THỰC HÀNH XỬ LÝ ẢNH SỐ - BÀI 4

Nội dung chính của bài thực hành thứ tư sẽ tập trung vào các thuật toán dò biên gián tiếp. Các thuật toán này dựa vào khái niệm liên thông, bao gồm *liên thông* – 4 và *liên thông* – 8 (xem lại định nghĩa lý thuyết).

Để áp dụng việc dò biên gián tiếp, ta thường chuyển ảnh về nhị phân (0-1), sau đó dò trên ảnh nhị phân đã thu được. Để chuyển về ảnh nhị phân, ta có thể phân ngưỡng tự động (OSTU).

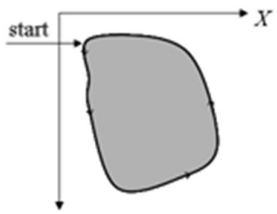
Các thuật toán dò biên gián tiếp dựa trên chu tuyến (đường khép kín bao xung quanh đối tượng), và vì vậy cần xác định giá trị nào thuộc về đối tượng, và giá trị nào là của nền ảnh.

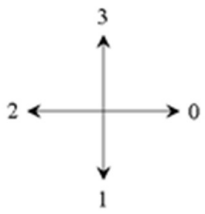
THUẬT TOÁN

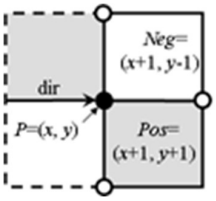
1. TÌM CHU TUYẾN:

Xác định cặp điểm nền – đối tượng để làm điểm xuất phát. Lặp lại quá trình tìm cặp điểm (nền-đối tượng điểm theo cho đến khi gặp lại điểm xuất phát.

Mã giả:








```
P.X=Start.X; P.Y=Start.Y; direction=1; // P is the current point
do
{ Pos=P+Positive[direction];          // Pos is the positive pixel
  Neg=P+Negative[direction];          // Neg is the negative pixel
  if (image[Pos/2]==foreground)       // Pos/2 is in standard raster
    direction=(direction+1) MOD 4;    // positive turn
  else
    if (image[Neg/2]==background)     // Neg/2 is in standard raster
      direction=(direction+3) MOD 4; // negative turn
  P=P+step[direction];                // a move in the new direction
} while( P.X!=Start.X || P.Y!=Start.Y);
```

a. Code xây dựng theo thuật toán:

Code đầy đủ có thể tham khảo tệp boundary_processing.py đính kèm.

Sửa đoạn code đọc ảnh để chạy thử với ảnh Example_01.png

b. Sử dụng thư viện OpenCV:

```
cv2.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
```

Ví dụ:

```
import numpy as np
import cv2 as cv
im = cv.imread('D:\\Teach_n_Train\\ImageProcessing\\code\\practice2\\Example_01.png')
imgray = cv.cvtColor(im, cv.COLOR_BGR2GRAY)
ret, thresh = cv.threshold(imgray, 127, 255, 0)
contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
```

```
cv.drawContours(imgray, contours, -1, (0,255,0), 3)
cv.drawContours(imgray, contours, 3, (0,255,0), 3)

cv.imshow('Test contours', imgray)
k = cv.waitKey(0)
if k == 27: # wait for ESC key to exit
    cv.destroyAllWindows()
```

2. THUẬT TOÁN TÌM XƯƠNG CỦA ĐỐI TƯỢNG

Xương: Cho một đối tượng, nếu thu hẹp đối tượng tự 2 phía đối xứng thì đường còn lại trên đối tượng (khi bước thu hẹp từ 2 phía đã gặp nhau) là xương của đối tượng.

Từ định nghĩa có thể thấy có 2 cách tìm xương: Dựa vào việc làm mảnh ảnh và không dựa vào việc làm mảnh ảnh.

Bài tập thực hành: Dựa vào làm mảnh ảnh: hãy sử dụng các phép toán hình thái (đã học và đã được cung cấp code) để làm các đối tượng trên ảnh mảnh dần cho đến khi đối tượng chỉ còn một nét mảnh (mỗi điểm của đối tượng có nhiều nhất 2 điểm kề cũng thuộc đối tượng).

Sử dụng thư viện SciKit Image: from skimage.morphology import skeletonize

Sau đó sử dụng: skeleton = skeletonize(image)

Ví dụ:

```
from skimage.morphology import skeletonize
from skimage import data
import matplotlib.pyplot as plt
from skimage.util import invert

# Invert the horse image
image = invert(data.horse())

# perform skeletonization
skeleton = skeletonize(image)

# display results
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(8, 4),
                         sharex=True, sharey=True)

ax = axes.ravel()

ax[0].imshow(image, cmap=plt.cm.gray)
ax[0].axis('off')
ax[0].set_title('original', fontsize=20)

ax[1].imshow(skeleton, cmap=plt.cm.gray)
ax[1].axis('off')
ax[1].set_title('skeleton', fontsize=20)

fig.tight_layout()
plt.show()
```

Bản chất của phương thức trên là sử dụng phép toán hình thái để làm mỏng đối tượng ảnh (Thuật toán Zang).

Tham khảo thêm phương thức dựa trên thuật toán Lee như sau:

```
import matplotlib.pyplot as plt
from skimage.morphology import skeletonize

blobs = data.binary_blobs(200, blob_size_fraction=.2,
                          volume_fraction=.35, seed=1)

skeleton = skeletonize(blobs)
skeleton_lee = skeletonize(blobs, method='lee')

fig, axes = plt.subplots(1, 3, figsize=(8, 4), sharex=True, sharey=True)
ax = axes.ravel()

ax[0].imshow(blobs, cmap=plt.cm.gray)
ax[0].set_title('original')
ax[0].axis('off')

ax[1].imshow(skeleton, cmap=plt.cm.gray)
ax[1].set_title('skeletonize')
ax[1].axis('off')

ax[2].imshow(skeleton_lee, cmap=plt.cm.gray)
ax[2].set_title('skeletonize (Lee 94)')
ax[2].axis('off')

fig.tight_layout()
plt.show()
```

Và phương thức dựa trên thuật toán trục trung vị (không làm mảnh ảnh).

```
from skimage.morphology import medial_axis, skeletonize


# Generate the data
blobs = data.binary_blobs(200, blob_size_fraction=.2,
                          volume_fraction=.35, seed=1)


# Compute the medial axis (skeleton) and the distance transform
skel, distance = medial_axis(blobs, return_distance=True)


# Compare with other skeletonization algorithms
skeleton = skeletonize(blobs)
skeleton_lee = skeletonize(blobs, method='lee')


# Distance to the background for pixels of the skeleton
dist_on_skel = distance * skel


fig, axes = plt.subplots(2, 2, figsize=(8, 8), sharex=True, sharey=True)
ax = axes.ravel()


ax[0].imshow(blobs, cmap=plt.cm.gray)
ax[0].set_title('original')
ax[0].axis('off')
```

```
ax[1].imshow(dist_on_skel, cmap='magma')
ax[1].contour(blobs, [0.5], colors='w')
ax[1].set_title('medial_axis')
ax[1].axis('off')


ax[2].imshow(skeleton, cmap=plt.cm.gray)
ax[2].set_title('skeletonize')
ax[2].axis('off')


ax[3].imshow(skeleton_lee, cmap=plt.cm.gray)
ax[3].set_title("skeletonize (Lee 94)")
ax[3].axis('off')


fig.tight_layout()
plt.show()
```