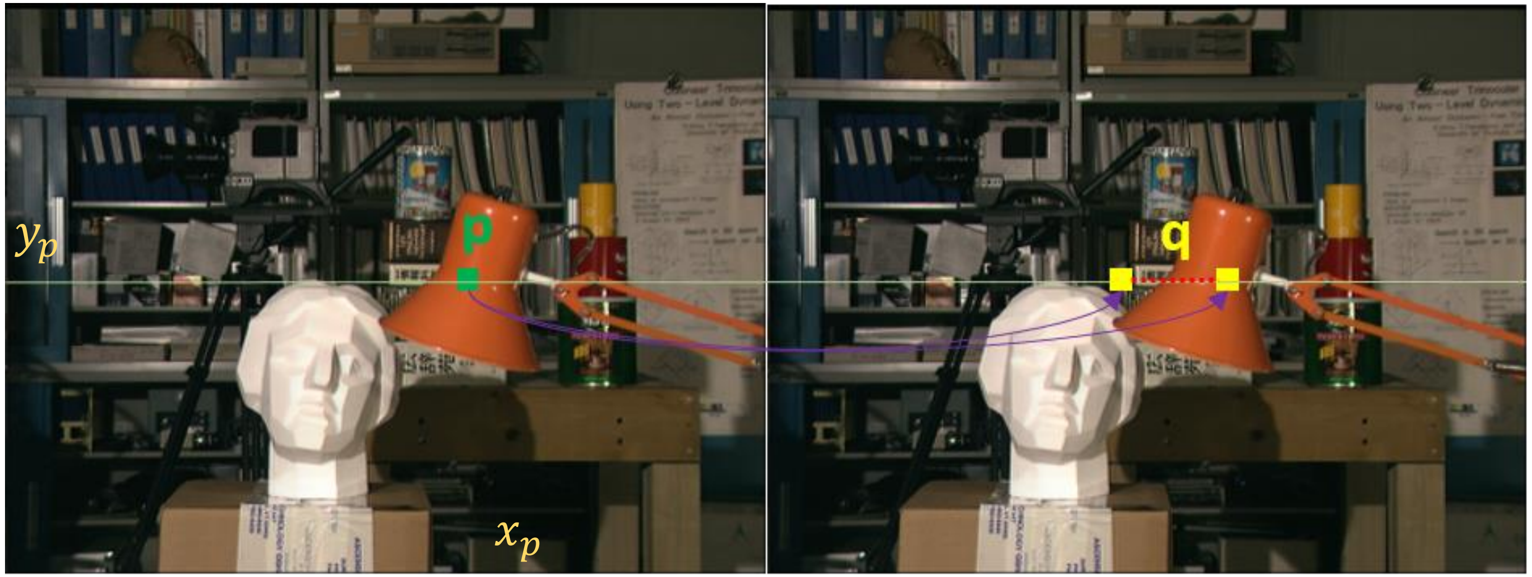
L is the left image

R is the right image

$L(\boldsymbol{p})$ is the (vector) value of **p**

$$\boldsymbol{p} = \begin{bmatrix} x_p \\ y_p \end{bmatrix} \qquad d \in D$$

$$\boldsymbol{d} = \begin{bmatrix} d \\ 0 \end{bmatrix} \qquad C_1(\boldsymbol{p}, \boldsymbol{d}) = |L(\boldsymbol{p}) - R(\boldsymbol{p} - \boldsymbol{d})|$$



Left Image

Right Image

Finding d so that $C_1(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{d})$ is minimum.

$$d = \underset{d \in D}{\operatorname{argmin}}\left(C_1(\boldsymbol{p}, \boldsymbol{d})\right)$$

Then, d is the value for the pixel **p** in disparity map

Stereo Matching

❖ Simple Method : Result

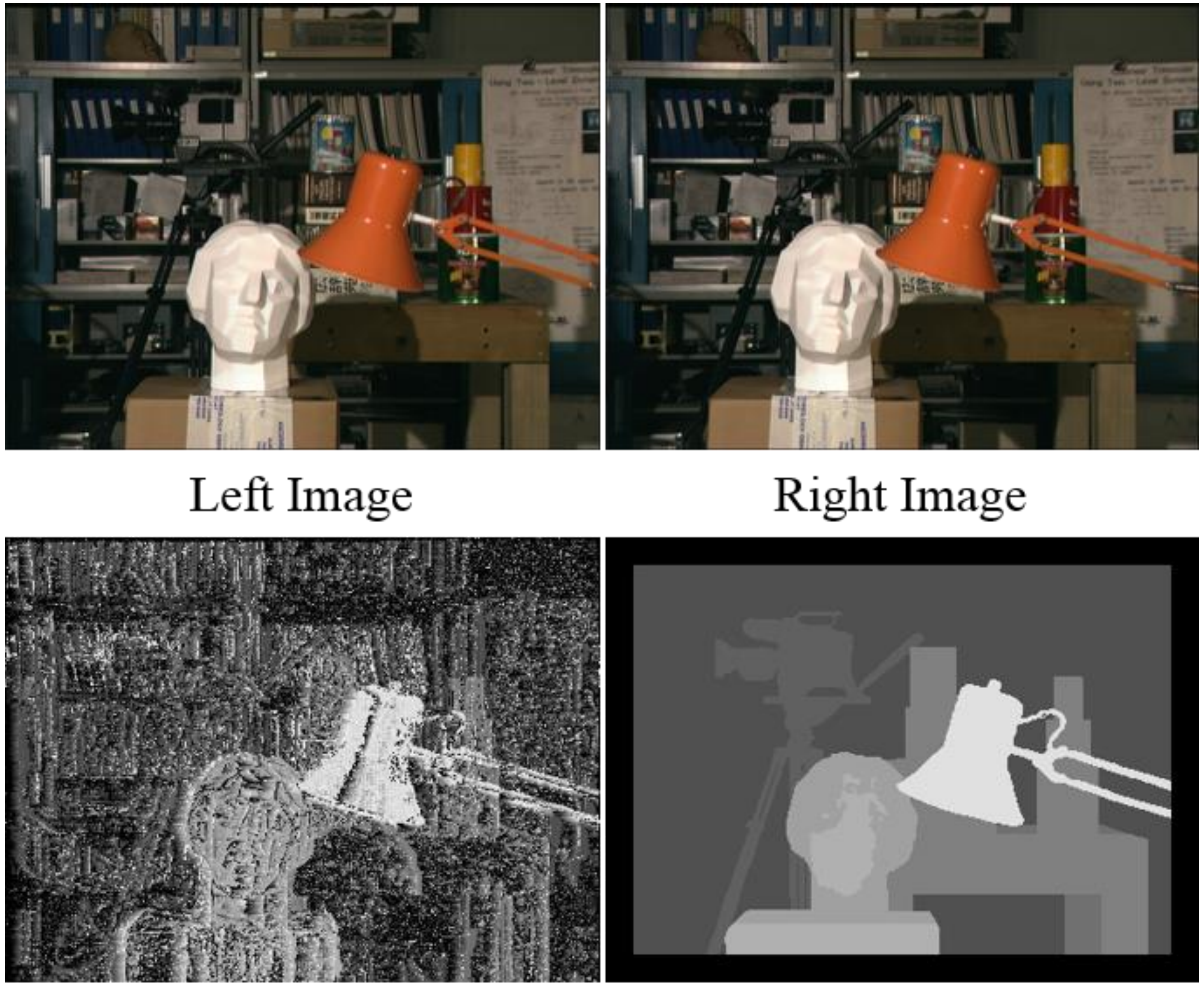
$$\boldsymbol{d} = \begin{bmatrix} d \\ 0 \end{bmatrix} \qquad d \in D$$

