**《数字图像处理概论》大作业**

**作 业 名 称： 图像处理系统软件**

**专 业 计算机科学与技术学院软件工程**

**组 序 号 2**

**组 长 姓 名 王艺璇**

**组 长 学 号 2021212077**

**联 系 方 式 17792522360**

|  |  |  |  |
| --- | --- | --- | --- |
| **成绩** |  | **评阅人** | **夏勇** |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **1** | **2** | **3** | **4** | **5** | **6** | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | | 学号 | 姓名 | 贡献权重 | 成绩 | | 2021212077 | 王艺璇 | 1 |  | | | | |

**哈尔滨工业大学计算机学院**

# 报告正文

## 0 小组分工介绍

本小组成员只有王艺璇一人，全部的十项功能模块的实现以及报告的撰写均由王艺璇一人独立完成。

## 1 系统设计

本系统用于对图像进行特定处理，包括将真彩图转换为灰度图、将灰度图划分为三个颜色通道的二值图、对灰度图进行反色、对给定图片绘制直方图、对直方图进行均衡化、对图像使用平均值滤波、对图像使用中值滤波、放缩图片、平移图片、对图片镜像、旋转图片、根据给定域值分割图片、根据迭代域值分割图片、根据Otsu算法获取域值分割图片、种子增长、区域分割、边缘检测、利用霍夫变换找到直线、利用区域标记分割区域和轮廓提取的功能。采用DevC++编辑器利用C++语言进行编程，利用cmd命令行进行交互，用户根据命令行提示输入指定数据，即可执行对应的模块的功能。

主菜单界面如下：

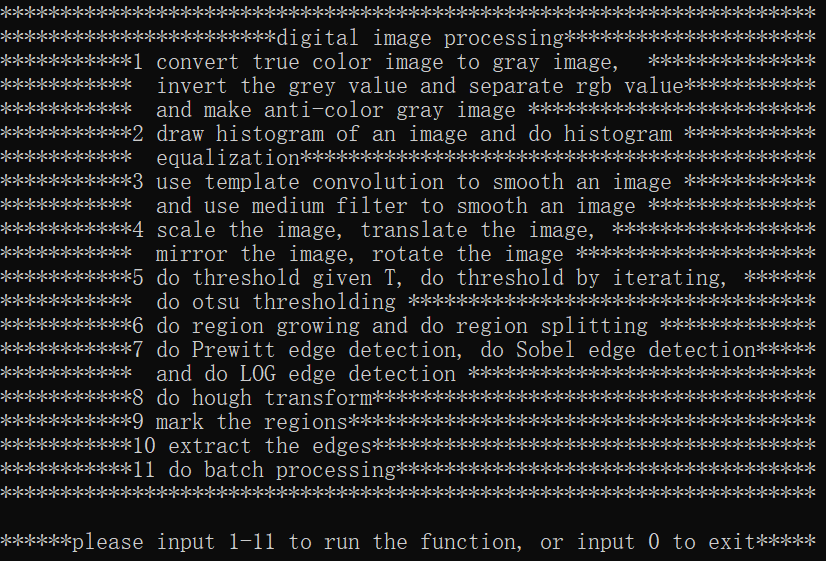


图1 主菜单界面

批处理参数已设置在函数内部，即将读入文件的位置和输出文件的位置写成固定的绝对位置。

运行界面截图：

（1）本系统支持多次输入1-11的数字来执行对应功能，直到输入0后表示“退出该系统”才会终止执行。

（2）当输入1时，执行将真彩图转换为灰度图、将灰度图划分为三个颜色通道写出、对灰度图进行反色的操作。系统在执行完成后给出“功能1已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（3）当输入2时，执行对给定图片绘制直方图、对直方图进行均衡化的操作。系统在执行完成后给出“功能2已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（4）当输入3时，执行对图像使用平均值滤波、对图像使用中值滤波的操作。系统在执行完成后给出“功能3已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（5）当输入4时，执行放缩图片、平移图片、对图片镜像、旋转图片的操作。系统在执行完成后给出“功能4已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（6）当输入5时，执行根据给定域值分割图片、根据迭代域值分割图片、根据Otsu算法获取域值分割图片的操作。系统在执行完成后给出“功能5已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（7）当输入6时，执行种子增长和区域分割的操作。系统在执行完成后给出“功能6已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（8）当输入7时，执行利用Prewitt算子进行边缘检测、利用Sobel算子进行边缘检测、利用LOG算子进行边缘检测的操作。系统在执行完成后给出“功能7已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（9）当输入8时，执行利用霍夫变换找到直线的操作。系统在执行完成后给出“功能8已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（10）当输入9时，执行利用区域标记分割区域的操作。系统在执行完成后给出“功能9已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（11）当输入10时，执行轮廓提取的操作。系统在执行完成后给出“功能10已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

