IMAGE PROCESSING
Image filtering and enhancement

NGUYỄN HẢI TRIỀU[1]

[1]Bộ môn Kỹ thuật phần mềm,

Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, December 2023

1. SHARPENING (**HIGHPASS**) SPATIAL FILTERS
   - Đạo hàm bậc 1 - Đạo hàm bậc 2
   - Ý nghĩa của đạo hàm trong phát hiện biên ảnh
   - Using first-order derivatives for image sharpening–**the gradient**
   - Robert cross gradient
   - Sobel operators: Lọc Sobel
   - Lọc đạo hàm bậc 2 - Bộ lọc Laplacian

Lọc sắc nét ảnh trong miền không gian

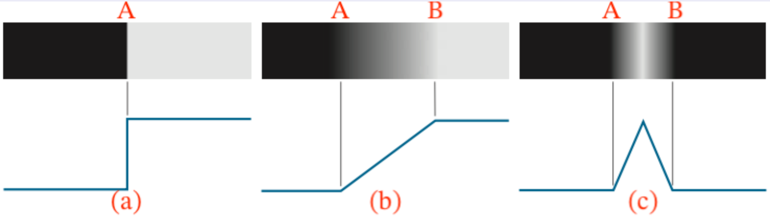
Tổng quan

Trong phần này chúng ta sẽ tìm hiểu:

1. Thế nào là Biên ảnh?
2. Mục đích của lọc sắc nét ảnh
3. Đạo hàm bậc 1,2 trong xử lý ảnh
4. Bộ lọc đạo hàm bậc 1
   - Bộ lọc Robert Cross Gradient
   - Bộ lọc Sobel
5. Bộ lọc đạo học bậc 2: Bộ lọc Laplacian (được đặt tên theo Pierre-Simon Laplace: một nhà toán học và nhà thiên văn học người Pháp)

Biên ảnh (edge)

- là tập hợp các pixel mà tại đó giá trị mức xám (cường độ) thay đổi nhanh hoặc đột ngột trong một bức ảnh
- mục đích của **lọc sắc nét ảnh là làm nổi bật các chi tiết trong ảnh, tức là làm nổi các biên ảnh**
- lọc sắc nét cũng được gọi là *kỹ thuật làm nổi biên ảnh, xác định sự thay đổi mức xám các pixel trong ảnh*



Hình 1: Phân loại biên ảnh: (a) biên nhảy vọt (**step/skip**); (b) biên thoai thoải (**Ramp**); (c) biên kiểu mái nhà (**Roof**).

Foundation

Trong toán học, *để xác định độ biến thiên của hàm số thì sử dụng đạo hàm bậc 1 hoặc đạo hàm bậc 2* (first- and second-order derivatives). Xét hàm một biến $f(x)$, ta có công thức xấp xỉ

- đạo hàm riêng bậc 1:

$$\partial f / \partial x \approx f(x+1) - f(x) \tag{1}$$

- đạo hàm riêng bậc 2:

$$\partial^2 f / \partial^2 x \approx f(x+1) + f(x-1) - 2f(x) \tag{2}$$

Lưu ý: để đơn giản, trong chương này chúng ta có thể thay dấu $\approx$ trong công thức eqs. (1) and (2) thành dấu $=$. Để tìm được công thức eqs. (1) and (2), sinh viên đọc chapter 10 tài liệu [1]

Nhắc lại chapter 10 tài liệu [1]

Derivatives of a digital function are defined in terms of finite differences. We obtain an approximation to the first-order derivative at an arbitrary point x of a one-dimensional function $f(x)$ by **expanding the function** $f(x + \Delta x)$ **into a Taylor series about** x

$$\begin{aligned} f(x+\Delta x) = & \; f(x) + \Delta x \frac{\partial f(x)}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \frac{(\Delta x)^3}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \ldots \\ = & \; \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!}\frac{\partial^n f(x)}{\partial x^n} \end{aligned} \tag{3}$$

For our purposes, Δx **is measured in pixel units**. Following the convention in the presentation, $\Delta x = 1$ for the sample preceding x and $\Delta x = -1$ for the sample following x (**1 pixel tương đương** $\Delta x = 1$, On an image, we cannot let Δx go to zero).

Nhắc lại chapter 10 tài liệu [1]

When $\Delta x = 1$ (sự khác nhau của các giá trị sau so với giá trị hiện tại), Eq.3 becomes

$$\begin{aligned} f(x+1) = & \; f(x) + \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \frac{1}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \ldots \\ = & \; \sum_{n=0}^{\infty} \frac{1}{n!}\frac{\partial^n f(x)}{\partial x^n} \end{aligned} \tag{4}$$

Similarly, when $\Delta x = -1$ (sự khác nhau của các giá trị trước so với giá trị hiện tại),

$$\begin{aligned} f(x-1) = & \; f(x) - \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial x^2} - \frac{1}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \ldots \\ = & \; \sum_{n=0}^{\infty} \frac{(-1)^n}{n!}\frac{\partial^n f(x)}{\partial x^n} \end{aligned} \tag{5}$$

Nhắc lại chapter 10 tài liệu [1]

We **compute intensity differences** using just a few terms of the Taylor series. For first-order derivatives we **use only the linear terms**.