$$C_1(\boldsymbol{p}, \boldsymbol{d}) = |L(\boldsymbol{p}) - R(\boldsymbol{p} - \boldsymbol{d})|$$

Finding d so that $C_1(\boldsymbol{p}, \boldsymbol{d})$ is minimum.

$$d = \underset{d \in D}{\operatorname{argmin}}\big(C_1(\boldsymbol{p}, \boldsymbol{d})\big)$$

Then, d is the value for the pixel **p** in disparity map



Left Image | Right Image

Disparity Map | Ground Truth

Stereo Matching

❖ Method 1: Implementation

$$\boldsymbol{d} = \begin{bmatrix} d \\ 0 \end{bmatrix} \qquad d \in D$$

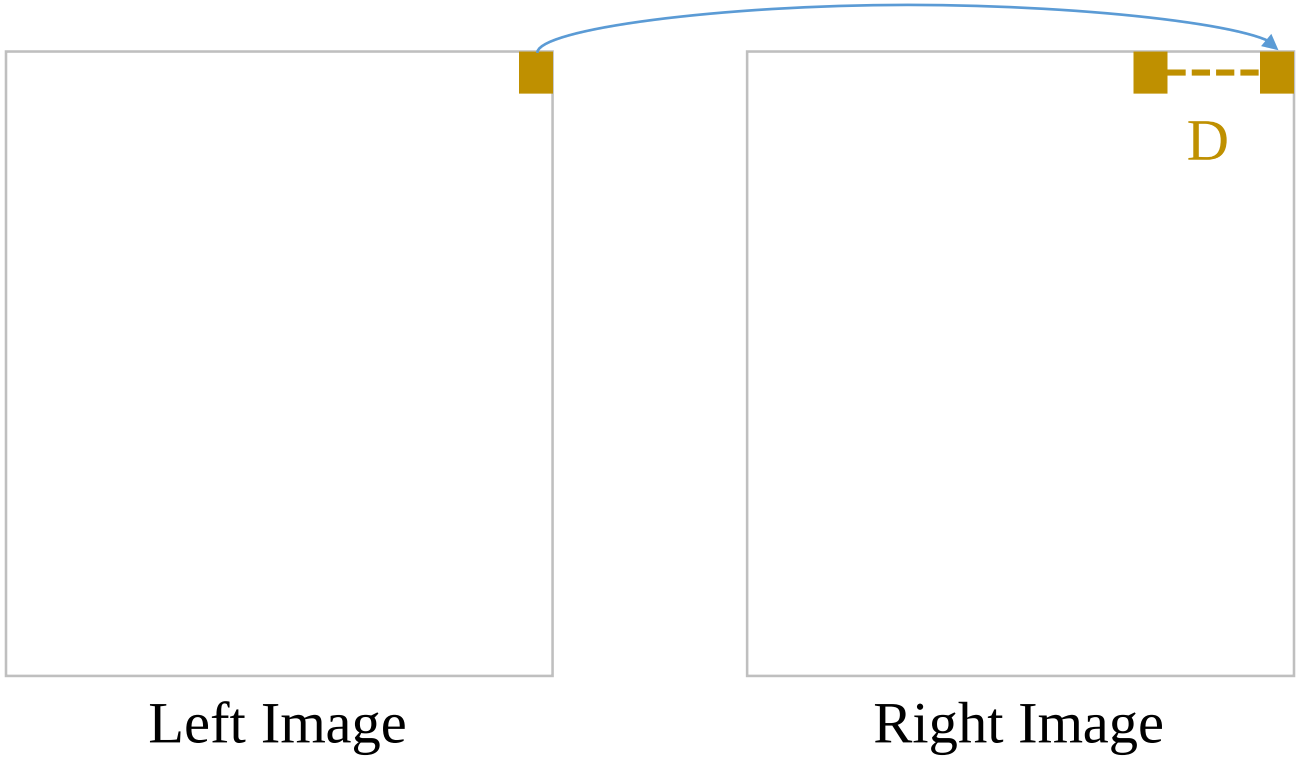
$$C_1(\boldsymbol{p}, \boldsymbol{d}) = |L(\boldsymbol{p}) - R(\boldsymbol{p} - \boldsymbol{d})|$$

