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

Chương 3: Tách biên và phát hiện biên

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

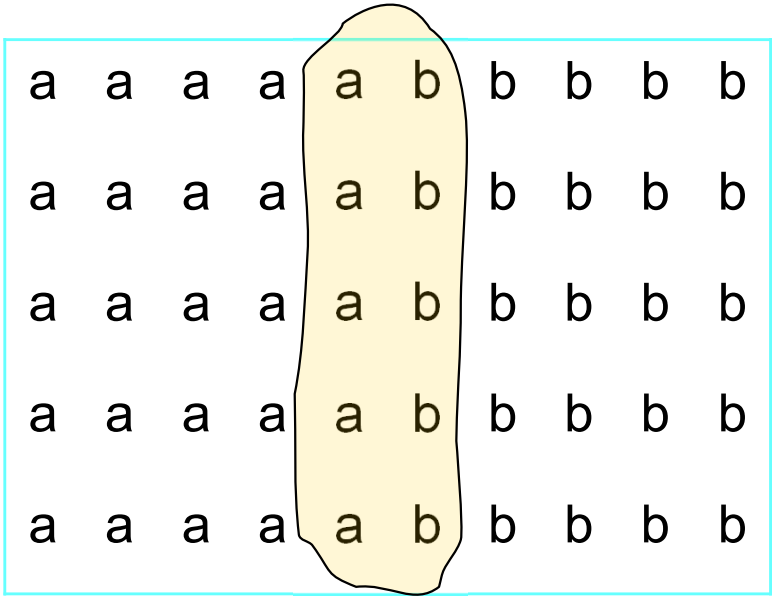
Đại học Vinh
Viện Kỹ thuật Công nghệ

ĐẠI HỌC VINH - 2022

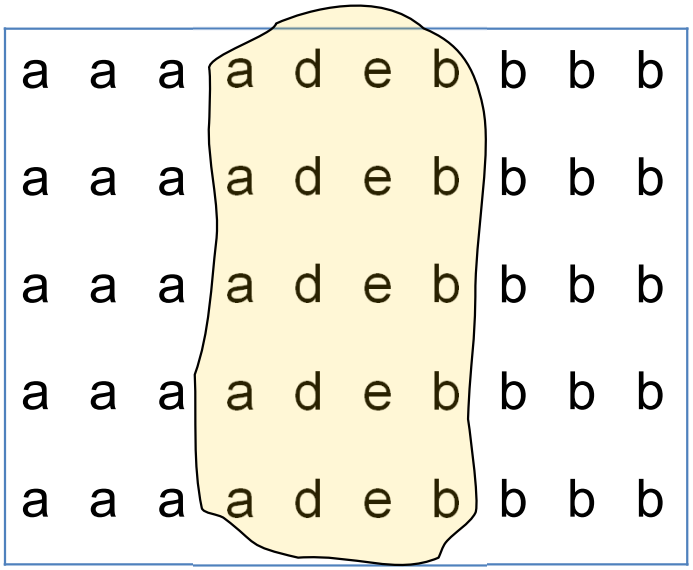
1. Khái niệm biên ảnh

- Biên là một trong những đặc tính quan trọng nhất của ảnh giúp chúng ta phân tích và hiểu ảnh.
- Điểm biên: Một điểm ảnh được coi là điểm biên nếu có sự thay đổi nhanh hoặc đột ngột về mức xám (hoặc màu).
- Tập hợp các điểm biên tạo thành biên hay đường bao (boundary) của ảnh. Như vậy, biên là đường ranh giới giữa các chi tiết ảnh hay các thực thể (đối tượng) ảnh.
- Ví dụ, trong ảnh nhị phân, điểm đen được gọi là điểm biên nếu lân cận của nó có ít nhất một điểm trắng.

