实践作业1——基于聚类的图像分割

1. 实现说明：

按照实验要求采用了Kmeans聚类和HAC聚类算法进行图像分割。

1、Kmeans聚类

首先，使用python实现了一个kmeans函数，传入参数data，k，其中data代表需进行图像分割的图像数据，假设原数据的高宽分别为height，width，则data为(height\*width, 3)，k为需要将数据通过聚类算法分成几类。函数中，首先随机从data中抽取k个像素点作为聚类中心，存储到centers的列表中。clusters为存储k个类的像素点的列表，label为长度height\*width的列表，存储每个像素点的label值，与data对应。通过遍历data中的每个像素点，计算该像素点到centers中每个聚类中心的距离，并将该像素点合并到距离最小的簇中。遍历完所有的像素点之后，更新centers中的像素值，更新为每个簇中的像素点的平均值。一直循环下去，直到新的聚类中心和原来的聚类中心没有区别才停止。

然后就是对聚类的结果进行上色处理，我们想要的是背景变为[0, 0, 0]的黑色，聚类出来的结果变为[255, 255, 255]的白色。所以将上面kmeans函数返回的label进行遍历，将label[0]像素点所在的那一簇作为背景类，因为label[0]是图片的左上角的点。这样就可以将背景和前景通过黑色和白色的不同表示区分开来。

2、HAC聚类算法

HAC聚类算法实际上是通过计算不同类别点的相似度创建一颗有层次的树结构，在这颗树中，树的底层是原始数据点，顶层是一个聚类的根节点，本次实现的HAC聚类算法使用的自底向下的方法。所以首先实现ClusterNode类，该类作为聚类树的节点，类中保存该节点所属的簇，该节点的左右节点，两个节点的距离，以及这个节点的叶子节点个数。之后实现了一个HAC聚类的类Hierarchical，初始化传入参数k，为聚类的簇的个数。类中包含三个函数fit()，calc\_label()，leaf\_traversal()。fit()函数首先初始化节点为ClusterNode类，然后遍历所有的节点，计算两个节点之间的距离，并选择一个最小距离的两个点将其合并为同一个簇。并且将合并的那一个簇作为一个新节点初始化为ClusterNode类，并将那两个节点删除。一直循环下去，直到聚类的簇的个数等于k的值。之后通过calc\_label()，leaf\_traversal()，将聚类树中的节点打上标签。leaf\_traversal()为递归遍历树节点。

然后就是取labels中的第一个类作为背景类，将与背景类相同的label值的像素变为黑色，其余变为白色。

但是需要注意的是，HAC聚类算法的时间复杂度为O(n^3)，时间复杂度比较高，如果使用实验给的原始图像，就会非常的缓慢，因此，将原图resize为包含1000到2000个像素点的图片，在对其进行聚类。

1. 结果截图：

Kmeans聚类的分割结果图如图1所示，指标结果如图2所示。

图1 kmeans聚类





图2 kmeans聚类指标

HAC聚类的分割结果图如图3所示，指标结果如图4所示。

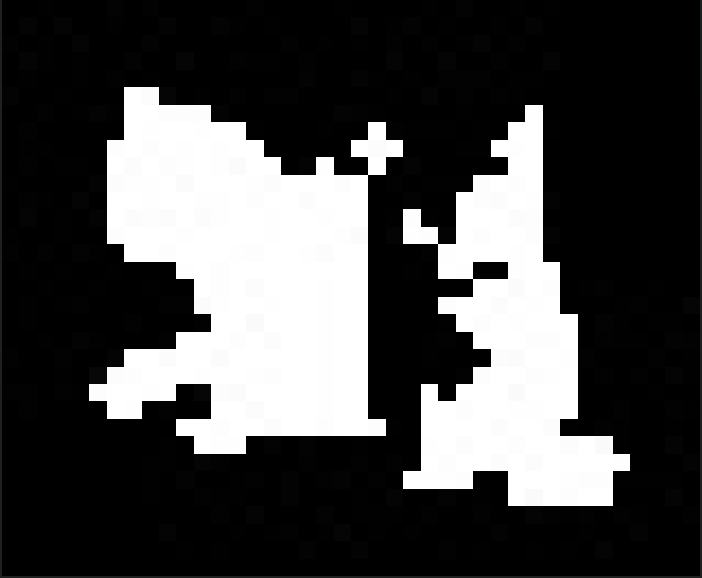
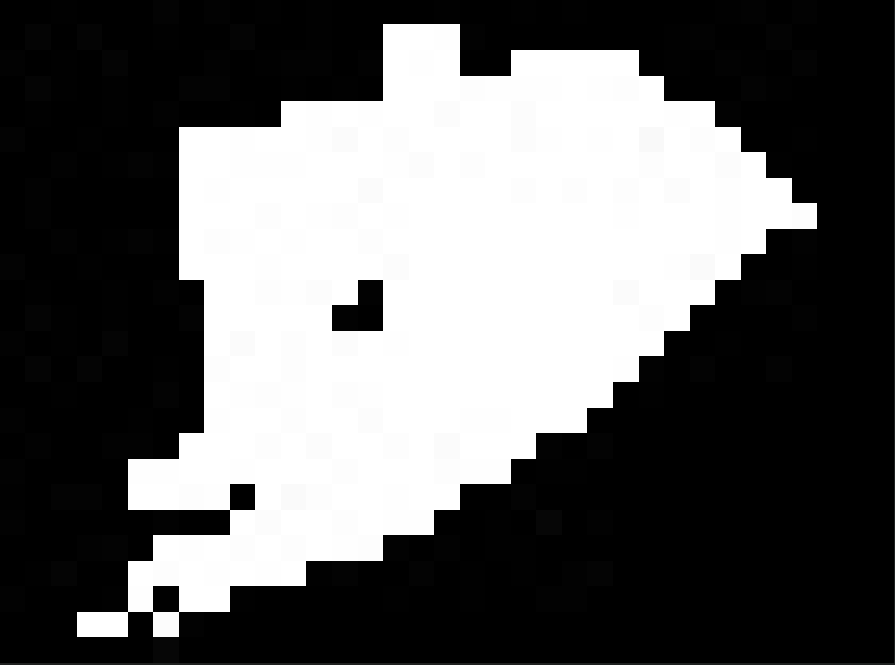
 