（12）当输入11时，执行以上的全部的操作。系统在执行完成后给出“功能11已经执行完成，并且可以前往homework\_11文件夹查看，请继续操作”的提示信息。

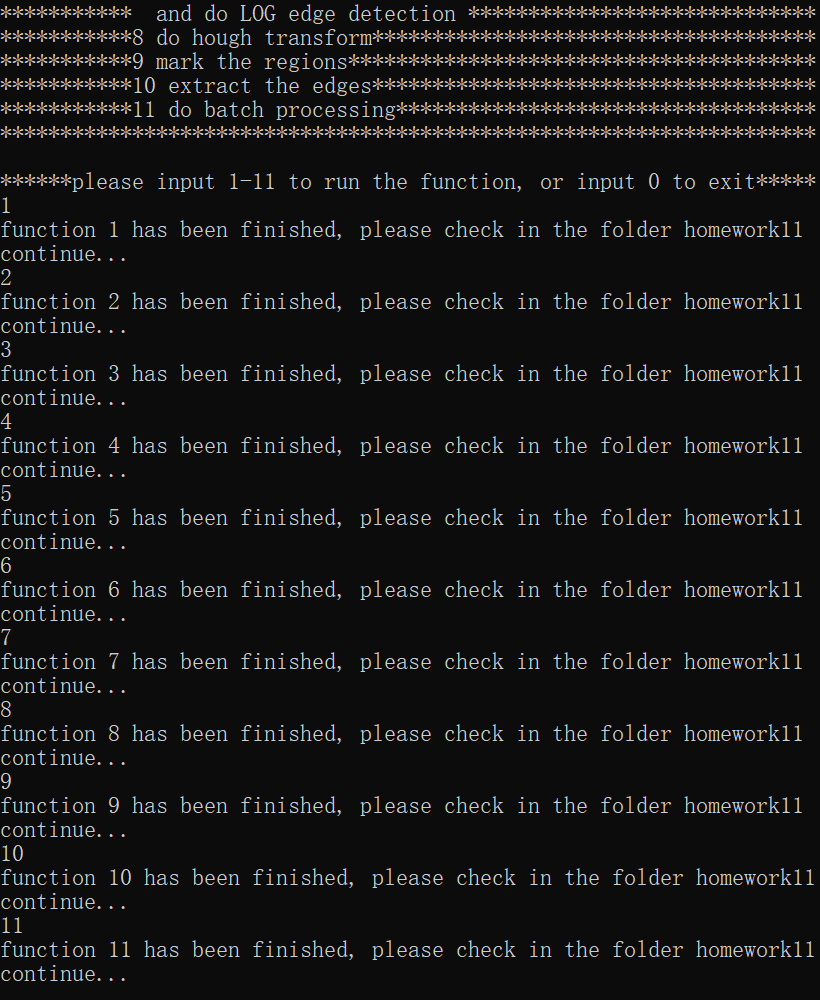


图2运行界面

## 2 功能模块设计与实现

### 2.1文件处理

#### 基本原理：

（1）文件读写的处理：

BMP图像文件包含位图文件头、位图信息头、调色板、实际位图数据四部分，在读取时需要对其每一部分用结构体进行封装，并根据位图信息头的数据为实际位图数据分配合适大小的内存空间。值得注意的是，在计算每行的字节数时需要判断字节对齐的问题，对8位灰度图找到最接近4的倍数的数，对24位真彩图找到最接近24的倍数的数。输出8位灰度图时，一定要给调色板赋值，否则会出现错误。

（2）真彩图转换成为灰度图的原理：

灰度值=0.299\*R+0.587\*G+0.114\*B，其中R、G、B对应真彩图的三个颜色通道的R、G、B的值。

（3）将灰度图划分为三个颜色通道的原理

灰度图是按照B、G、R的顺序对每一个点的颜色值进行存储，当要提取其中一种颜色值时，令该通道数值为原图中该通道的数值而其他通道为0即可，此时灰度图只会呈现出不为0的像素颜色，即实现了根据颜色通道划分的效果。

（4）对灰度图进行反色的原理

灰度图的调色板的数值从0-255对应256种颜色，利用255减去该像素点在该通道的颜色值即可获得该像素点在该通道的相反的（互补）的颜色。

#### 输入图像：

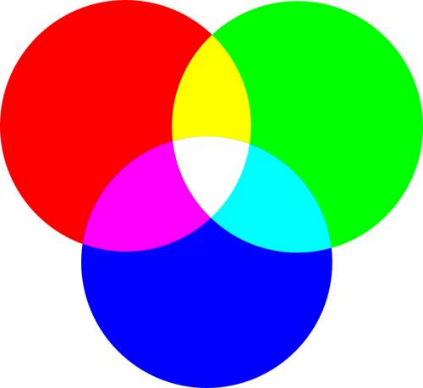


图3 rgb.bmp展示

#### 结果图像：

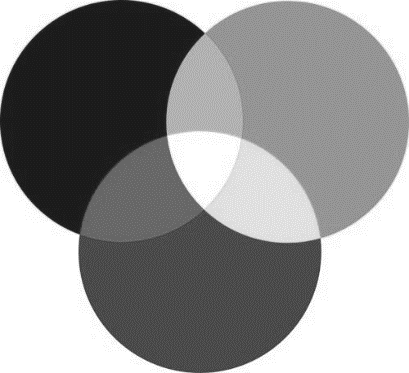
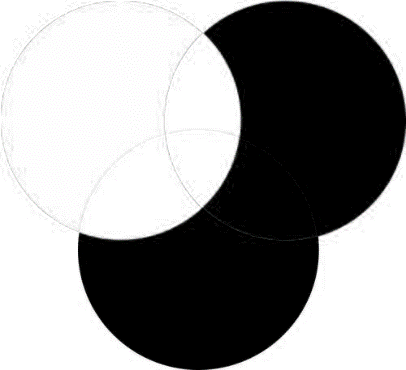
 

图4 rgb\_to\_grey.bmp展示 图5 rgb\_extract\_red.bmp展示

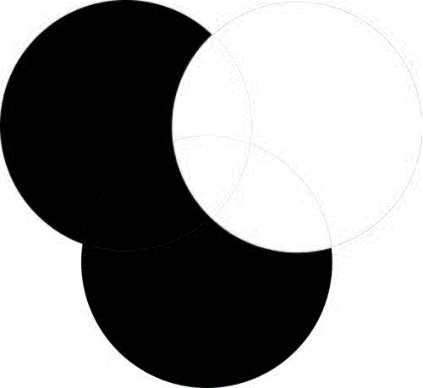
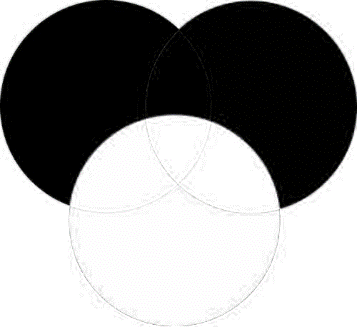
 