Khái niệm biên ảnh

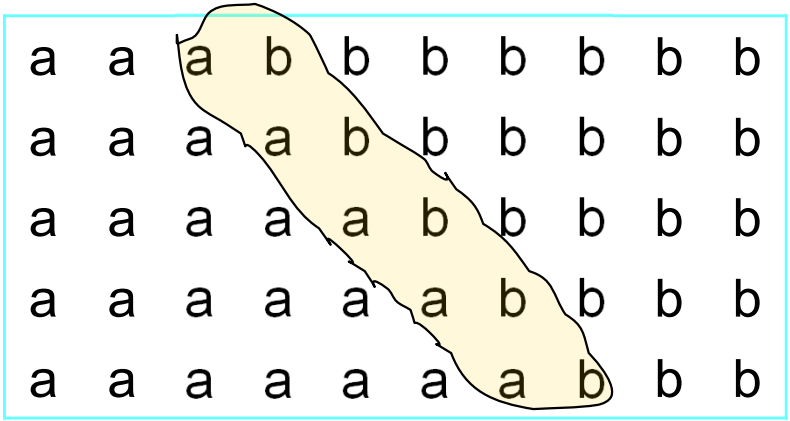


Biên bước dọc – chuyển tiếp đơn

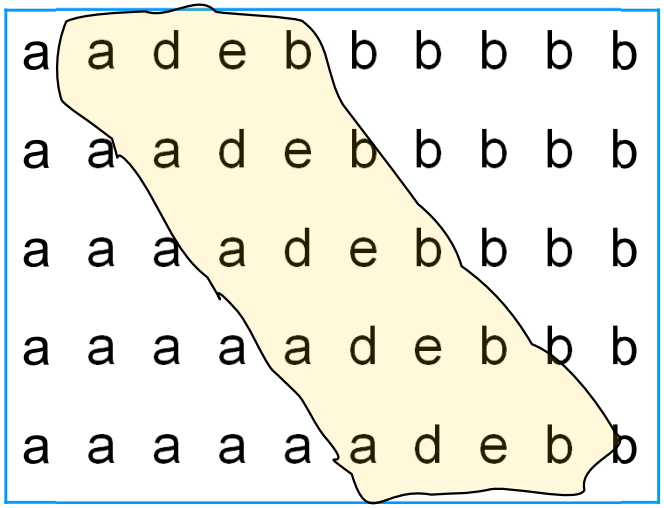


Biên bước dọc – chuyển tiếp mịn

Khái niệm biên ảnh

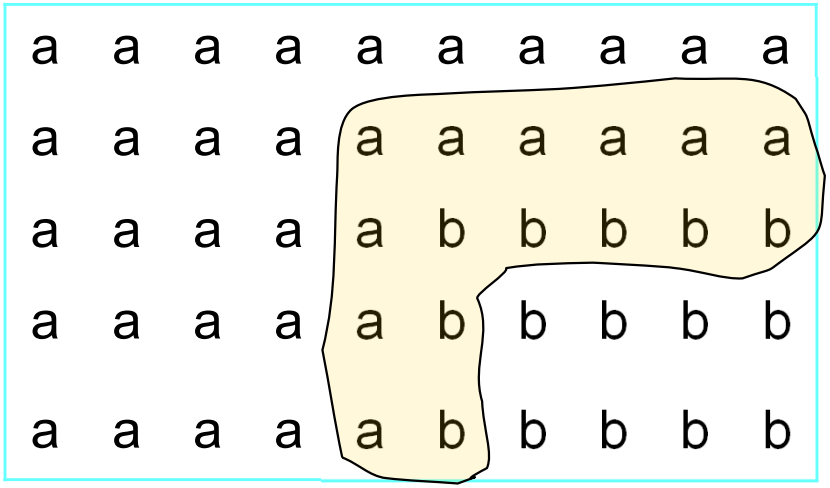


Biên bước chéo – chuyển tiếp đơn

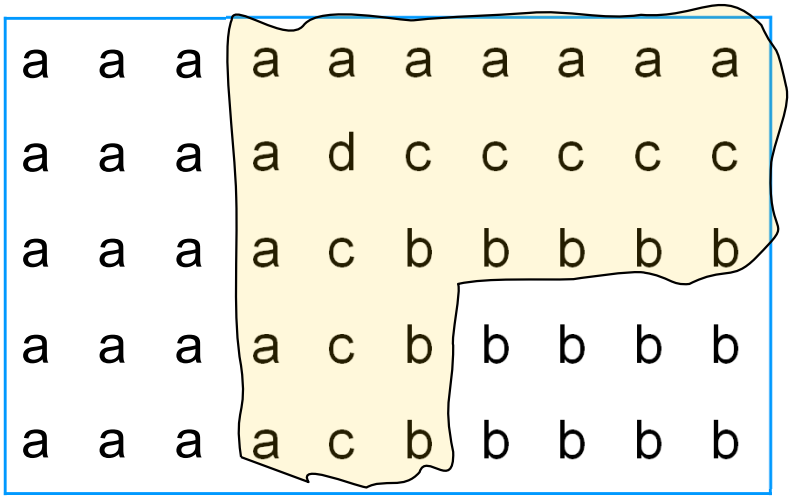


Biên bước chéo – chuyển tiếp mịn

Khái niệm biên ảnh



Biên góc – chuyển tiếp đơn



Biên góc – chuyển tiếp mịn

2. Các kĩ thuật phát hiện biên

- Có 2 phương pháp phát hiện biên:
- **Phát hiện biên trực tiếp**
- Phương pháp này chủ yếu dựa vào sự biến thiên độ sáng của đểm ảnh để làm nổi biên bằng kĩ thuật đạo hàm.
  - Phương pháp Gradient: lấy đạo hàm bậc nhất của f(x,y)
  - phương pháp Laplace: lấy đạo hàm bậc hai của f(x,y)
- **Phát hiện biên gián tiếp**
  - Phân chia ảnh thành các vùng → đường phân cách giữa các vùng đó chính là biên ➔ Kỹ thuật phân vùng ảnh

Các kĩ thuật phát hiện biên

- Phát hiện biên Gradient: Sobel, Prewitt, Robert
- Phát hiện biên Laplace
- Phát hiện biên Canny

Phương pháp Gradient

- Gradient là 1 vectơ có các thành phần biểu thị tốc độ thay đổi mức xám của điểm ảnh (theo 2 hướng x,y đối với ảnh 2 chiều)
- PP Gradient là PP dò biên cục bộ, dựa vào cực đại của đạo hàm bậc nhất
- Với hàm G(x, y), gradient của G tại tọa độ (x, y) được định nghĩa là một vectơ cột:

$$G(x,\ y) = \begin{bmatrix} \frac{\partial f(x,\ y)}{\partial x} \\ \frac{\partial f(x,\ y)}{\partial y} \end{bmatrix} = \begin{bmatrix} G_x \\ G_y \end{bmatrix}$$

