数字图像处理第五次实验图像分割报告（文末附代码）

1 问题

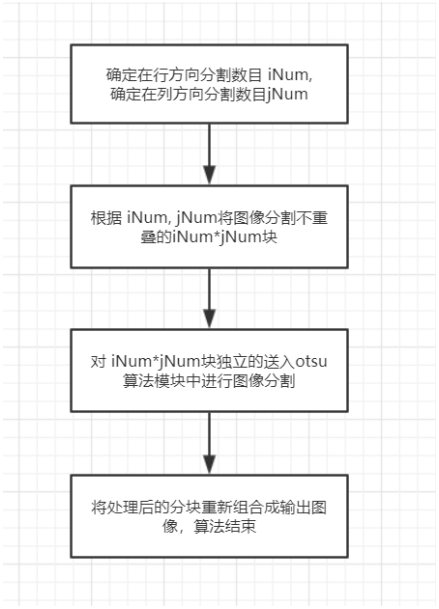
设计图像分割算法完成 SegImg.bmp 图像的分割，分割结果“米粒”像素标 记为 255，“背景”像素标记为 0。

2 问题分析

本次实验进行图像分割的实验，观察整幅图像我发现该图像仅有米粒和背景，而且每个米粒的颜色相近（也就是灰度值相近），因此可以采用单阈值分割的方法来实现图像分割。在所有单阈值分割方法中，我选用已被证明的利用最大化类间方差的 otsu 算法计算阈值。

同时再观察整幅图像，我发现该图像在垂直方向上受光照的影响明显，图像的灰度图存在被光照污染的可能，为了解决这个问题我采用了分块的方案，通过分块我让受光照影响相近的区域为一个整体做图像分割，不同分块之间不重叠，通过这种方案我很好地解决了光照对图像分割的影响。

