《计算机视觉》期末作业实验报告

18342057 刘倍延

本作业测试图像为{57.png, 157.png, ..., 957.png}。

文件结构：

```
-CV_final
    |--data
        |--test
            |--cluster_fig：聚类展示图
            |--gt
            |--imgs
            |--rgb_feature_data：rgb特征矩阵文件
            |--seg：图像分割结果
            |--seg_gt：图像分割标记结果
        |--train：训练集
            |--gt
            |--imgs
            |--rgb_feature_data：rgb特征矩阵文件
            |--seg：图像分割结果
            |--seg_gt：图像分割标记结果
    |--raw_data：题目提供的原始数据
    |--report：报告
        |--img：报告图像
        |--Report.pdf
    |--src：源代码
        |--q1：问题一源代码
        |--q2：问题二源代码
        |--q3：问题三源代码
```

题目1

1.1 题目要求

实现基于 Graph-based image segmentation 的图像分割方法，通过设定恰当的阈值将每张图分割为 50~100 个区域，同时修改算法要求任一分割区域的像素个数不能少于 20 个 。结合 GT 中给定的前景 mask，将每一个分割区域标记为前景或背景 。同时要求对测试图像子集生成相应处理图像的前景标注并计算生成的前景 mask 和 GT 前景 mask 的 IOU 比例。

1.2 实现原理

本题目中，对P. Felzenszwalb, D. Huttenlocher论文的复现参考了Github中作者salaee的开源实现；区域标记部分为自己实现。

1.2.1 基于图的图像分割

图像分割功能实现于`main.py`中的`segment()`函数中。首先分别对输入图像的RGB三个通道进行高斯平滑，随后根据输入图像构造图，然后进行分割，最后将较小的区域进行合并。

图的构造实现于`main.py`中的`build_graph()`函数中。该函数主要构造了一个大小为（width * height * 4, 3）的`edges`矩阵用于表示图。`edges`矩阵记录了所有相邻像素点相连构成的所有边。对`edges`矩阵中的一个元素`i`，`edges[i,0]`和`edges[i,1]`通过简单的哈希`int(y*width+x)`表示边`i`的两个顶点，`edges[i, 2]`记录该边的权值。边权值为RGB空间中两顶点的距离。

基于图的分割实现于`segment_graph.py`中的`segment_graph()`函数中。该函数主要构造一个并查集森林。依据P. Felzenszwalb, D. Huttenlocher论文中所提出的方法，该函数首先对`edges`矩阵中的元素按边权值进行降序排序，并初始化并查集`u`和各区域阈值`threshold`。随后，函数遍历每一条边，如果由这条边连接的两个区域的阈值均大于该边的权重，则将两个区域进行合并。

并查集森林定义于`disjoint_set.py`中的`universe`数据结构中。`universe`主要维护一个大小为(num_vertices, 3)的`elts`矩阵（num_vertices 为像素节点数量）；对节点`i`，`elts[i,0]`记录它在所属区域的并查集中所处的层数，`elts[i,1]`记录其所属区域的大小，`elts[i,2]`记录它的父节点。`universe`主要包括三个方法，`find`方法寻找一个节点`x`所属区域的并查集的根节点，用以表示它所属的区域；`join`方法合并根节点`x`和`y`所代表的两个区域；`all_comp`方法查找当前记录的所有区域，返回一个记录所有区域根节点的列表。

1.2.2 k 值选择

根据题目要求，每张图需要分割为 50~100 个区域；而区域大小同设定的 k 值相关。为寻找合适的 k 值，在`main.py`的`select_k`函数中逐步尝试从 80 到 100 的 k 值设置，对每个 k 值进行 100 次分割，统计每个 k 值中不符合要求的区域数量。根据统计结果， k = 80 时不符护要求的区域数量最小，故随后实验中 k 取 80。

```
(tensorflow) C:\Users\Barytes\Desktop\CV_final\src\q1>python main.py
50      7
55      7
60      6
65      3
70      2
75      1
80      1
85      2
90      2
95      3
(tensorflow) C:\Users\Barytes\Desktop\CV_final\src\q1>python main.py
74      35
75      34
76      32
77      31
78      28
79      28
80      28
```

1.2.3 区域标记

根据题目要求对各区域进行前景或背景的标记，该功能主要实现于`main.py`的`mark_comp()`函数中。函数首先对输入图像进行分割，得到分割后的并查集森林`u`。随后它统计森林中每一个区域在gt中被标记为前景(255)的像素数量，据此判断每一个区域属于前景或后景。最后，该函数将前景区域的所有像素置为255，后景区域所有像素置为0，生成一个新的gt图像gt_seg，并保存于`gt_seg/`文件夹中。

以57.png为例，其gt和gt_seg如下所示：

1.2.4 计算iou

该部分实现于`main.py`中的`calc_iou`函数中。对每一张图像，将其gt和gt_seg逐像素进行对比，统计在gt和gt_seg中均为前景的像素数量为r1 ∩ r2，在gt或gt_seg中为前景的像素数量为 r1 ∪ r2。

1.3 结果

iou结果已经呈现于`data/test`和`data/train`文件夹中的`iou.txt`中。测试图像的iou比例如下所示：

```
pic r1&r2   r1|r2   iou
157.png 1610.0  2948.0  0.5461329715061058
257.png 4871.0  5758.0  0.8459534560611324
357.png 1992.0  3085.0  0.6457050243111832
457.png 7169.0  9597.0  0.7470042721683859
557.png 1628.0  4743.0  0.3432426734134514
57.png  4689.0  5899.0  0.7948804882183421
657.png 9386.0  11299.0 0.8306929816797947
757.png 6173.0  8337.0  0.7404342089480629
857.png 6329.0  8927.0  0.7089727792091408
957.png 339.0   1494.0  0.22690763052208834
```

