实验报告

16340282 袁之浩

1.形态学处理

算法描述：

膨胀（dilation）：定义一个结构模板，模板以中心点对准待处理像素，逐点扫描像素，如果模板中有一个以上的 1 与其对应的像素点相同，待处理像素为 1，否则为 0.

腐蚀（erosion）：同样以结构模板扫描图像，模板中心对准待处理像素。如果模板中所有 1 点与其对应像素相同，则待处理像素为 1，否则为 0.

开变换：先腐蚀，然后膨胀。

闭变换：先膨胀，然后腐蚀。

代码实现：先在 myDilate.m 和 myErode.m 中实现可变换结构中心的膨胀和腐蚀变换，再在 P1.m 中把它们组合起来。

实验结果：

(a)

```
res1 =

     0     0     0     0     0     0     0
     1     1     1     1     0     0     0
     0     1     1     1     0     0     0
     0     1     1     1     1     0     0
     1     1     1     1     1     1     0
     1     1     1     1     1     0     0
     1     1     1     1     1     1     0
     0     0     0     0     0     0     0
```

(b)

```
res2 =

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     1     1     0     0     0
     0     0     1     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

(c)

```
res3 =

     0     0     1     1     0     0     0
     0     0     1     1     1     0     0
     0     0     0     1     1     0     0
     0     0     1     1     1     1     0
     0     0     1     1     1     1     1
     0     1     1     1     1     1     0
     0     1     1     1     1     1     1
     0     0     0     0     0     0     0
```

(d)

```
res4 =

     0     0     0     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     1     0     0
     0     0     0     1     1     0     0
     0     0     0     1     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

(e)

```
res5 =

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     1     1     1     1     0     0     0
     1     1     1     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

res6 =

     0     0     0     1     0     0     0
     0     0     0     1     1     0     0
     0     0     0     0     1     0     0
     0     0     0     1     1     1     0
     0     0     0     1     1     1     0
     0     0     0     1     1     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

```
res7 =

     0     0     0     0     0     0     0
     1     1     0     0     0     0     0
     0     1     0     0     0     0     0
     0     1     1     0     0     0     0
     1     1     1     1     0     0     0
     1     1     1     0     0     0     0
     1     1     1     1     0     0     0
     0     0     0     0     0     0     0

res8 =

     0     0     0     1     0     0     0
     0     0     0     1     1     0     0
     0     0     0     0     1     0     0
     0     0     0     1     1     1     0
     0     0     0     1     1     1     0
     0     0     1     1     1     1     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

(f)

2. 图像二值化

算法描述：

基本全局阈值算法：

1. 选择图像灰度极值的平均值作为阈值 T 的初始值

2. 用 T 分割图像，生成两组像素：G1 由所有灰度值大于等于 T 的像素组成，G2 由所有灰度值小于 T 的像素组成

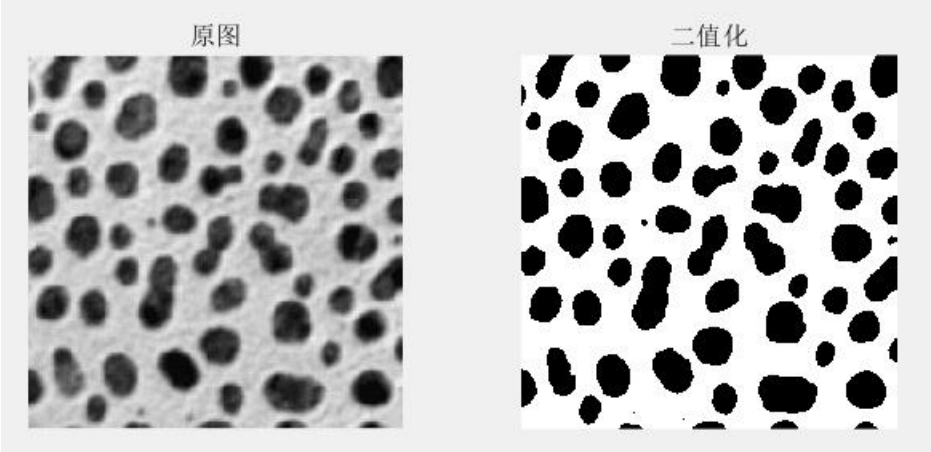
3. 对区域 G1 和 G2 中的所有像素计算平均灰度值 u1 和 u2

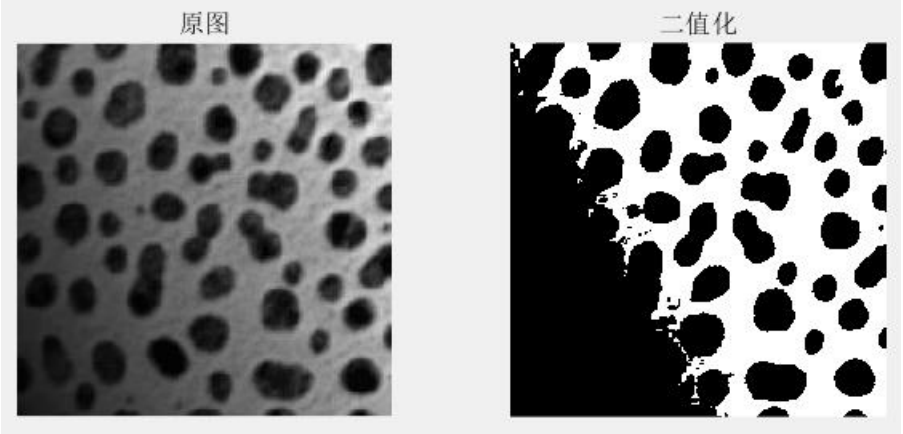
4. 计算新的阈值 T=1/2（u1+u2）

5. 重复步骤 2-4，直到逐次迭代所得的 T 值之差不变

代码实现：算法实现在 threshold.m，P2.m 中调用处理图像

实验结果：





可以看到这种算法对光照均匀的图像效果很好，但是对光照不均匀的图像效果并不理想。

解决方法是采用基本自适应阈值算法，可以解决单一全局阈值存在的问题。

具体步骤是将图像进一步细分为子图像，并对不同的子图像采用刚才的算法处理。在这里我将原图分成了 26*26 的小块，代码实现在 P3.m 中。效果较好。