A gentle introduction to Spatial Filtering.

Report Lab week 3-4.

Ngọc Thuận - IPSAL LAB

18, December 2022

Tóm tắt nội dung

Bài trước chúng ta đã đề cập đến các biến đổi trên miền không gian, và đã tiếp cập với một số phép biến đổi cơ bản trực tiếp lên điểm ảnh (Intensity transformation). Bài này chúng ta sẽ tiếp tục về các phép biến đổi trên miền không gian, cụ thể là phần còn lại (Spatial Filtering). Dưới đây là đôi nét cơ bản về lọc không gian và một số phép lọc cơ bản, đồng thời cũng là phần báo cáo Lab tuần của tôi!

Mục lục

1 Giới thiệu về lọc không gian

1.1 Chung chung

Ta có thể hiểu một bộ lọc không gian gồm hai thành phần:

1. Lân cận (*neighborhood*)
2. Toán tử (*predefined operation*)

Bộ lọc sẽ tạo ra một pixel mới với tọa độ trùng với tọa độ tâm của *neighborhood* và giá trị là kết quả của toán tử.

Phân loại dựa trên *predefined operation* ta có thể phân loại thành:

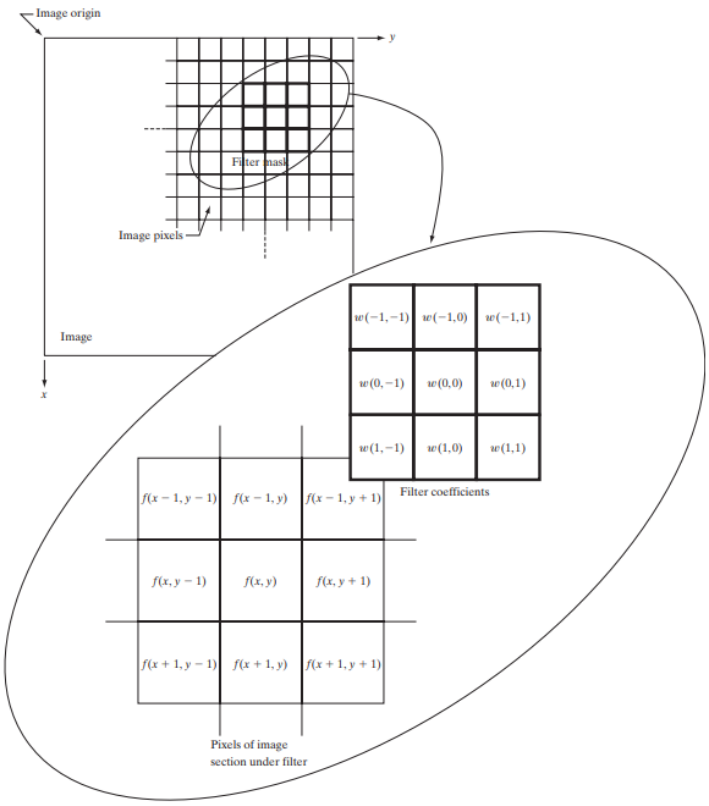
- Bộ lọc tuyến tính (*Linear spatial filter*)
- Bộ lọc phi tuyến tính (*NonLinear spatial filter*)

Lưu ý: Trong giới hạn bài này ta chỉ làm việc với bộ lọc tuyến tính!

Thông thường khi nhắc tới *Linear spatial filter* của một bức ảnh kích thước $M \times N$ với bộ lọc có kích thước $m \times n$ ta sẽ hiểu là:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{1}$$

x, y thay đổi khi w đi qua từng *pixel* trong *f*.
Hình 1 cho thấy cách làm việc của bộ lọc tuyến tính!



Hình 1: The mechanics of linear spatial filtering using a filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering

1.2 Spatial Correlation and Convolution

Đây là hai phép toán quan trọng, nhưng dễ gây nhầm lẫn.

