TRƯỜNG ĐẠI HỌC VINH
VINH UNIVERSITY
Nơi tạo dựng tương lai cho tuổi trẻ

Chương 2: Các phép toán cơ bản và phương pháp xử lý ảnh số

ThS. Nguyễn Thị Minh Tâm
Email: tamntm@vinhuni.edu.vn

Đại học Vinh
Viện Kỹ thuật Công nghệ

ĐẠI HỌC VINH - 2022

1. Các phép toán trên điểm ảnh

- Khái niệm
  - Thông thường mỗi điểm ảnh có 2 đặc trưng: **vị trí và giá trị màu**. Khi xử lý một điểm ảnh ta chỉ quan tâm tới giá trị màu của nó mà không quan tâm tới vị trí của nó.
  - Do đó, hai điểm ảnh có vị trí khác nhau, nhưng nếu có cùng giá trị màu thì đầu ra sẽ bằng nhau.
  - Nếu ta có giá trị màu đầu vào là u, thì giá trị đầu ra v = ϕ(u).
  - Về bản chất xử lý điểm ảnh là một ánh xạ:

$$f : X(m,n) \rightarrow Y(m,m)$$

$$u(m,n) \rightarrow v(m,n) = \phi(u(m,n))$$

  - Hay ta có thể viết: Y(m,n) = ϕ(X(m,n))

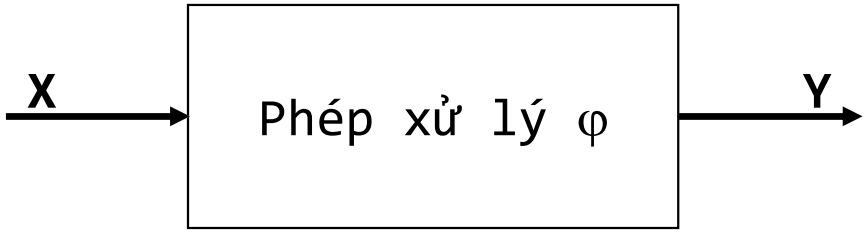
Các phép toán trên điểm ảnh

- Phép biến đổi ảnh bất kì về ảnh nhị phân:

$$Y(m,n) = \begin{cases} 1 \;\; if \;\; X(m,n) \geq \theta \;(\text{nguong}) \\ 0 \;\; if \;\; X(m,n) < \theta \end{cases}$$

Toán tử tuyến tính

- Gọi X(m,n) là ảnh vào và Y(m,n) là ảnh ra của phép xử lý có dạng:



- Khi đó, φ được gọi là toán tử tuyến tính nếu thoả mãn:

$$\varphi(a(X1(m,n))+b(X2(m,n))) = aY1(m,n)+bY2(m,n)$$

với

$$Y1(m,n) = \varphi(X1(m,n))$$

$$Y2(m,n) = \varphi(X2(m,n)).$$

2. Điều chỉnh độ sáng và độ tương phản trong ảnh

- Giả sử f là một hàm biểu diễn cho một ảnh nào đó, f(x,y) là giá trị của pixel trong ảnh vị trí (x,y).
- Đặt g(x,y) = αf(x,y) + β.
- α và β còn được gọi là tham số gain và bias, hoặc tham số để điều chỉnh contrast (độ tương phản) và brightness (độ sáng)
  - Nếu α > 1, thì ta nói ảnh g(x,y) có độ tương phản gấp α lần so với ảnh f(x,y).
  - Nếu β > 0 ta nói độ sáng của ảnh g(x,y) đã thay đổi một lượng là β. Dựa vào công thức trên ta có chương trình thay đổi độ sáng và tương phản của ảnh như sau:

```
alpha= 1  beta= 0      --> no change
0 < alpha < 1        --> lower contrast
alpha > 1            --> higher contrast
-127 < beta < +127   --> good range for brightness values
```