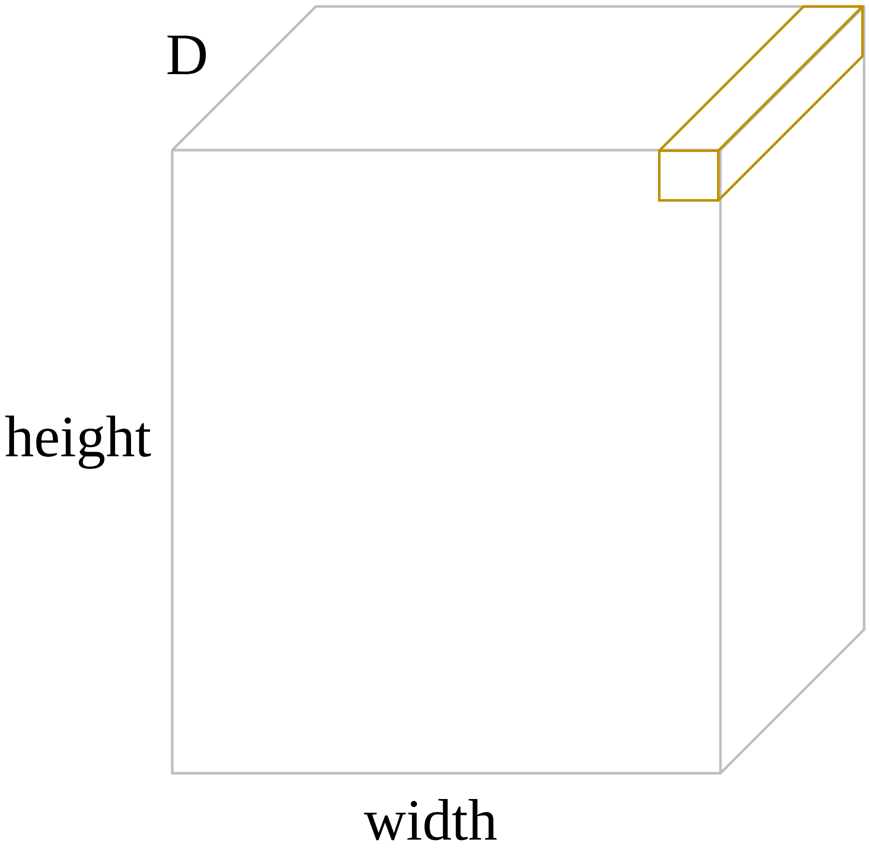
Finding d so that $C_1(\boldsymbol{p}, \boldsymbol{d})$ is minimum.

$$d = \underset{d \in D}{\operatorname{argmin}}\left(C_1(\boldsymbol{p}, \boldsymbol{d})\right)$$

Then, d is the value for the pixel **p** in disparity map

```python
# disparity map
depth = np.zeros((height, width), np.uint8)
scale = 255 / disparity_range

for y in range(height):
    for x in range(width):

        # find argmin_d
        disparity = 0
        cost_min  = abs(left[y, x] - right[y, x])
        for d in range(disparity_range):
            if (x - d) < 0:
                cost = 255
            else:
                cost = abs(left[y, x] - right[y, x - d])

            # update cost_min
            if cost < cost_min:
                cost_min  = cost
                disparity = d

        # set to the disparity map
        depth[y, x] = disparity*scale

# save
cv2.imwrite('images/disparity_ad.png', depth)
```



