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

Phép nhân chập không gian với ảnh số

- Tích chập (convolution) là một kỹ thuật quan trọng xử lý ảnh. Nó có mặt trong hầu hết các thuật toán làm mờ (Gausian Blur), hay làm rõ các đường (edge detector). Trong nhận dạng ảnh (deep learning image processing), tích chập là một tầng biến đổi ma trận đầu vào để làm rõ và tách ra các đặc tính của hình ảnh mà vẫn bảo toàn tính tương quan không gian giữa đầu ra và đầu vào
- (CNN:Convolution Neural Network)

2

Phép nhân chập không gian với ảnh số

- Cho X(m, n) là ảnh đầu vào với m ∈ [0,M-1], n ∈ [0,N-1]
- H(k, l) là mặt nạ của phép nhân chập k∈[0, K-1], l∈[0,L-1] (thường K = L << M, N).
- Phép nhân chập được định nghĩa như sau:

$$Y(m,n) = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} H(k,l) * X(m-k, n-l)$$

3

Phép nhân chập tại tâm với mặt nạ lẻ

- Có thể thực hiện bằng cách sau:
  - Quay mặt nạ H một góc 180^0 ta được H*. (đảo cột trái và phải)
  - Để tính Y[m,n] ta đặt tâm của H* trùng với điểm X[m,n], lúc này Y[m,n] bằng tổng các tích tương ứng của các phần tử H* và X.
- Ví dụ: Giả sử có ảnh X[M, N] và mặt nạ H[K,L]

$$X[M,N] = \begin{bmatrix} 5 & 7 & 9 & 4 & 3 \\ 0 & 1 & 0 & 5 & 7 \\ 1 & 4 & 7 & 9 & 4 \\ 8 & 2 & 1 & 3 & 6 \\ 5 & 0 & 1 & 2 & 3 \end{bmatrix} \qquad H[K,L] = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

4

Phép nhân chập tại tâm với mặt nạ lẻ

$$X[M,N]=\begin{bmatrix}5&7&9&4&3\\0&1&0&5&7\\1&4&7&9&4\\8&2&1&3&6\\5&0&1&2&3\end{bmatrix}$$

$$H[K,L]=\begin{bmatrix}1&1&-1\\1&2&-1\\1&1&-1\end{bmatrix}$$

- Quay H góc 180^0 ta có:

$$H^*[K,L]=\begin{bmatrix}-1&1&1\\-1&2&1\\-1&1&1\end{bmatrix}$$

- Sau khi giữ lại biên ảnh và tính các phần tử không phải biên ta có:

$$Y[M,N]=X[M,N]*H[K,L]=\begin{bmatrix}5&7&9&4&3\\0&23&22&21&7\\1&10&25&35&4\\8&3&18&21&6\\5&0&1&2&3\end{bmatrix}$$

5

$$X[M,N]=\begin{bmatrix}5&7&9&4&3\\0&1&0&5&7\\1&4&7&9&4\\8&2&1&3&6\\5&0&1&2&3\end{bmatrix}$$

$$H^*[K,L]=\begin{bmatrix}-1&1&1\\-1&2&1\\-1&1&1\end{bmatrix}$$

6

$$H[K,L]=\begin{bmatrix}1 & 1 & -1\\ 1 & 2 & -1\\ 1 & 1 & -1\end{bmatrix}$$

$$X[M,N]=\begin{bmatrix}5 & 7 & 9 & 4 & 3\\ 0 & 1 & 0 & 5 & 7\\ 1 & 4 & 7 & 9 & 4\\ 8 & 2 & 1 & 3 & 6\\ 5 & 0 & 1 & 2 & 3\end{bmatrix} \quad H^{*}[K,L]=\begin{bmatrix}-1 & 1 & 1\\ -1 & 2 & 1\\ -1 & 1 & 1\end{bmatrix}$$

7

Phép nhân chập không gian với ảnh số

- Công thức nhân chập:

$$Y(m,n)=\sum_{k=0}^{K-1}\sum_{l=0}^{L-1}H(k,l)*X(m-k,n-l)$$

- Rõ ràng, một điểm ảnh ở đầu ra phụ thuộc vào H(*k, l*), điểm ảnh X(*m, n*) và các lân cận của nó. Mặt khác, theo công thức trên thì nhân chập H ⊗ X có độ phức tạp tính toán rất cao, nên người ta thường dùng nhân chập H_{KxL} có kích thước hữu hạn và lẻ như 3 x 3, 5 x 5, 7 x 7.