Ví dụ độ sáng và độ tương phản

Ảnh ban đầu

Ảnh thay đổi
alpha=1.8, beta=10

Hàm thay đổi độ sáng và độ tương phản

```
import cv2

image = cv2.imread('1.jpg')

alpha = 1.5 # Contrast control (1.0-3.0)
beta = 0 # Brightness control (0-100)

adjusted = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)

cv2.imshow('original', image)
cv2.imshow('adjusted', adjusted)
cv2.waitKey()
```

- Bài tập:
- Viết chương trình tạo trackbar để thay đổi độ sáng và độ tương phản của ảnh
- Hiện ảnh trên matplotlib
- Viết chương trình biến đổi ảnh thành ảnh nhị phân, Tạo trackbar để thay đổi giá trị ngưỡng

3. Biến đổi ảnh âm bản (Image negatives)

g(x,y) = L - f(x,y)

Trong đó L là mức sáng cao nhất của ảnh

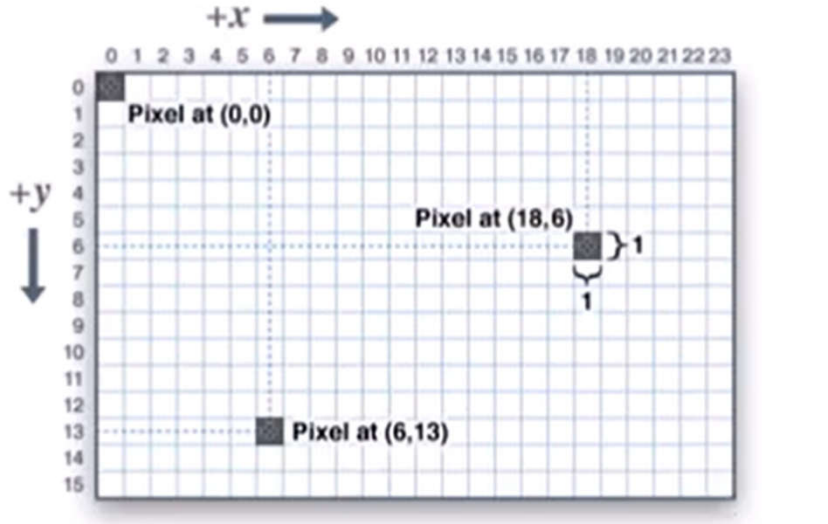
```
1. import cv2
2. import numpy as np

3. image = cv2.imread('lena.jpg',0)

4. cv2.imshow('Original Image', image)
5. neg_img = 255 - image
6. cv2.imshow('Negative Image', neg_img)
7. cv2.waitKey(0)
8. cv2.destroyAllWindows()
```

Các phép biến đổi hình học của ảnh

- Phóng đại ảnh
- Cắt ảnh
- Phép tịnh tiến
- Phép quay
- Phép biến đổi affine

Một số lưu ý trong xử lý ảnh với opencv

- **Lưu ý:**
  - trong opencv, kích thước ảnh được lưu trữ theo thứ tự là height,width,depth → Để lấy chiều cao và rộng của ảnh: height, width = img.shape[:2]
  - Các thuật toán xử lý ảnh trong opencv xử lý theo thứ tự width*height
  - Hệ trục tọa độ ảnh:



Các biến đổi hình học

- Các phép biến đổi hình học là tập hợp các phép biến đổi hình ảnh từ một hình dạng này sang một hình dạng khác thông qua việc làm thay đổi phương, chiều, góc, cạnh mà không làm thay đổi nội dung chính của bức ảnh.
- Mỗi một phép biến đổi hình học sẽ được xác định bởi một ma trận dịch chuyển M (translation matrix) . Khi đó bất kì 1 điểm có tọa độ (x,y) trên ảnh gốc thông qua phép biến đổi T sẽ có tọa độ trong không gian mới sau dịch chuyển là T(x,y) theo công thức:

$$T(x,y) = \mathbf{M}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{bmatrix}$$

- Một hàm số T: $R^n \rightarrow R^n$ được coi là một phép biến đổi tuyến tính nếu nó thỏa mãn 2 tính chất sau:
- Tính chất cộng tính: $T(\vec{u}+\vec{v}) = T(\vec{u}) + T(\vec{v})$
- Tính chất nhân tính: $T(\lambda\vec{x}) = \lambda T(\vec{x})$
- Ta nhận thấy tính chất cộng hoàn toàn có thể được suy ra trực tiếp từ phép nhân ma trận M(A+B) = M(A)+M(B). Trong đó M là ma trận biến đổi và A,B là các tọa độ điểm

➢ Vậy để xác định một phép biến đổi hình học ta sẽ cần phải xác định được ma trận dịch chuyển của nó là gì?