- |G(x,y)| được gọi là độ lớn của Vectơ Gradient của ảnh. Ta có:

$$|G(x,y)| = \sqrt{G_x^2 + G_y^2} \quad \text{hoặc} \quad |G(x,y)| = |G_x| + |G_y|$$

Các toán tử Gradient

- Cho ảnh X(m,n), ta có:

$$\overrightarrow{X'} = \overrightarrow{X'_m} + \overrightarrow{X'_n} \quad \Rightarrow |X'| = \sqrt{(X'_m)^2 + (X'_n)^2}$$

- Tuy nhiên, trong xử lý ảnh để đơn giản phép tính mà không làm thay đổi bản chất của đạo hàm người ta thường tính như sau:

$$|X'(m,n)| = |X'_m| + |X'_n|$$

- X(m,n) là biên khi và chỉ khi X’(m,n) $\geq \theta$ (ngưỡng)

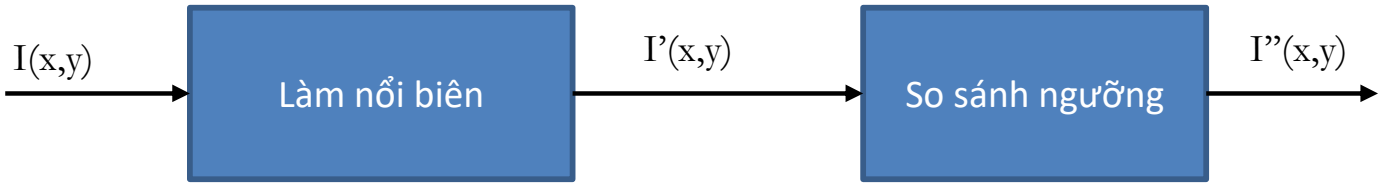
Các toán tử Gradient

- Trong xử lý ảnh để tính các đạo hàm thành phần người ta tìm cặp mặt nạ (H_m,H_n) sao cho:
- $X'_m(m,n) = X(m,n) * H_m$
- $X'_n(m,n) = X(m,n) * H_n$
- Cặp (H_m, H_n): toán tử Gradient

Quy trình tìm điểm biên

```
for(m = 0;m < M;m++)
    for(n = 0;n < N;n++)
        { Tìm  Gm = X(m,n) * Hm;
               Gn = X(m,n) * Hn
               G =|Gm| + |Gn|;
          Nếu (G >= θ)
               Y(m,n) = L; // biên
          else
               Y(m,n) = 0; // nền
        }
```

- Các công đoạn phát hiện biên theo kỹ thuật Gradient



- Thực tế, việc làm nổi biên là nhân chập ảnh I với một mặt nạ (ma trận)

Một số toán tử Gradient - Toán tử Prewitt

- Toán tử Prewitt

$$H_x(k,l) = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}; \qquad H_y(k,l) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