图6 rgb\_extract\_green.bmp展示 图7 rgb\_extract\_blue.bmp展示

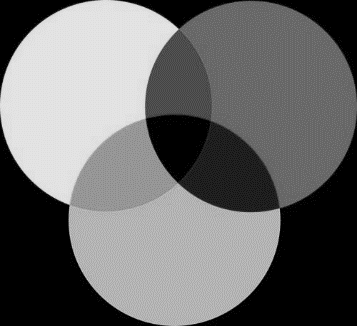


图8 rgb\_to\_antigrey.bmp 展示

#### 结果分析：

（1）真彩图转换为灰度图的结果为图4；

（2）将灰度图划分为红色、绿色、蓝色三个颜色通道灰度图分别为图5、图6、图7；

（3）对灰度图进行反色的结果为图8。

### 2.2直方图处理

#### 基本原理：

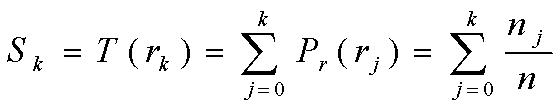
（1）绘制直方图的原理

直方图统计的是图像内各个灰度级出现的次数，横坐标是图像中各像素点的灰度级，纵坐标是具有该灰度级。统计灰度图中每一个颜色的数量和最大的颜色出现次数。为了画图的美观，即使图像尽量呈现为一个正方形，在绘制时用最大的颜色出现次数除以256（颜色种类）来作为最大高度，其余所有颜色均除以这个比值，即等比例缩小。

（2）对直方图进行均衡化的原理

均衡化的目的是将原始图像的直方图变为均衡分布的的形式，将非均匀灰度概率密度分布图像，通过变换，变成一幅具有均匀概率密度分布的目的图像。

设一幅图像的像素总数为n，分为L个灰度级，其中：nk：表示第K个灰度级出现的个数。p(rk)=nk/n：第K个灰度级出现的概率（0<=rk<=1, k=0,1,2,...,L-1）。

求得公式如下：

计算步骤如下：

①求出图像中所包含的灰度级rk,一般范围在[0,1]之间，也可以在[0,L-1]；

②统计各灰度级的像素数目nk（k=0,1,2,...,L-1）；

③计算图像直方图；

④计算变换函数；

⑤用变换函数计算映射后输出的灰度级sk；

⑥统计映射后新的灰度级sk的像素数目nk；

⑦计算输出图像的直方图。

#### 输入图像：

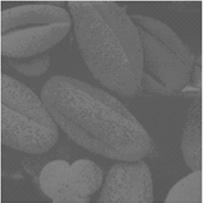


图9 dim.bmp

#### 结果图像：

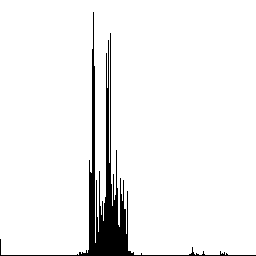
 

图10 dim\_hist\_normalization.bmp展示 图11 dim\_equalization.bmp展示

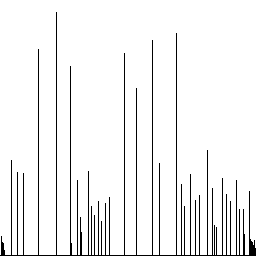


图12 dim\_hist\_equalization.bmp

#### 结果分析：

（1）对原图直接进行直方图绘制的结果为图10；

（2）对原图进行直方图均衡化后的结果为图11；

（3）对均衡化后的图片进行直方图绘制的结果为图12。

### 2.3空间域滤波

#### 基本原理：

（1）对图像使用平均值滤波的原理：

平均值滤波是指用当前像素点周围n\*n个像素值的均值来代替当前像素值。使用该方法遍历处理图像内的每一个像素点可完成整幅图像的均值滤波。

当对某一像素点进行均值滤波时，考虑需要对周围多少个像素点去取平均值。通常情况下，以该当前像素为中心，对行数和列数相等的一块区域内的所有像素点的像素取平均值。例如,我们可以以当前像素点的像素周围3x3区域内所有像素点的像素取平均值，也可以对周围5x5区域内所有像素点的像素值取平均值。计算得到新值以后，将新值作为当前像素点平均值滤波后的像素值，依次对图中每一个像素点进行如上操作即可得到当前图像的平均值滤波结果。

（2）对图像使用中值滤波的原理：

中值滤波是指用当前像素点周围n\*n个像素值的中位数的值来代替当前像素值。使用该方法遍历处理图像内的每一个像素点可完成整幅图像的中值滤波。

当对某一像素点进行中值滤波时，考虑需要对周围多少个像素点去取平均值。通常情况下，以该当前像素为中心，对行数和列数相等的一块区域内的所有像素点的像素取中位数，达到让周围的像素值接近真实值，从而消除孤立的噪声点的效果。例如,我们可以以当前像素点的像素周围3x3区域内所有像素点的像素取中位数，也可以对周围5x5区域内所有像素点的像素值取中位数。计算得到新值以后，将新值作为当前像素点中值滤波后的像素值，依次对图中每一个像素点进行如上操作即可得到当前图像的中值滤波结果。

（3）对图像边缘的处理：

①对边上、角上的点的像素不处理。

②扩充图像，复制原图像的边界的像素来填充扩充的图像边界，使得在图像边界处也可计算。特别地，对角上的点的像素值可以赋值为任意相邻边上的最近邻点的像素值。

#### 输入图像：

图13 lena.bmp展示 图14 noise2.bmp展示

#### 结果图像：



图15 lena\_neighborhood\_average\_filtering.bmp展示



图16 lena\_neighborhood\_average\_filtering\_9.bmp展示



图17 lena\_neighborhood\_average\_filtering\_9\_cov.bmp展示



图18 lena\_neighborhood\_average\_filtering\_25.bmp展示



图19 noise2\_median\_filtering.bmp展示

#### 结果分析：

（1）对图像使用平均值滤波的结果为图15、图16、图17、图18，区别在于：图15是对比了扫描3\*3的范围和5\*5的范围来为一个像素点来赋值后的更好的效果，图16是采取了较为繁复的算法的代码来生成的扫描3\*3的范围的平均值滤波结果，图17是调用了自己编写的卷积算法的代码来生成的扫描3\*3的范围的平均值滤波结果，图18是扫描5\*5的范围的平均值滤波结果。