综上，本次图像分割实验采用的分割方案如下：

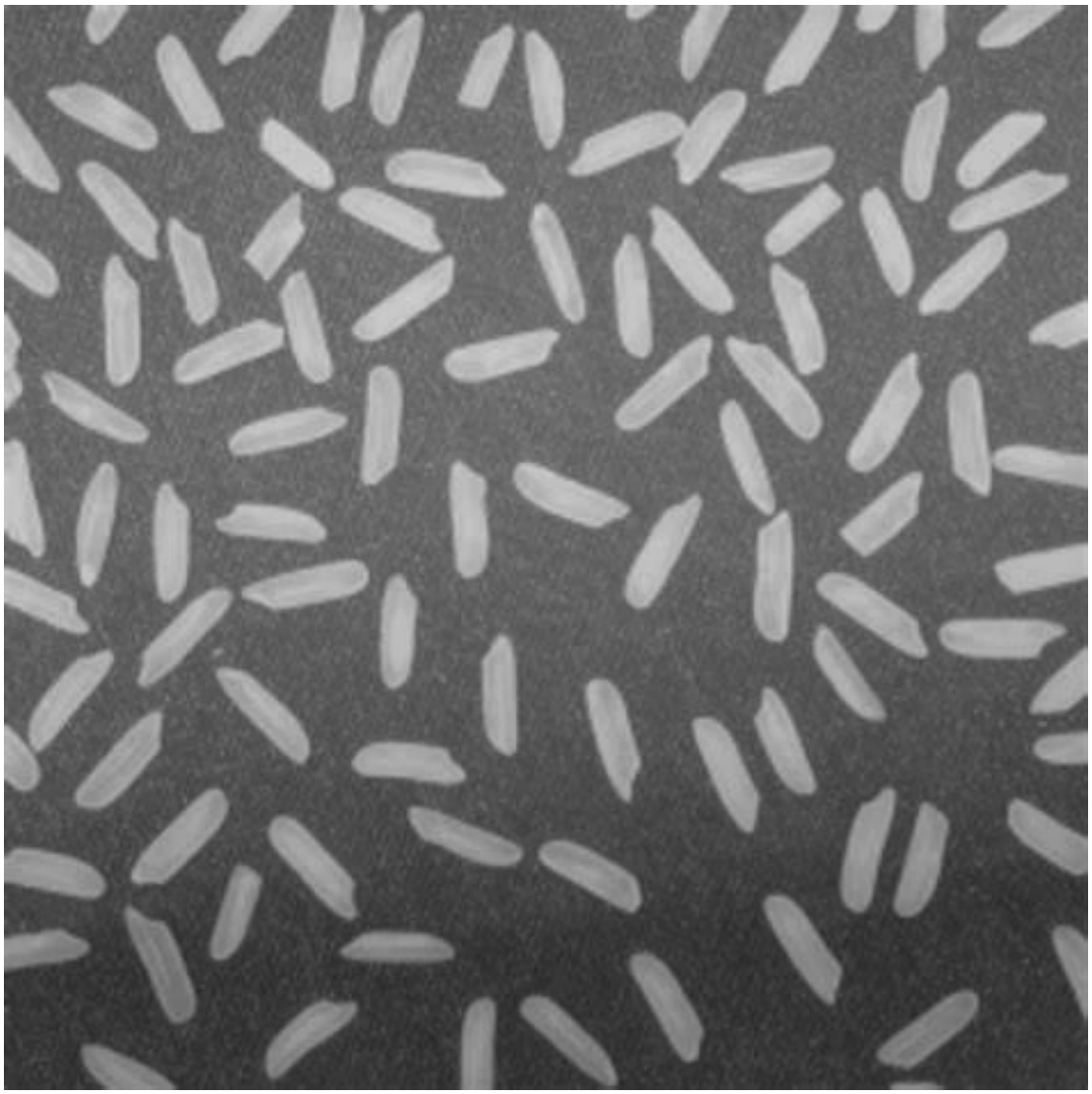



其中 iNum 表示在行方向 iNum 等分，jNum 表示在列方向 jNum 等分

