

**Matlab 图 像 处 理 编 程 实 践**

**第 一 次 大 作 业 报 告**

|  |  |
| --- | --- |
| 课程名称： | 课程综合实践Ⅰ |
| 学 院： | 计算机科学与技术学院 |
| 系： | 软件工程系 |
| 专 业： | 软件工程 |

2022 年 7 月 5 日

## 实验任务简介

实验目的：熟练使用MATLAB进行计算机图像处理，设计算法实现书法字相似度识别。

任务简介：calligraphy文件夹中给出若干书法字，对其进行处理。编写一个程序使得选择其中一幅字可以自动播放与其相似度最高的20个单字。

## 程序框架与技术细节

总体框架：

1. 预处理模块(pretreatment.m)
   1. 首先新建处理后图片存储的文件夹
   2. 遍历所有图片进行处理：转二值->降噪(bwareaopen和中值滤波)->切割图片->统一大小(imresize)并保存至treat1文件夹->结构化处理(bwmorph)并保存至treat2文件夹
2. 识别搜索模块（searchAll.m）
   1. 利用uigetfile读取想要进行比较的文件
   2. 与treat1文件夹中所有文件进行比较并返回两个向量val1和val2
   3. 与treat2文件夹中所有文件进行比较并返回三个向量val3，val4和val5
   4. 对上述5个向量进行加权形成最终向量val。
   5. 根据最终向量排序、除干扰输出。

各功能模块细节介绍：

1. 处理图片（pretreatfunction函数）
2. 转二值：先判断其是不是RGB图，如果是RGB图先转成灰度图，然后利用imbinarize函数转为二值图。
3. 降噪：bwareaopen函数直接降噪，噪声判定设置为图片总大小的千分之一。

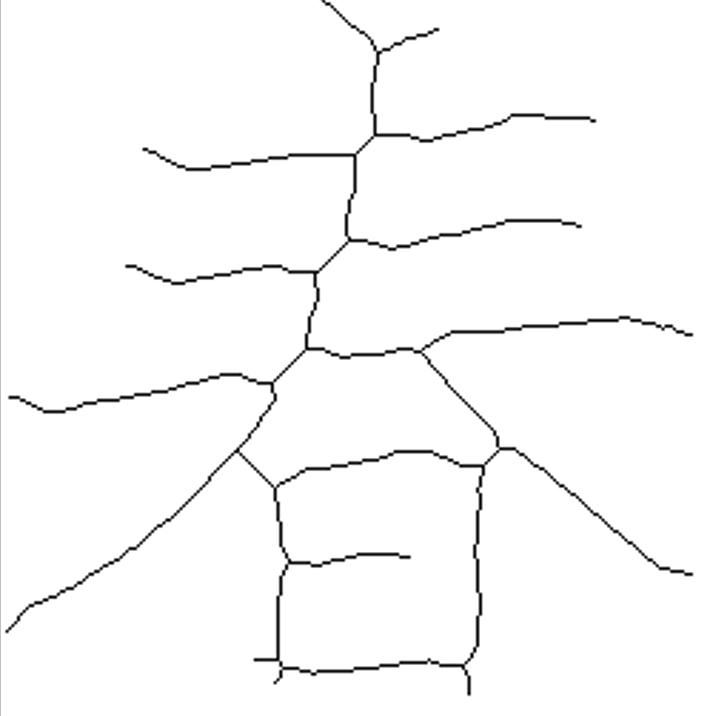
medfilt2函数实现中值滤波降噪。

1. 切割图片：先通过循环获得图像的上下左右边界，这里我规定黑色占比大于整条边的0.05、小于整条边的0.9（防止有图片本身含边框）时判定为文字主体。

然后判断图片黑色是否占50%以下，若在一半以上判定为阴文，需要转成阳文，做取反处理。

最后利用imcrop函数进行切割。

1. 统一大小：用imresize将图片统一为256\*256大小。**输出在treat1文件夹。**
2. 结构化处理：先将图片取反，然后用bwmorph函数的’thin’方法获得字体的骨架并**输出在treat2文件夹**。