Một số toán tử Gradient - Toán tử Sobel

- Toán tử Sobel

$$H_x(k,l) = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}; H_y(k,l) = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Sử dụng hàm tách biên Sobel trong openCV

Các bước:

- Dùng hàm cv2.GaussianBlur() để loại bớt nhiễu
- Chuyển ảnh sang ảnh xám
- Áp dụng hàm Sobel nhân tích chập ảnh gốc với kernel 3x3 theo hướng x và y
- Lấy giá trị tuyệt đối
- Tính tổng trọng số của 2 mảng (2 ảnh)
- Hiện kết quả

Sử dụng hàm tách biên Sobel trong openCV

1. Dùng hàm cv2.GaussianBlur() để loại bớt nhiễu, làm mịn ảnh trước để thuật toán phát hiện biên làm việc tốt hơn:

   cv.GaussianBlur(src, (3, 3), 0)

2. Chuyển ảnh sang ảnh xám: cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
3. Áp dụng hàm Sobel nhân tích chập ảnh gốc với kernel 3x3 theo hướng x và y:

   cv.Sobel(src, ddepth, dx, dy,ksize)

   - src_gray: ảnh đầu vào (8 bit: CV_8U hay np.uint8)
   - ddepth: độ sâu của ảnh đầu ra, thường đặt cv2.CV_16S, cv2.CV_64F
   - x_order: Bậc của đạo hàm theo hướng x (xorder=1, yorder=0)
   - y_order: Bậc của đạo hàm theo hướng y (xorder=0, yorder=1)

- **grad_x = cv2.Sobel(src_gray, cv2.CV_64F, 1, 0, ksize)**
- **grad_y = cv2.Sobel(src_gray, cv2.CV_64F, 0, 1, ksize)**

Sử dụng hàm tách biên Sobel trong openCV

4. Lấy giá trị tuyệt đối và chuyển đổi kết quả thành 8-bit:
   cv2.convertScaleAbs(src)

- **abs_grad_x = cv2.convertScaleAbs(grad_x)**
- **abs_grad_y = cv2.convertScaleAbs(grad_y)**

5. Tính tổng trọng số của 2 mảng (2 ảnh):

– cv.addWeighted(src1, alpha, src2, beta, gamma)
  # src1*alpha+src2*beta+gamma
– **grad = cv.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)**

6. Hiện kết quả: **cv2.imshow(grad)**

cvtype values in OPENCV

CV_8U - 8-bit unsigned integers (0..255)

CV_8S - 8-bit signed integers (-128..127)

CV_16U - 16-bit unsigned integers (0..65535)

CV_16S - 16-bit signed integers (-32768..32767)

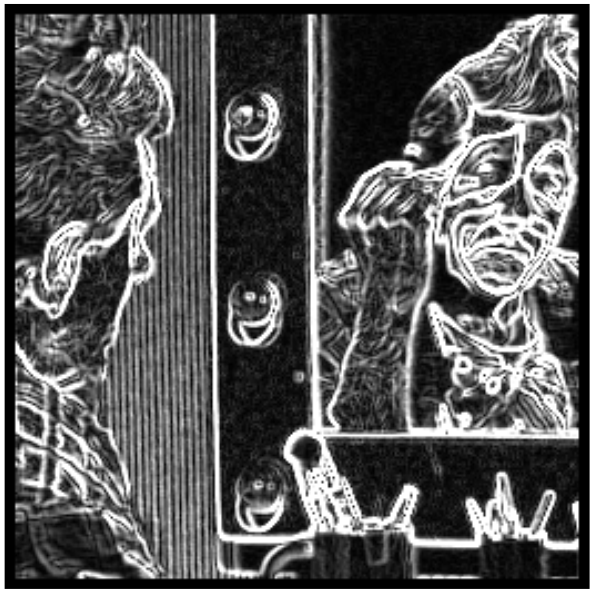
CV_32S - 32-bit signed integers (-2147483648..2147483647)

CV_32F - 32-bit floating-point numbers (-FLT_MAX..FLT_MAX, INF, NAN)

CV_64F - 64-bit floating-point numbers (-DBL_MAX..DBL_MAX, INF, NAN)