Phóng đại ảnh (Scale ảnh)

- Scale ảnh là việc chúng ta thay đổi kích thước dài, rộng của ảnh mà không làm thay đổi tính chất song song của các đoạn thẳng trên ảnh gốc so với các trục tọa độ X và Y.
- Theo định nghĩa về phép biến đổi hình học thì một biến đổi phóng đại các chiều (x, y) theo hệ số (a_1,a_2) sẽ có ma trận dịch chuyển M là ma trận đường chéo. Tức là ma trận vuông có đường chéo chính là [a_1,a_2] và các phần tử còn lại bằng 0. Khi đó phép dịch chuyển sẽ là:

$$T(x,y) = \mathbf{M}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_1 x \\ a_2 y \end{bmatrix}$$

Phóng đại ảnh (Scale ảnh)

- Thay đổi tỉ lệ hình ảnh có ích trong nhiều ứng dụng xử lý hình ảnh cũng như học máy.
  - Phóng to hoặc thu nhỏ hình ảnh để đáp ứng các yêu cầu về kích thước.
  - Giảm số lượng pixel từ một hình ảnh → có thể làm giảm thời gian đào tạo mạng nơ-ron vì số lượng pixel trong hình ảnh nhiều hơn làm tang số lượng nút đầu vào, do đó làm tăng độ phức tạp của mô hình.
- OpenCV cung cấp cho chúng ta một số phương pháp nội suy để thay đổi kích thước hình ảnh.

Phóng đại ảnh (Scale ảnh)

- Kích thước của hình ảnh có thể thay đổi theo kích thước đưa ra hoặc theo hệ số tỷ lệ.
- Lệnh chỉnh kích thước: cv2.resize()

cv2.resize(src,dsize,dst,fx,fy,interpolation)

  - src: ảnh đầu vào
  - dsize: kích thước mới (width, height).
  - fx, fy: dùng để biến đổi theo tỉ lệ ảnh thay vì kích thước tuyệt đối.
  - interpolation: phép nội suy
    - cv2.INTER_AREA thu nhỏ,
    - cv2.INTER_CUBIC (chậm nhưng hiệu quả hơn)
    - cv2.INTER_LINEAR (default) phóng to.

Phóng đại ảnh

- Ví dụ: **cv2.resize(src,dsize,dst,fx,fy,interpolation)**

```
1. img_down = cv2.resize(image, None,fx=0.6,fy=0.6,interpolation= cv2.INTER_LINEAR)
2. img_up = cv2.resize(image, None,fx= 1.2,fy= 1.2, interpolation=cv2.INTER_LINEAR)
3. res = cv2.resize(img, (int(width / 2), int(height / 2)), interpolation =
                                                        cv2.INTER_CUBIC)
```

- **Thay đổi kích thước ảnh theo kích thước mới:**

```
img_res = cv2.resize(img,(400,300))
```

Cắt ảnh - Crop

- Cắt ảnh nhằm loại bỏ tất cả các đối tượng hoặc vùng ảnh không mong muốn khỏi hình ảnh, hoặc để làm nổi bật một đặc trưng cụ thể của hình ảnh.
- Không có hàm cụ thể nào để cắt ảnh bằng OpenCV → thực hiện cắt mảng.
- Mọi hình ảnh được đọc vào sẽ được lưu trữ trong một mảng 2D (cho mỗi kênh màu). → Chỉ cần đưa ra chiều cao và chiều rộng của khu vực sẽ được cắt.
- `cropped_img = img[start_row:end_row, start_col:end_col]`
- Ví dụ: `new_image = img[80:280, 150:330]`