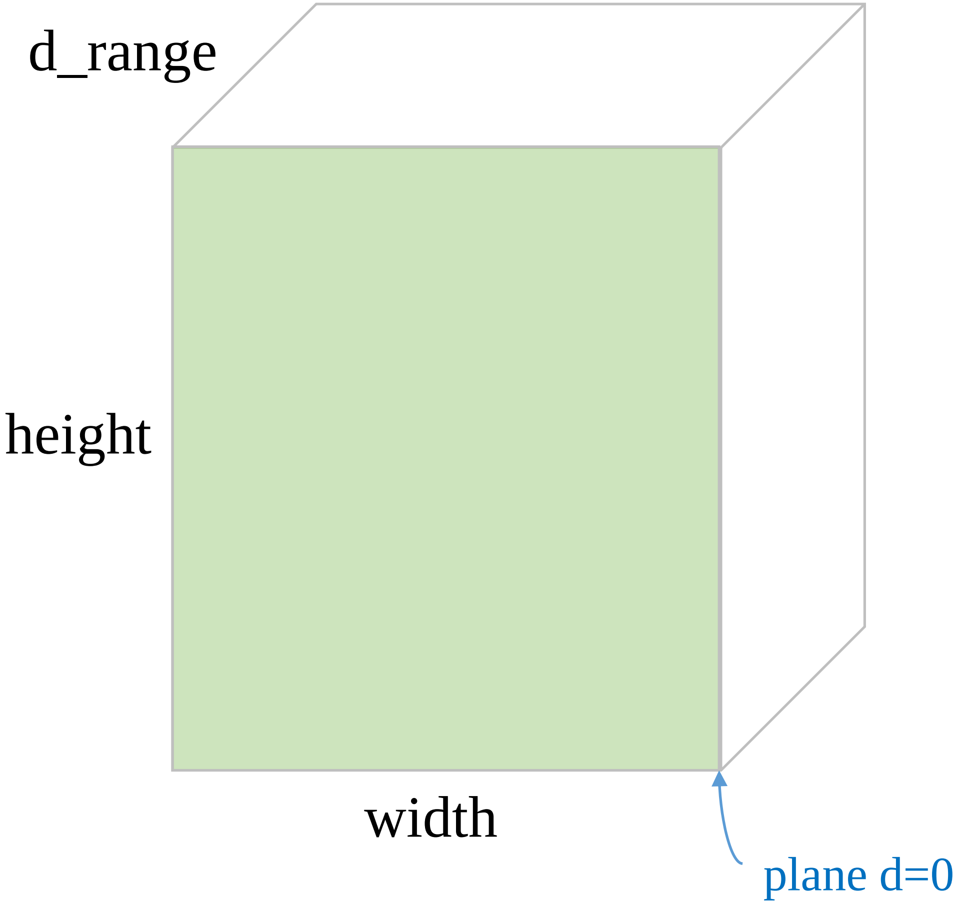


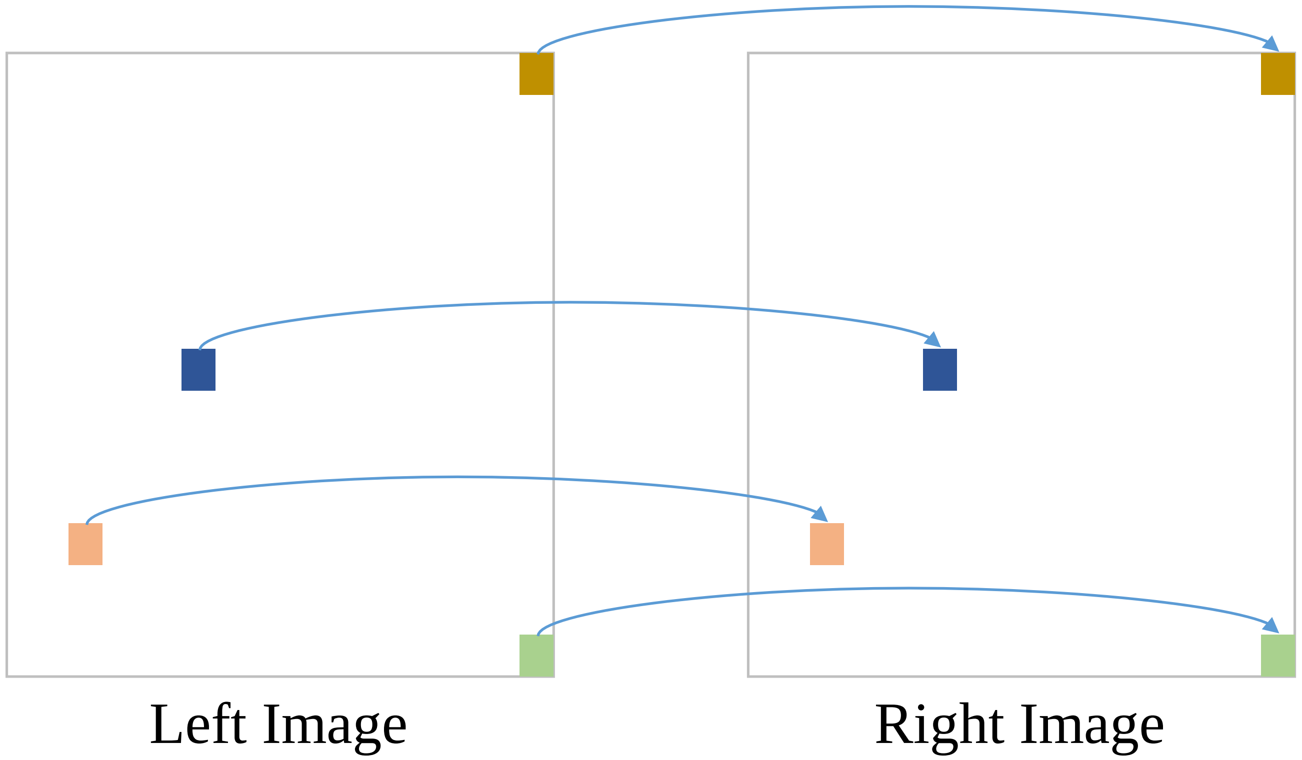
Normal approach

Each pixel p in the left image has D candidate pixel q in the right image.

Then, D cost values are computed from D pairs (**p**, **q**)