Ví dụ minh họa thuật toán Gradient

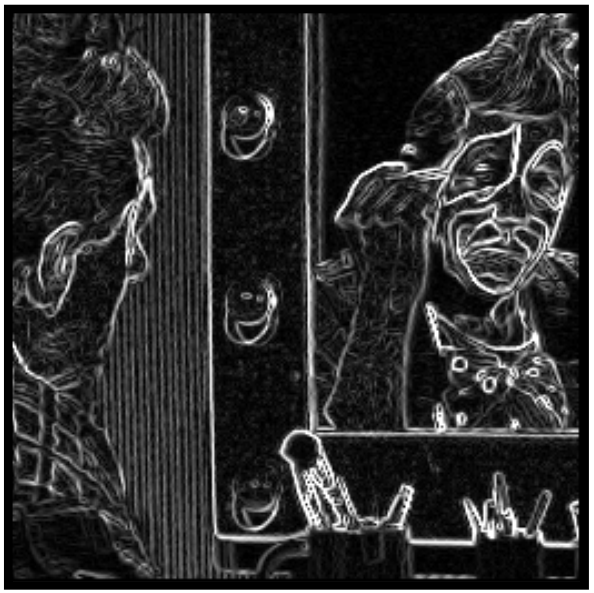
ảnh gốc



Sau khi áp dụng Sobel

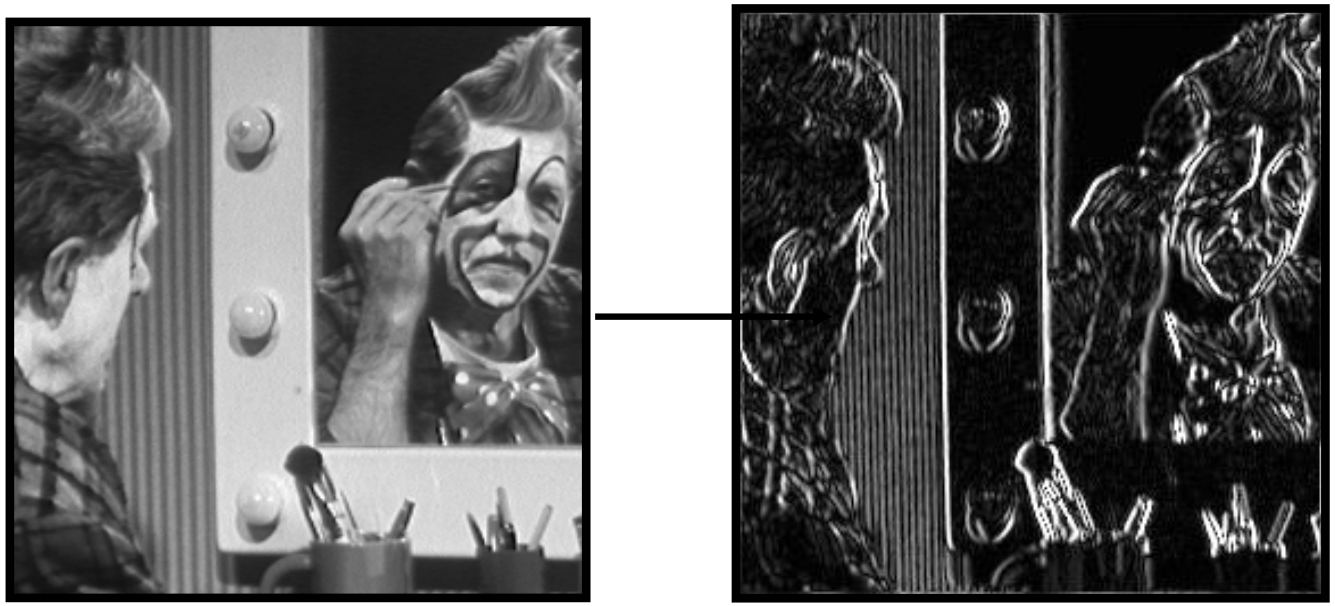
Ví dụ minh họa thuật toán Gradient

Ảnh gốc



Sau khi áp dụng Roberts

Ví dụ minh họa thuật toán Gradient



ảnh gốc

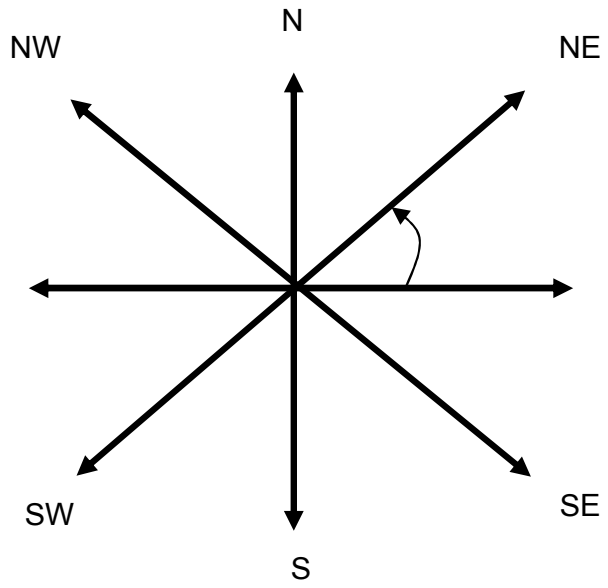
ảnh sau khi dùng toán tử Prewitt

- Tổng tất cả các hệ số trong mặt nạ bằng 0. Điều này nhằm làm cho đáp ứng tại những vùng cấp xám không thay đổi có giá trị bằng 0.

```
int i, j, M, N, l, k, tam1, tam2, r, K = 3, nguong = 200;
BYTE Y[1000][1000];
M = pDoc->biHeader.biHeight;
N = pDoc->biHeader.biWidth;
int Hm[3][3] = { { 1,0,-1 },{ 1,0,-1 },{ 1,0,-1 } };
int Hn[3][3] = { { 1,1,1 },{ 0,0,0 },{ -1,-1,-1 } };
r = (K - 1) / 2;
for (i = r; i<M - r; i++)
    for (j = r; j<N - r; j++)
    {
        tam1 = tam2 = 0;
        for (k = 0; k<K; k++)
            for (l = 0; l<K; l++)
