图像分割

一、 实验目的

读入一幅图像（可以是灰度图像和 RGB 图像），用 K-Means 算法对该图像进行分割，并最终对比显示分割前后的图像。

二、 实验环境

编程语言：Python3.7

操作系统：Windows10

代码编辑器：vscode

调用 Python 库：sys、numpy、matplotlib

三、 算法原理

图像限速点的颜色和位置反映了基本的信息，通过使用 K-Means 算法对像素点聚类可以得到若干簇以及对应的均值，将所有属于该簇的像素点的颜色值用均值来替换，可以起到图像分割的效果。

K-Means 图像分割算法的流程如下：

1、 随机生成 K 个像素点的像素值作为聚类中心（对应到 K 个簇）。

2、 计算图像所有的像素点到这 K 个聚类中心距离（这里的距离函数后面会详细介绍），并选取最近的中心点作为其所属的簇。

3、 分别重新计算 K 个簇的均值作为新的聚类中心，如果新的聚类中心和之前的一样，则终止聚类，否则跳转到第二步。

4、 根据最终得到的 K 个聚类中心和簇，将所有像素点的像素值用聚类中心的值来代替，进而得到分割后的图像。

要想有较好的图像分割效果需要合适的距离计算函数，这里采用了颜色空间的欧氏距离：$dist = \sqrt{\left(\sum_{i=1}^{n} c_i^2\right)}$其中 n 表示颜色通道（灰度图像为 1，RGB 图像为 3），c_i表示对应通道的颜色值。

四、 代码流程

K-Means 算法的实现流程：

1、 首先用矩阵 clusters 来保存原图像中每个像素点所属的聚类（用 0~K 随机数进行初始化），用 means 矩阵来保存 k 个聚类中心

```
clusters = np.random.randint(0, k, img.shape[: 2])
means = np.empty((k, 1 if len(img.shape) == 2 else img.shape[2]))
```

2、 接着开始进行 K 均值聚类的算法流程，先计算 K 个簇的均值作为聚类中心，然后对每个像素点计算和它最近的聚类中心，并将对应的 clusters 矩阵置为相应的均值。同时还要对这次聚类是否有像素点改变了聚类中心进行记忆，如果有改变，则循环进行该步骤，否则聚类终止。

```
while True:
    for i in range(k):
        index = np.where(clusters == i)
        if index[0].size > 0:
            means[i] = np.mean(img[np.where(clusters == i)], axis=0)

    changed = False
```

```
    for x in range(clusters.shape[0]):
        for y in range(clusters.shape[1]):
            dists = dist(x, y, img, means)
            pos = np.argmin(dists)
            if pos != clusters[x, y]:
                clusters[x, y] = pos
                changed = True

    if not changed:
        break
```

3、 最后根据 clusters 矩阵生成新的分割后的图像

```
target = np.uint8(means)[clusters].reshape(img.shape)
```

五、 实验结果

影响 K-Means 图像分割算法效果的因素有很多，比如 K 值的选取，距离计算函数的选取等。这里只是简单的对图像分割进行了建模，因此分割效果很一般。下面展示在不同图像类别和 k 值选取的时候的图像分割结果。

Figure 1 对 RGB 图像分别用 K=3 和 K=5 的分割效果

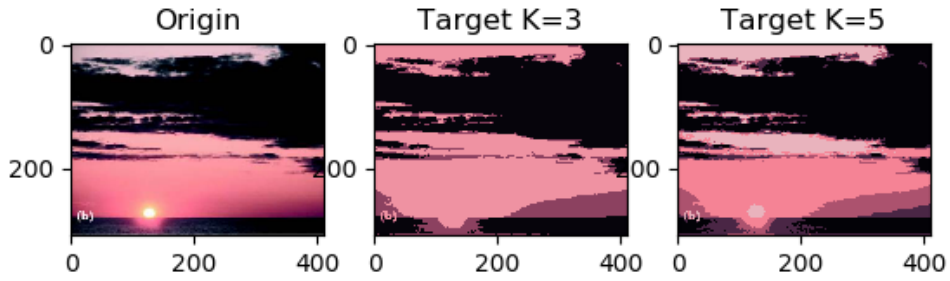
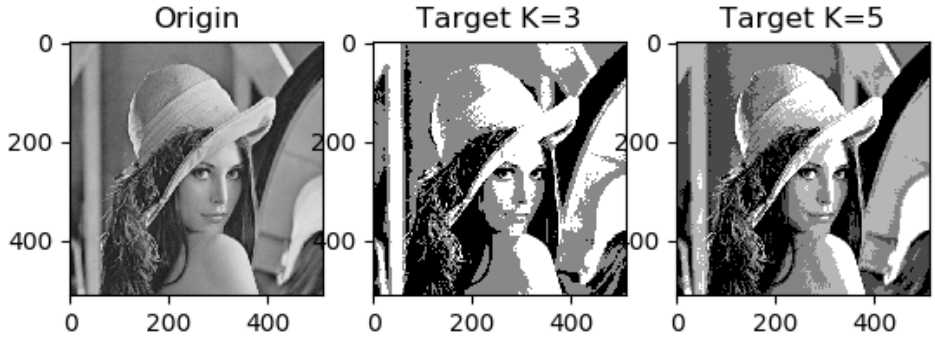


Figure 2 对灰度图像分别用 K=3 和 K=5 的分割效果



六、 总结分析

K-Means 算法是聚类算法，也可以用于图像分割，但是存在计算复杂度过高，分割效果不明显的缺点，一个更好的选择是 Mean-Shift 算法。在这个实验中通过 K-Means 算法的图像分割程序编写进一步掌握了聚类算法的使用和加深了对图像的理解，在后面的实验中会对图像分割问题进一步探讨，用更好的分割算法进行建模。