1.2.1 Tương quan - Correlation

Correlation của bộ lọc *w(x,y)* có kích thước *m x n* với ảnh *f(x,y)* được kí hiệu là:

$$w(x,y) \otimes f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

1.2.2 Tích chập - Convolution

Khá giống định nghĩa tích chập trong giải tích,

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau$$

Trong trường hợp rời rạc,

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n-m]$$

Convolution của bộ lọc *w(x,y)* có kích thước *m x n* với ảnh *f(x,y)* được kí hiệu là:

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Hai phép toán trên thật ra cũng không khác nhau nhiều lắm, để có được một trong hai, rõ ràng ta chỉ cần đảo bộ lọc của thằng còn lại. Nên thuật ngữ 'Tích chập' được dùng chung chung cho cả hai, và điều này đôi khi gây nhầm lẫn.

Lưu ý: Trong phạm vi bài này ta sẽ dùng (1). Và tôi sẽ dùng thuật ngữ 'tích chập' chung cho tất cả các bộ lọc tuyến tính

1.3 Tích chập hai chiều - Two dimensional Convolution (2D Convolution)

Trước khi đi vào xử lí ảnh, ta sẽ lướt sơ qua về kiểu dáng và cách làm việc với tích chập.

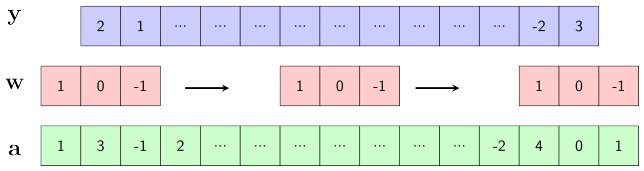
1.3.1 Tích chập một chiều

Giả sử ta có đầu vào và bộ lọc lần lượt là $\mathbf{a} \in \mathbb{R}^N$ và $\mathbf{w} \in \mathbb{R}^f$ (f là số tự nhiên bất kì, thường là số lẻ). Khi đó đầu ra là một vector $\mathbf{y}$ với từng phần tử được tính bởi

$$y_n = \sum_{i=0}^{f-1} a_{n+i} w_i$$

Quá trình đó có thể được thực hiện như sau:

1. Đặt bộ lọc $\mathbf{w}$ vào vị trí tương ứng với f phần tử đầu tiên của $\mathbf{a}$.
2. Nhân từng phần tử tương ứng của $\mathbf{w}$ với $\mathbf{a}$ rồi cộng các kết quả lại để được phần tử tương ứng của $\mathbf{y}$.
3. *Trượt* bộ lọc $\mathbf{w}$ một bước sang bên phải. Nếu phần tử cuối cùng của bộ lọc không vượt ra ngoài phần tử cuối cùng của tín hiệu, quay lại Bước 2. Ngược lại, dừng tính toán.



Hình 2: Cách tính tích chập một chiều, machinelearningcoban.com

Trong ví dụ Hình 2

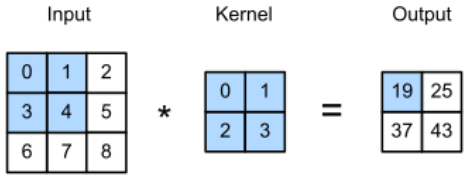
$$y_0 = a_0 w_0 + a_1 w_1 + a_2 w_2 = 1 - (-1) = 2$$

$$y_1 = a_1 w_0 + a_2 w_1 + a_3 w_2 = 3 - 2 = 2$$

Ngoài ra, ta còn hai bước quan trọng khác, có thể cần chú ý là *'Bước trượt - Stride'* và *'Thêm lề - Padding'*, các bạn có thể tham khảo thêm tại:
Bài 37: Tích chập hai chiều - machinelearningcoban.com

1.3.2 Tích chập hai chiều

Là cái ta đã nói ở trên, về cơ bản là tương tự như tích chập một chiều, bộ lọc trượt khắp tín hiệu hai chiều đầu vào theo thứ tự từ trái qua phải, từ trên xuống dưới. Tại mỗi vị trí, các giá trị của bộ lọc và đầu vào tương ứng được nhân với nhau rồi cộng vào lại để thu được kết quả đầu ra.
Giới hạn ở bài này, tôi chỉ dừng lại ở ví dụ về tích chập hai chiều đơn kênh, SISO. Nếu muốn tham khảo thêm các bạn có thể truy cập link bên trên.
Hình 3 minh họa cho cách tính tích chập hai chiều.