Stereo Matching



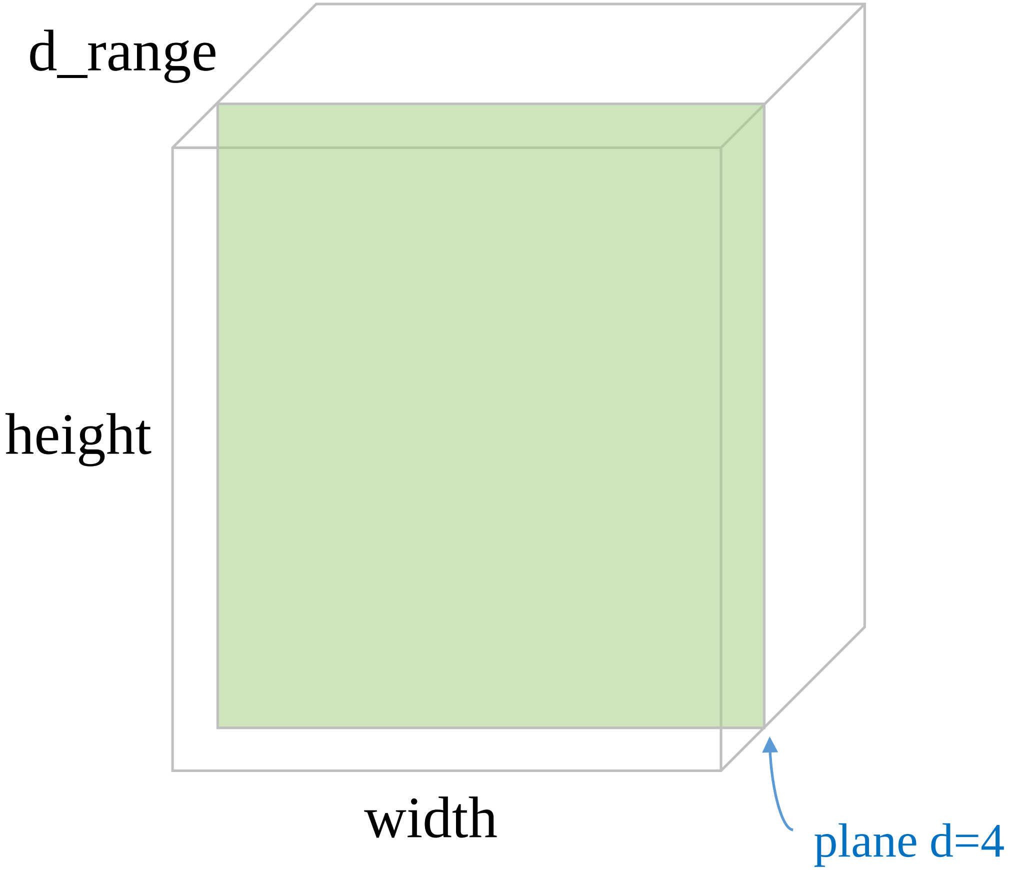


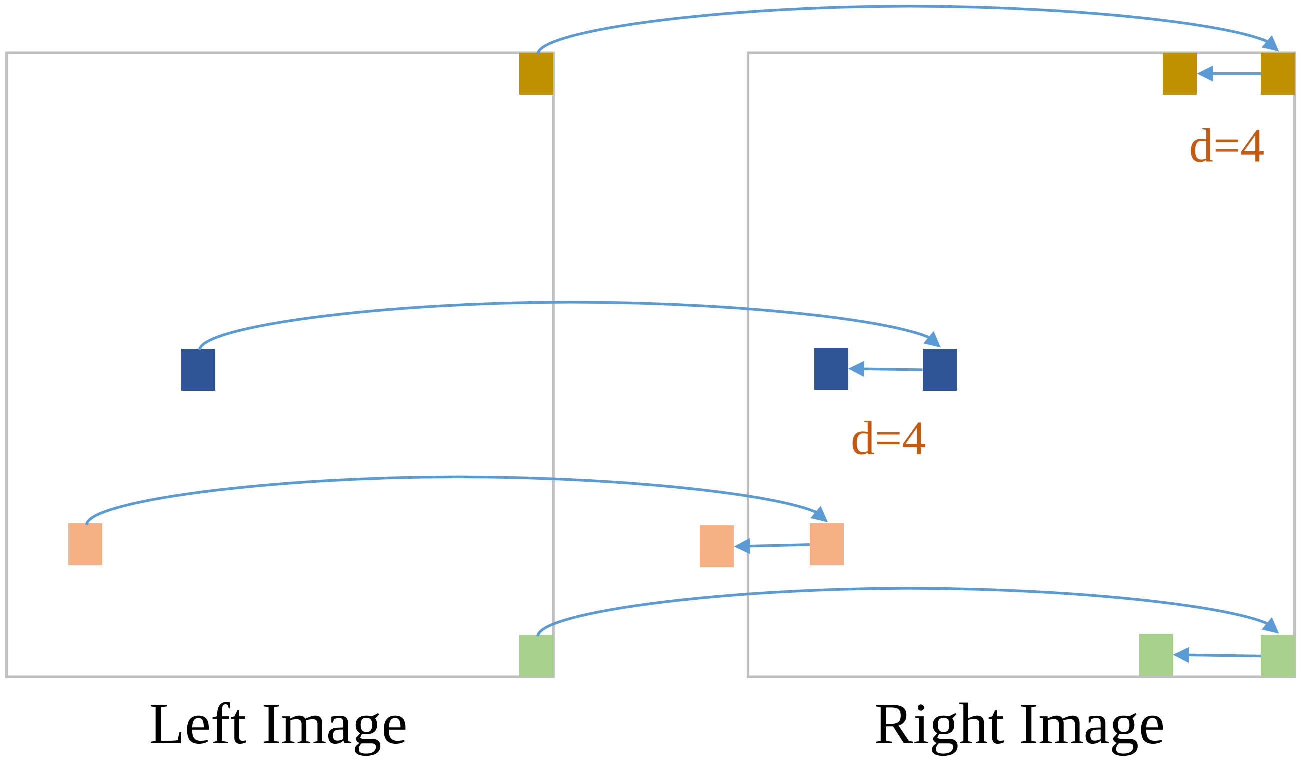
Cost values for a disparity d=0

Given C_d is a cost plane for a disparity d

$$C_0 = |L - R|$$

Stereo Matching

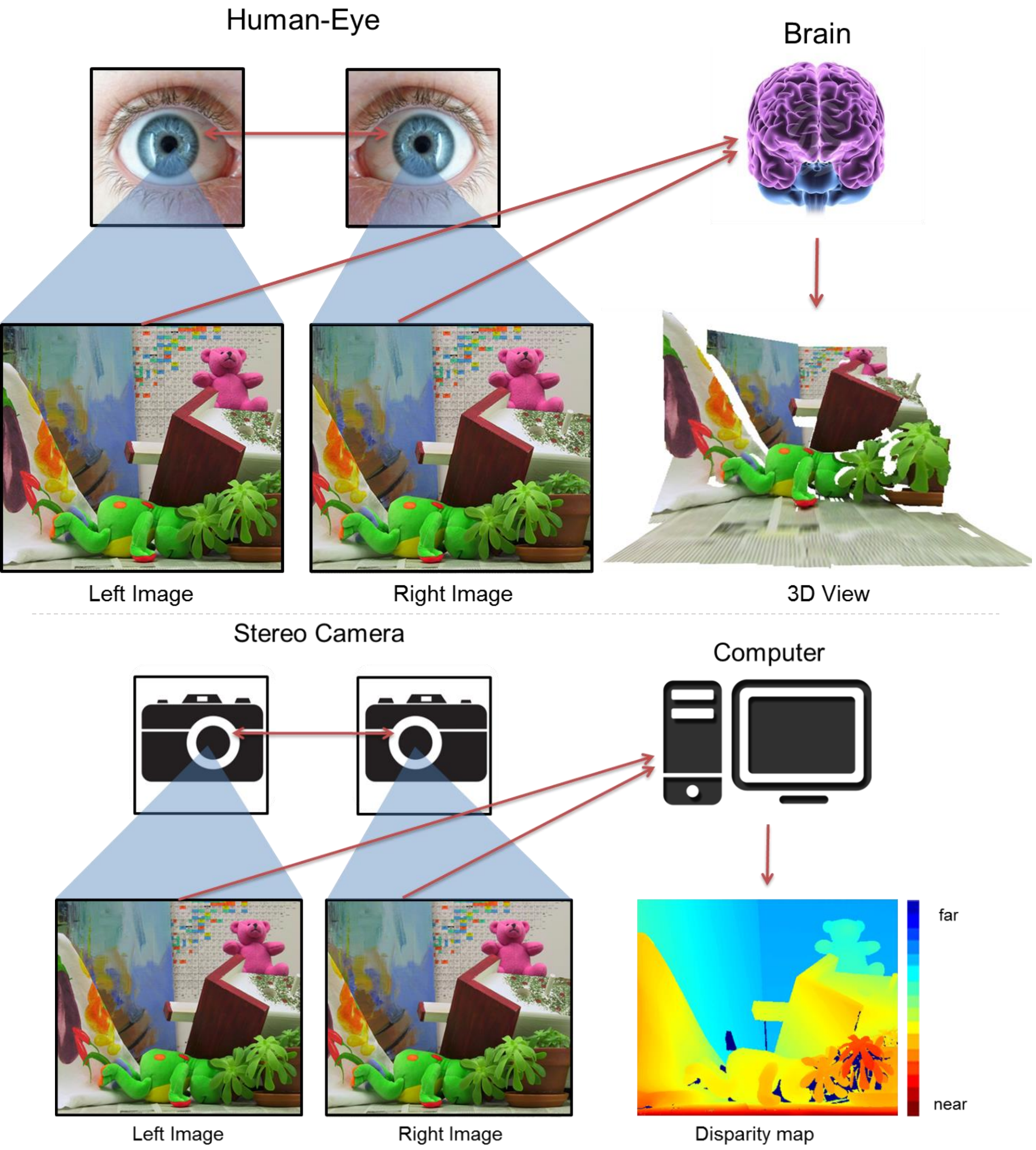


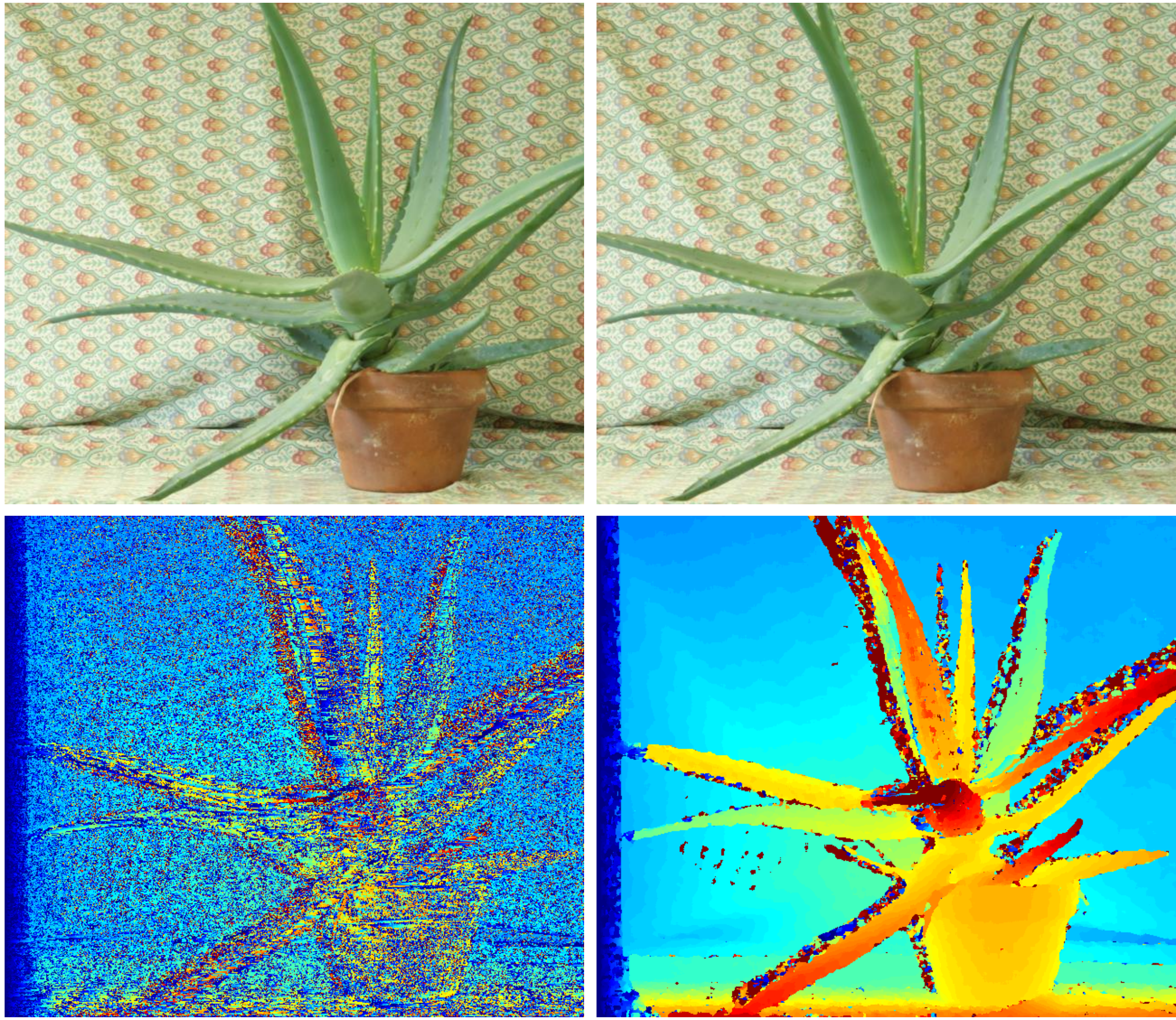


Cost values for a disparity d=4

Mean

❖ Stereo matching





Outline

- Central limit theorem
- Mean and its Applications
- Median and its Applications
- Variance and its Applications
- Quick Introduction to Correlation

Median

❖ Definition

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

❖ Definition

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

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 $m = \left(S_{\left(\frac{N}{2}\right)} + S_{\left(\frac{N}{2}+1\right)}\right)/2$

```python
def calculate_median(numbers):
    N = len(numbers)
    numbers.sort()
    if N%2 == 0:
        m1 = N/2
        m2 = (N/2) + 1
        m1 = int(m1)-1
        m2 = int(m2)-1
        median = (numbers[m1] + numbers[m2])/2
    else:
        m = (N+1)/2
        m = int(m)-1
        median = numbers[m]
    return median
```

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 clo
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

- Central limit theorem
- Mean and its Applications
- Median and its Applications
- Variance and its Applications
- Quick Introduction to Correlation

Variance

❖ Definition

Formula:

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

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

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

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

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

$$var(X) = \frac{1}{5}[(5-5)^2 + (3-5)^2 + (6-5)^2 + (7-5)^2 + (4-5)^2]$$
$$= \frac{1}{5}(0+4+1+4+1)=2$$

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

Ignore the differences between population and sample!!!