            {
                tam1 += Hm[k][l] * pDoc->Data[i - k + r][j - l + r];
                tam2 += Hn[k][l] * pDoc->Data[i - k + r][j - l + r];
            }
        if ((abs(tam1) + abs(tam2)) >= nguong)
            Y[i][j] = 255;
        else
        {
            Y[i][j] = 0;
        }
    }
for (i = 0; i<M; i++)
    for (j = 0; j<N; j++)
        pDoc->Data[i][j] = Y[i][j];
Invalidate();
```

Toán tử la bàn

- Mỗi điểm ảnh có 8 lân cận, ta mong muốn 1 trong 8 hướng gây ra đột biến, đó chính là điểm biên.
- Kirsh đề xuất đo Gradient theo 8 hướng, mỗi hướng cách nhau 45^0. Ta gọi đây là toán tử Kirsh hay toán tử la bàn.



Toán tử la bàn

- Để tìm đạo hàm của ảnh người ta lấy max của 8 đạo hàm:

$$G(m,n)=\max_{k=0}^{7}\{|G_k(m,n)|\} \quad \text{voi } G_k(m,n)=H_k*X$$

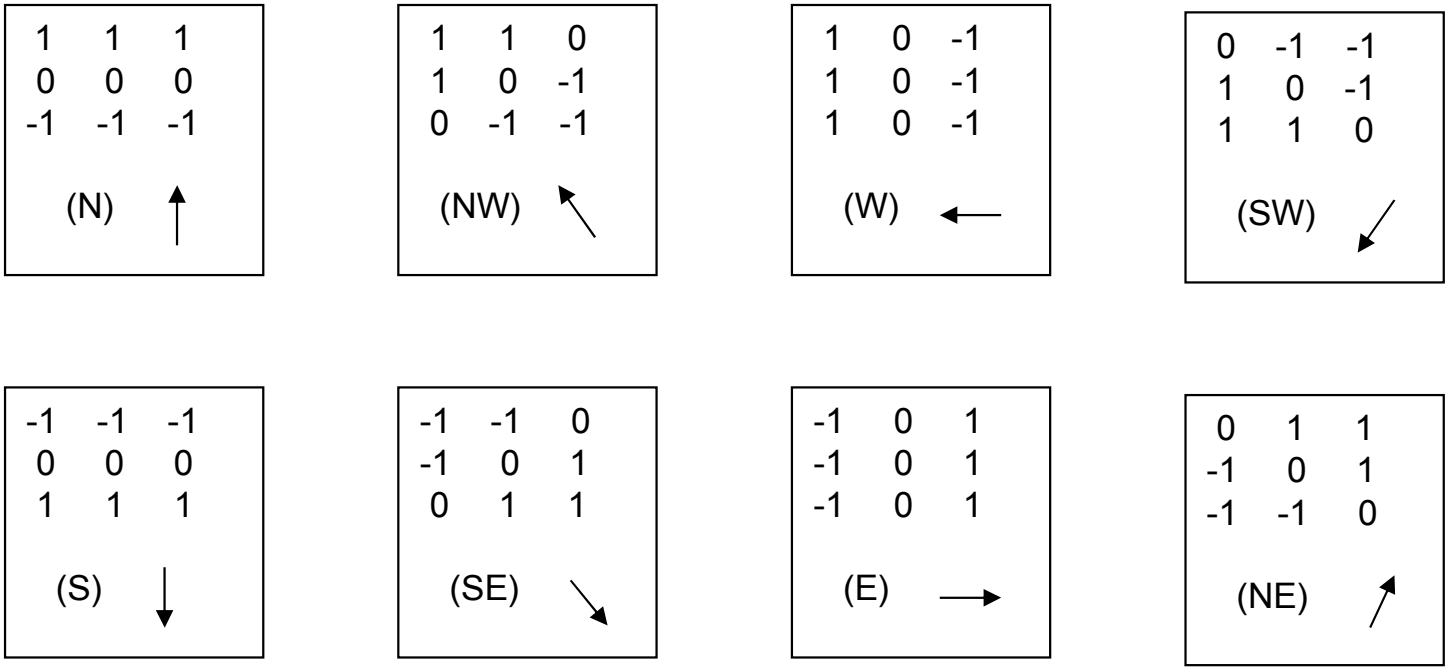
- Để tính đạo hàm theo hướng kπ/4 ta thực hiện:

$$X'_{\frac{k\pi}{4}}(m,n)=X(m,n)*H_{\frac{k\pi}{4}}$$

- Toán tử la bàn gồm 8 mặt nạ.
  - Mặt nạ theo hướng 0^0 (k=0) trùng với H_n của toán tử Gradient
  - Các mặt nạ còn lại nhận được bằng cách quay H_0 các góc π/4

Toán tử la bàn

- Giả sử ta có H_0 ≡ toán tử Prewitt ta có toán tử la bàn như sau:



Kỹ thuật phát hiện biên Laplace

- Các phương pháp sử dụng gradient đạo hàm bậc nhất làm việc tốt khi độ sáng thay đổi rõ nét.
- Khi mức xám thay đổi chậm, miền chuyển tiếp trải rộng, phương pháp sử dụng đạo hàm bậc hai lại cho kết quả tốt hơn, đạo hàm bậc hai ta sử dụng toán tử laplace.
- Ta có:

$$\left|X''(m,n)\right| = \left|X''_m(m,n)\right| + \left|X''_n(m,n)\right|$$

Kỹ thuật phát hiện biên Laplace

- Trong không gian rời rạc, đạo hàm bậc hai thực chất là phép nhân chập
- Ta có công thức tính:

$$X''(m,n) = X(m,n) * H(k,l)$$

  - Với H(k,l): toán tử Laplace
- X(m,n) là biên khi và chỉ khi $X''(m,n) \leq \theta$

Một số mặt nạ Laplace

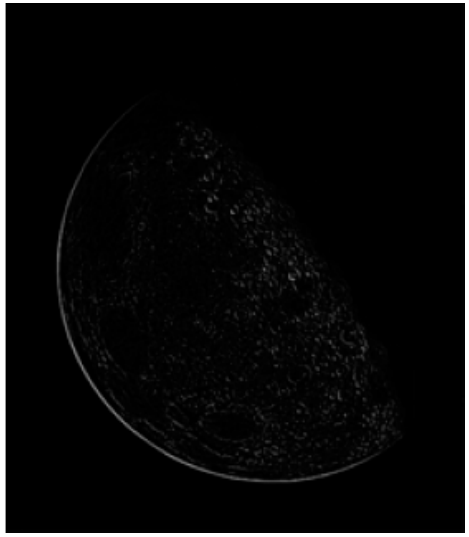
- ;

$$1.\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \qquad 2.\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

;

$$3.\begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

Kết quả minh họa



Nhận xét

- Kỹ thuật sử dụng đạo hàm bậc hai thường cho ta đường biên mảnh, và ta gọi là đường biên độ rộng 1 pixel.
- Tuy nhiên, kỹ thuật đạo hàm bậc hai rất nhạy cảm với nhiễu vì đạo hàm bậc hai thường không ổn định.