Hình 3: Minh họa tích chập hai chiều, diveintodeeplearning

1.4 Thiết kế bộ lọc

Đây không phải là vấn đề gì lớn, nhưng tôi muốn nhắc lại một tí kiến thức về giải tích và phần trước.

Chúng ta biết rằng, trong xử lí ảnh, tín hiệu đầu vào của ta là ảnh số, trong đó giá trị của ảnh được thể hiện qua *pixel*, và tùy thuộc vào lượng *bit* được sử dụng mà giá trị của một điểm ảnh có thể thuộc vào những khoảng khác nhau. Và việc sử dụng một bộ lọc không gian có thể dẫn đến thay đổi những giá trị điểm ảnh, điều đó có thể sẽ khiến kết quả không được như mong muốn.

Để giải quyết việc đó, ta nhắc lại một kết quả quen thuộc trong môn giải tích:

Giả sử *f(x)* liên tục trên $[a, b]$, có $\alpha + \beta = 1$, α, β không âm , khi đó: Tồn tại một số $c \in [a, b]$ thỏa mãn:

$$f(c) = \alpha f(a) + \beta f(b)$$

Chứng minh
Đặt $\alpha f(a) + \beta f(b) = \mu$.
Do $0 \leq \alpha, \beta \geq 1$, suy ra $\mu \leq max_{[a,b]} f(x)$ và $\mu \geq min_{[a,b]} f(x)$ (*)
Mà, $f(x)$ lại liên tục suy ra $c \in [c, d]$ với c, d là argmin, argmax $f(x)$, hiển nhiên $[c, d] \subset [a, b]$.
Từ đó ta có điều phải chứng minh!
□ **Q.E.D.**

Trên là một kết quả hay và quan trọng. Nhưng ở đây (*) là cái chúng ta cần!
Kết luận: Mỗi bộ lọc, để đảm bảo đầu ra, cần chuẩn hóa sao cho tổng các phần tử của bộ lọc là bằng 1.
Chúng ta sẽ thấy rõ điều này ở mục 2.1 Gaussian Filter - bộ lọc Gauss

Hình 4 ví dụ cho một số bộ lọc hai chiều!

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Hình 4: Hai bộ lọc làm mịn 3 x 3

2 Smoothing Spatial Filters

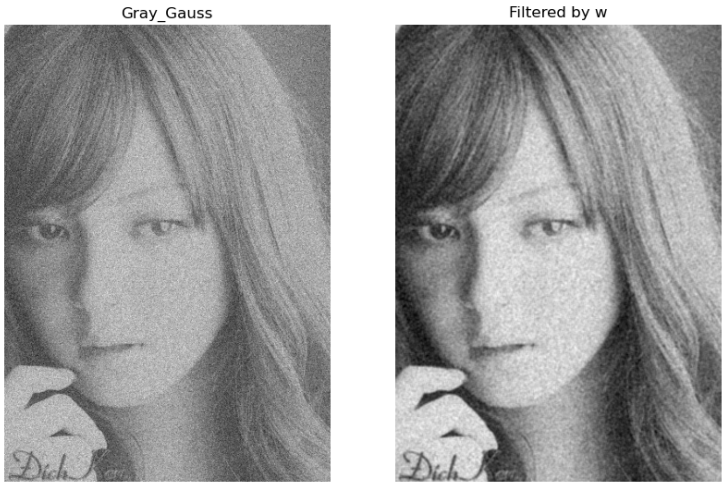
Dịch ra tiếng việt là bộ lọc không gian làm mịn, nghe cứ kì kì.