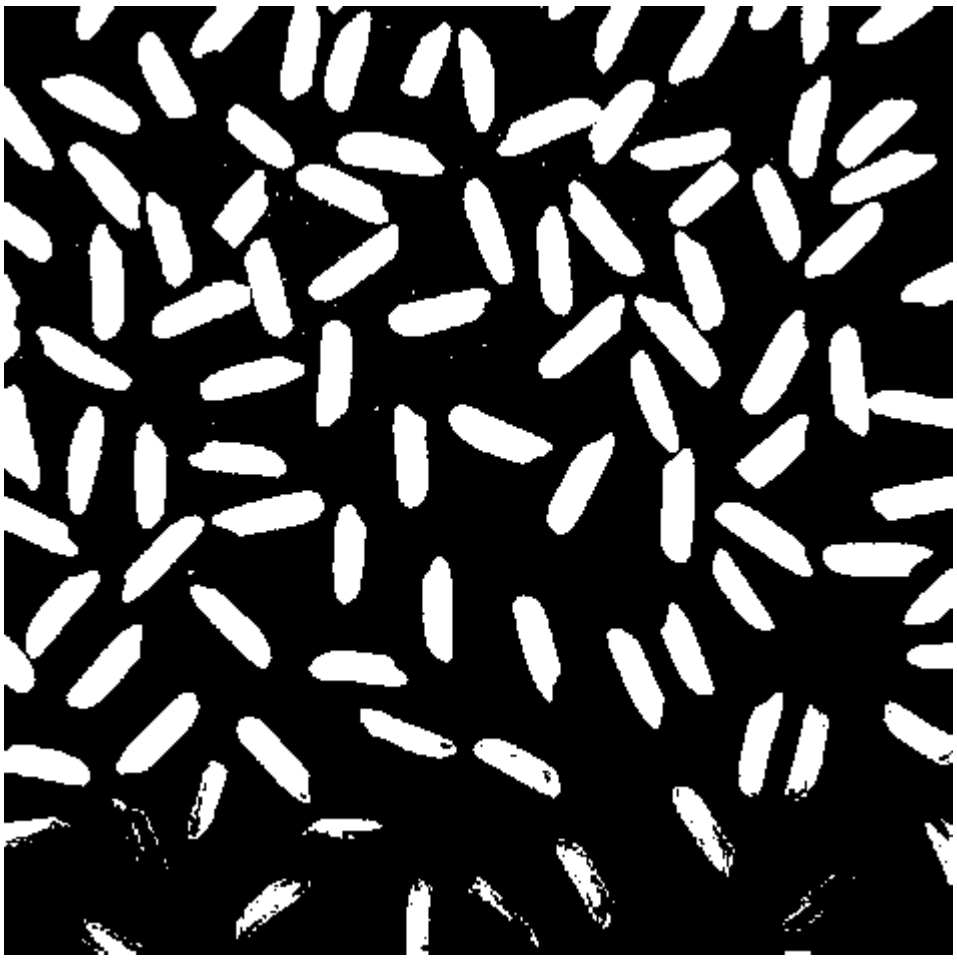
在接下来的实验中，我将展示 直接使用 otsu 和 分块使用 otsu 的区别，并对实验结果进行分析。