（2）对图像使用中值滤波的结果为图19，每次扫描3\*3的范围来为一个像素点赋值，当遇到边界之外的点时直接赋予边界上的点的像素值。

### 2.4图像变换

#### 基本原理：

（1）放缩图片的原理：

放缩图片时采用后向映射的逻辑，即从目标图像的每个位置出发，根据不同的插值方法寻找其在对应的原图中的位置的像素点的颜色值来填充该目标图像的像素点的颜色值，以此可以保证目标图像中不会出现“空洞”。

放大过程对于目标图像中的每一个像素点(x,y)，我们只需要将它映射到原图像的(x / 1.5,y / 1.5)位置，通过原图像已有的像素值插值得到(x / 1.5,y / 1.5)的像素值即可。

缩小过程对目标图像中每一个像素点(x,y)，我们只需将它映射到原图像的(x \* 2,y \* 2)位置，通过原图像已有的像素值插值得到(x \* 2,y \* 2)的像素值即可。

其中，插值方法主要有最近邻插值、线性插值、曲线插值三种。最邻近插值法最为简单，但效果较差；线性插值相对复杂一些，效果较好，应用最为广泛；曲线插值很复杂，较少采用。

①最近邻插值：取坐标上最靠近的像素；

②线性插值：使用原图中两个值来构造所求坐标处的值；

③双线性插值：。

（2）平移图片的原理：

图像平移，就是把图像中像素的坐标（x,y）进行处理，设x,y方向的平移量分别为dx,dy，平移后的坐标为（x1,y1），则计算公式为： ，用矩阵表示为： 。而对于已经移出图片边界的区域不做处理，即在遍历时直接跳过这些点。

（3）对图片镜像的原理：

图片的镜像分为水平镜像和垂直镜像两种。

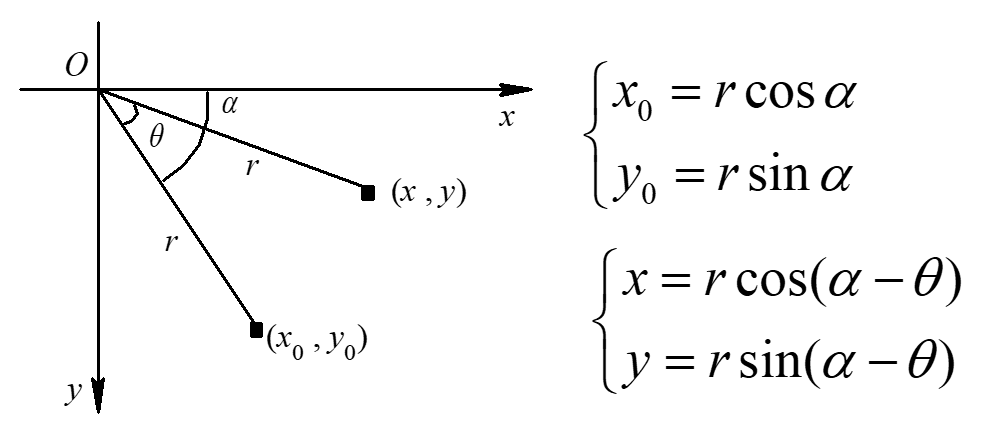
①水平镜像: 将图像的左右部分以图像的垂直中轴线为对称线进行镜像；

②垂直镜像:将图像的上下部分以图像的水平中轴线为对称线进行镜像。

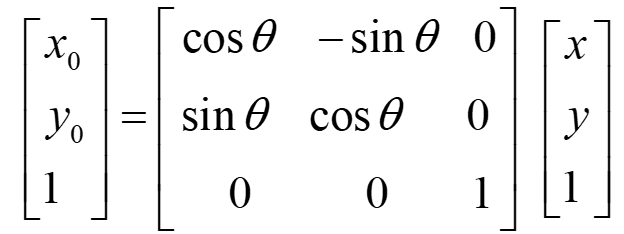
设点P0(x0, y0)进行镜像后的对应点为P(x, y)，图像高度为H，宽度为W。

原图像中点P0(x0, y0)经过水平镜像后坐标将变为 ；原图像中点P0(x0, y0)经过垂直镜像后坐标将变为。

（4）旋转图片的原理：

设点P0(x0, y0)旋转θ角后的对应点为P(x, y)，如图所示。

写成矩阵形式的逆运算为：



#### 输入图像：



图20 lena.bmp展示

#### 结果图像：

图21lena\_bigger.bmp展示 图22 lena\_smaller.bmp展示

图23 lena\_translation.bmp展示 图24 lena\_mirror.bmp展示

图25 lena\_upside\_down.bmp展示 图26 lena\_rotate.bmp展示

#### 结果分析：

（1）对图片放大后的结果为图21；

（2）对图片缩小后的结果为图22；

（3）对图片平移后的结果为图23；

（4）对图片左右镜像后的结果为图24；

（5）对图片上下翻转后的结果为图25；

（6）对图片旋转30°后的结果为图26。

### 2.5阈值分割

#### 基本原理：

（1）根据给定域值分割图片的原理：

将整个图像的灰度阈值设置为常数。其公式为：，其中f(x,y)为点(x,y)的像素值，g(x,y) 为分割后的图像，T为全局阈值。通常通过直方图来获取全局阈值。

（2）根据迭代域值分割图片的原理：

迭代域值分割方法适用于背景和对象在图像中占据的面积相近的情况。

最开始选择一个阈值作为初始估计值（一般选择将所有点的像素值排序后的中位数），然后按某种策略不断地改进这一估计值，直到满足给定的准则为止。在迭代过程中，关键之处在于选择什么样的阈值改进策略，好的阈值的改进策略应该具备两个特征，一是能够快速收敛，二是在每一个迭代过程中，新产生阈值优于上一次的阈值。

