画像処理 最終課題

情報科学部ディジタルメディア学科

3年Dクラス 15K1118 白石祐大

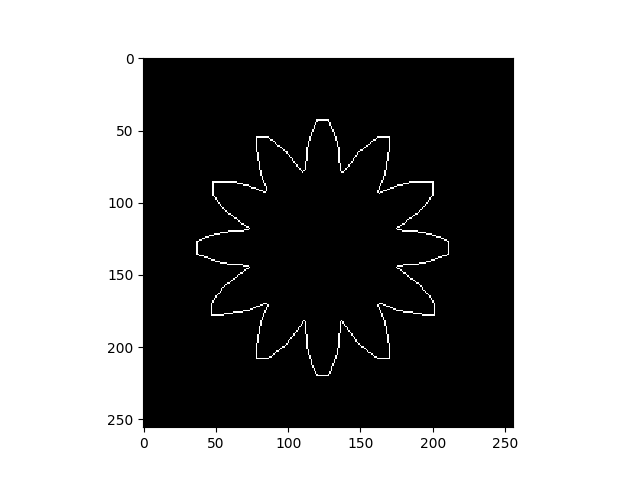
1. **課題**

今回の課題は「チェインコードが与えられたときに、それによって囲まれる領域内の画素数を求める(図形が与えられるわけではない)。画素数にはチェインコード自体の画素も含む」というものである。

1. **実装・結果**

最初に思いついた方法は、まず任意の大きさの値が0の行列を用意し任意の点からchain codeを当てはめていって、そのあてはめた画素値を255とする。

次に、行列を左上から右下の順で見ていき、chain codeをあてはめた画素が出てきたら次に出てきたchai codeまでの画素から最初に出てきた画素の座標の差分を取り、それを全行に対して行えば求められると考えた。しかし、この方法ではchain codeの値が横方向に動くものだったときや、同じ行に複数のchain codeを当てはめた画素が出てくるときなどに対応できない。そこでインターネットを用いて調べたところ、chain code内の画素数を調べる方法が見つかったので、それを用いて実装した。以下に結果を示す。





1. **考察**

今回用いた方法では、chain code内に別のchain codeが存在しても、そのchain codeがわかっていれば同じ全体からそのchain code内の画素数を引けば簡単に求められる。

Chain code内の画素数（面積）を求めるのはつまり、外積を求めることと同意ではないかと今回の課題を進めていく上で考えた。Chain codeの各値に対応するx, y方向の移動をそれぞれ2つのベクトルに格納し、その2つベクトルの外積を求めることでchain code内の画素数は求められる。今回は自分で外積のプログラムを作成し試しに行ってみたがうまく行かず、結局ネットに挙げっているコードを参考にした。今後機会があったら自分で最後まで考え実装してみたいと考えた。

1. **ソースコード**

"""This is a test program."""

import numpy as np

import matplotlib.pyplot as plt

class Image:

def \_\_init\_\_(self):

self.img = np.zeros((256, 256))

self.x, self.y = 120, 220

self.chainCode = ''

with open('flower\_chain\_code.txt', encoding='utf-8') as f:

for line in f:

self.chainCode = line

self.chainCode = list(self.chainCode)

*# self.chainCode = ['0', '0', '0', '2', '2', '2', '4', '4', '4', '6', '6', '6']*

self.area = 0

self.Ref\_CIX = np.array([[1, 1, 0, -1, -1, -1, 0, 1]])

self.Ref\_CIY = np.array([[0, 1, 1, 1, 0, -1, -1, -1]])

def chain\_code(self):

self.img[self.y][self.x] = 255

for d in self.chainCode:

if d == '0':

self.x = self.x + 1

self.img[self.y][self.x] = 255

elif d == '1':

self.x = self.x + 1

self.y = self.y + 1

self.img[self.y][self.x] = 255

elif d == '2':

self.y = self.y + 1

self.img[self.y][self.x] = 255

elif d == '3':

self.x = self.x - 1

self.y = self.y + 1

self.img[self.y][self.x] = 255

elif d == '4':

self.x = self.x - 1

self.img[self.y][self.x] = 255

elif d == '5':

self.x = self.x - 1

self.y = self.y - 1

self.img[self.y][self.x] = 255

elif d == '6':

self.y = self.y - 1

self.img[self.y][self.x] = 255

elif d == '7':

self.x = self.x + 1

self.y = self.y - 1

self.img[self.y][self.x] = 255

def show(self):

plt.imshow(self.img, 'gray')

def calculate\_area(self):

size = (1, len(self.chainCode))

CIX = np.zeros(size)

CIY = np.zeros(size)

YI = np.zeros(size)

A = np.zeros(size)

for i in range(size[1]):

CIX[0][i] = self.Ref\_CIX[0][int(self.chainCode[i])]

CIY[0][i] = self.Ref\_CIY[0][int(self.chainCode[i])]

for i in range(size[1]):

if i != 0:

YI[0][i] = YI[0][i - 1] + CIY[0][i]

A[0][i] = CIX[0][i] \* (YI[0][i - 1] + CIY[0][i] / 2)

else:

YI[0][i] = 0 + CIY[0][i]

A[0][i] = CIX[0][i] \* (0 + CIY[0][i] / 2)

print('chain code内の画素数: {}'.format(abs(sum(A[0]))))

self.area = abs(sum(A[0])) + len(self.chainCode)

print('chain codeも含めた画素数: {}'.format(self.area))

img = Image()

img.chain\_code()

img.calculate\_area()

img.show()

plt.show()

1. **参考**

<https://jp.mathworks.com/matlabcentral/fileexchange/37866-area-calculation-of-8-direction-chain-code?s_tid=gn_loc_drop>

<https://stackoverflow.com/questions/16342200/how-can-i-calculate-the-area-of-an-object-by-using-its-contour-chain-code>