3 实验结果：

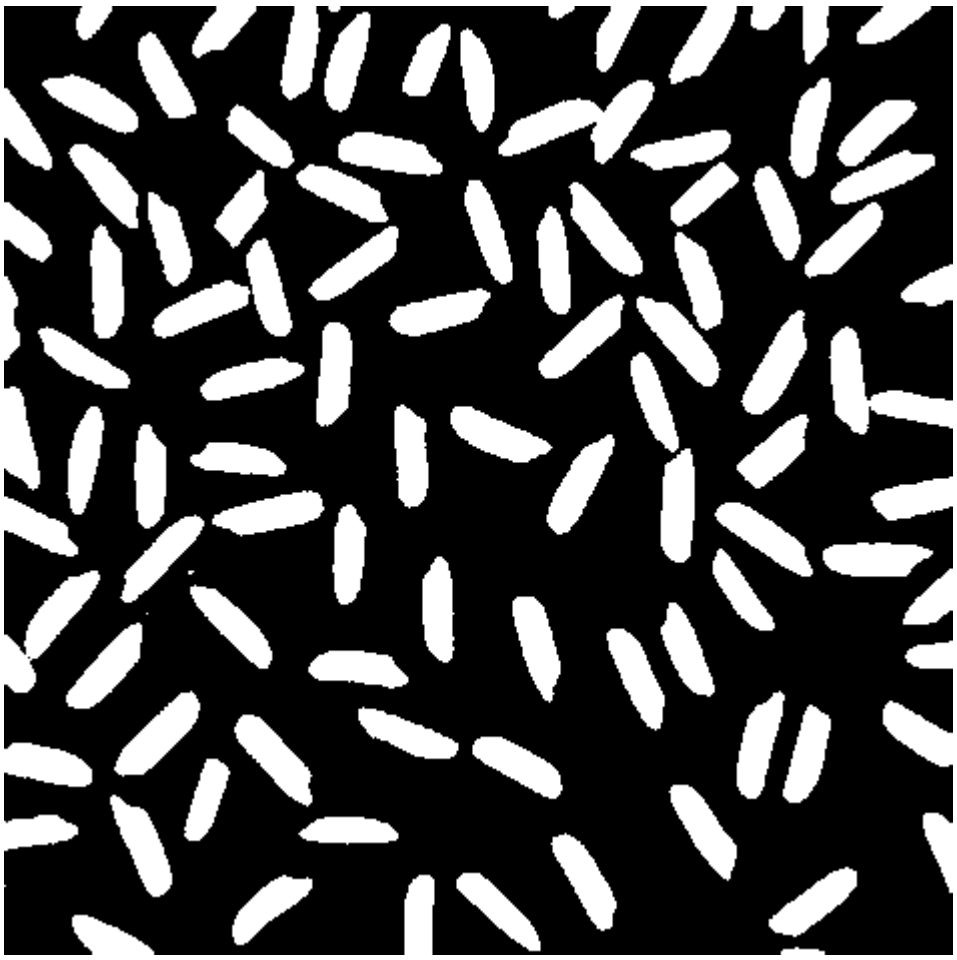
原图像： （命名为 test.bmp）



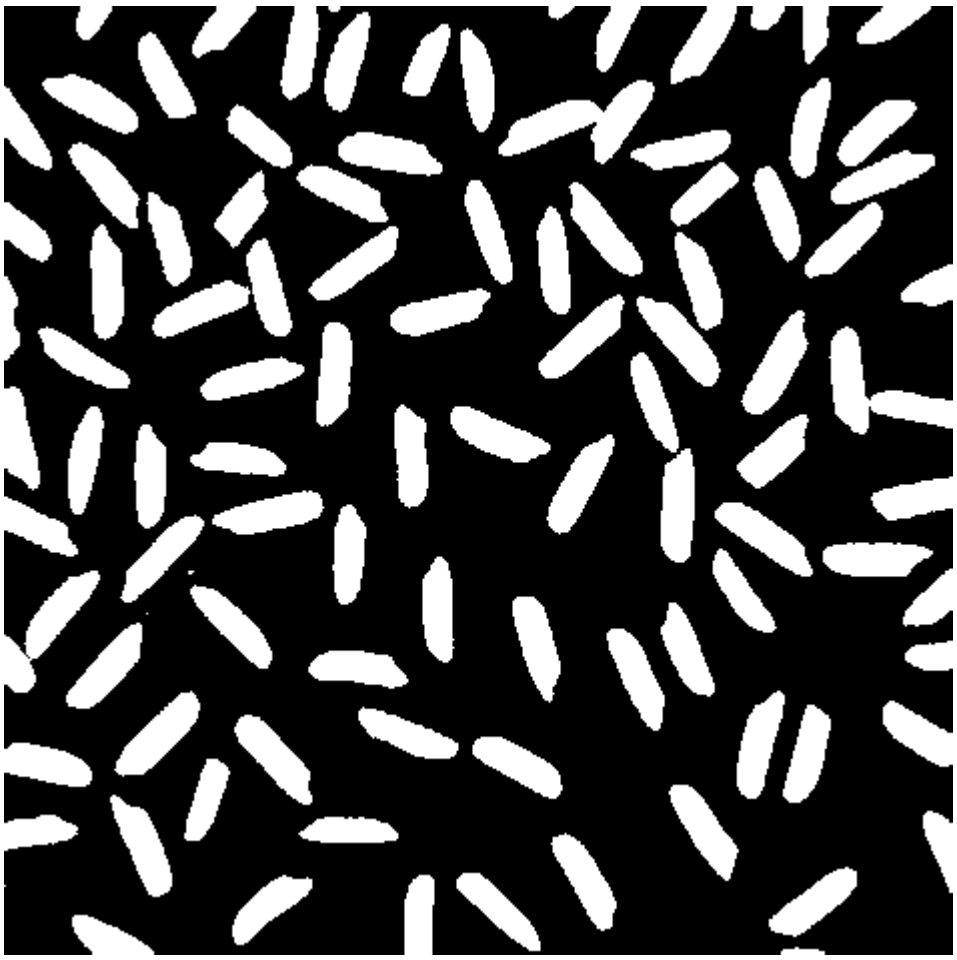
iNum = 1, jNum=1 时的结果（不进行分块）



iNum = 8, jNum =1:



iNum =8, jNum=8:



4 实验结果

分析实验结果，我们可以看到如果不分块直接应用 otsu 算法，图片底部的分割效果很不好。观察原图，我们可以看到原图的底部受光照影响最大。光照导致底部的直方图和中上部的直方图混合，导致整幅图像的直方图的波谷不是特别明显，所以底部的分割效果不尽如人意，同时导致中上部分出现了噪声（小白点）。

从原图上来看，原图主要在垂直方向光照变化明显，因此我设置 iNum=8, jNum =1 只在垂直方向进行分块，让光照均匀的区域单独进行 otsu 分割。从实验结果上看极大地改善了分割效果。对比原图可以说很好地完成了分割任务。这是因为当我们分块之后，不同光照强度区域不会相互影响，从而提升了分割效果