Cắt ảnh - Crop

```
import cv2
import numpy as np
img = cv2.imread('test.jpg')
print(img.shape)
cv2.imshow("original", img)
cropped_image = img[80:280, 150:330]
cv2.imshow("cropped", cropped_image)
cv2.imwrite("Cropped Image.jpg", cropped_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Dịch chuyển ảnh (Translation)

- Dịch chuyển ảnh đến các vị trí khác nhau.
- Ví dụ dịch tới các góc trái, phải, ở giữa, bên trên, bên dưới.
- Phép dịch chuyển sẽ giữ nguyên tính chất song song của các đoạn thẳng sau dịch chuyển đối với các trục X hoặc Y nếu trước dịch chuyển chúng cũng song song với một trong hai trục này.

Dịch chuyển ảnh (Translation)

- Dịch chuyển hình ảnh theo trục x và trục y một khoảng t_x,t_y.
- Ma trận dịch chuyển có dạng:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

- Khi đó ta có:

$$T(x,y) = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$$

- Như vậy mỗi điểm tọa độ (x,y) đã được dịch chuyển tới một tọa độ mới là ($x+t_x$, $y+t_y$)
- Tx > 0 : dịch sang phải, Tx <0 : dịch sang trái
- Ty > 0 : dịch xuống dưới, Ty <0 : dịch lên trên

Dịch chuyển ảnh

- Trong opencv, hàm dịch chuyển: cv2.warpAffine()
- Đầu vào là ma trận dịch chuyển M và bức ảnh gốc ta thu được kết quả là ảnh sau dịch chuyển
- Ví dụ:

```
# Dịch chuyển hình ảnh xuống góc dưới bên phải
tx, ty = (200, 200)
M1 = np.array([[1, 0, tx], [0, 1, ty]], dtype=np.float32)
tran1 = cv2.warpAffine(img, M1, (cols, rows))

# Dịch chuyển hình ảnh xuống góc dưới bên trái
M2 = np.array([[1, 0, -tx], [0, 1, ty]], dtype=np.float32)
tran2 = cv2.warpAffine(img, M2, (cols, rows))
```

Xoay ảnh (Rotation)

- Xoay ảnh theo một góc xác định quanh một điểm nào đó.
- Phép xoay sẽ không đảm bảo tính chất song song với các trục X hoặc Y như phép dịch chuyển nhưng nó sẽ bảo toàn độ lớn góc.
- Nếu 3 điểm bất kì tại ảnh gốc tạo thành một tam giác thì khi biến đổi qua phép xoay ảnh, chúng sẽ tạo thành một tam giác đồng dạng với tam giác ban đầu.
- Phép xoay của một hình ảnh tương ứng với một góc đạt được bằng một ma trận dịch chuyển M như sau:

$$\mathbf{M} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

Bài tập

- Tìm hiểu về tích chập trong xử lý ảnh (Convolution)
- Các phương pháp lọc ảnh (Image Filtering)
  - Thuật toán
  - Tác dụng: tốt đối với ảnh nào
  - Sử dụng hàm trong OpenCV: hiểu rõ từng tham số
- Tuần sau trình bày

- Tìm hiểu để thực hành: Viết chữ, vẽ các hình lên ảnh (đường thẳng, đường tròn, chữ nhật, elip, đa giác,...)

Xoay ảnh (Rotation)

- Ngoài ra OpenCV hỗ trợ một phép xoay phóng đại (scaled rotation): vừa biến đổi ảnh theo phép xoay theo tâm xác định và điều chỉnh lại kích thước ảnh sau xoay.
- Ma trận dịch chuyển được đưa ra như sau:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)c_x - \beta c_y \\ -\beta & \alpha & \beta c_x + (1-\alpha)c_y \end{bmatrix}$$

$$\alpha = scale.\, cos(\theta) \beta = scale.\, sin(\theta)$$

- (cx,cy) là tọa độ tâm của phép xoay và scale là độ phóng đại.

Xoay ảnh (Rotation)

- Để tìm ra ma trận transform có thể sử dụng hàm:

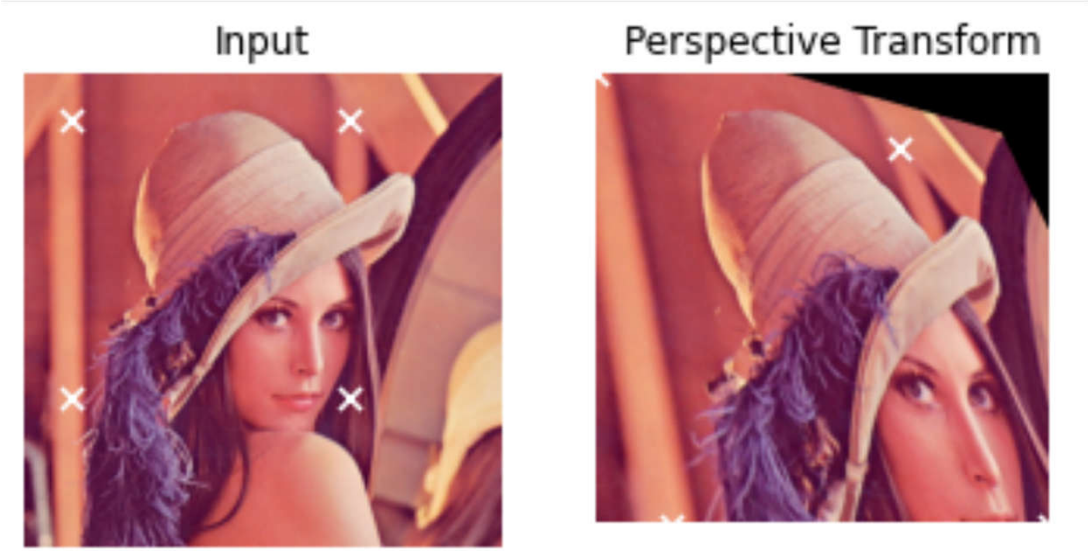
cv2.getRotationMatrix2D(center, angle, scale)

- Tham số vào: center: tâm của phép xoay, angle: góc xoay, scale tỉ lệ phóng đại kích thước ảnh.
- Sau đó dùng hàm xoay ảnh: **cv2.warpAffine(src, M, dsize)**
- **Ví dụ:**