- The **forward difference** is obtained from Eq.4

$$\frac{\partial f(x)}{\partial x} = f'(x) \approx f(x+1) - f(x) \tag{6}$$

- The **backward difference** is similarly obtained by keeping only the linear terms in Eq.5:

$$\frac{\partial f(x)}{\partial x} = f'(x) \approx f(x) - f(x-1) \tag{7}$$

Nhắc lại chapter 10 tài liệu [1]

We **compute intensity differences** using just a few terms of the Taylor series. For first-order derivatives we **use only the linear terms**.

- The **central difference** is obtained by subtracting Eq.5 from Eq.4:

$$\frac{\partial f(x)}{\partial x} = f'(x) \approx \frac{f(x+1) - f(x-1)}{2} \tag{8}$$

In general, the **more terms we use** from the Taylor series to represent a derivative, the **more accurate the approximation will be**. *Từ các công thức eqs. (6) to (8), ta thấy rằng* ***central difference cho xấp xỉ sai số nhỏ nhất*** *khi chỉ sử dụng các hệ số tuyến tính trong công thức Taylor.*

Nhắc lại chapter 10 tài liệu [1]

The second order derivative based on a central difference, $\partial^2 f/\partial x^2$, is obtained by adding Eqs.4 and 5:

$$\frac{\partial^2 f}{\partial x^2} = f''(x) \approx f(x+1) - 2f(x) + f(x-1) \tag{9}$$

To obtain the **third order, central derivative we need one more point on either side of** x. That is, we need the Taylor expansions for $f(x+2)$ and $f(x-2)$, which we obtain from Eqs. 4 and 5 with $\Delta x = 2$ and $\Delta x = -2$.

Bài tập: SV chứng minh rằng

$$\frac{\partial^3 f}{\partial x^3} = \quad f'''(x) \approx \frac{f(x+2)-2f(x+1)+0f(x)+2f(x-1)-f(x-2)}{2}$$

Ý nghĩa của đạo hàm trong phát hiện biên ảnh

Derivatives of a digital function are defined in terms of differences. There are various ways to define these differences. However, **we require that any definition we use for a *first derivative***:

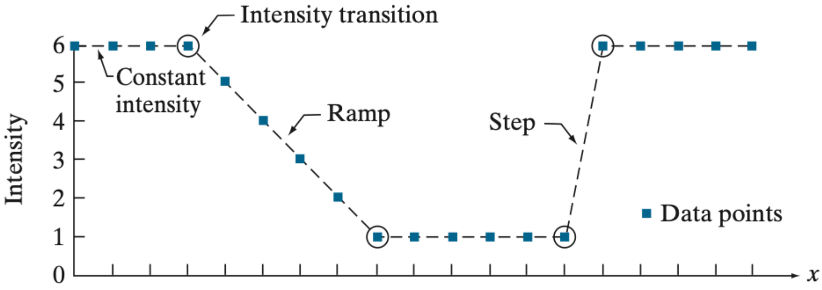
1. Must be zero in areas of constant intensity.
2. Must be nonzero at the onset of an intensity step or ramp.
3. Must be nonzero along intensity ramps.

Similarly, **any definition of a *second derivative***

1. Must be zero in areas of constant intensity.
2. Must be nonzero at the onset **and end** of an intensity step or ramp.
3. **Must be zero** along intensity ramps.

Ý nghĩa của đạo hàm trong phát hiện biên ảnh

(a)

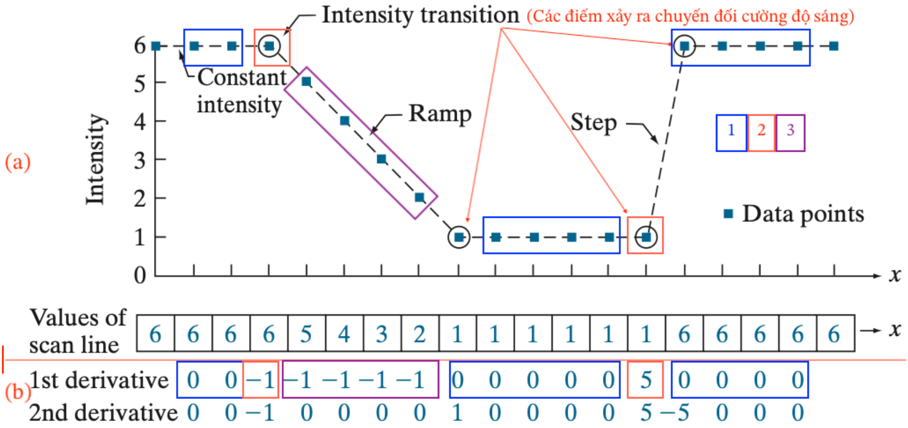


(b)

Values of scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	

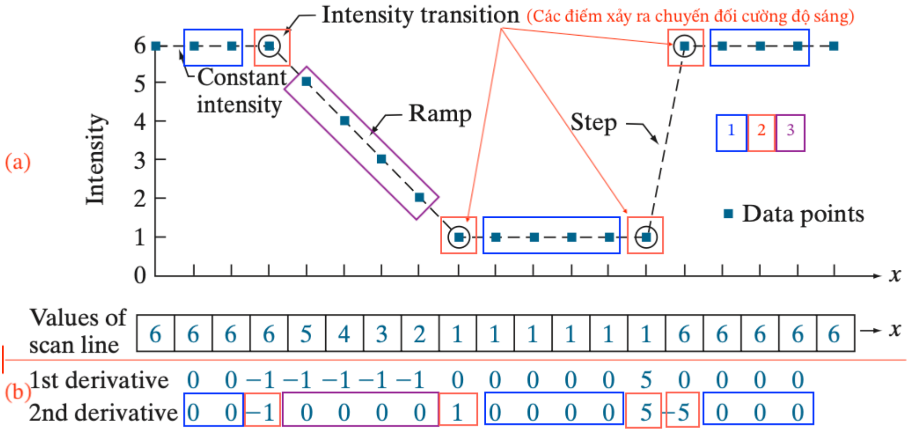
Hình 2: (a) A section of a horizontal scan line from an image, showing ramp and step edges, as well as constant segments. (b)Values of the scan line and its derivatives.

Ý nghĩa của đạo hàm trong phát hiện biên ảnh



Hình 3: **first derivative**: 1. Must be zero in areas of constant intensity; 2. Must be nonzero at the onset of an intensity step or ramp; 3. Must be nonzero along intensity ramps.

Ý nghĩa của đạo hàm trong phát hiện biên ảnh



Hình 4: **second derivative**: 1. Must be zero in areas of constant intensity; 2. Must be nonzero at the onset and end of an intensity step or ramp; 3. Must be zero along intensity ramps.

Lọc đạo hàm bậc 1: **THE GRADIENT**

First derivatives in image processing are implemented **using the magnitude of the gradient**. The gradient of an image f at coordinates (x, y) is defined as **the two-dimensional column vector**

$$grad(f) = \nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \tag{10}$$

Độ lớn của vector ∇f là **tốc độ thay đổi cường độ sáng lớn nhất tại tọa độ** (x, y) **theo hướng gradient**. The magnitude (length) of vector ∇f , denoted as $M(x, y)$ (the *vector norm* notation $\|\nabla f\|$ is also used frequently), where

$$M(x, y) = \|\nabla f\| = mag(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

In some implementations, it is more suitable computationally to approximate the squares and square root operations by absolute values:

$$M(x, y) = \|\nabla f\| \approx |g_x| + |g_y| \tag{11}$$

This expression **still preserves the relative changes in intensity**, but *the isotropic property is lost in general*. The most popular kernels used to approximate the gradient are isotropic at multiples of 90^o, so nothing of significance is lost in using the Eq.(11).

Robert cross gradient

Để đơn giản, chúng ta kí hiệu toạ độ các pixel của một vùng ảnh 3×3 như Hình 5. Trong đó, $z_5 = f(x, y)$, $z_6 = f(x+1, y)$, $z_8 = f(x, y+1)$, $z_9 = f(x+1, y+1)$... tương tự cho các z khác.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Hình 5: A 3×3 region of an image, where the z_s are intensity values.

Từ công thức Gradient vector Eq.10, kết hợp với công thức xấp xỉ đạo hàm forward difference Eq.6, ta thu được:

$$\nabla f \approx \begin{bmatrix} f(x+1, y) - f(x, y) \\ f(x, y+1) - f(x, y) \end{bmatrix} = \begin{bmatrix} z_6 - z_5 \\ z_8 - z_5 \end{bmatrix} \qquad (12)$$

Robert đề xuất bộ lọc kích thước 2×2

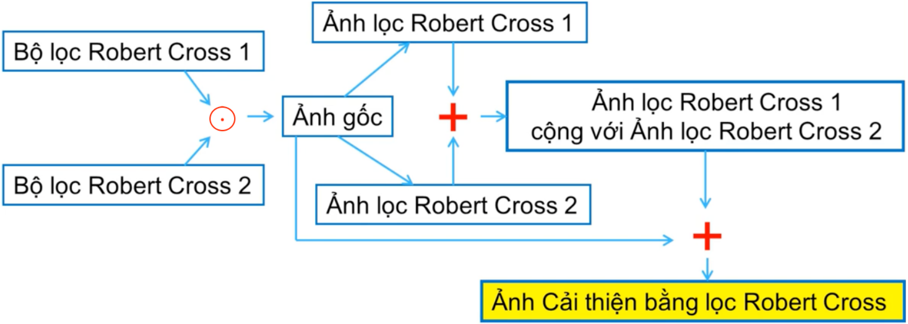
Two other definitions, proposed by Roberts [1965] in the early development of digital image processing, use cross differences:

- Theo hướng chéo thứ nhất: $g_x = z_9 - z_5 \Rightarrow$ **Bộ lọc cho hướng chéo thứ nhất**: $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$. *Giải thích: lấy bộ lọc thứ nhất nhân tích chập với vùng ảnh* $\begin{bmatrix} z_5 & z_6 \\ z_8 & z_9 \end{bmatrix}$ *ta tìm được công thức của* g_x.
- Theo hướng chéo thứ hai: $g_y = z_8 - z_6 \Rightarrow$ **Bộ lọc cho hướng chéo thứ hai**: $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

Bộ lọc Robert cross gradient kích thước 3×3

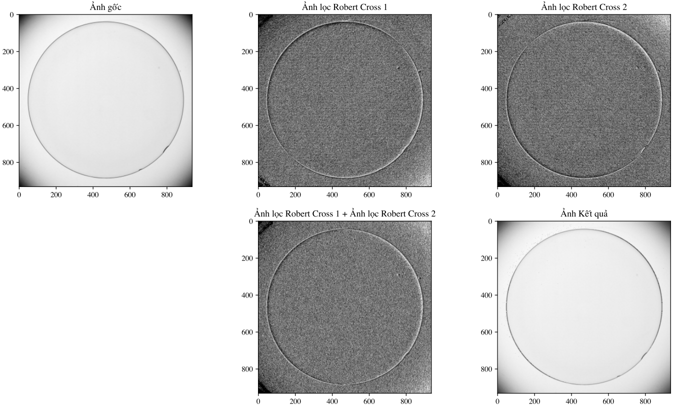
Mở rộng kích thước bộ lọc

- **Bộ lọc cho hướng chéo thứ nhất**: $G_{cross1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.
- **Bộ lọc cho hướng chéo thứ hai**:$G_{cross2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$.



Hình 6: Các bước thực hiện lọc sắc nét ảnh bằng bộ lọc Robert cross-gradient [3, 1]

Các bước thực hiện làm sắc nét ảnh bằng Bộ lọc Robert Cross Gradient



Hình 7: Kết quả minh hoạ kỹ thuật lọc sắc nét ảnh sử dụng Robert Cross Gradient cho ảnh chụp một Kính áp tròng.

Sobel operators: Lọc Sobel

Approximations to g_x and g_y using a 3×3 neighborhood centered on z_5 are as follows:

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \\ (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \end{bmatrix} \tag{13}$$

- *The difference between the third and first rows* of the 3×3 image region approximates the partial derivative in the x-direction.
- *The difference between the third and first columns* approximates the partial derivative in the y-direction.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

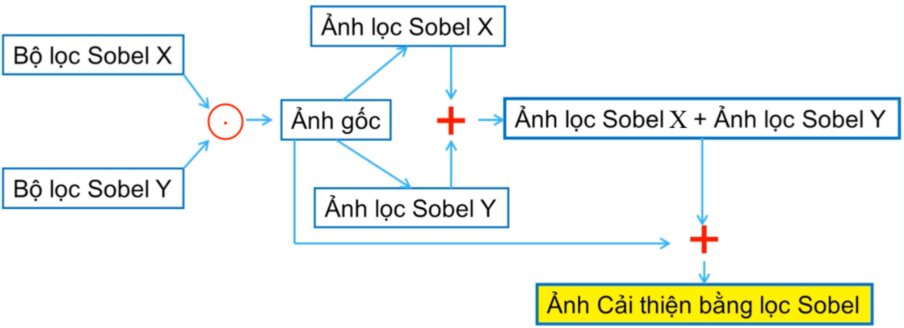
Bộ lọc Sobel theo hướng x, y

- Bộ lọc theo hướng x: $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$.
- Bộ lọc theo hướng y: $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$.

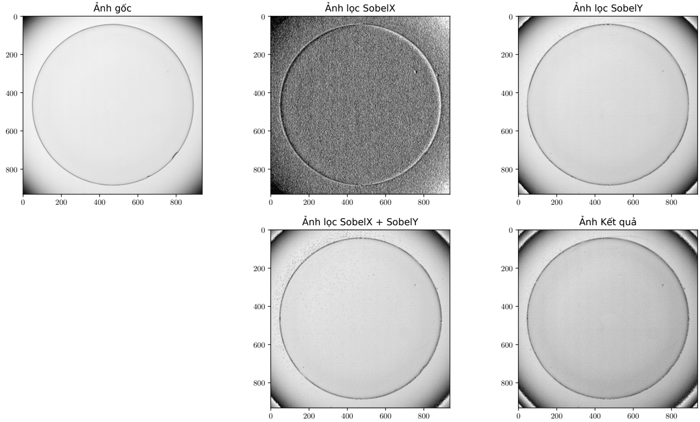
Giải thích: lấy bộ lọc theo hướng x nhân tích chập với vùng ảnh $f(x, y)$ ta tìm được cộng thức của g_x, nhân tích chập tương tự với bộ lọc theo hướng y.

Nhận xét

- Tổng các phần tử quanh điểm trung tâm bằng 0.
- Để lọc ảnh, chúng ta sử dụng cả 2 toán tử và kết hợp (cộng) kết quả lại với nhau.



Hình 8: Các bước thực hiện lọc sắc nét ảnh bằng bộ lọc Sobel [3, 1].



Hình 9: Kết quả minh hoạ kỹ thuật lọc sắc nét ảnh sử dụng Sobel operators cho ảnh chụp một Kính áp tròng.

Using the second derivative for image sharpening-**the laplacian**

The simplest isotropic derivative operator is the **Laplacian**, which, for an image $f(x, y)$ of two variables, is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}. \tag{14}$$

Từ công thức xấp xỉ đạo hàm bậc 2 ở Eq.9, ta có đạo hàm riêng bậc 2 theo 2 phương x-y là:

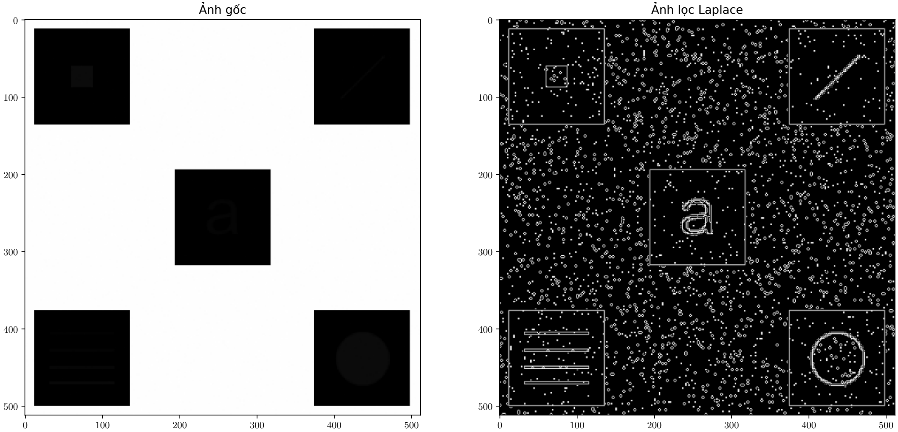
$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) - 2f(x, y) + f(x-1, y)$$
$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) - 2f(x, y) + f(x, y-1)$$

Thay đạo hàm riêng bậc 2 theo 2 hướng $x - y$ vào phương trình Laplacian 14, thu được

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{15}$$

Bộ lọc Laplacian

Tương tự như xây dựng bộ lọc Robert, Sobel, chúng ta dễ dàng xây dựng được bộ lọc Laplacian như sau $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$. *Giải thích: lấy bộ lọc Laplacian nhân tích chập với vùng ảnh $f(x,y)$ ta tìm được biểu diễn của $\nabla^2 f(x,y)$.*

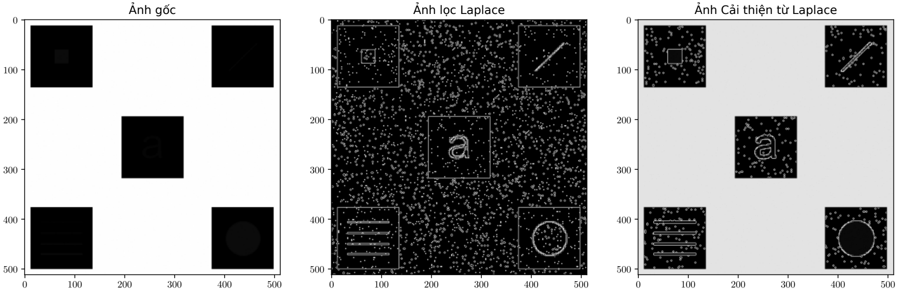


Hình 10: Áp dụng bộ lọc Laplacian vào một ảnh, chúng ta nhận được một ảnh **làm nổi biên và các đường nét không liên tục** khác và **được đặt trên nền tối** (*it highlights sharp intensity transitions in an image and de-emphasizes regions of slowly varying intensities, all superimposed on a dark, featureless background*).

Kết quả của lọc Laplacian không phải là một ảnh cải thiện. Để thu được ảnh cải thiện lọc sắc nét $g(x, y)$ từ ảnh gốc ban đầu $f(x, y)$ ta sử dụng công thức sau:

$$g(x, y) = f(x, y) + c\big[\nabla^2 f(x, y)\big], \tag{16}$$

với $c = -1$ hoặc $c = 1$, c phụ thuộc vào dấu của hệ số trung tâm trong bộ lọc Laplacian. Ví dụ trong bộ lọc Laplacian đã nói ở phía trước, hệ số trung tâm bằng -4, vậy nên $c = -1$.



Hình 11: Ảnh cải thiện = ảnh gốc - ảnh qua bộ lọc Laplacian

Các biến thể của bộ lọc Laplacian

Biến thể thứ nhất của Laplacian

Dựa vào bộ lọc Laplacian chuẩn từ phương trình Eq.15, **thay vì lấy đạo hàm theo hướng $x - y$, ta sẽ lấy đạo hàm theo 2 đường chéo chính (thứ 1) và phụ (thứ 2)**. Kết quả biến thể thứ nhất là ***tổng*** *của bộ lọc Laplacian chuẩn từ phương trình Eq.15 cộng với bộ lọc theo hướng đạo hàm riêng theo 2 hướng chéo.*

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{17}$$

Các biến thể của bộ lọc Laplacian

Biến thể thứ hai của Laplacian

Dựa vào bộ lọc Laplacian chuẩn từ phương trình Eq.15, chúng ta sẽ tiến hành **đổi dấu đạo hàm** để thu được bộ lọc biến thể thứ 2.

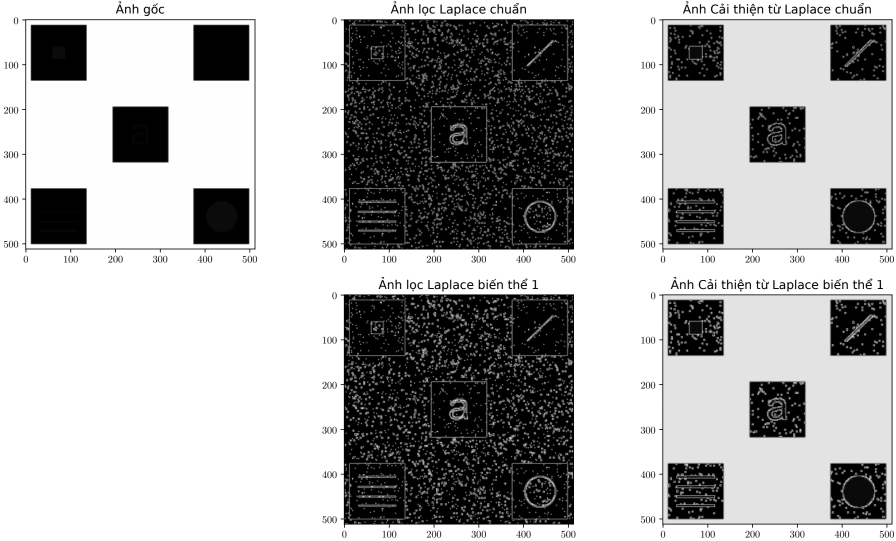
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \underline{\textbf{đổi dấu đạo hàm}} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (18)$$

Các biến thể của bộ lọc Laplacian

Biến thể thứ ba của Laplacian

Áp dụng ý tưởng của biến thể bộ lọc thứ 2 ở (18) vào biến thể thứ nhất của bộ lọc Laplacian ở (17), ta thu được biến thể thứ ba:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \xrightarrow{\textbf{đổi dấu đạo hàm}} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{19}$$



Hình 12: Kết quả minh hoạ khi áp dụng các bộ lọc sắc nét đạo hàm bậc 2 Laplacian và các biến thể của nó.

Nhận xét

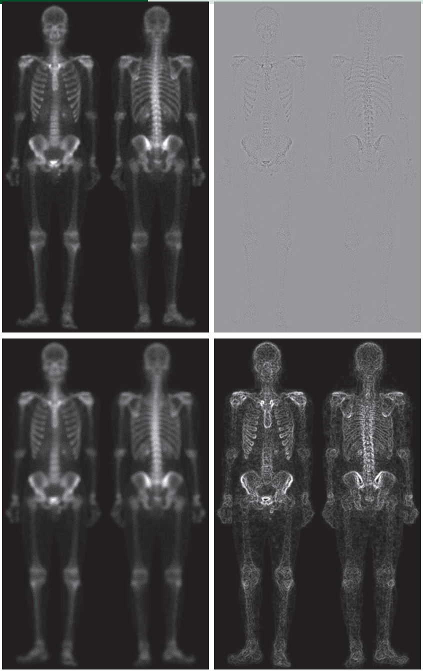
So sánh đạo hàm bậc 1 và bậc 2:

- Các đạo hàm bậc 1 thường tạo ra các biên mỏng hơn
- Các đạo hàm bậc 2 có đáp ứng mạnh hơn với các chi tiết nét, chẳng hạn như các đường mảnh
- Đạo hàm bậc 1 có đáp ứng mạnh hơn với bước thay đổi độ sáng
- Đạo hàm bậc 2 tạo ra đáp ứng kép ở bước thay đối độ xám
- Đạo hàm bậc 2 thường được sử dụng hơn đạo hàm bậc 1: đáp ứng mạnh hơn với các chi tiết nét

Kết luận

Thành công của cải thiện ảnh không thể đạt được với một phương pháp đơn lẻ. Chúng ta **kết hợp các kỹ thuật khác nhau** để đạt được kết quả cuối cùng.

Lọc đạo hàm bậc 2 - Bộ lọc Laplacian



a	b
c	d

FIGURE 3.57 (a) Image of whole body bone scan. (b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel gradient of image (a). (Original image courtesy of G.E. Medical Systems.)

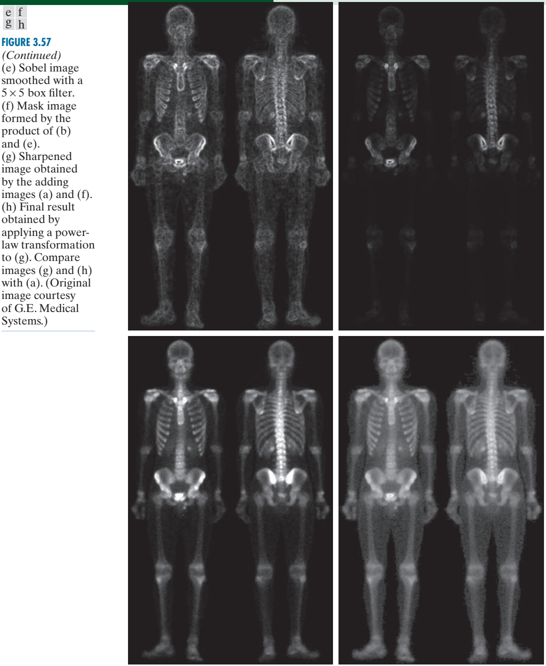


FIGURE 3.57 *(Continued)* (e) Sobel image smoothed with a 5×5 box filter. (f) Mask image formed by the product of (b) and (e). (g) Sharpened image obtained by the adding images (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare images (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)

Code minh hoạ I

```
import matplotlib.pyplot as plt
import cv2
import numpy as np
plt.rcParams.update({"text.usetex":True})
class highpass_filter(object):
  '''Các bộ lọc cơ bản trong highpass filter'''
  def __init__(self,image) -> None:
    '''constructor:
      - Ảnh đầu vào
    '''
    self.img=image
  def init_kernel(self, kernel_name:str):
    '''
    Nhập vào tên kernel:
      - RobertCross: xem kết quả lọc sắc nét bằng bộ lọc RobertCrossGradient
      - RobertCross1: xem Lọc RobertCross theo hướng 1
      - RobertCross2: xem Lọc RobertCross theo hướng 2
      - Sobel: Xem kết quả lọc sắc nét bằng bộ lọc Sobel
      - SobelX: xem Lọc Sobel theo hướng X
```

Code minh hoạ II

```
        - SobelX: xem Lọc Sobel theo hướng Y
        - Laplacian: Xem kết quả lọc sắc nét bằng bộ lọc Laplacian chuẩn
        - Laplacian1: Xem kết quả lọc sắc nét bằng bộ lọc Laplacian biến thể 1
        - Laplacian2: Xem kết quả lọc sắc nét bằng bộ lọc Laplacian biến thể 2
        - Laplacian3: Xem kết quả lọc sắc nét bằng bộ lọc Laplacian biến thể 3
      '''
      if kernel_name=="RobertCross":
        return self.show_SharpenedRobert()
      elif kernel_name=="RobertCross1":
        return self.show_RobertCrossGradient1()
      elif kernel_name=="RobertCross2":
        return self.show_RobertCrossGradient2()
      elif kernel_name=="Sobel":
        return self.show_SharpenedSobel()
      elif kernel_name=="SobelX":
        return self.show_SobelX()
      elif kernel_name=="SobelY":
        return self.show_SobelY()
      elif kernel_name=="Laplacian":
        return self.show_SharpenedLaplacian()
```

Code minh hoạ III

```
    elif kernel_name=="LaplacianV1":
      return self.show_SharpenedLaplacianV1()
    elif kernel_name=="LaplacianV2":
      return self.show_SharpenedLaplacianV2()
    elif kernel_name=="LaplacianV3":
      return self.show_SharpenedLaplacianV3()
    else:
      return self.img
  def Convolution2D(self,kernel):
    m, n = self.img.shape
    img_new = np.zeros([m, n])
    for i in range(1, m-1):
      for j in range(1, n-1):
        temp= self.img[i-1, j-1]  * kernel[0, 0]\
          + self.img[i, j-1]    * kernel[0, 1]\
          + self.img[i+1, j - 1] * kernel[0, 2]\
          + self.img[i-1, j]    * kernel[1, 0]\
          + self.img[i, j]      * kernel[1, 1]\
          + self.img[i+1, j]    * kernel[1, 2]\
          + self.img[i - 1, j+1] * kernel[2, 0]\
```

Code minh hoạ IV

```
            +  self.img[i, j + 1]  * kernel[2, 1]\
            +  self.img[i + 1, j + 1]* kernel[2, 2]
          img_new[i, j]= temp
      img_new = img_new.astype(np.uint8)
      return img_new
    def show_RobertCrossGradient1(self):
      '''Kết quả RobertCrossGradient theo hướng thứ nhất'''
      G_cross1 = np.array(([0, 0, 0], [0,-1, 0], [0, 0, 1]), dtype="float")
      return self.Convolution2D(G_cross1)
    def show_RobertCrossGradient2(self):
      '''Kết quả RobertCrossGradient theo hướng thứ nhất'''
      G_cross2 = np.array(([0, 0, 0], [0, 0,-1], [0, 1, 0]), dtype="float")
      return self.Convolution2D(G_cross2)
    def show_SharpenedRobert(self):
      '''Kết quả Lọc sắc nét sử dụng bộ lọc RobertCrossGradient theo 2 hướng
           chéo'''
      return
           self.show_RobertCrossGradient1()+self.show_RobertCrossGradient2()+self.img
    def show_SobelX(self):
      '''Bộ lọc Sobel theo hướng X'''
```

Code minh hoạ V

```
      SobelX = np.array(([-1,-2,-1], [ 0, 0, 0], [ 1, 2, 1]), dtype="float")
      return self.Convolution2D(SobelX)
    def show_SobelY(self):
      '''Bộ lọc Sobel theo hướng Y'''
      SobelY = np.array(([-1, 0, 1], [-2, 0, 2], [ 1, 0, 1]), dtype="float")
      return self.Convolution2D(SobelY)
    def show_SharpenedSobel(self):
      '''Bộ lọc Sobel'''
      return self.show_SobelX()+self.show_SobelY()+self.img
    def show_SharpenedLaplacian(self):
      '''Bộ lọc Laplacian chuẩn'''
      Laplacian_kerner = np.array(([0, 1, 0], [1, -4, 1], [0, 1, 0]),
           dtype="float")
      return self.img-self.Convolution2D(Laplacian_kerner)
    def show_SharpenedLaplacianV1(self):
      '''Bộ lọc Laplacian Biến thể V1'''
      Laplacian_kerner = np.array(([1, 1, 1], [1, -8, 1], [1, 1, 1]),
           dtype="float")
      return self.img-self.Convolution2D(Laplacian_kerner)
    def show_SharpenedLaplacianV2(self):
```