①选择图像灰度的中值作为初始阈值Ti=T0；

②利用阈值Ti把图像分割成两部分区域，R1和R2，并计算其灰度均值

；

③计算新的阈值Ti+1，；

④重复步骤②③，直到Ti+1和Ti的值差别小于某个给定值。

（3）根据Otsu算法获取域值分割图片的原理：

Otsu法确定最佳阈值的准则是使阈值分割后各个像素类的类内方差最小。另一种确定阈值的准则是使得阈值分割后的像素类的类间方差最大。这两种准则是等价的，因为类间方差与类内方差之和即整幅图像的方差，是一个常数。分割的目的就是要使类别之间的差别最大，类内之间的差别最小。

设图像总像素数为N，灰度级总数为L，灰度值为i的像素数为Ni。令ω(k)和μ(k)分别表示从灰度级0到灰度级k的像素的出现概率和平均灰度，分别表示为 ，由此可见，所有像素的总概率为ω(L-1)=1，图像的平均灰度为μT=μ(L-1)。

设有M-1个阈值(0≤t1＜t2＜…＜tM-1≤L-1)，将图像分成M个像素类Cj(Cj∈[tj-1+1,…,tj]；j=1,2,…,M；t0=0,tM=L-1)，则Cj的出现概率ωj、平均灰度μj和方差σj2为 ，

，，由此可得类内方差为，各类的类间方差为，将使类内方差最小或使类间方差最大的阈值组(t1,t2,…,tM－1)作为M阈值化的最佳阈值组。

若取M为2，即分割成2类，则可用上述方法求出二值化的最佳阈值。由此可得类间方差为，此外，还有，从而类间方差可以变换为：，类内方差的最小化与类间方差的最大化是一致的优化目标。

#### 输入图像：



图27 lena.bmp展示

#### 结果图像：

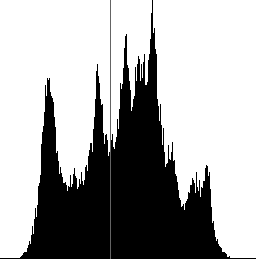
 

图28 lena\_hist.bmp展示 图29 lena\_manually\_threshold.bmp展示

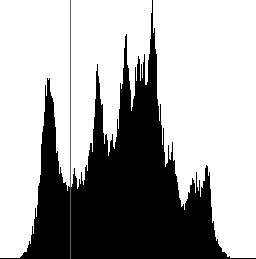
 

图30 lena\_hist70.bmp展示 图31 lena\_manually\_threshold70.bmp展示

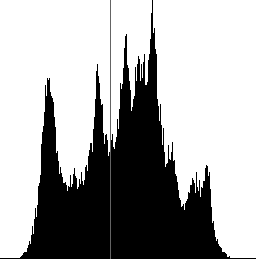
 

图32 lena\_hist110.bmp展示 图33 lena\_manually\_threshold110.bmp展示

图34 lena\_iteration.bmp展示 图35 lena\_iteration5.bmp展示

图36 lena\_iteration10.bmp展示 图37 lena\_otsu.bmp展示

#### 结果分析：

（1）根据给定域值分割图片的结果为图28、图29、图30、图31、图32、图33。其中，图28为域值是70和域值是110的两种情况下分隔效果更好的域值的直方图上的标记，图29是根据图28的给定域值进行区域分割的结果，图30为域值是域值是70的情况下的直方图上的标记，图31是根据图30的给定域值进行区域分割的结果，图32是域值是110的情况下的直方图上的标记，图33是根据图32的给定域值进行区域分割的结果。

（2）根据迭代域值分隔图片的结果为图34、图35、图36。其中，图34为以前后两次平均像素颜色值的差小于等于5作为终止条件和以前后两次平均像素颜色值的差小于等于10作为终止条件的效果更好的绝对值之差作为域值的区域分割的结果，图35是以前后两次平均像素颜色值的差小于等于5作为终止条件的区域分割的结果，图36是以前后两次平均像素颜色值的差小于等于10作为终止条件的区域分割的结果。

（3）根据Otsu算法获取域值分割图片的结果为图37。

### 2.6基于区域的分割

#### 基本原理：

（1）种子增长的原理：

区域生长可以根据预先定义的生长规则将像素或者小区域不断组合为更大区域的过程。具体地，区域生长是从一组初始种子点出发，通过预先定义的区域生长规则，将与种子点性质相似的领域像素不断添加到每个种子点上，并且满足区域生长的终止条件时形成最终生长区域的过程。

形式化地定义如下：令I表示图像,H表示具有相同性质的谓词,图像分割把I分解成n个区域 Ri, i＝1,2,…,n,满足：



即分割区域要覆盖整个图像且各区域互不重叠，每个区域具有相同的性质，相邻的两个区域性质相异从而不能合并为一个区域。

（2）区域分割的原理：

如果区域的某些特性差别比较大,即不满足一致性准则时,则区域应该采用分裂法。分裂过程从图像的最大区域开始，一般情况下，是从整幅图像开始。注意确定分裂准则(一致性准则)，确定分裂方法，即如何分裂区域，使得分裂后的子区域的特性尽可能都满足一致性准则值。算法如下：

①形成初始区域；

②对图像的每一个区域Ri，计算P(Ri)，即是否可视为同一个区域。如果P(Ri)=FALSE，即区域Ri应当被继续细分，则沿着某一合适的边界分裂区域；

③重复步骤②，当没有区域需分裂时，算法结束。

#### 输入图像：

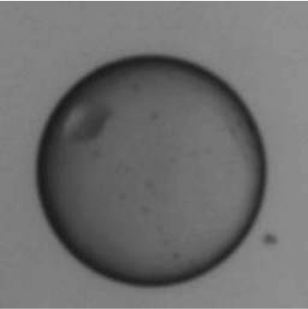
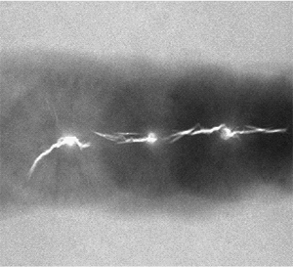
 