图3 HAC聚类





图4 HAC聚类指标

1. 运行说明：

在运行文件时需要注意修改图片路径。

在运行HAC.ipynb之前，要先对图片进行resize操作，因为HAC聚类算法的时间复杂度太高。

实践作业2——图像拼接

1. 实验说明：

使用opencv库中自带的SIFT算法，对两张图片提取sift特征点。在两幅图像中进行最近邻搜索进行特征匹配，并规定最大比较次数为50。匹配的结果记为match，并且遍历match，过滤掉一些匹配度不强的特征点。如果匹配的特征点个数大于设定好的阈值，则针对第二张图片，计算视角变换矩阵，对第二章图片就行视角变换处理，以更好的和第一张图片进行拼接。使用cv2.findHomography计算出从图像一到图像二的变换矩阵M和掩码mask。然后再使用cv2.warpPerspective对图像二进行视角变换，并将它扩大为两张图片宽的和，结尾warpImg。

最后开始处理图像一和warpImg的左右两端再什么地方重叠。记录好重叠的左右两边的列的索引，之后对图像一和warpImg进行加权拼接。在二者没有重合的地方则不需要加权处理，直接拼接即可，在二者重合的地方，计算出当前列数的权重，之后将两张图片的像素进行加权求和，实验证明，这样进行加权处理，效果会好很多。

除此之外，还实现了将两张图像的特征点就行匹配后的结果图展示，在match.py文件中实现。

1. 结果截图：

实验结果如下图所示。图5为图像拼接的最终结果。图6为两张图像的特征点匹配的结果展示。



图5 图像拼接结果

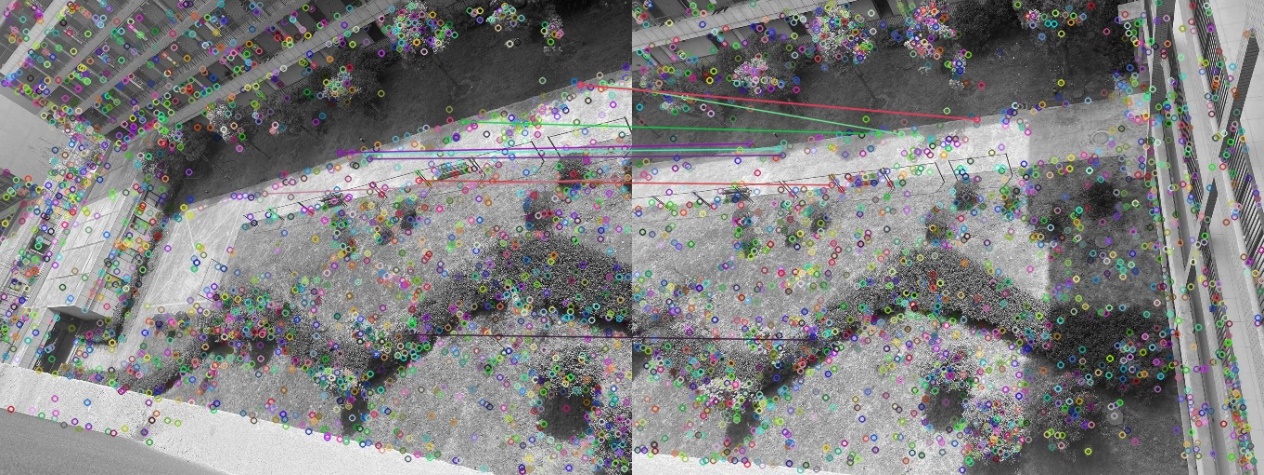


图6 特征点匹配展示

1. 运行说明：

运行SIFT. ipynb即可获得图片拼接的结果。

运行match. ipynb可以获得特征点匹配的结果展示。