Đầu ra của bộ lọc không gian làm mịn tuyến tính chỉ đơn giản là trung bình (có trọng số) của các pixel trong vùng lân cận được xác định bởi bộ lọc. Dễ thấy rằng việc này sẽ làm mờ đi cạnh và làm mất đi nhiều chi tiết nhỏ của ảnh. Trọng số ở đây, có thể hiểu là càng gần *center* giá trị càng lớn.

Có một vài bộ lọc khá phổ biến, ví dụ như Hình 4, cho kết quả khá ổn.
Hình 5 là minh họa cho bộ lọc bên phải Hình 4. Tuy nhiên với kích thước bộ

Hình 5: Ảnh xám với bộ lọc bên phải Hình 4

lọc khá nhỏ ta không thấy rõ, sự khác biệt, Hình 6 cho ta một cái nhìn rõ ràng hơn, cũng như cho ta thấy một công dụng khác của bộ lọc không gian là 'khử



Hình 6: Ảnh xám + nhiễu, sử dụng bộ lọc bên trái Hình 4

nhiễu'.

2.1 Gaussian Filter

Phân phối chuẩn nhiều chiều - Phân phối chuẩn hai chiều

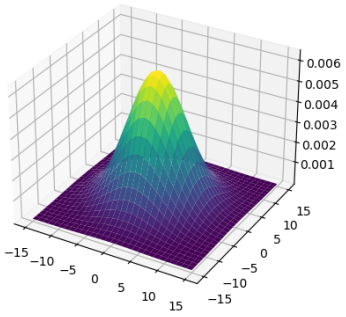
Phân phối chuẩn nhiều chiều (*Multivariate Normal Distribution*), giả sử là D chiều, *Vector kì vọng* μ và *Covariance matrix* Σ

$$p(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right) \tag{2}$$

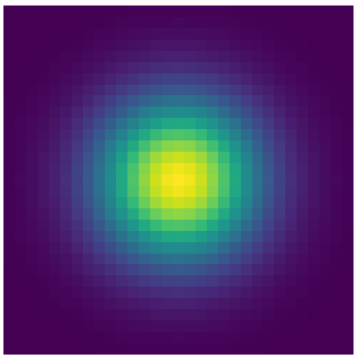
Tuy nhiên, trong ảnh đơn kênh, ta chỉ cần dùng đến 2 chiều, hơn nữa, bộ lọc ta coi trọng số, khoảng cách từ lân cận đến tâm bằng nhau nếu có cùng khoảng cách. Do đó các hệ số tương quan đều bằng 0, và hai chiều có cùng phương sai σ^2. Khi đó phương trình (2) có thể viết lại như sau:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{3}$$

Hình 7 minh họa cho phân phối chuẩn hai chiều trong không gian.



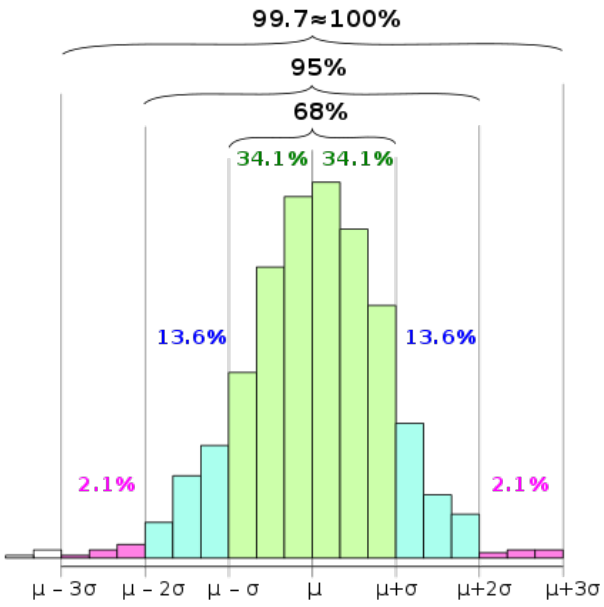
(a) Đồ thị của phân phối chuẩn hai chiều, $\sigma = 5$



