华 中 科 技 大 学

图 像 处 理 作 业

图像分割

院　　系: 人工智能与自动化学院

专 业 班 级: 自 动 化　1903　班

学 生 姓 名: 李　　子　　奥

学 生 学 号: U 2 0 1 9 1 4 6 2 9

指 导 教 师: 谭　　山

2022 年 7 月 9 日

目　　录

1 图像分割原理

1.1 Otsu 阈值法

Otsu 阈值法是一种自适应的阈值确定的方法，它根据图像的灰度数，将图像分成两个部分。按灰度级分成 2 个部分，使得两个部分之间的灰度值差异最大，即最大化类间方差。

假设 $\{0, 1, 2, \ldots, L-1\}$ 表示图像中的 L 个灰度等级[1]，每个灰度等级的像素占总像素的比例为 p_i，则 p_i 满足关系式：

$$\sum_{i=0}^{L-1} p_i = 1 \tag{1.1}$$

假设我们选择了一阈值 $k, 0 < k < L-1$ 将图像分割成了两部分：C_1 和 C_2，则某一像素属于 C_1 类的概率为：

$$P_1 = P(C_1) = \sum_{i=0}^{k} p_i \tag{1.2}$$

相应的，某一像素属于 C_2 类的概率为：

$$P_2 = P(C_2) = 1 - P_1 = \sum_{i=k+1}^{L-1} p_i \tag{1.3}$$

C_1 类中的像素的平均灰度值为：

$$\begin{aligned} m_1(k) &= \sum_{i=0}^{k} iP(i \mid C_1) \\ &= \sum_{i=0}^{k} iP(C_1 \mid i)\, P(i)/P(C_1) \\ &= \frac{1}{P(C_1)} \sum_{i=0}^{k} iP(i) = \frac{1}{P_1} \sum_{i=0}^{k} ip_i \end{aligned} \tag{1.4}$$

C_2 类中的像素的平均灰度值为：

$$m_2(k) = \frac{1}{P(C_2)} \sum_{i=k+1}^{L-1} ip_i = \frac{1}{P_2} \sum_{i=k+1}^{L-1} ip_i \tag{1.5}$$

[1]在本实验中，$L = 256$

整张图像的平均灰度值为:

$$m_G = \sum_{i=0}^{L-1} ip_i \tag{1.6}$$

用 σ_B^2 表示 C_1 类与 C_2 类的类间方差，则 σ_B^2 的定义为:

$$\sigma_B^2 = P_1 (m_1 - m_G)^2 + P_2 (m_2 - m_G)^2 \tag{1.7}$$

通过枚举不同的阈值 k，找到使类间方差 σ_B^2 最大的阈值 k^*。k^* 将图像划分成的两部分就是 Ostu 阈值法的划分结果。

1.2 区域增长法

区域增长法是一种有效的图像分割方法，其基本思想如下:

1. 指定一个或多个点作为种子点，将所有种子点加入集合 C_1 中
2. 判断种子点邻域内的点和种子点是否属于同一个物体。若是，将该点加入集合 C_1 中
3. 重复操作 2，直至都没有新的点可以被加入至集合 C_1。此时图像被分为了两个部分: C_1 与 $C_2 = \bar{C_1}$

从区域增长法的算法流程可以看出，区域增长法的分割结果非常依赖于:

- 判断某一点是否应被包含在区域中的标准
- 判断某一点是否为一已知种子点邻域的标准

在我的实验中, 判断某一点是否为一已知种子点邻域的标准为 4 邻域判别: 若某一点在一已知点的正上方、正下方、正左方或正右方，且两点间距离为 1，则该点位于这一一直点的邻域内，如图1.1所示。

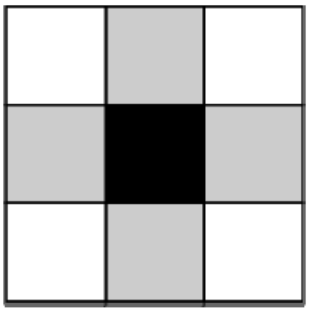

图 1.1 4 邻域判别标准

假设某点 x 位于一已知种子点 x_0 的邻域内，则当他们的像素值满足下式时，点 x 被划分到 x_0 的集合内。

$$I(x) \in [I(x_0) - c \cdot \sigma, I(x_0) + c \cdot \sigma] \tag{1.8}$$

其中 σ 为整张图片的标准差，c 是一需手动设定的超参数。

2 图像分割实验

2.1 参数选择

使用 Otsu 法进行图像分割不需要选择超参数，但是使用区域增长法进行图像分割需要确定两个超参数：种子点个数和判断某一点是否应被包含在区域内的标准 c。表2.1为区域增长法参数选择结果。