图38 bubble.bmp展示 图39 gap.bmp展示

#### 结果图像：

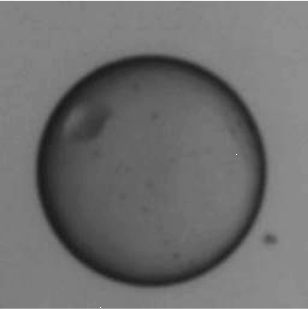
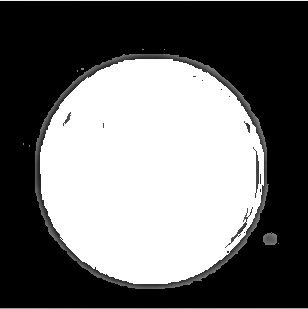
 

图40 bubble\_to\_grey.bmp展示 图41 bubble\_seed\_growth.bmp展示

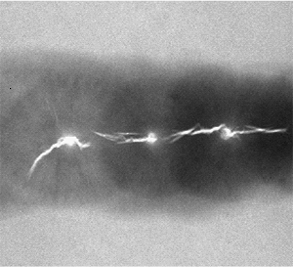
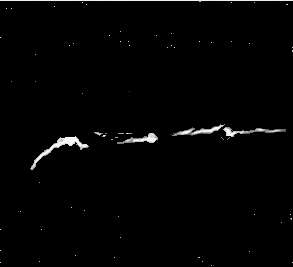
 

图42 gap\_to\_grey.bmp展示 图43 gap\_seed\_growth.bmp展示

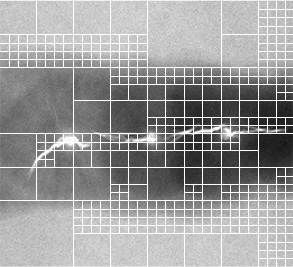


图44 gap\_aera\_division.bmp展示

#### 结果分析：

（1）图38从真彩图转换为灰度图的结果为图40，图40经过种子增长得到图41，其中种子点标记在图40的两个白点位置；

（2）图39从真彩图转换为灰度图的结果为图42，图42经过种子增长得到图43，其中种子点标记在图42的左上角的黑点位置，图44是图42经过区域分割之后的结果。

### 2.7边缘检测

#### 基本原理：

在边缘处，灰度和结构等信息的产生突变.边缘是一个区域的结束,也是另一个区域的开始,利用该特征可以分割图像。沿边缘方向像素变化平缓,垂直于边缘方向像素变化剧烈.边缘上的这种变化可以用微分算子检测出来,通常用一阶或二阶导数来检测边缘。

在边缘部分，像素值出现“跳跃”或者较大的变化，如果在此边缘部分求一阶导数，就会看到极值的出现，而在一阶导数为极值的地方，二阶导数为0，基于这个原理，就可以进行边缘检测。

梯度算子是一阶导数算子：。

（1）根据Prewitt算子进行边缘检测的原理：

假设要处理的图像为 I ，在两个方向求导：比如，模板大小为3时，

水平方向：将图像 I 与奇数大小的模板进行卷积，结果为Gx，有Gx=\*I；

垂直方向：将图像 I 与奇数大小的模板进行卷积，结果为Gy，Gy=\*I；

那么在图像的每一点处，结合以上两个结果可以求出：|G|=|Gx|+|Gy|；

统计极大值的位置，就是图像的边缘。

（2）根据Sobel算子进行边缘检测的原理：

Sobel算子对于像素的位置的影响做了加权，因此与Prewitt算子相比效果更好。假设要处理的图像为 I ，在两个方向求导：

水平方向：将图像 I 与奇数大小的模板进行卷积，结果为Gx，Gx=\*I；

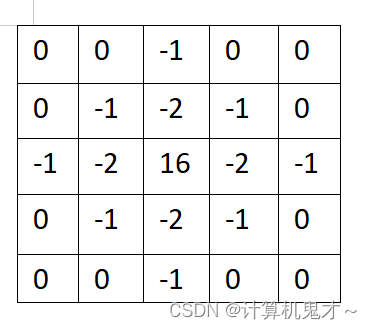
垂直方向：将图像 I 与奇数大小的模板进行卷积，结果为Gy，Gy=\*I；

那么在图像的每一点处，结合以上两个结果可以求出：|G|=|Gx|+|Gy|；

统计极大值的位置，就是图像的边缘。

（3）根据LOG算子进行边缘检测的原理：

该算子首先用高斯函数对图像作平滑滤波处理，然后才使用Laplacian算子检测边缘。因此克服了Laplacian算子抗噪声能力比较差的缺点，但是在抑制噪声的同时也可能将原有的比较尖锐的边缘也平滑掉了，造成这些尖锐边缘无法检被测到。

，其一个5\*5模版如下：。

#### 输入图像：



图45 lena.bmp展示

#### 结果图像：

图46 lena\_prewitt.bmp展示 图47 lena\_sobel.bmp展示



图48 lena\_log.bmp展示

#### 结果分析：

（1）根据Prewitt算子进行边缘检测的结果为图46；

（2）根据Sobel算子进行边缘检测的结果为图47；

（3）根据LOG算子进行边缘检测的结果为图48。

### 2.8霍夫变换

#### 基本原理：

利用霍夫变换找到直线的原理：

霍夫变换是一种在图像中寻找直线、圆形以及其他简单形状的方法。霍夫变换优点在于受噪声和曲线间断的影响较小。在已知曲线形状的条件下，霍夫变换实际上是利用分散的边缘点进行曲线逼近，它也可看成是一种聚类分析技术。

笛卡尔空间的一条直线映射到霍夫空间就是一个点，同理，霍夫空间的一个点就是笛卡尔空间的一条直线；笛卡尔空间的一个点映射到霍夫空间就是一条直线，同理，霍夫空间的一条直线就是笛卡尔空间的一个点；同理，笛卡尔空间中的多个共线点，就对应霍夫空间的多个共点线，而这个共点就是笛卡尔空间的线的k、b。故要找笛卡尔空间中的线条，我们就把笛卡尔空间转换到霍夫空间，在霍夫空间找交点最多的点，其霍夫坐标值就是笛卡尔空间中的线条的k和b。

用霍夫变换找原图中的线的基本思路是：将原图从笛卡尔空间映射到霍夫空间，然后找霍夫空间中尽可能多直线的交汇点的点，这个点的坐标就是原图中的直线的斜率和截距。这样我们就得到了原图中的直线了。

霍夫变换的基本步骤是：

①在参数空间建立一个二维数组A,数组的第一维的范围为图像空间中直线斜率的可能范围(amin, amax),第二维为图像空间中直线截距的可能范围(bmin, bmax),且开始时把数组初始化为零.

