小作业三：图像分割

一、实验要求

• 读入一幅图像

• 进行图像分割

• 算法不限，建议采用K-means或mean shift

• 显示结果

二、算法原理

1.基本思想

使用K-means算法进行图像分割，是经典的基于划分的聚类方法。以空间中的k个点为中心进行聚类，对最靠近它们的对象归类，类别数为k。不断迭代，逐次更新各聚类中心的值，直至得到最好的聚类结果。

最终的k个聚类具有以下特点：

- 各聚类本身尽可能的紧凑，而各聚类之间尽可能的分开。
- 该算法的最大优势在于简洁和快速。

2.算法描述

假设要把样本集分为k个类别
（1）初始时随机地从样本集 D={X1,X2,...,Xm} 中选择k个点作为k个类的初始聚类中心；
（2）在第 i 次迭代中，对任意一个样本点，求其到聚类k个中心的距离，将该样本点归到距离最短的聚类中心所在的类；
（3）利用均值等方法更新该类的聚类中心；
（4）对于所有的k个聚类中心，如果利用2，3步的迭代法更新后，损失函数低于设定的阈值，则迭代结束，否则继续迭代。

应用K-means算法时应当注意这些局限性：

- **需要人工预先确定初始K值**
- **K-means只能收敛到局部最优，受初始聚类中心的选择影响很大**
- **当集群存在不同尺寸，不同密度，非球形的簇时，K-means存在问题，不适合发现具有非凸形状的簇**
- **对噪声数据和异常值很敏感**

三、基本思路

编写以下函数：

1.K-means函数，函数的参数是输入图像，聚类中心数目，迭代阈值，返回所有像素的标签矩阵。

```
def k_means(input_signal, center_num, threshold):
  input_signal_cp = np.copy(input_signal) # 输入信号的副本
  input_row, input_col = input_signal_cp.shape # 输入图像的尺寸
  pixls_labels = np.zeros((input_row, input_col)) # 储存所有像素标签
  # 随机初始聚类中心行标与列标
  initial_center_row_num = [i for i in range(input_row)]
  random.shuffle(initial_center_row_num)
  initial_center_row_num = initial_center_row_num[:center_num]
  initial_center_col_num = [i for i in range(input_col)]
  random.shuffle(initial_center_col_num)
```

```
  initial_center_col_num = initial_center_col_num[:center_num]
  # 得到当前的聚类中心
  present_center = []
  for i in range(center_num):
    present_center.append(input_signal_cp[initial_center_row_num[i],
initial_center_row_num[i]])
  pixls_labels = classifer(input_signal_cp, present_center)
  num = 0 # 用于记录迭代次数
  while True:
    pre_centet = present_center.copy() # 储存前一次的聚类中心
    # 计算当前聚类中心
    for n in range(center_num):
      temp = np.where(pixls_labels == n)
      present_center[n] = sum(input_signal_cp[temp].astype(int)) /
len(input_signal_cp[temp])
    # 根据当前聚类中心分类
    pixls_labels = classifer(input_signal_cp, present_center)
    # 计算上一次聚类中心与当前聚类中心的差异
    loss = loss_function(present_center, pre_centet)
    num = num + 1
    print("Step:"+ str(num) + "  Loss:" + str(loss))
    # 当损失小于迭代阈值时，结束迭代
    if loss <= threshold:
      break
  return pixls_labels
```