Variance

Formula

mean

$$E(X) = \sum_{i=1}^{N} X_i P_X(X_i)$$

variance

$$var(X) = E\left(\left(X - E(X)\right)^2\right)$$

$$= \sum_{i=1}^{N} \left(X_i - E(X)\right)^2 P_X(X_i)$$

Standard deviation

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

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

$$E(X) = 5 \times \frac{1}{5} + 3 \times \frac{1}{5} + 6 \times \frac{1}{5} + 7 \times \frac{1}{5} + 4 \times \frac{1}{5}$$

$$= 5$$

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

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

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

Variance

Formula

mean

$$E(X) = \sum_{i=1}^{N} X_i P_X(X_i)$$

variance

$$var(X) = E\left(\left(X - E(X)\right)^2\right)$$
$$= \sum_{i=1}^{N} \left(X_i - E(X)\right)^2 P_X(X_i)$$

Standard deviation

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

$$var(X) = \sum_{i=1}^{N} \left(X_i - E(X)\right)^2 P_X(X_i)$$
$$= \sum_{i=1}^{N} \left(X_i^2 - 2X_i E(X) + E(X)^2\right) P_X(X_i)$$
$$= \sum_{i=1}^{N} X_i^2 P_X(X_i) - \sum_{i=1}^{N} 2X_i E(X) P_X(X_i)$$
$$+ \sum_{i=1}^{N} E(X)^2 P_X(X_i)$$
$$= E(X^2) - 2E(X)\left[\sum_{i=1}^{N} X_i P_X(X_i)\right] + E(X)^2$$
$$= E(X^2) - \left(E(X)\right)^2$$

Variance

```python
import numpy as np

x = np.array([5, 3, 6, 7, 4])

var = (x - x.mean())**2
var = np.mean(var)
print(var)

sdv = np.sqrt(var)
print(sdv)
```

```
2.0
1.4142135623730951
```

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

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

def caculate_variance(numbers):
    mean = calculate_mean(numbers)