8

Phép nhân chập không gian với ảnh số

- Nếu K = L : lẻ thì phép nhân chập được thực hiện:

$$Y(m,n) = \sum_{k=0}^{2Lc}\sum_{l=0}^{2Lc} H(k,l) * X(m-k+Lc, n-l+Lc) \quad \text{Voi Lc} = \frac{L-1}{2}$$

- Xử lý trường hợp các điểm biên:
  - Mở rộng ảnh và cho các điểm ngoài biên có giá trị bằng 0 ⇒ kích thước ảnh thay đổi ⇒ ít dùng
  - Chọn phương án giữ nguyên biên và tính kết quả cho các điểm không thuộc biên (do các điểm phía ngoài thường không mang nhiều ý nghĩa)

9

Các toán tử không gian

- Ảnh thu nhận có nhiễu → cần phải loại bỏ nhiễu.
- Ảnh không sắc nét, bị mờ, hoặc cần làm rõ các chi tiết như biên → cần làm sắc nét, nổi biên.
- Các toán tử không gian dùng trong tăng cường ảnh được phân nhóm theo công dụng: làm trơn nhiễu, nổi biên...

➔ Sử dụng các bộ lọc tuyến tính hay lọc phi tuyến

10

Mô hình biểu diễn ảnh bị nhiễu

- Giả sử X(m,n) là ảnh số thu nhận được
- S(m,n) là thông tin có ích của ảnh (ảnh không bị nhiễu)
- N(m,n) là nhiễu ngẫu nhiên tác động lên ảnh.
- Ta có thể biểu diễn như sau:

$$X(m,n) = S(m,n) + N(m,n)$$

  - Nếu tồn tại $\exists m,n$ sao cho $N[m,n] \ll S$ thì gọi là nhiễu cộng
  - Nếu tồn tại $\exists m,n$ sao cho $N[m,n] \gg S$ thì gọi là nhiễu xung

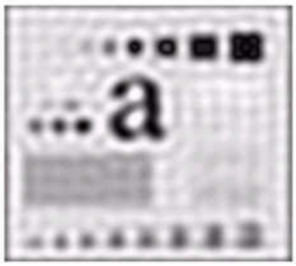
➢ Khử nhiễu cộng: Bộ lọc thông thấp

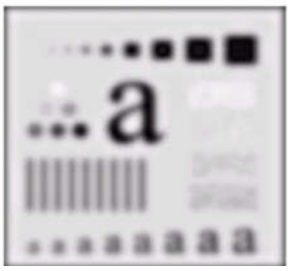
➢ Khử nhiễu xung: Bộ lọc thông cao

11

Bộ lọc thông thấp

- Dùng bộ lọc thông thấp để khử nhiễu cộng.
- Lọc thông thấp chỉ cho những thành phần tần số thấp đi qua và loại bỏ những thành phần tần số cao.
- Ảnh sau khi lọc sẽ trơn mịn nhưng không được sắc nét bằng ảnh đầu vào.