2.classifer函数，通过当前的聚类中心，给输入图像分类，参数为输入图像和聚类中心，返回标签矩阵。分类的依据就是计算每个像素与所有聚类中心的差平方，对于某点来说，把差异最小的中心点的类别作为这个点的类别，并用标签矩阵存储起来。

```
def classifer(intput_signal, center):
  input_row, input_col= intput_signal.shape # 输入图像的尺寸
  pixls_labels = np.zeros((input_row, input_col)) # 储存所有像素标签
  pixl_distance_t = [] # 单个元素与所有聚类中心的距离，临时用
  for i in range(input_row):
    for j in range(input_col):
      # 计算每个像素与所有聚类中心的差平方
      for k in range(len(center)):
        distance_t = np.sum(abs((intput_signal[i, j]).astype(int) -
center[k].astype(int))**2)
        pixl_distance_t.append(distance_t)
      # 差异最小则为该类
      pixls_labels[i, j] = int(pixl_distance_t.index(min(pixl_distance_t)))
      # 清空该list，为下一个像素点做准备
      pixl_distance_t = []
  return pixls_labels
```

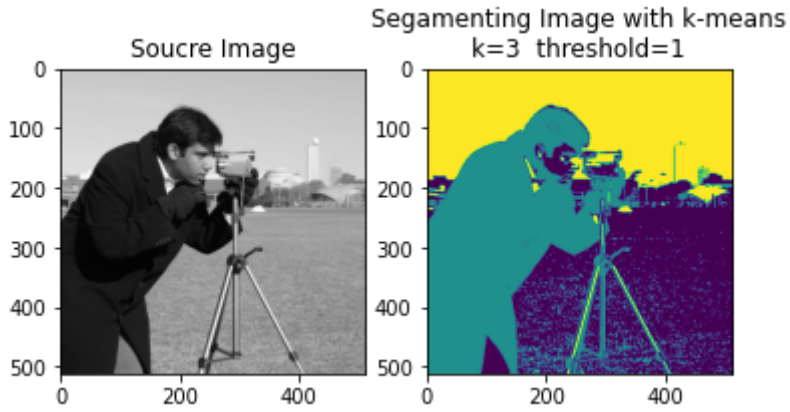
3.loss_function函数，计算上一次与当前聚类中的差异（像素差的平方和），参数为当前聚类中心和上一次聚类中心，返回损失值，当损失函数小于我们规定的阈值时，我们认为分类结束了。

```
def loss_function(present_center, pre_center):
  present_center = np.array(present_center)
  pre_center = np.array(pre_center)
  return np.sum((present_center - pre_center)**2)
```

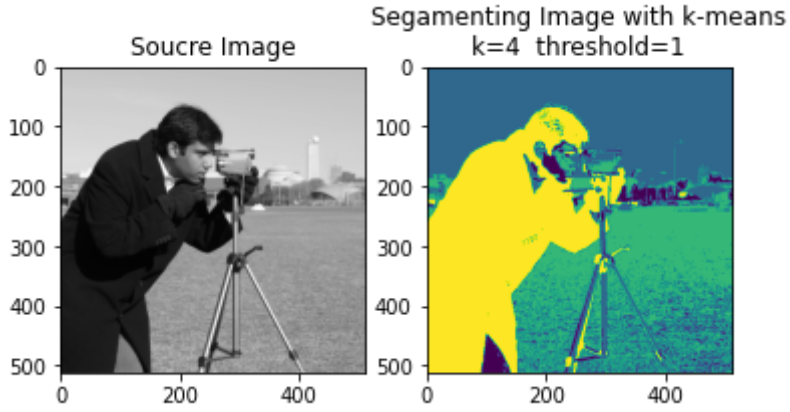
四、实验结果

我们使用灰度camera图像（通常用于分割和去噪的示例）来对算法进行测试。

设置 k = 3 时，人物及其相机，草坪和天空被分割成了三类，分割效果不错。



设置 k = 4 时，需要迭代更多的次数且分割的效果没有 k = 3 时的效果好。



因此对于这副图像，我们令 k = 3，迭代阈值为1，即迭代更新后上一次与当前聚类中的差异（像素差的平方和），若小于1则停止迭代。

参考资料

(32条消息) 聚类算法实例：K-Means实现图像分割*wangqiangianya*的博客-*CSDN*博客k-means图像分割

python基于K-means聚类算法的图像分割*python*脚本之家 (jb51.net)