# 实验四 基于分水岭算法的图像分割方法

## 一、实验目的

用 OpenCV 编写一个基于分水岭算法的图像分割程序能对自然图像进行分割，强化和巩固学生对图像分割知识的掌握和灵活应用。

## 二、实验内容

（1）利用 opencv-python 读取灰度图像，并进行噪声消除；

（2）对图像进行形态学处理，区分背景区域和未知区域；

（3）使用距离变换获取确定的前景色，并对确定的确定前景图像进行标注；

（4）基于 opencv-python 实现分水岭算法对预处理的结果图像进行分割；

（5）要求给出过程中二值图像、背景区域、前景区域、未知区域、结果的过程效果图，并对给出的演示代码添加注释。

## 三、实验过程

1. 获取图像，并把原图像转成二值图。

对于图像处理的许多应用，颜色信息无助于我们识别重要的边缘或其他特征。有例外。如果在灰度图像中难以检测到色相的边缘（像素值的阶跃变化），或者如果我们需要识别已知色调的物体（绿色叶子前面的橙色水果），则颜色信息可能是有用。如果我们不需要颜色，那么我们可以认为它是噪音。此外，使用灰度图像进行图像处理可以简化问题，在灰度图像中，分水岭算法相当容易概念化，因为我们可以将两个空间维度和一个亮度维度想象为具有丘陵，山谷，集水盆地，山脊等的 3D 图像。“峰值亮度”仅仅是山峰在我们的灰度图像的三维可视化。有许多算法可以用直观的“物理”解释来帮助我们思考问题。在 RGB，HSI，Lab 和其他色彩空间中，这种可视化更加困难，因为标准人类大脑无法容易地将其视觉化。因此需要进行图像灰度化。

2. 进行膨胀操作，可以得到大部分都是背景的区域，即“ 确定背景 B ”

使用形态学的膨账操作能够将图像内的前景“ 膨胀放大 ”。当图像内的前景被放大后，背景就会被“ 压缩 ”，所以此时得到的背景信息一定小于实际背景的，那么此时的背景自然可以确定为原背景的一部分，为不包含前景的“确定背景”，简称 B。

3. 使用距离变换获取确定的前景色

当图像内的各个子图没有连接时，可以直接使用形态学的腐蚀操作确定前景对象，但是如果图像内的子图连接在一起时，就很难确定前景对象了。此时，借助于距离变换函数 cv2.distanceTransform()可以方便地将前景对象提取出来。

距离变换函数 cv2.distanceTransform（）计算二值图像内任意点到最近背景点的距离。一般情况下，该函数计算的是图像内非零值像素点到最近的零值像素点的距离，即计算二值图像中所以像素点距离其最近的值为 0 的像素点的距离。当然，如果有的像素点本身的值为 0，则这个距离也为 0。

距离变换函数 cv2.distanceTransform（）的计算结果反映了各个像素与背景（值为 0 的像素点）的距离关系。通常情况下：

- 如果前景对象的中心（质心）距离 值为 0 的像素点 距离较远，会得到一个较大的值
- 如果前景对象的边缘 距离 值为 0 的像素点 较近，会得到一个较小的值。

如果对上述计算结果进行阈值化，就可以得到图像内子图的中心、骨架等信息。

4. 计算未知区域 UN (UN = O - B - F）

图像中有了确定前景 F 和 确定背景 B，剩下区域的就是未知区域 UN 了。这部分区域正是分水岭算法要进一步明确的区域。

针对一幅图像 O，通过以下关系能够得到未知区域 UN:

未知区域 UN = 图像 O - 确定背景 B - 确定前景 F

对上述表达式进行整理，可以得到：

未知区域 UN =（图像 O - 确定背景 B）- 确定前景 F

上式中的 “ 图像 O - 确定背景 B ” ，可以通过对图像进行形态学的膨胀操作得到。

5. 利用函数 cv2.connectedComponents（）对原始图像进行标注

明确了确定前景后，就可以对确定的确定前景图像进行标注了。在 OpenCV 中，可以使用函数 cv2.connectedComponents（）进行标注，该函数会将背景标注为 0，将其他的对象使用从 1 开始的正整数标注。函数 cv2.connectedComponents（）的语法格式为：

```
retval, labels = cv2. connectedComponents (image)
```

参数解释：

image 为 8 位单通道的待标注图像；retval 为返回的标注的数量。

labels 为标注的结果图像。

6. 使用分水岭函数完成对图像的分割

完成上述处理后，就可以使用分水岭算法对预处理的结果图像进行分割。在 OpenCV 中，实现分水岭算法的函数是 cv2.watershed（），其语法格式为：

```
markers = cv2.watershed(image,markers)
```

参数解释：

image 是输入图像，必须是 8 位三通道的图像。在对图像使用 cv2.watershed（）函数处理之前，必须先用正数大致勾画出图像中的期望分割区域。每一个分割的区域会被标注为 1、2、3 等。对于尚未确定的区域，需要将它们标注为 0。我们可以将标注区域理解为进行分水岭算法分割的“种子”区域。

代码演示：

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r'D:/coin1.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
ishow = img.copy()
ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
kernel = np.ones((3,3), np.uint8)
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)

sure_bg = cv2.dilate(opening, kernel, iterations=3)

dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform, 0.7*dist_transform.max(),255,0)

sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)


ret, markers = cv2.connectedComponents(sure_fg)
markers = markers + 1
markers[unknown==255]=0

markers = cv2.watershed(img, markers)
img[markers == -1] = [255, 0, 0]
cv2.imshow('result',img)
plt.subplot(241)
plt.imshow(ishow)
plt.axis("off")
plt.subplot(122)
plt.imshow(img)
plt.axis('off')
plt.show()

cv2.waitKey()
cv2.destroyAllWindows()
```

结果参考：