（上图从左到右依次是原图、treat1中的二值图、treat2中的骨架图）

1. 相似度分析（三个method函数中获取五个向量）

**Val1**（method1\_overlap）：对两个二值图进行异或操作获得重叠度，计算相似度。

**Val2**（method2\_edgedetect）：边缘检测算法，将两个二值图利用edge函数的sobel算法取出笔画边缘，并将新边缘图分为16\*16的格子，对每个格子中的颜色直方图进行计数，利用公式计算其余弦相似度。

**Val3**（method3\_DIYvector）：将两个二值图直接分为16\*16的格子，对每个格子中的颜色直方图进行计数，利用corr2函数计算其相似度。

**Val4**（method3\_DIYvector）：通过比较256\*256的图片内第121-136行内黑色占比是否小于平均黑色占比的一半来判断该字是否为左右结构。

**Val5**（method3\_DIYvector）：将两幅字分别横切为16分，对每一份中的颜色进行计数，计算两幅字的相关系数作为相似度。。

1. 向量加权处理

5个向量按照1::0.4:1:0.3:0.3的比例进行加权形成最终向量。

1. 排序除干扰输出

排序：按照val的大小对存储图像名字的数组进行排序

除干扰：用bwlabel函数进行图像内连通性判断，若为下图类似的图像，则跳过此图。这类图连通对象数量将远远超过阈值，我将阈值定为50.

判断val向量值是否为NaN，若为NaN则也为干扰图像。

输出：通过路径找到原图并输出图像及其名称。

各函数功能介绍：

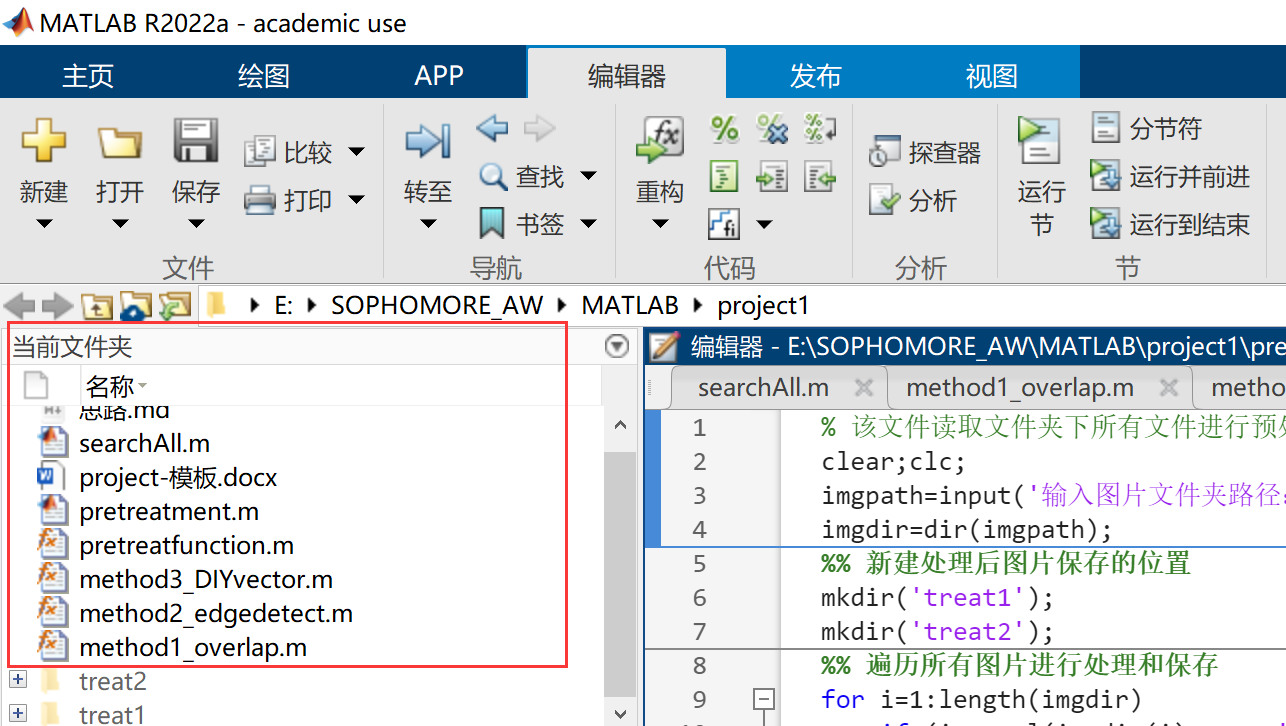
|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 输入变量 | 输出变量 | 功能 |
| pretreatfunction | 一个二值图数组  [img] | 处理后的两个二值图数组  [BW4,BW7] | 图像预处理 |
| method1\_overlap | 两个二值图数组  [BW,I] | 两个图片的相似度  val | 重叠法计算相似度 |
| method2\_edgedetect | 两个二值图数组  [BW,I] | 两个图片的相似度  val | 边缘检测计算相似度 |
| method3\_DIYvector | 两个二值图数组  [BW,I] | 两个图片的三种相似度  [val1,val2,val3] | 分格占比、结构分布、横纵切割计算相似度 |