(b) Kernel, 31 x 31

Hình 7: Phân phối chuẩn hai chiều

Quy tắc 3 sigma - *three sigma rule*



Hình 8: Quy tắc 3 sigma, wikipedia

Hình 8 minh họa cho quy tắc 3 sigma, bạn có thể thấy, $P(|x - \mu| < 3\sigma) = 99.7\%$ xấp xỉ 1. Theo nhận xét ở mục 1.4 ta rút ra được cách chọn kích thước bộ lọc: $size = (6n + 1, 6n + 1)$
Hình 9 cho thấy kết quả của bộ lọc Gauss.

Hình 9: Làm mờ sử dụng bộ lọc Gauss với bộ lọc Gauss trong Hình 7

Khử nhiễu

Như nhận xét trên, dẫu biết có thể sử dụng bộ lọc Gauss để khử nhiễu, nhưng đa số, bộ lọc Gauss tỏ ra kém hiệu quả trong khoản này!. Hình 10

2.2 Median-Filter

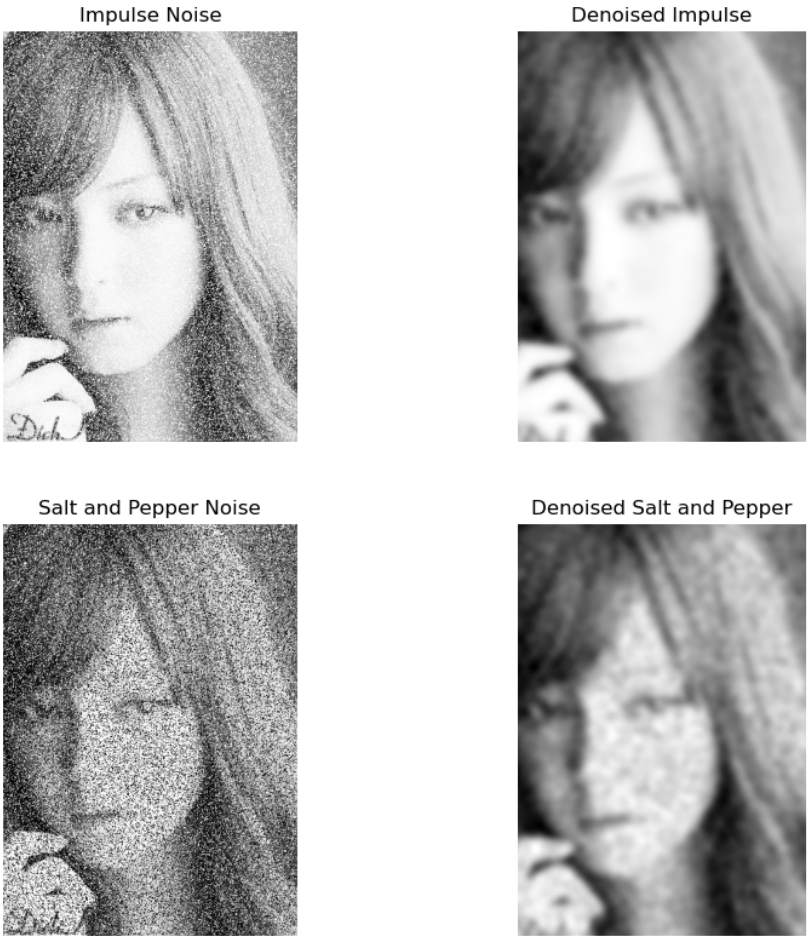
Ta thấy rằng bộ lọc Gauss đã không giải quyết triệt để được vấn đề khử nhiễu, cho nên bộ lọc *Median* ra đời để xử lí điều đó.