```
# Xoay ảnh kích thước 45 độ tại tâm của ảnh, độ phóng đại ảnh không đổi.
  M = cv2.getRotationMatrix2D(center = (height/2,width/2), angle=-45, scale=1)
  img1 = cv2.warpAffine(img, M, (height,width))
```

Biến đổi phối cảnh (Perspective Transform)

- Để biến đổi phối cảnh ta cần một ma trận biến đổi 3x3. Đường thẳng sẽ giữ nguyên là đường thẳng sau biến đổi.
- Để tìm ra ma trận biến đổi cần tìm ra 4 điểm trong ảnh đầu vào tương ứng với các điểm trong ảnh đầu ra. Trong số 4 điểm này, không có bất kì 3 điểm nào thẳng hàng.
- Ma trận biến đổi có thể được thiết lập thông qua hàm số cv2.getPerspectiveTransform.
- Áp dụng cv2.warpPerspective với ma trận biến đổi 3x3.



Biến đổi phối cảnh

- Ví dụ:

```
pts1 = np.float32([[50,50],[350,50],[50,350],[350,350]])
pts2 = np.float32([[0,0],[200,50],[50,300],[300,300]])

M = cv2.getPerspectiveTransform(pts1,pts2)

dst = cv2.warpPerspective(img,M,(300,300))
```

Tóm tắt

- Phóng ảnh: scale: cv2.resize
- Crop: img[hang:cot,hang cuoi:cot cuoi]
- Dịch ảnh:
  - Ma trận dịch chuyển: tx,ty
  - cv2.warpAffine
- Xoay ảnh:
  - Ma trận dịch chuyển:
  - Cv2.warpAffine
- Phối cảnh:
  - Ma trận dịch chuyển
  - cv2.warpPerspective

Thank you!

TRƯỜNG ĐẠI HỌC VINH
VINH UNIVERSITY

Nơi tạo dựng tương lai cho tuổi trẻ