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

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

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

    return variance
```

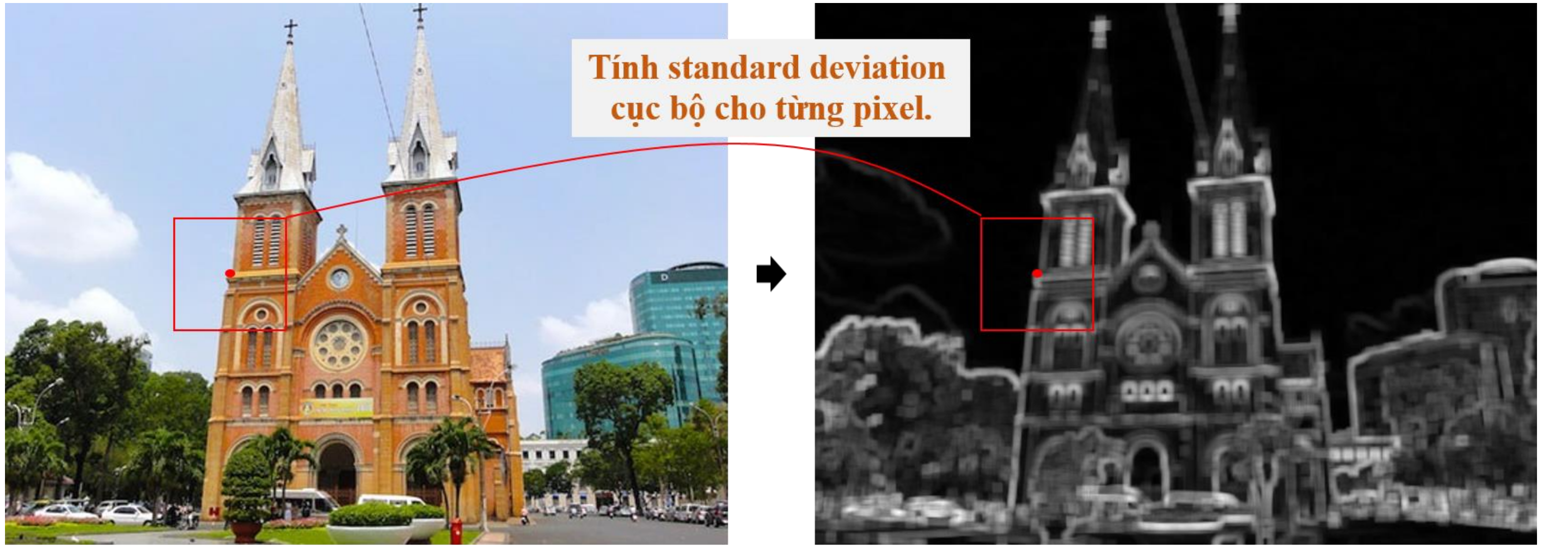
Variance

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

Tính standard deviation cục bộ cho từng pixel.

Image

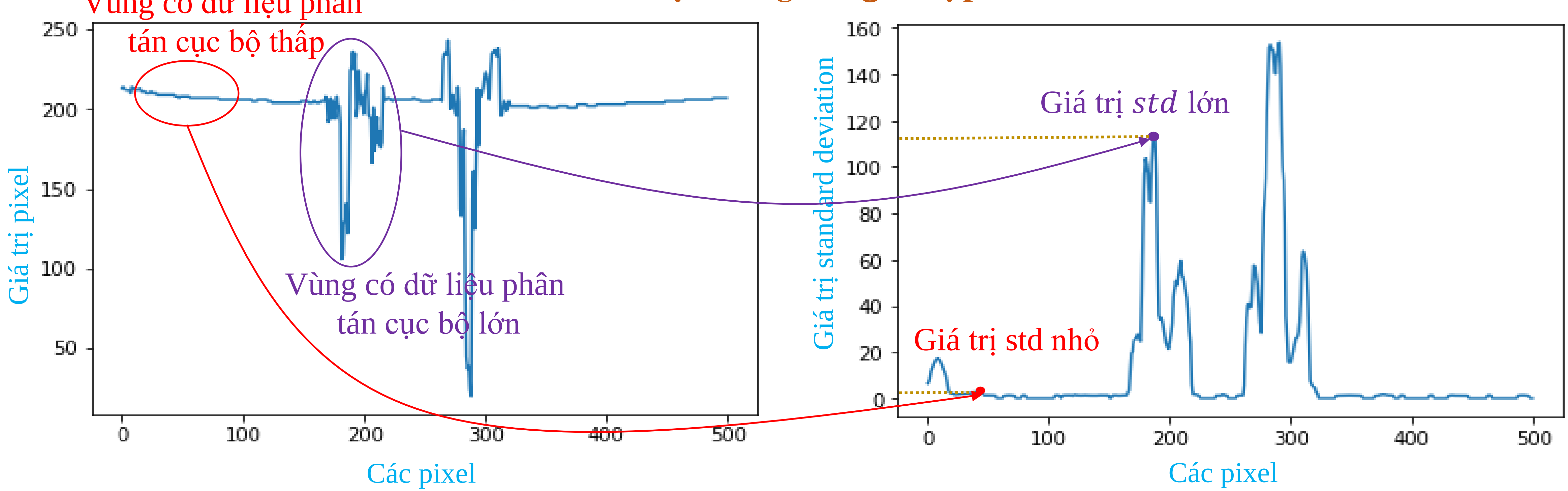
Image Texture



Ảnh gốc

Ảnh thông tin texture

Quan sát một mảng các giá trị pixel.

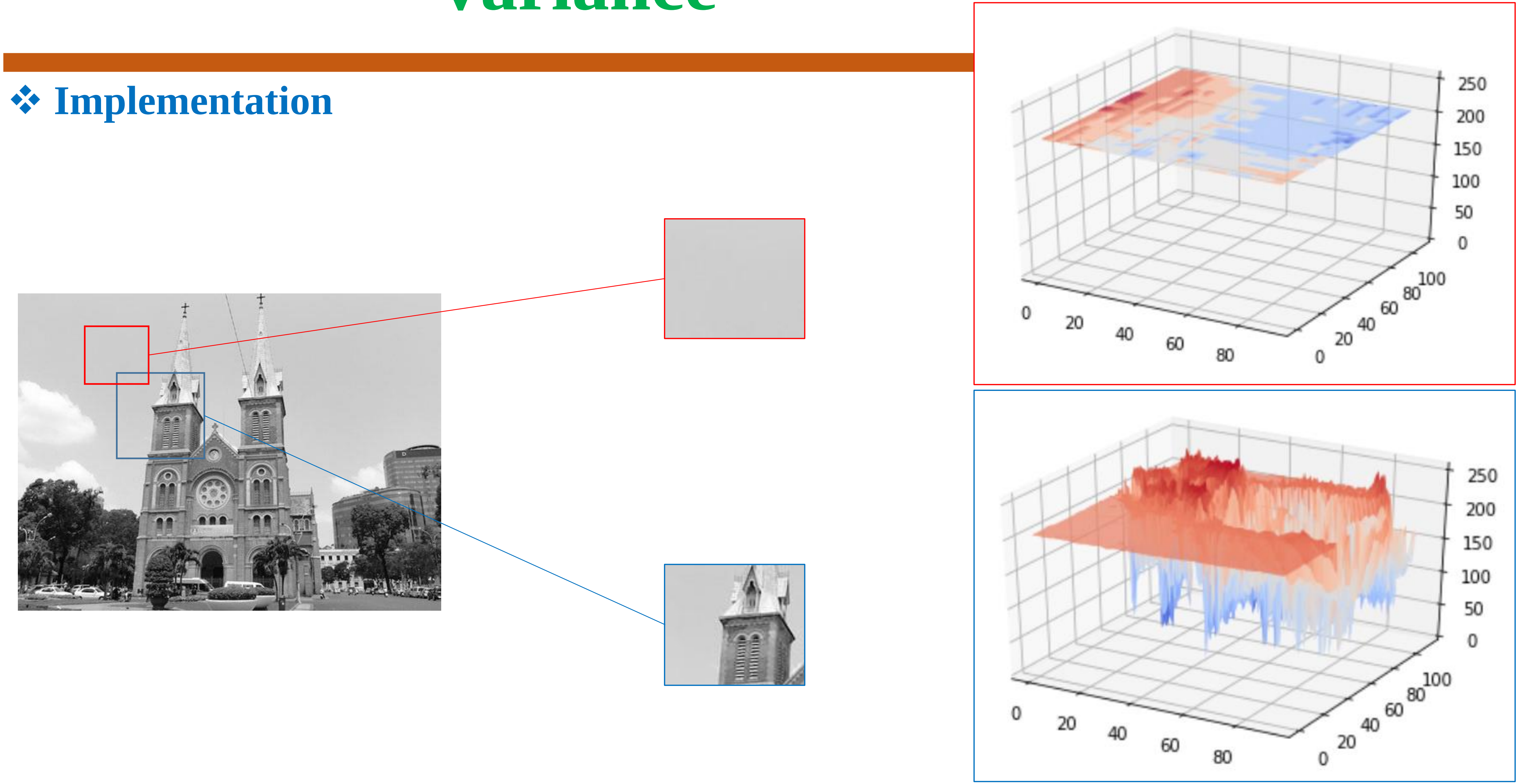


Variance

❖ 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

- Central limit theorem
- Mean and its Applications
- Median and its Applications
- Variance and its Applications
- Quick Introduction to Correlation

Correlation Coefficient

ignore the differences between population and sample

❖ Definition

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]

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

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

Mean-Demo Last Checkpoint: 9 giờ trước (unsaved changes)

```
    # Display
    cv2.imshow('img', img)

    # Stop if escape key is pressed
    if cv2.waitKey(33) == ord('a'):
        break

template = gray[300:400, 200:400]
w, h = template.shape[::-1]
cv2.imshow('template', template)
cv2.waitKey(0)

while True:
    # Read the frame and Convert to grayscale
    _, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

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

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

    # draw
    cv2.rectangle(img, top_left, bottom_right, (0, 255, 0), 2)


    # Display
    cv2.imshow('img', img)

    # Stop if escape key is pressed
    if cv2.waitKey(33) == ord('a'):
        break

# Release the VideoCapture object
cap.release()
cv2.destroyAllWindows()
```

In [ ]: