医 学 影 像 分 析（实验一）

题目：实验一：基于 DICOM 图像文件解析、读取和可视化

学生姓名：王天也

学　　号：21S010051

所在学院：经济与管理学院

专业名称：管理科学与工程

2022 年 5 月

目　录

1 实验一，基于最大类间自适应阈值选择方法

1.1 实验目的

熟悉常见的 DICOM 格式，能显示医学图像，并能进行基本的医学图像处理操作（窗宽窗位变换、阈值分割）。

1.2 DICOM 医学图像

DICOM 是医学图像和相关信息的国际标准，它定义了满足临床需要的可用于数据交换的医学图像格式，被广泛用于放射、成像的诊疗诊断设备。也就是说，在医院放射科使用的设备上读取的影像基本都是 DICOM 格式，包括 CT、MRI、超声等。

DICOM 格式的图像，每一张都带有除像素（体素）信息之外的大量信息。其信息组成主要有以下几部分：

Patient：病人信息

Study：诊疗信息，如检查时间、检查部位、ID 等

Series：序列号、图像坐标信息等。图像坐标信息主要关注：

SliceThickness：层厚，即每张切片之间的间距

ImagePositionPatient：该张切片坐标原点（图像左上角，二维图中向下为 y 轴，向右为 x 轴）在空间中的坐标（x，y，z）

ImageOritentationPatient：六元组，当前切片的 x、y 轴与解剖学坐标系间的关系。此处涉及矢状面、冠状面、横断面

简单的记忆方式是：矢状面即一支箭射中人所形成的面，正中二分人体。冠状面可想象古代官帽，从头顶向地面的切面。横断面即腰斩平面。在下图中，红色切面为矢状面，紫色切面为冠状面，绿色切面为横断面。

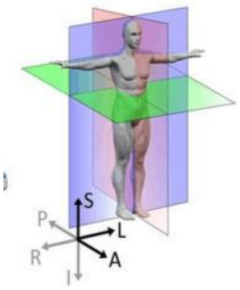

图 1.1　DICOM 图像

1. 3 实验原理

最大类间自适应阈值选择方法

对图像的灰度值进行划分，找到阈值是阈值两边的灰度值的方差达到最大。方差越大，代表通过阈值区分出来的两部分灰度值差别越大。该方法适用于分割的图像与周围环境的灰度值差别较大的情况。缺点是对图像的噪声比较敏感，并且只能进行单个图像的分割。如果需要分割的图像与环境大小相差较大，则很难分出来。

公式：

（1）背景占像素占比：

$$\omega_1 = \frac{N_1}{S}$$

其中 N_1 代表前景像素的个数，S 代表图像总的像素数。

(2) 前景像素占比

$$\omega_2 = \frac{N_2}{S}$$

(3) 背景像素的平均灰度值：

$$\nu_1 = \sum_{i=0}^{t} i * \frac{P_i}{N_1} \sum_{i=0}^{t} \frac{(i * P_i/S}{N_1/S} = \frac{\nu_{tl}}{\omega_2}$$

其中，P_i 表示的是背景中，灰度值为 i 的像素点的个数，ν_{tl} 表示的是背景像素相对于整个图像的灰度的数学期望值

（4）前景像素的平均灰度值

$$\nu_2 = \sum_{i=t+1}^{M} i * \frac{P_t}{N_2} = \sum_{i=t+1}^{M} \frac{i * P_i/S}{N_2/S} = \frac{\nu_{t2}}{\omega_2} = \frac{\nu - \nu_{t1}}{\omega_2}$$

其中，ν_{t2} 表示的是前景像素相对于整个图像的灰度到的数字期望值。

（5）0-M 灰度范围内的图像的灰度均值：

$$\nu = \sum_{i=0}^{t} i * \frac{P_i}{S} + \sum_{i=i+1}^{M} i * \frac{P_i}{S} = \nu_{t1} + \nu_{t2} = \omega_1\nu_1 + \omega_2\nu_2$$

(6) 类间方差：

$$G = \omega_1(\nu - \nu_1)^2 + \omega_2(\nu_1 - \nu_2)^2$$

1. 4 实验流程

1. 准备适合于最大类间的阈值选择法的图片

本次实验选择了皮肤病的临床医学图像作为实验图像。病变区域相较于环境差别比较大。并且噪音稀少，比较容易利用该阈值方法进行分割。

实验图像如图所示1.1：2. 将原始 RGB 图像转为灰度图像，转换之后的图像如图所

图 1.2　原始医学图像

示1.2

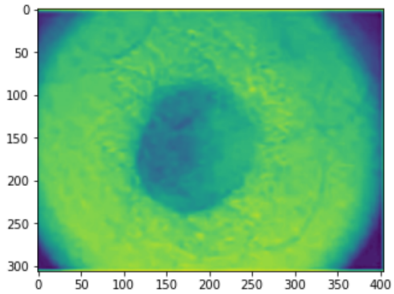

图 1.3　灰度图像

我们发现直方图有明显的一个凸峰，并且仅有一个，所以该图像非常适用于使用最大类间自适应阈值法进行分割。

3. 通过最大类间自适应阈值选择，找出该图像的最大类间

找出的最大类间的灰度阈值为 146.

4. 使用找出的阈值对图像进行分割，小于该阈值的设定为 0，大于该阈值的设定为 255.

得到的图像结果如图所示1.3：

我们可以从图像中很清晰的看出，中间病变的地方被分割出来。

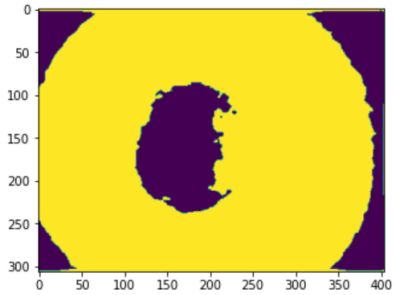

图 1.4　图像分割

1.5 最大类间自适应阈值法代码块

Listing 1.1: python 代码

```python
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('1.png')
# hist是256x1数组，每个值对应于该图像中具有相应像素值的像素数
hist = cv.calcHist([img],[0],None,[256],[0,256])
# 绘制直方图
plt.plot(hist)
plt.show()
########################################
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread('1.png', 0)
plt.imshow(img)
plt.show()
#####################################
def OTSU(img_gray, GrayScale):
    assert img_gray.ndim == 2, "must input a gary_img"  # shape有几个数字，ndim就是多少
    img_gray = np.array(img_gray).ravel().astype(np.uint8)
    u1 = 0.0  # 背景像素的平均灰度值
    u2 = 0.0  # 前景像素的平均灰度值
    th = 0.0

    # 总的像素数目
```

```
27      PixSum = img_gray.size
28      # 各个灰度值的像素数目
29      PixCount = np.zeros(GrayScale)
30      # 各灰度值所占总像素数的比例
31      PixRate = np.zeros(GrayScale)
32      # 统计各个灰度值的像素个数
33      for i in  range(PixSum):
34          # 默认灰度图像的像素值范围为GrayScale
35          Pixvalue = img_gray[i]
36          PixCount[Pixvalue] = PixCount[Pixvalue] + 1
37
38      # 确定各个灰度值对应的像素点的个数在所有的像素点中的比例。
39      for j in  range(GrayScale):
40          PixRate[j] = PixCount[j] * 1.0 / PixSum
41      Max_var = 0
42      # 确定最大类间方差对应的阈值
43      for i in  range(1, GrayScale):  # 从1开始是为了避免w1为0.
44          u1_tem = 0.0
45          u2_tem = 0.0
46          # 背景像素的比列
47          w1 = np. sum(PixRate[:i])
48          # 前景像素的比例
49          w2 = 1.0 - w1
50          if w1 == 0 or w2 == 0:
51              pass
52          else:  # 背景像素的平均灰度值
53              for m in  range(i):
54                  u1_tem = u1_tem + PixRate[m] * m
55              u1 = u1_tem * 1.0 / w1
56              # 前景像素的平均灰度值
57              for n in  range(i, GrayScale):
58                  u2_tem = u2_tem + PixRate[n] * n
59              u2 = u2_tem / w2
60              # print(u1)
61              # 类间方差公式: G=w1*w2*(u1-u2)**2
62              tem_var = w1 * w2 * np.power((u1 - u2), 2)
63              # print(tem_var)
64              # 判断当前类间方差是否为最大值。
65              if Max_var < tem_var:
66                  Max_var = tem_var  # 深拷贝, Max_var与tem_var占用不同的内存空间。
```

```
                th = i
    return th


th = OTSU(img, 256)
print("使用numpy的方法: " +  str(th))  # 结果为 146

#############################################################

ret,img_seg = cv2.threshold(img_gray,146,256,cv2.THRESH_BINARY)
plt.imshow(img_seg)
plt.show()

#################################################
```