12

Bộ lọc thông thấp

- Ý tưởng chính: cộng tất cả các điểm ảnh trong một mặt nạ lọc, sau đó chia cho tổng số điểm ảnh trong mặt nạ đó
- Đặc điểm của bộ lọc thông thấp: tổng các phần tử của kenel bằng 1.
- **Bộ lọc tuyến tính:**

❑ Bộ lọc trung bình hay Bộ lọc hộp (Box Filter):

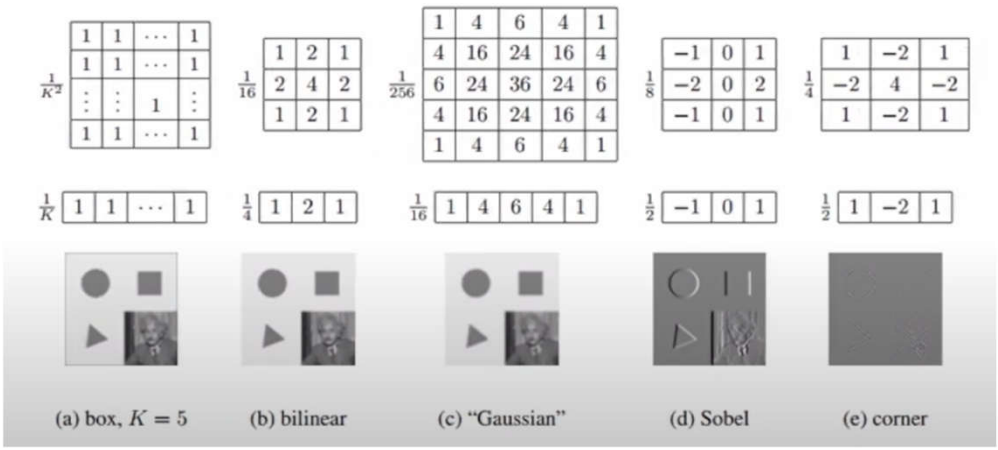
$$H = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

❑ Bộ lọc Gaussian: $H = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$

13

Bộ lọc tuyến tính

- Một số kenel



14

Hàm lọc làm mờ ảnh trong OpenCV

- Hàm tích chập 2D: Tích chập 1 kenel với 1 ảnh

  `cv2.filter2D(src, ddepth, kernel)`

- src: hình ảnh gốc
- ddepth: độ sâu của hình ảnh thu được. Giá trị -1 nếu hình ảnh kết quả sẽ có cùng độ sâu với ảnh gốc
- kernel: mặt nạ áp dụng cho việc lọc
  - Ví dụ Kenel:

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

15

Ví dụ một số kernel

- Một số Kernel:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Identity kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen kernel

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

Box blur

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Gaussian blurr kernel

16

cv2.filter2D()

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('opencv_logo.png')
kernel = np.ones((5,5),np.float32)/25
dst = cv.filter2D(img,-1,kernel)
plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(dst),plt.title('Averaging')
plt.xticks([]), plt.yticks([])
plt.show()
```

17

Một số hàm làm mờ ảnh trong openCV

- cv2.blur (src, ksize)
- cv2.boxFilter
- cv2. GaussianBlur

18

Ví dụ hàm blur

```
cv2.blur(src, ksize)
```

Ví dụ:

```
import cv2 as cv
img = cv.imread('test.jpg')
img_blur = cv2.blur(image, (5,5))
cv2.imshow('Original', image)
cv2.imshow('Blurred', img_blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

19

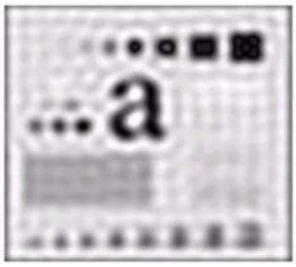
Bộ lọc Bilateral (bộ lọc song phương)