Trong thống kê, khi thu thập một lượng mẫu lớn về một sự kiện nào đó, việc đầu tiên nghĩ đến sẽ là lấy giá trị trung bình của chúng - đây chính xác là ý tưởng của *Gaussian Filter*. Tuy nhiên, có một số trường hợp, người ta kị sử dụng giá trị trung bình, đó là khi số liệu có tính phân hóa cao. Khi đó thì *Median* sẽ được sử dụng thay cho *kì vọng*.

Có thể lấy ví dụ như sau:

> "Có một bạn sinh viên đi thi vấn đáp môn 'Thị giác máy tính' trước hội đồng gồm 9 thầy cô. Sau khi đi thi số điểm bạn ấy nhận được lần lượt là 0, 0, 1, 2, 7, 8, 9, 10, 10. Điểm thì chỉ có một đầu điểm. Vậy có thể cho rằng bạn ý sẽ được bao nhiêu điểm?"

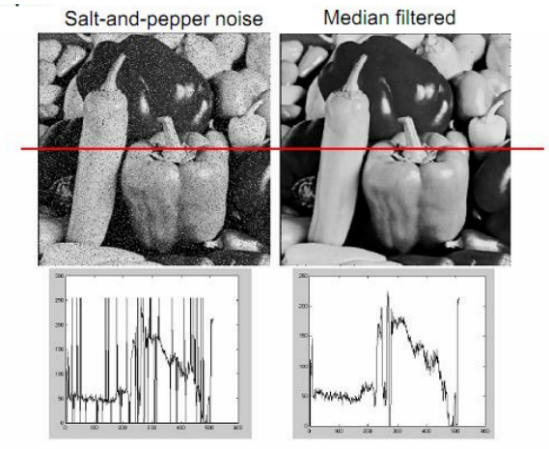
Dễ thấy rằng nếu tính theo kì vọng, số điểm trung bình của bạn sẽ là 5.2.



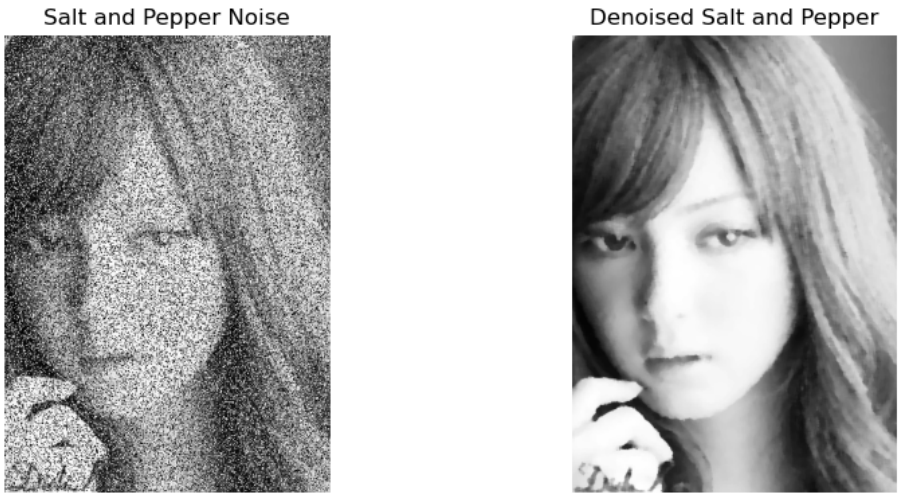
Hình 10: Bộ lọc Gauss tỏ ra kém hiệu quả trước các nhiễu đặc biệt

Nhưng có vẻ sẽ không hợp lí lắm, khi phần lớn thầy cô đều cho bạn điểm khá cao. Trong trường hợp này, ta sử dụng trung vị (*Median*) số điểm của bạn ấy có thể sẽ là 7 thì hợp lí hơn nhiều!
Tham khảo thêm tại Trung vị, wikipedia

Hình 11 cho ta minh họa về phép lọc *Median*, ta thấy lược đồ xám của ảnh gốc, rất rất 'lệch'. Sau khi sử dụng bộ lọc, kết quả hoàn toàn khiến chúng ta bất ngờ!
Tuy nhiên, ảnh ta vẫn không tránh khỏi việc bị mờ đi nhiều so với ảnh gốc! Kernel size càng lớn, khử nhiễu càng tốt, nhưng ảnh càng làm mờ ảnh!