## 程序运行示例

指导用户如何用这个程序，包括如何设置文件路径，如何设置输入，结果展示等。

1. 图像预处理

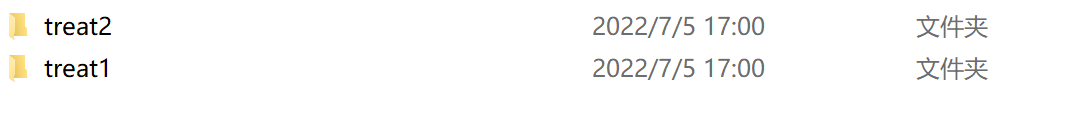
首先打开pretreatment.m文件，保证当前文件夹目录有脚本，运行。



输入calligraphy文件夹地址（注意包含calligraphy文件夹名）

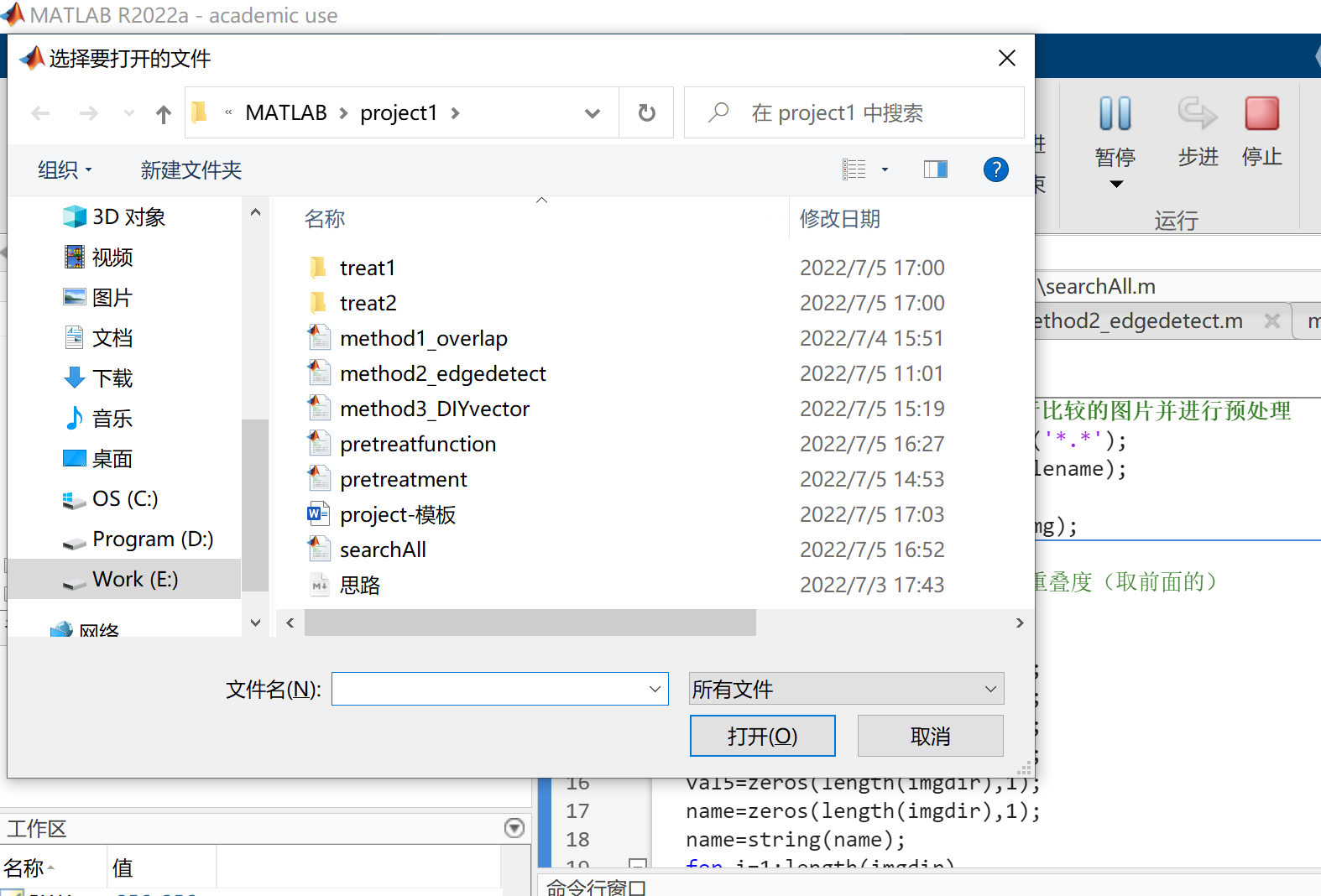


等待处理完毕，处理完成后在脚本同目录下会出现两个文件夹treat1和treat2，此为处理后的两种图片。

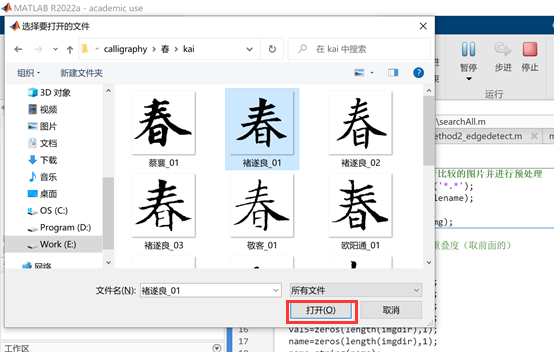


1. 相似度查询