图像名	1	2	3	4	5	6	b	lena	objs
种子点数	1	1	4	1	1	1	2	8	2
c	0.2	0.2	0.1	0.1	0.2	0.4	0.3	0.15	0.2

表 2.1 区域增长法图像分割参数选择

2.2 图像分割结果

在下文中，最左侧一列为原始图像，中间一列为使用 Ostu 法进行图像分割的结果，最右侧一列为使用区域增长法进行图像分割的结果。其中图像分割的边缘用红色实线标识；种子点用绿色叉形标识。

图 2.1 图像 1

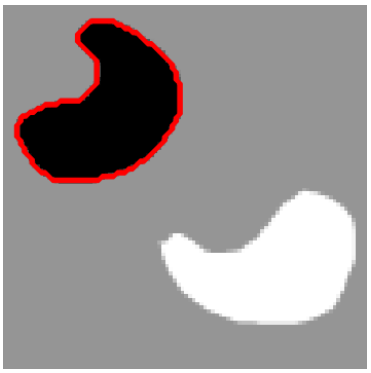

图 2.2 图像 1 Ostu 法

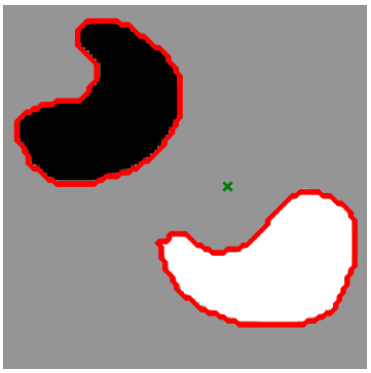

图 2.3 图像 1 区域增长法

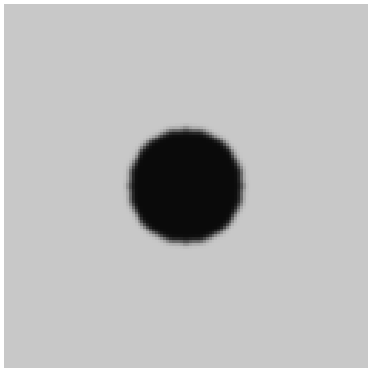

图 2.4 图像 2

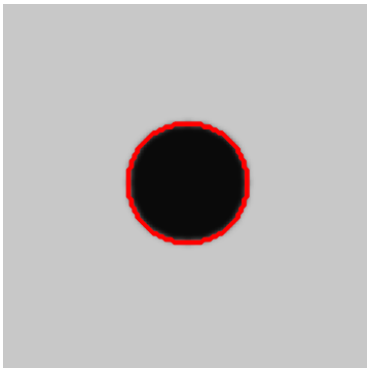

图 2.5 图像 2 Ostu 法

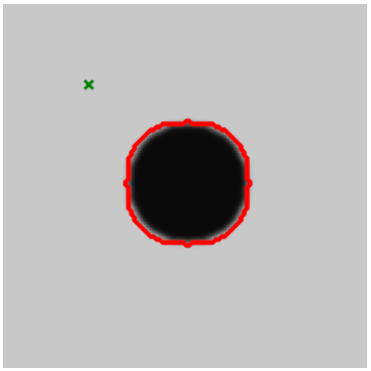

图 2.6 图像 2 区域增长法

图 2.7 图像 3

图 2.8 图像 3 Ostu 法

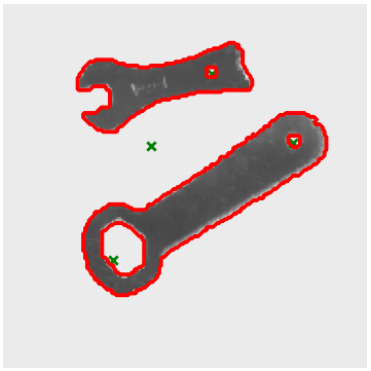

图 2.9 图像 3 区域增长法

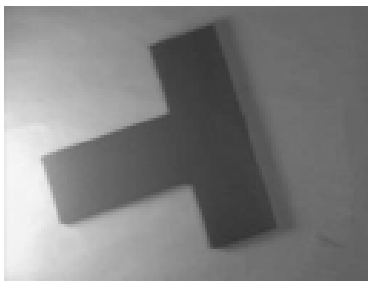

图 2.10 图像 4

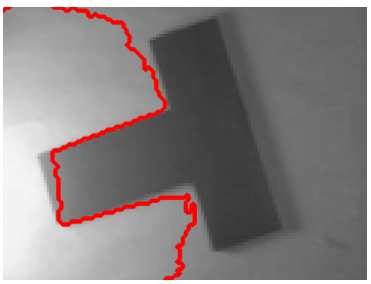

图 2.11 图像 4 Ostu 法

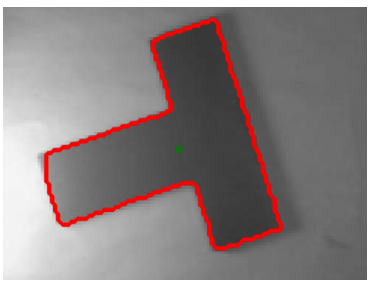

图 2.12 图像 4 区域增长法

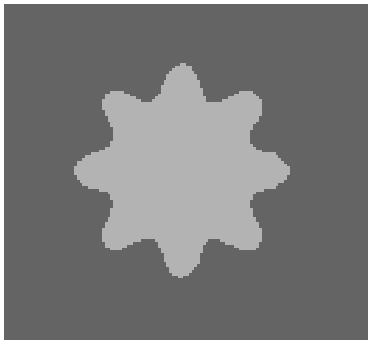

图 2.13 图像 5

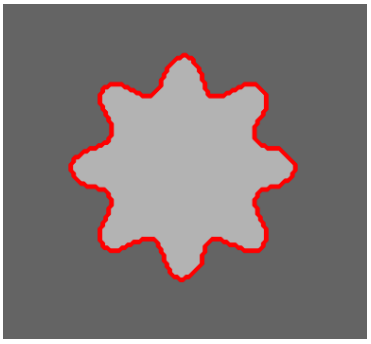

图 2.14 图像 5 Ostu 法

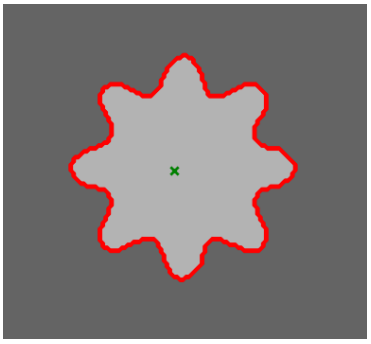

图 2.15 图像 5 区域增长法

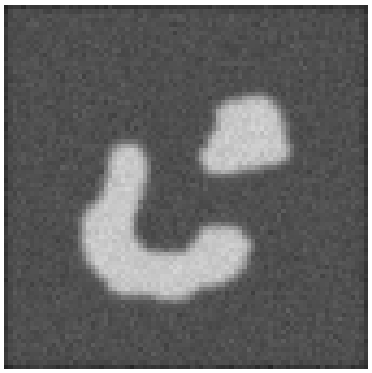

图 2.16 图像 6

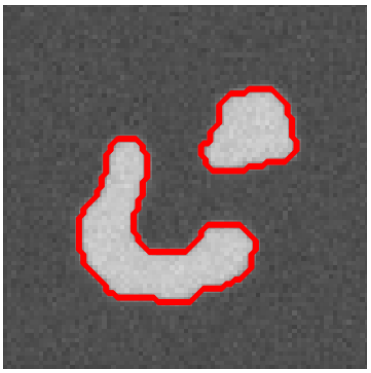

图 2.17 图像 6 Ostu 法

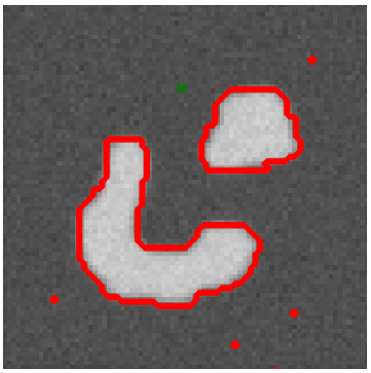

图 2.18 图像 6 区域增长法

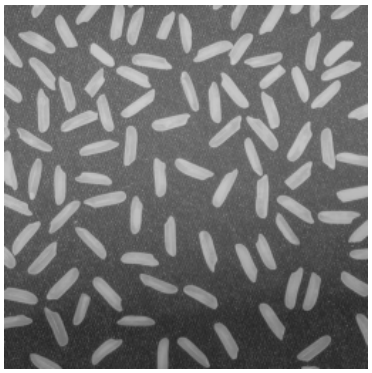

图 2.19 图像 b

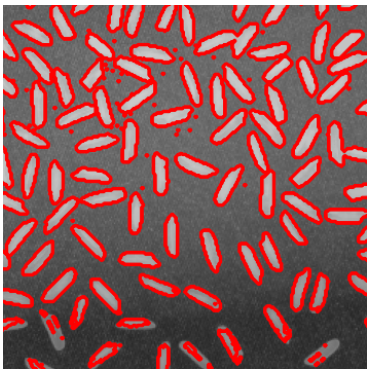

图 2.20 图像 b Ostu 法

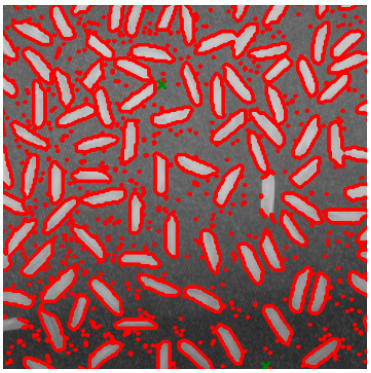

图 2.21 图像 b 区域增长法

图 2.22 图像 lena

图 2.23 图像 lena Ostu 法

图 2.24 图像 lena 区域增长法

图 2.25 图像 objs

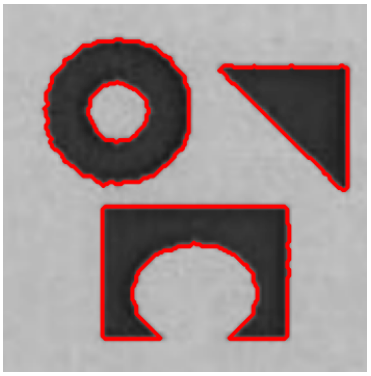

图 2.26 图像 objs Ostu 法

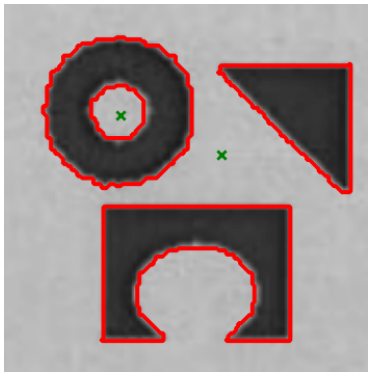

图 2.27 图像 objs 区域增长法

2.3 纯色填充实验结果

对于部分颜色复杂的图像，将分割边缘标识不利于观察分割的结果（如图像 lena），于是我将一类划分结果变为同一种颜色[1]，而维持另一类划分结果的颜色不变，这样能够更加方便观察复杂的图像的分割结果。

图 2.28 图像 1

图 2.29 图像 1 Ostu 法

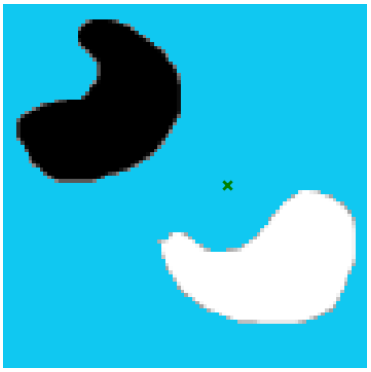

图 2.30 图像 1 区域增长法

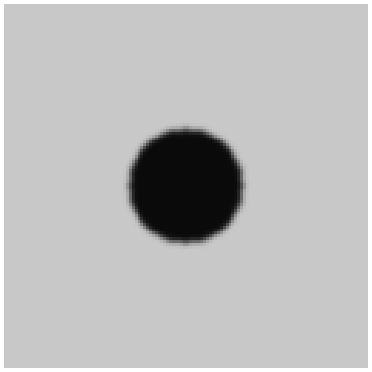

图 2.31 图像 2

图 2.32 图像 2 Ostu 法

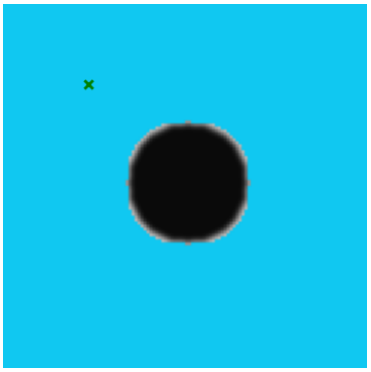

图 2.33 图像 2 区域增长法

图 2.34 图像 3

图 2.35 图像 3 Ostu 法

图 2.36 图像 3 区域增长法

[1]在我的实验中为蓝色

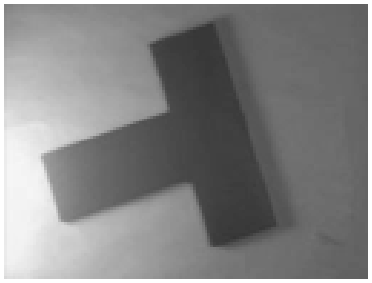

图 2.37 图像 4

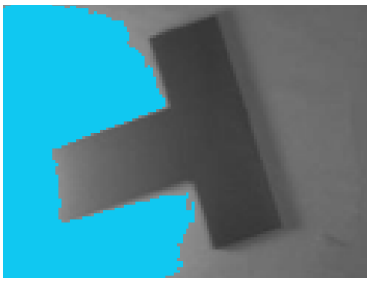

图 2.38 图像 4 Ostu 法

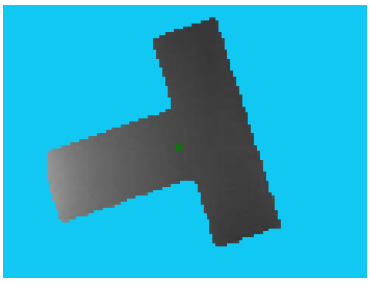

图 2.39 图像 4 区域增长法

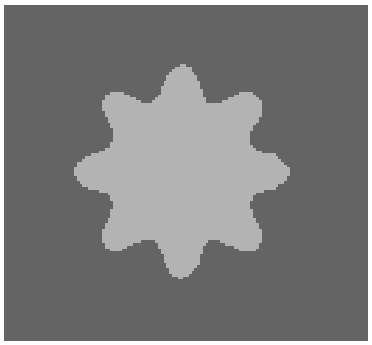

图 2.40 图像 5

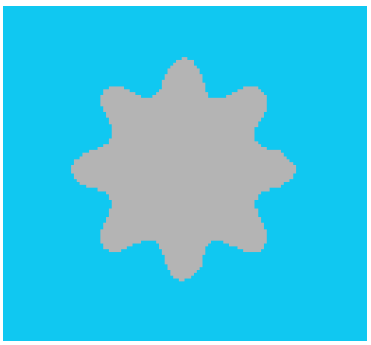

图 2.41 图像 5 Ostu 法

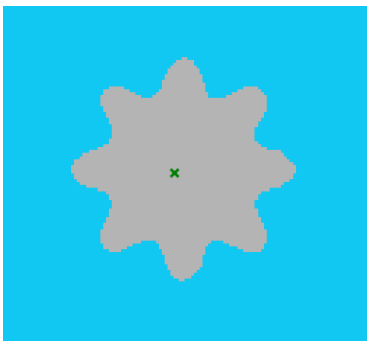

图 2.42 图像 5 区域增长法

图 2.43 图像 6

图 2.44 图像 6 Ostu 法

图 2.45 图像 6 区域增长法

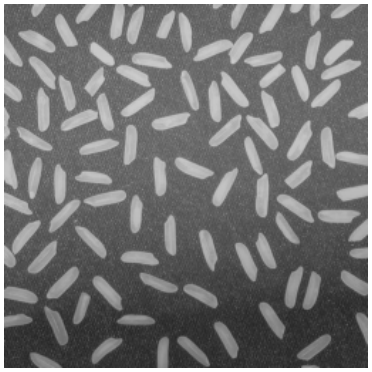

图 2.46 图像 b

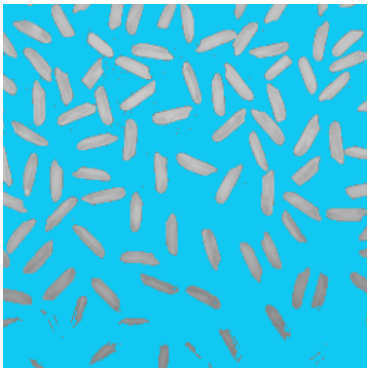

图 2.47 图像 b Ostu 法

图 2.48 图像 b 区域增长法

图 2.49 图像 lena

图 2.50 图像 lena Ostu 法

图 2.51 图像 lena 区域增长法

图 2.52 图像 objs

图 2.53 图像 objs Ostu 法

图 2.54 图像 objs 区域增长法

2.4 实验结论

对于颜色单一的图像，使用 Otsu 法进行图像分割更好，因为 Otsu 法既不用手动设定超参数与种子点，也能够取得比区域增长法更好的效果（见图像 6 与图像 b）。然而，由于 Otsu 法在整张图像上使用相同的阈值作为分割指标，Otsu 法

对于有多种颜色的图像可能会效果较差（如图像 1）。

对于颜色复杂（如多种颜色、渐变颜色等）的图像，使用区域增长法更好，因为区域增长法通过设定种子点，能够使不同颜色的部分都被划分到同一个区域中（如图像 1、图像 4 与图像 lena）。

附录 A　实验代码

```
from copy import deepcopy
import os
from matplotlib.image import imsave
import matplotlib.pyplot as plt
import numpy as np
import cv2

OutputDir = "output"
ALL_FILES = []
CURRENT_FILE = ""
IF_SHOW = False
Suffix = ".png"
SegmentMethod = ['otsu', 'growing'][1]

growing_config = {
    '1': {'seeds': [[50, 60]], 'c': 0.2},
    '2': {'seeds': [[30, 30]], 'c': 0.2},
    '3': {'seeds': [[80, 80], [140, 60], [42, 112], [78, 155]], 'c': 0.1},
    '4': {'seeds': [[50, 60]], 'c': 0.1},
    '5': {'seeds': [[60, 60]], 'c': 0.2},
    '6': {'seeds': [[20, 40]], 'c': 0.4},
    'b': {'seeds': [[60, 110], [250, 180]], 'c': 0.3},
    'lena': {'seeds': [[100, 450], [250, 460], [50, 100], [150, 50], [200,
        10], [300, 500], [20, 460], [50, 440]], 'c': 0.15},
    'objs': {'seeds': [[60, 60], [80, 110]], 'c': 0.2},
}

def show_pic(pic1: np.array, str1='', scale=True):
    global CURRENT_FILE
    pic1 = pic1.squeeze()

    filename = os.path.join(OutputDir, str1+'_'+CURRENT_FILE+Suffix)

    if scale:
        imsave(filename, pic1, cmap=plt.get_cmap('gray'),vmin=0,vmax=255)
    else:
        imsave(filename, pic1, cmap=plt.get_cmap('gray'))

    if IF_SHOW:
        if scale:
            plt.imshow(pic1, cmap=plt.get_cmap('gray'),vmin=0,vmax=255)
        else:
```

```
42              plt.imshow(pic1, cmap=plt.get_cmap('gray'))
43          plt.xticks([])
44          plt.yticks([])
45          plt.title(CURRENT_FILE)
46          plt.show()
47
48  def show_with_contour(pic: np.array, contour: np.array, seeds: np.array = None
        , str1=''):
49      global CURRENT_FILE
50      pic = pic.squeeze()
51
52      filename = os.path.join(OutputDir, 'c_'+str1+'_'+CURRENT_FILE+Suffix)
53
54      plt.imshow(pic, cmap=plt.get_cmap('gray'),vmin=0,vmax=255)
55      plt.contour(contour, colors='red')
56      if not seeds is None:
57          plt.scatter(seeds[:, 1], seeds[:, 0], color='green', marker='x',
                linewidths=2)
58      plt.xticks([])
59      plt.yticks([])
60      plt.savefig(filename, bbox_inches='tight', pad_inches=-0.1)
61      plt.close()
62
63      filename = os.path.join(OutputDir, 'cf_'+str1+'_'+CURRENT_FILE+Suffix)
64
65      pic = cv2.cvtColor(pic, cv2.COLOR_GRAY2RGB)
66      contour = np.repeat(contour[:,:,np.newaxis], 3, axis=-1)
67      pic = np.where(contour == 0, pic, np.array([16, 200, 241]))
68      plt.imshow(pic, vmin=0,vmax=255)
69      # plt.contour(contour, colors='red')
70
71      if not seeds is None:
72          plt.scatter(seeds[:, 1], seeds[:, 0], color='green', marker='x',
                linewidths=2)
73      plt.xticks([])
74      plt.yticks([])
75      plt.savefig(filename, bbox_inches='tight', pad_inches=-0.1)
76      plt.close()
77
78      if IF_SHOW:
79          plt.imshow(pic, cmap=plt.get_cmap('gray'),vmin=0,vmax=255)
80          plt.xticks([])
81          plt.yticks([])
82          plt.title(CURRENT_FILE)
83          plt.show()
84
85
86  def otsu(pic: np.array) -> np.array:
```

```
87      Ls = np.array(range(256))
88      cnt = np.bincount(pic.flatten(), minlength=256)
89      assert(sum(cnt) == pic.size)
90      p_cnt = cnt / pic.size
91
92      m_G = np.sum(Ls * p_cnt)
93
94      all_Sigma = np.zeros(256)
95      for k in range(1, 256):
96          L1, L2 = Ls[:k], Ls[k:]
97          p_cnt1, p_cnt2 = p_cnt[:k], p_cnt[k:]
98          P1, P2 = np.sum(p_cnt1), np.sum(p_cnt2)
99          if P1 == 0 or P2 == 0:
100             continue
101         m1 = np.sum(L1 * p_cnt1) / P1
102         m2 = np.sum(L2 * p_cnt2) / P2
103
104         sigma_B = P1 * ((m1 - m_G) ** 2) + P2 * ((m2 - m_G) ** 2)
105         all_Sigma[k] = sigma_B
106
107     k = all_Sigma.argmax()
108
109     new_pic = np.where(pic < k, 0, 1)
110     return new_pic
111
112
113 def growing(pic: np.array, seeds: np.array, c=0.2) -> np.array:
114     std = np.std(pic.flatten())
115     d = [[-1, 0], [0, 1], [1, 0], [0, -1]]
116     H, W = pic.shape
117
118     ret_pic = np.zeros_like(pic)
119     st = np.zeros_like(pic)
120     for seed in seeds:
121         st[seed[0], seed[1]] = 1
122     while len(seeds) > 0:
123         h, w = seeds[0]
124         ret_pic[h, w] = 1
125         seeds = seeds[1:]
126         intensity = pic[h, w]
127         for dh, dw in d:
128             nh, nw = h + dh, w + dw
129             if not ( nh >= 0 and nw >= 0 and nh < H and nw < W ):
130                 continue
131             if not st[nh, nw] == 0:
132                 continue
133             n_intensity = pic[nh, nw]
134             # print ('\t[{}, {}] = {}'.format(nh, nw, n_intensity))
```

```
135                 if np.abs(intensity - n_intensity) < c * std:
136                     st[nh, nw] = 1
137                     seeds = np.concatenate([seeds, np.array([[nh, nw]])], axis=0)
138         return ret_pic
139 
140 
141 def main():
142     global CURRENT_FILE
143 
144     _, _, ALL_FILES = list(os.walk("./data"))[0]
145     print('\033[1;32m', ALL_FILES, '\033[0m')
146 
147     for file in ALL_FILES:
148         CURRENT_FILE = os.path.splitext(os.path.basename(file))[0]
149         assert(CURRENT_FILE in growing_config)
150         print('\033[31m', 'Processing "' + CURRENT_FILE + '"', '\033[0m')
151 
152         pic = plt.imread(os.path.join("data", file))
153 
154         if SegmentMethod == 'otsu':
155             new_pic = otsu(pic)
156             show_with_contour(pic, new_pic, None, 'Otsu1')
157             show_with_contour(pic, 1-new_pic, None, 'Otsu2')
158         else:
159             seeds = np.array(growing_config[CURRENT_FILE]['seeds'])
160             c = growing_config[CURRENT_FILE]['c']
161             new_pic = growing(pic.astype(np.int32), seeds, c)
162 
163             show_with_contour(pic, new_pic, seeds, 'Growing1')
164             show_with_contour(pic, 1-new_pic, seeds, 'Growing2')
165 
166 if __name__ == "__main__":
167     main()
```