Hình 11: Lược đồ xám và ảnh. Sử dụng bộ lọc *Median*



Hình 12: Mờ! Tác dụng phụ của *Median Filter*

3 Sharpening Spatial Filters

3.1 Đạo hàm bậc 1 và Đạo hàm bậc 2

Chúng ta đã không còn xa lạ với khái niệm đạo hàm, khái niệm mà hầu như ta dùng đến ở mọi lĩnh vực. Ý nghĩa của đạo hàm của một hàm số, ngoài cho ta biết về độ dốc, còn cho ta biết về hướng tăng của hàm số đó!

Xuất phát từ định nghĩa:

$$f'(x) = \frac{f(x+\Delta) - f(x)}{\Delta}, \Delta \approx 0$$

Trong hàm số liên tục, Δ là lân cận của x, tuy nhiên trong ảnh lân cận của một điểm ảnh chính là điểm ảnh ngay sát nó.
Ta định nghĩa đạo hàm cho trường hợp rời rạc:

$$\frac{\partial f(x)}{\partial x} = f(x+1) - f(x)$$

Trên là đạo hàm phải, mang ý nghĩa là nhìn về phía trước. Tương tự, ta có định nghĩa đạo hàm bậc hai cho trường hợp rời rạc:

$$\frac{\partial^2 f(x)}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

hoàn toàn dễ suy ra từ khai triển *Taylor*

3.2 Toán tử Laplace - Laplacian operator

Toán tử *Laplace* được định nghĩa, như sau:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Trong trường hợp rời rạc:

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

Dựa vào đó, ta có các bộ lọc tương ứng: Hình 13 minh họa về các bộ lọc *Laplace*.

3.3 Làm nét ảnh bằng bộ lọc Laplace

$$g(x, y) = f(x, y) + c\left[\nabla^2 f(x, y)\right] \tag{4}$$

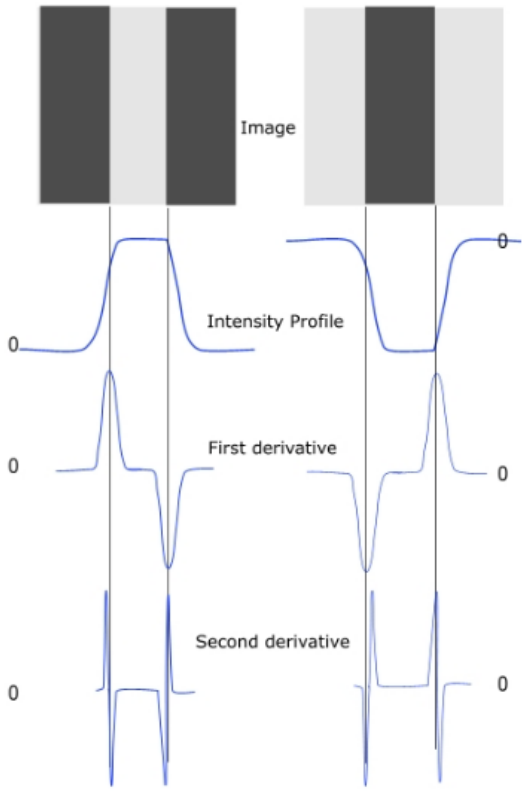
trong đó c là hằng số, hoặc là 1 hoặc -1, phụ thuộc vào phần tử ở giữa bộ lọc. Xem lại mục (1.4).
Thật vậy, ta có nhận xét rằng, có thể kết hợp 4 vào thành một bộ lọc!!! Ví dụ:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -5 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -9 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1
0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

Hình 13: Các bộ lọc Laplace, hai bộ lọc bên phải, là có tính cả đường chéo, khi đó hệ số của tâm sẽ là ±8

Hình 14: Kết quả với bộ lọc Laplace, sử dụng phiên bản kết hợp với bộ lọc chéo. cv2



Hình 15: Đạo hàm cấp một, cấp hai.

3.4 Thảo luận

Tôi cũng như bạn, cũng thắc mắc tại sao lại có công thức (4), tôi đã cố gắng suy nghĩ, và cũng có đôi chút suy đoán.
Trước tiên, hãy nhìn vào Hình 15.
Những điểm có mức sáng thay đổi đột ngột sẽ là cạnh (*edges*). Và nó sẽ ứng với vị trị mà đạo hàm bậc nhất đạt cực đại, hay đạo hàm bậc hai đạt 0. Qua toán tử đạo hàm, nhưng điểm có mức sáng bằng nhau sẽ bị chuyển về 0, tương tự

với đạo hàm bậc 2.
Do đó nếu ta lấy ảnh gốc trừ đi ảnh đạo hàm cấp hai, cạnh và các điểm lân cận cùng mức sáng sẽ không thay đổi, nhưng lân cận của cạnh, sẽ càng tương phản. Dẫn đến ta trông ảnh ta như sắc nét hơn!
Tuy nhiên, toán tử của ta là toán tử *Laplace* với việc cộng hai đạo hàm riêng nên suy luận trên cũng không hoàn toàn đúng!

Đó là lí thuyết nhưng trong thực tế:

1. Các điểm lân cận, dù gần nhau nhưng qua toán tử đạo hàm, điểm ảnh mới chưa chắc đã bằng 0. Lí do này khiến ảnh bị '*tối hơn*' nếu sử dụng công thức (4).

2. Nếu không sử dụng toán tử kết hợp, mà sử dụng công thức (4) thì rất dễ gây ra sai sót, bởi bộ lọc *Laplace* không đảm bảo về tính trung bình của điểm ảnh (*tổng các phần tử khác 1*).

Để khắc phục điều này, có hai cách:

1. Chuẩn hóa ảnh sau khi dùng toán tử *Laplace* rồi nhân với 255 làm tròn. Trước khi dùng công thức (4). Nếu ảnh tối thì ta có thể làm sáng lại bằng các phương pháp của bài trước!

2. Dùng thẳng công thức thu gọn, kết hợp với hàm *filter2D* của thư viện *cv2*.

3. Trong trường hợp muốn dùng hằng số c khác 1, -1 thì đừng quên mục 1.4

Tuy nhiên những khắc phục này chưa triển để. Hình 16. Và trong việc này thư viện *cv2* làm tốt hơn, so với tự làm theo lí thuyết!

Ngoài cách làm nét này bạn có thể tham khảo thêm *Highboost Filtering* cũng khá tương tự.

4 Mã nguồn

Mã nguồn của bài này có thể tham khảo tại đây: *My Github* , *Link colab* Tuy nhiên phần làm nét tôi không làm tốt cho lắm, ảnh ở đây là vừa làm lại để đối chiếu, trong mã nguồn trên sẽ không có!
Trong bài tới ta sẽ đi sâu hơn về việc xác định cạnh!

Hình 16: Phỏng theo lí thuyết, tuy nhiên chưa quá hiệu quả.

5 Tài liệu tham khảo

Tài liệu

[1] Truong. PV, Thao. TT *Silde bài giảng tuần 4*, Đại học Bách Khoa Hà Nội.

[2] R.C. Gonzalez and R.E. Woods *Digital image processing (*3^{rd} *editon)*, Prentice Hall, 2008.

[3] Tiep. VK Bài 37: Tích chập hai chiều, machinelearningcoban.com, 2018.

[4] D2l 6.2 Phép tích chập cho ảnh, d2l.aivivn.com.

[5] `https://viblo.asia/p/tuan-5-gradient-and-edge-bJzKmOGwl9N`

[6] Wikipedia Trung vị

[7] Wikipedia Three sigma rule