Kỹ thuật phát hiện biên Laplacian

Các bước:

- Tải hình ảnh
- Áp dụng bộ lọc Gaussian loại bỏ nhiễu.
  - cv2.GaussianBlur(img, (3, 3), 0)
- Chuyển đổi hình ảnh gốc sang ảnh xám
  - cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
- Áp dụng toán tử Laplacian cho ảnh xám
  - cv2.Laplacian(img_gray, ddepth, ksize=kernel_size)
- Lấy giá trị tuyệt đối và chuyển đổi kết quả thành 8-bit
  - cv2.convertScaleAbs(result)
- Hiển thị kết quả

Kỹ thuật phát hiện biên Canny

- Thuật toán Canny tương đối tốt, đưa ra đường biên mảnh, phát hiện chính xác điểm biên với điểm nhiễu
- Câu lệnh: result = cv2.Canny(image, edges, lower_threshold, upper_threshold)
- low_threshold
- **Các bước:**

- Loại bỏ nhiễu bằng bộ lọc Gauss.
- Tách biên bằng hàm Canny
- Canny sử dụng 2 ngưỡng: ngưỡng trên và ngưỡng dưới
  - Nếu gradient pixel > ngưỡng trên, pixel thuộc cạnh biên
  - Nếu giá trị gradient pixel < ngưỡng dưới, loại bỏ pixel.
  - Nếu gradient pixel nằm giữa hai ngưỡng, thì nó sẽ chỉ được chấp nhận nếu nó được kết nối với một pixel cao hơn ngưỡng trên.
  - Canny đề xuất tỷ lệ trên:dưới là 2:1 và 3:1 (Ví dụ: upper=4,lower=2 tỉ lệ 2:1)

- Tìm hiểu phân vùng ảnh (segmentation), contour
  - Tìm hiểu thuật toán
  - Nêu rõ các thành phần trong lời gọi hàm
  - Viết chương trình demo

Phân vùng ảnh

Contour

- Contour là “tập các điểm-liên-tục tạo thành một đường cong (curve) (boundary), và không có khoảng hở trong đường cong đó.
- Trong một contour, các điểm có cùng /gần xấu xỉ một giá trị màu, hoặc cùng mật độ.
- Contour là một công cụ hữu ích được dùng để phân tích hình dạng đối tượng, phát hiện đối tượng và nhận dạng đối tượng”.

Contour của ảnh

- Hàm tìm Contour của ảnh:

contours, hierarchy = cv2.findContours(image, mode, method)

Trong đó:

- **contours**: Danh sách các contour có trong bức ảnh nhị phân. Mỗi một contour được lưu trữ dưới dạng vector các điểm
- **hierarchy**: Danh sách các vector, chứa mối quan hệ giữa các contour.
- **image**: ảnh nhị phân 8 bit (thu được bằng cách sử dụng các hàm threshold, adaptiveThreshold, Canny trên ảnh gốc)
- **mode**: các kiểu trích xuất contour, có các dạng sau: **RETR_TREE**, RETR_EXTERNAL, RETR_LIST, RETR_CCOMP, RETR_FLOODFILL.
- **method**: các phương pháp xấp xỉ contour. Có các phương thức sau: **CHAIN_APPROX_SIMPLE**, CHAIN_APPROX_NONE, CHAIN_APPROX_TC89_L1, CHAIN_APPROX_TC89_KCOS.

Contour của ảnh

- Hàm vẽ contour:
- drawContours(image, contours, contourIdx, color, thickness)
- Trong đó:
  - imgage: ảnh, có thể là ảnh grayscale hoặc ảnh màu.
  - contours: danh sách các contour, nếu để số âm: vẽ tất cả
  - contourIndex: chỉ định contour được vẽ, thông thường chúng ta để -1
  - color: Giá trị màu của contour chúng ta muốn vẽ, ở dạng BGR, vd vẽ contour màu xanh lá cây thì set là (0,255,0).
  - thickness : độ dày của đường contour cần vẽ, giá trị thickness là số âm thì sẽ vẽ hình được tô màu

Ví dụ Contour của ảnh

```
import cv2
import numpy as np
import random as rng #Để tạo số ngẫu nhiên, lấy màu vẽ contour
img = cv2.imread('colorfull_ballon.jpg')

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_gray = cv2.blur(img_gray, (3,3))

lower_threshold = 90
# Phát hiện biên, đổi sang ảnh nhị phân
canny_output = cv2.Canny(img_gray, lower_threshold, lower_threshold  * 2)
# Find contours
contours, _ = cv2.findContours(canny_output, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
print(contours)
# Draw contours
img_out = np.zeros((canny_output.shape[0], canny_output.shape[1], 3), dtype=np.uint8) #Tạo ảnh đen để vẽ contour
for i in range(len(contours)):
    color = (rng.randint(0,256), rng.randint(0,256), rng.randint(0,256))
    cv2.drawContours(img_out, contours, i, color, 2)

cv2.imshow('Original',img)
cv2.imshow('Contours', img_out)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Thank you!

TRƯỜNG ĐẠI HỌC VINH
VINH UNIVERSITY

Nơi tạo dựng tương lai cho tuổi trẻ