而当我们对列也进行分块的时候（设置 jNum=8）我们看到分割效果和上一步相近。这是因为原图在列方向（横向）的光照基本均匀，所以得到的效果几乎相同。

除上述，我们还看到分块之后的图像分割结果仍然存在噪声，这是因为原图本来就存在噪声的原因，这可以通过先对图像进行滤波降噪处理，再进行图像分割来解决。

（所有的实验结果在 result 下，根据 参数 iNum,jNum 的顺序对图片进行了命名）

5 代码

```
import os
import numpy as np
import cv2 as cv

def otsu(input):
```

```
    '''

    :param input: 输入图片是一个numpy 数组, 深度为1

    :return: output: 分割之后的结果
    '''
    hist,edges =  np.histogram(input,256,[0,256],density=True ) #hist 是概率密度分布,计算概率分布, 这一步自己实现也很简单只需要遍历 input 统计即可
    cdf = hist.cumsum()                                          # cdf 将概率密度求和, cdf[k] 就是 i 从 0 到 k (包括k ) pi 的求和
    mean_global =  np.sum(np.arange(256)*hist )                  #计算全局均值
    var_k_array = np.zeros(256)
    for k in range(256):
        mean_k = np.sum(np.arange(k+1)*hist[0:k+1] ) #计算 m_k  ;m_k 为  i 从 0 到 k (包括k) i*pi 的求和
        if cdf[k] ==1 or cdf[k] ==0:
            continue
        temp_var = np.power((mean_global*cdf[k]-mean_k),2)/(cdf[k]*(1-cdf[k]))
        var_k_array[k] = temp_var
    max_var_betweet_class = np.max(var_k_array)      # 求出最大的类间方差
    threshold  = np.average(np.where(var_k_array==max_var_betweet_class))
# 将所有等于类间最大方差的k值 求平均 计算出分割阈值
    mask = (input > threshold)              #输入的图片 所有灰度值 > 阈值的设置为 true
    output = np.zeros(input.shape, dtype = np.uint8)      #初始化0矩阵
    output[mask]=255                                # 将 =1 的值设置为 255
    return output
```

```
def devide_block_ostu( input, i_divided_num, j_divided_num ):
    '''

    将图片分块用 ostu 算法处理 返回分割后的图像

    :param input:   输入灰度图, numpy 数组

    :param i_devided_num:    i 轴等分数

    :param j_divided_num:    j 轴等分数

    :return:
    '''
    output = np.zeros(input.shape,dtype = np.uint8)
    block_height = input.shape[0]// i_divided_num
    block_width = input.shape[1]//j_divided_num

    #分块执行 ostu 算法

    for i in range(i_divided_num):
        i_begin = i * block_height
        i_end = i_begin + block_height if i < i_divided_num - 1 else
input.shape[0]  # 最后一次取全图 防止遗漏

        for j in range(j_divided_num):
            j_begin = j * block_width
            j_end = j_begin + block_width if j < j_divided_num - 1 else
input.shape[1]  # 最后一次取全图 防止遗漏

            output[i_begin:i_end,j_begin:j_end] =
otsu(  input[i_begin:i_end,j_begin:j_end] )
    return output


if __name__ =='__main__':
    BASE_PATH = os.path.dirname(os.path.abspath(__file__))
    RESULT_PATH = os.path.join(BASE_PATH, "result")
    if not os.path.exists(RESULT_PATH):
        os.mkdir(RESULT_PATH)
    import argparse

    parser = argparse.ArgumentParser()
    parser.add_argument('--img', help="the path of the input img")
    parser.add_argument('--iNum', help='the num of in')
    parser.add_argument('--jNum', help='the num of jn')
    # parser.add_argument('--kernal_size', help='the size of the kernal')
```

```
    args = parser.parse_args()

    ################## 图片读取模块

    img_name = os.path.split(args.img)[-1]
    img_name, ext = os.path.splitext(img_name)

    input = cv.imread(args.img, flags=cv.IMREAD_GRAYSCALE)  # 读取灰度图片

    # input = cv.imread(args.img, flags=cv.IMREAD_COLOR) #读取 rgb 图片

    ############################ 处理模块

    output = devide_block_ostu(input,int(args.iNum),int(args.jNum))

    #########图片输出模块

    img_name = img_name + "_segmentation_" + str(args.iNum)+ "_" +
str(args.jNum)
    img_name = img_name + ext
    img_name = os.path.join(RESULT_PATH, img_name)
    cv.imwrite(img_name, output)
```