②然后对图像空间中的点用Hough变换计算出所有的a,b值,每计算出一对a,b 值,就对数组中对应的元素A(a,b)加1.计算结束后, A(a,b)的值就是图像空间中落在以a为斜率,b为截距的直线上点的数目。

当图像空间中有直线为竖直线时,斜率a为无穷大,此时,参数空间可采用极坐标。在极坐标系下，其实是一样的：极坐标系内的一个点映射到霍夫空间中是一条直线(曲线)，极坐标系内的一条直线映射到霍夫空间中是一个点，只不过霍夫空间不再是[k,b]的参数，而是[ρ,θ]。只要我们求得霍夫空间中的交点位置，就得到了原坐标系下的直线了。

#### 输入图像：

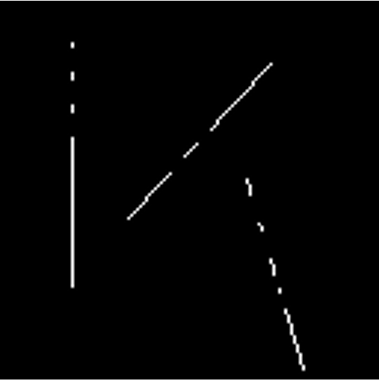


图49 draw.bmp展示

#### 结果图像：

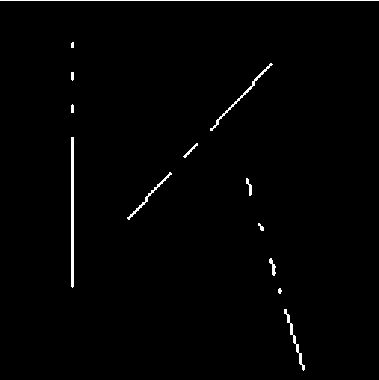
 

图50 draw\_to\_2.bmp展示 图51 draw\_hough.bmp展示

#### 结果分析：

从真彩图转换为灰度图的结果为图50，图50经过霍夫变换得到图51，其中用灰色线标识出霍夫变换检测出的直线。

### 2.9区域标记

#### 基本原理：

利用区域标记分割区域的原理：

图像分割的结果通常是一幅二值图像，所有的目标区域都被赋予同一种灰度值。如果图像中有多个目标区域，并且希望分析各个目标的大小、形状等特征时，就需要对区域加以区分。

区域标记是指对图像中同一连通区域的所有像素赋予相同的标记，不同的连通区域赋予不同的标记。 常用区域标记方法有两种：递归标记和序贯标记。

（1）递归标记（非常费时，适用于并行机处理）

①从左到右，从上到下逐行逐列扫描图像，寻找没有标记的目标点P，给该点分配一个新的标记。

②递归分配同一标记给与P点具有相同灰度的邻域像素。

③直到相互连接的像素均标记完毕，一个连通区域就标上了同样的记号。

④重复步骤①②和③，寻找未标记的目标点并递归分配同一标记给其邻域目标点；若找不到未标记的目标点，则图像标记完毕。

（2）序贯标记

①从左到右、从上到下扫描图像，寻找未标记的目标点P。

②如果P点的左、左上、上、右上4个邻点都是背景点，则赋予像素P一个新的标记；如果4个邻点中有1个已标记的目标像素，则把该像素的标记赋给当前像素P；如果4个邻点中有2个不同的标记，则把其中的1个标记赋给当前像素P，并把这两个标记记入一个等价表，表明它们等价。

③第二次扫描图像，将每个标记修改为它在等价表中的最小标记。

序贯标记算法通常要求对图像进行两次扫描。由于该算法一次仅运算图像的两行，因此当图像以文件形式存储且内存空间不允许把整幅图像全部载入时，也能使用该算法。 它在第二次扫描图像时，利用等价表给同一连通区域的所有像素分配唯一的标记。但是，当图像中的目标区域十分不规则时，会导致庞大的等价表。

#### 输入图像：

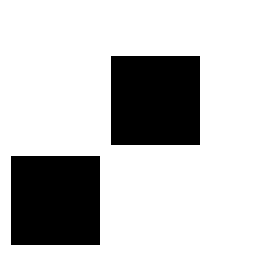
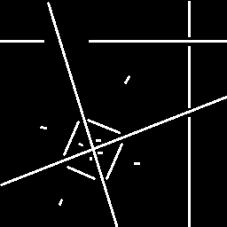
 

图52 region\_mark.bmp展示 图53 pic.bmp展示



图54 col.bmp展示

#### 结果图像：

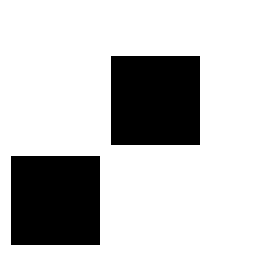
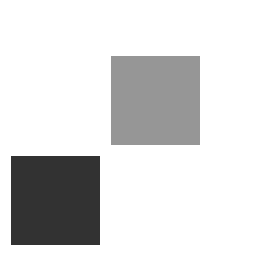
 

图55 region\_mark\_to2.bmp展示 图56 region\_mark\_zone\_marker.bmp展示

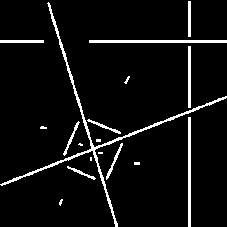
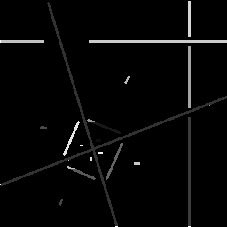
 