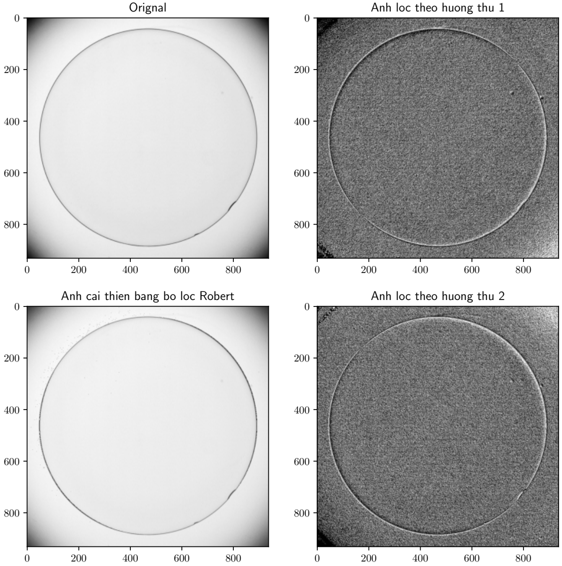
Code minh hoạ VI

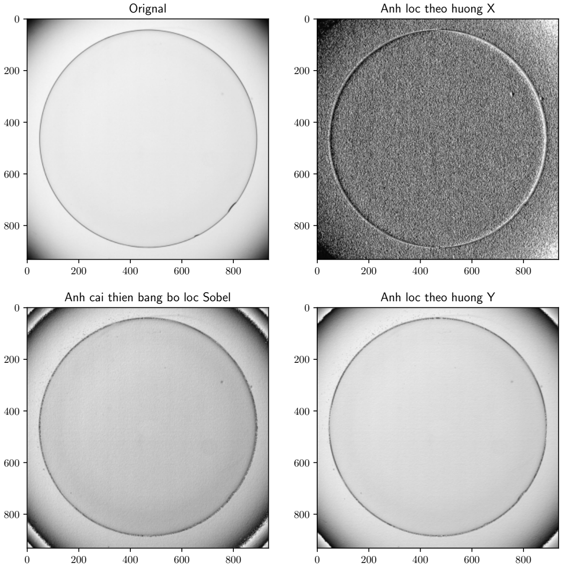
```
    '''Bộ lọc Laplacian Biến thể V2'''
    Laplacian_kerner = np.array(([0, -1, 0], [-1, 4, -1], [0, -1, 0]),
         dtype="float")
    return self.img+self.Convolution2D(Laplacian_kerner)
  def show_SharpenedLaplacianV3(self):
    '''Bộ lọc Laplacian Biến thể V3'''
    Laplacian_kerner = np.array(([-1, -1, -1], [-1, 8, -1], [-1, -1, -1]),
         dtype="float")
    return self.img+self.Convolution2D(Laplacian_kerner)


if __name__=='__main__':
  # Đọc và hiển thị ảnh gốc
  image = cv2.imread('./figures/len.tif', 0)
  fig=plt.figure(figsize=(9, 9))
  ax=fig.subplots(2,2)
  # hien thi anh goc
  ax[0,0].set_title("Orignal")
  ax[0,0].imshow(image,cmap="gray")
  ## Su dung bo loc RobertCrossGradient
```

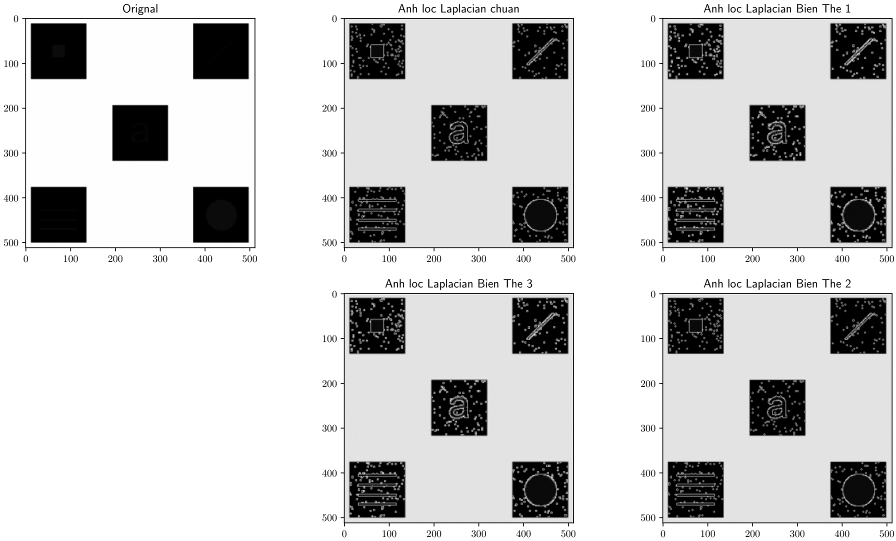
Code minh hoạ VII

```
# hien thi anh loc Robert theo huong thu 1
Robertcross1=highpass_filter(image).init_kernel("RobertCross1")
ax[0,1].imshow(Robertcross1,cmap='gray')
ax[0,1].set_title("Anh loc theo huong thu 1")
# hien thi anh loc Robert theo huong thu 2
Robertcross2=highpass_filter(image).init_kernel("RobertCross2")
ax[1,1].imshow(Robertcross2,cmap='gray')
ax[1,1].set_title("Anh loc theo huong thu 2")
# Ket qua anh loc sac net bang bo loc Robert
Robertcross=highpass_filter(image).init_kernel("RobertCross")
ax[1,0].imshow(Robertcross,cmap='gray')
ax[1,0].set_title("Anh cai thien bang bo loc Robert")
plt.savefig("filter_Robert.pdf",bbox_inches='tight')
plt.show()
```

Code minh hoạ VIII




Orignal
Anh loc theo huong X
Anh cai thien bang bo loc Sobel
Anh loc theo huong Y
0
200
400
600
800


Orignal
Anh loc Laplacian chuan
Anh loc Laplacian Bien The 1
Anh loc Laplacian Bien The 3
Anh loc Laplacian Bien The 2

Bài tập về nhà

Tăng cường ảnh "*whole-body-skeleton-scan.ppm*"

Yêu cầu sinh viên code lại các bước trong FIGURE 3.57, 3.58 cho ảnh đính kèm để thu được kết quả cuối cùng.

Tài liệu tham khảo

- Rafael C. Gonzalez, Richard E. Woods
  Digital Image Processing (2018), Fourth Edition, Global Edition, Pearson.
- Ravishankar Chityala, Sridevi Pudipeddi
  Image Processing and Acquisition using Python (2020), Second Edition, Chapman & Hall/CRC.
- Phạm Nguyễn Minh Nhựt
  Xử lý ảnh (2021), Trường Đại học Công nghệ Thông tin và Truyền thông Việt - Hàn.