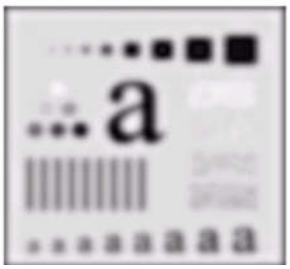
- Bộ lọc hiệu quả cao trong việc loạt bỏ nhiễu mà vẫn giữ lại được các đường viền (cạnh) trong ảnh
- `cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace)`
  - src: ảnh gốc
  - d: đường kính của vùng lân cận pixel được sử dụng để lọc
  - sigmaColor và sigmaSpace xác định độ lệch chuẩn của phân bố cường độ màu (1D) và phân bố không gian (2D) tương ứng
    - sigmaColor xác định phân phối Gaussian một chiều, chỉ định mức độ mà sự khác biệt về cường độ pixel có thể được chấp nhận
    - sigmaSpace xác định phạm vi không gian của mặt nạ, theo cả hướng x và y (giống như bộ lọc mờ Gaussian)
- Ví dụ:
- `bilateral_img = cv2.bilateralFilter(src=image, d=9, sigmaColor=75, sigmaSpace=75)`
- Viết gọn: `cv2.bilateralFilter(img,9,75,75)`

20

Bộ lọc thông cao

- Lọc thông cao chỉ cho những thành phần tần số cao đi qua và loại bỏ những thành phần tấn số thấp.
- Ảnh sau khi lọc sẽ có dải mức xám không đa dạng tại những vùng trơn mịn, nhưng sắc nét hơn so với ảnh đầu vào.





21

Bộ lọc thông cao

- Bộ lọc phi tuyến khử nhiễu xung
- Bộ lọc trung vị (Median Filter)
  - Bộ lọc làm trơn (smooth filtering), làm mờ ảnh, loại bỏ các đối tượng nhỏ trong ảnh (nhiễu muối tiêu)

22

Lọc trung vị

- Phương pháp thực hiện:
  - Đặt một cửa sổ hình vuông có kích thước lẻ (3x3, 5x5, ...) lên ảnh ban đầu sao cho tâm của cửa sổ này trùng với (m,n)
  - Sắp xếp các phần tử trong cửa sổ theo thứ tự tăng dần (hoặc giảm dần), phần tử trung vị là giá trị nằm chính giữa dãy đã sắp xếp
  - Giá trị mới tại điểm (m,n) chính là phần tử trung vị trong cửa sổ trên

23

Ví dụ: lọc trung vị

- Ta có ảnh đầu vào

$$X[M,N]=\begin{bmatrix}1 & 5 & 7 & 9 & 10\\ 6 & 20 & 7 & 8 & 5\\ 4 & 3 & 5 & 8 & 7\\ 9 & 4 & 6 & 50 & 3\\ 5 & 7 & 4 & 6 & 5\end{bmatrix}$$

- Dùng cửa sổ kích thước 3 x 3
- Giả sử tính giá trị Y(1,1)
- Các phần tử trong cửa sổ: 1, 5, 7, 6, 20,7,4,3,5.
- Sắp xếp dãy theo thứ tự tăng dần ta có: 1, 3, 4, 5, **5**, 6, 7, 7, 20.
- Phần tử trung vị là 5. Vậy Y(1,1) = 5
- Sau khi áp dụng cửa sổ trên ảnh ta thu được kết quả:

$$Y[M,N]=\begin{bmatrix}1 & 5 & 7 & 9 & 10\\ 6 & 5 & 7 & 7 & 5\\ 4 & 6 & 7 & 7 & 7\\ 9 & 5 & 6 & 6 & 3\\ 5 & 7 & 4 & 6 & 5\end{bmatrix}$$

24

- Bài tập:
- 1. Phép chiếu phối cảnh (perspective transform): tự lấy điểm bằng chuột
- 2. Làm slide báo cáo phần: tách biên ảnh, tìm contour ảnh, có code minh họa
- Nộp cả code và slide vào elearning (hạn nộp 1/3)

25

Hàm bộ lọc trung vị trong opencv

- medianBlur(src, ksize)
  - src: hình ảnh gốc
  - ksize: mặt nạ, phải là số nguyên dương và lẻ
- Ví dụ:
- median_img = cv2.medianBlur(image, 5)

- Đọc thêm: Smoothing filter + opencv

26

Thank you!

TRƯỜNG ĐẠI HỌC VINH
VINH UNIVERSITY

Nơi tạo dựng tương lai cho tuổi trẻ

27