图57 pic\_to2.bmp展示 图58 pic\_zone\_marker.bmp展示

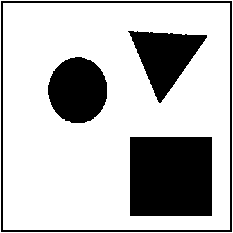
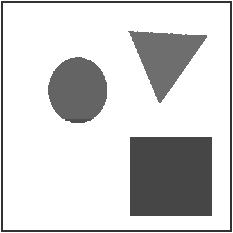
 

图59 col\_to2.bmp展示 图60 col\_marker.bmp展示

#### 结果分析：

（1）图52从真彩图转换为灰度图的结果为图55，图55经过区域标记得到图56，其中用不同的颜色标识出不同的区域。

（2）图53从真彩图转换为灰度图的结果为图57，图57经过区域标记得到图58，其中用不同的颜色标识出不同的区域。

（3）图54从真彩图转换为灰度图的结果为图59，图59经过区域标记得到图60，其中用不同的颜色标识出不同的区域。

### 2.10轮廓提取

#### 基本原理：

轮廓提取的原理：

对于一副背景为白色，目标为黑色的[二值图像](https://so.csdn.net/so/search?q=%E4%BA%8C%E5%80%BC%E5%9B%BE%E5%83%8F&spm=1001.2101.3001.7020)，如果在图中找到一点是黑色，且它的8领域也均为黑色，就说明该点是目标图像的内部点，将其置为白色，表现为掏空状态；否则保持黑色不变，该点是目标的边界点。整幅图像按此原理处理后，便得到轮廓。

#### 输入图像：

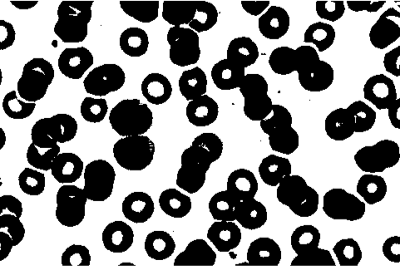
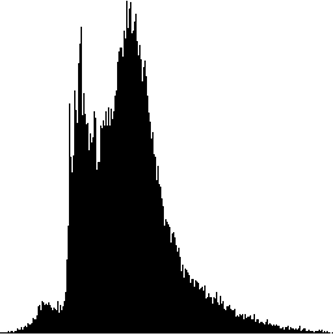
 

图61 circle.bmp展示 图62 hist.bmp展示

#### 结果图像：

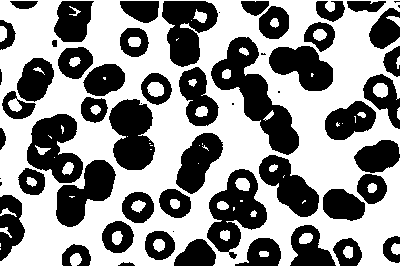
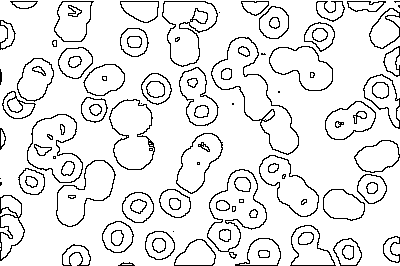
 

图63 circle\_to\_2.bmp展示 图64 circle\_extraction.bmp展示

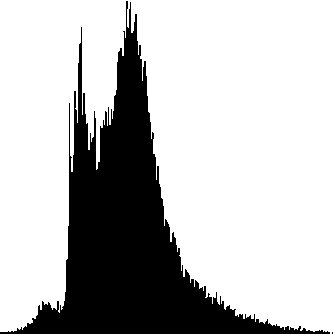
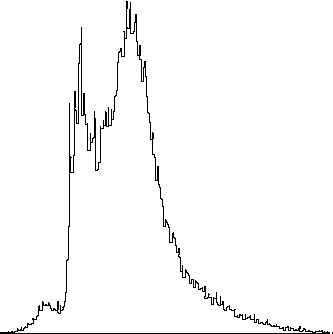
 

图65 hist\_to\_2.bmp展示 图66 hist\_extraction.bmp展示

#### 结果分析：

（1）图61从真彩图转换为灰度图的结果为图63，图63经过轮廓提取得到图64，其中用黑色标识出轮廓。

（2）图62从真彩图转换为灰度图的结果为图65，图65经过轮廓提取得到图66，其中用黑色标识出轮廓。

## 3 总结

通过本门课程的学习，我系统地掌握了部分有关数字图像处理的内容。通过完成本门课程的平时作业和本次大作业，我可以熟练运用BMP文件处理、直方图处理、空间域滤波、图像变换、阈值分割、基于区域的分割、边缘检测、Hough变换、区域标记和轮廓提取的相关知识和算法，使得我对于如何对给定图像进行特定处理有了一定的了解，并且锻炼了我实现各种图像处理功能的能力。

在本门课程的学习中，令我影响最深刻的是经常需要用代码来实现指定算法，但是我总会在此问题上犯难。我总会有算法似乎很简洁，清晰明了，但是自己就是无法实现的痛苦，以及有好几次作业都时逢考试周，我不得不在之后的时间将其补回来。在此期间，从文件读取操作时对文件头部分字节长度的不理解，到之后霍夫变换时很难直接给出准确的编码思路，非常感谢夏勇老师以及同学的帮助，我的代码能力提升良多，并且对算法也有了更深刻的认知。我非常认可这样的授课方式，本来只学习算法理论并不能对其有十分深刻的了解，很有可能过几天直接抛之脑后；但是当真正用代码实现时，我会真正地去思考的原理基于什么样的数据结构、什么样的函数去实现，去切身测验它的可行性，大大增强了这门课的价值。

## 4 格式评价

## 5 程序评价

## 6 附加题：车牌文字提取