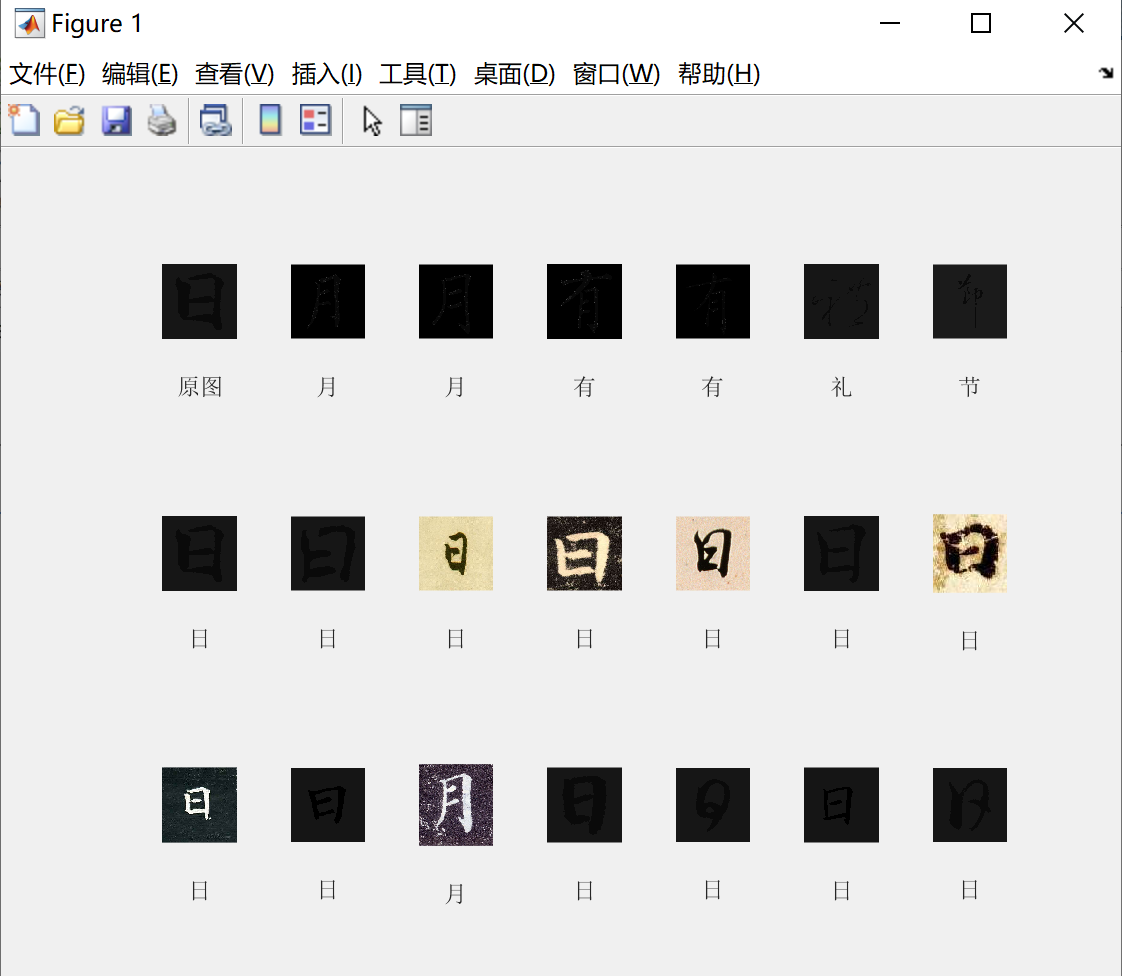
打开searchAll.m文件，运行，此时弹出一个窗口。



在该窗口中选择未处理的文件夹calligraphy下的一张图片，点击打开。



等待程序运行十秒左右即可，弹出窗口中有21张图片，第一张为原图，后20张为程序识别与其最为接近的20张。



## 实验结果分析

结果分析，不同方法的对比等

|  |  |  |
| --- | --- | --- |
| 选择图片 | 运行结果 | 正确率 |
| 楷书 |  | 100% |
| 楷书 |  | 100% |
| 其他 |  | 90% |
| 草书 |  | 90% |
| 篆书 |  | 85% |
| 楷书 |  | 80% |
| 楷书 |  | 80% |
| 草书 |  | 75% |

结果分析：

总体正确率较高，一些清晰的字可以达到95%及以上的正确率，对于特征不明显的字体（如草书）正确率稍低，但大多在50%以上。

不同方法对比：

Val1向量可以清晰的给出与原图重叠度较高的图片，对于楷书等位置相对固定的字体辨识正确率很高。

Val2向量基于边缘检测和余弦相似度，可以明显区分笔画较多的字体和笔画少的字体。

Val3向量以格子化的色块为切入点，是基于模糊查询的高阶算法，，对位置相对固定且笔画多少区分度明显的字体（如隶书）辨识度高。

Val4向量用来判断字体是否为左右结构，起辅助作用。

Val5向量是val4向量的模糊查询版本，将图像进行横纵分割查询。