题目2

2.1 题目要求

对训练集中的每一张图提取512维归一化 RGB 颜色直方图特征，同时结合问题 1，对训练集中的每张图进行分割（分割为 50~100 个区域），对得到的每一个分割区域提取512维归一化 RGB 颜色直方图特征，将每一个区域的特征定义为区域颜色直方图和全图颜色直方图的拼接，因此区域特征的维度为 2*512=1024 维，采用 PCA 算法对特征进行降维取前 50 维。训练集中的每张图被分割为 50~100 个区域，每个区域可以提取 50 维特征，且根据问题 1，每个区域可以被标注为类别 1（前景：该区域 50%以上像素为前景）或 0（背景：该区域 50%以上像素为背景），选用任意分类算法（SVM， Softmax，随机森林， KNN 等）进行学习得到分类模型。 最后在测试集上对每一张图进行测试（将图像分割为 50~100 个区域，对每个区域提取同样特征并分类），根据测试图像的 GT， 分析测试集区域预测的准确率。

2.2 实现原理

本题目是一个分类任务，其完成主要分为两部分；第一部分为从训练集和测试集图像中提取RGB特征生成数据集，第二部分为使用分类算法对数据集进行分类。本实验使用的分类方法为随机森林。

2.2.1 vertice-pixel哈希表

本部分在`disjoint_set.py`中新增了一个辅助数据结构`vp_hash_table`；由于题目1中将像素节点哈希为`int(y*width+x)`以便在并查集中记录，因此新增该哈希表以便对节点坐标和其哈希值进行转换。该数据结构维护一个长度为`height*width+width`的列表，每个元素记录对应的坐标元组`(x,y)`。它提供两个方法，`vertice2pix`方法返回输入的节点哈希值的对应坐标，`pix2vertice`返回输入节点坐标的哈希值。

2.2.2 生成数据集

数据集生成实现于`data_generation.py`的`make_blobs()`函数中。对每一张图片，它首先进行分割，得到其并查集`u`，随后根据`u`计算该图像对应的特征矩阵，再根据该图像对应的gt_seg计算其标签向量。将所有图片的特征矩阵和标签向量进行拼接，经过PCA算法降维后得到数据集。

2.2.2.1 计算特征矩阵和RGB特征

RGB直方图特征的计算实现于`data_generation.py`的`calcRgbHistFeature`函数中。该函数支持计算图像的RGB直方图特征向量，以及计算图像经过遮罩后的区域的RGB直方图特征。对没有输入遮罩的图像，该函数将图像变形为`(height*width,3)`的一列像素；对输入了遮罩的图像，调用`reshapeMaskedImg`函数，将遮罩区域的像素置于一个列表中返回。由此，需要计算的部分被重塑为一列像素。对所有需要计算的像素，统计其在8 * 8 * 8的RGB颜色空间中各点的像素数量，并返回所得特征向量。

计算特征矩阵功能实现于`data_generation.py`的`calcFeatureMatrix()`方法中。该方法对一张图像计算其特征矩阵。首先对输入图像计算512维归一化RGB直方图特征。随后分割图像得到的每一个区域计算512维归一化RGB直方图特征，将上述两个512维直方图特征拼接为1024维向量作为该区域的特征向量；最后将所有区域的特征向量拼接为该图像的特征矩阵并返回。

计算一个区域的RGB直方图特征首先需要根据该区域生成一个同图像等大的遮罩，区域内的像素为白色（255），其余像素为黑色（0）；随后将该遮罩同图像一起传入`calcRgbHistFeature`函数中。计算遮罩的功能实现于`comp2Mask()`函数中。

计算所得的RGB直方图特征矩阵通过np.save以npy文件形式保存于`rgb_feature_data`文件夹中。

2.2.2.2 计算标签向量

该功能实现于`data_generation.py`的`calcLabelVec`函数中。对图像标签向量的计算需要借助对应的gt_seg。对分割图像得到的每一个区域，若其根节点在gt_seg中的像素值为255，则该区域被标记为前景（1），否则标记为后景（0）。函数返回计算得到的标签向量。

2.2.2.3 PCA降维

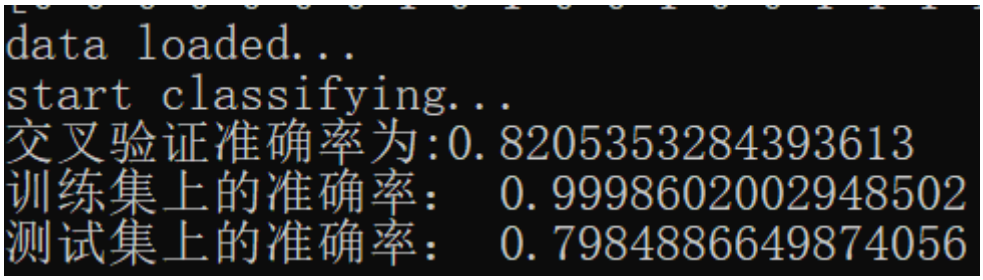
pca降维使用了scikit-learn库中的PCA方法，设置n_components = 50；对上述步骤得到的宽度为1024维的特征矩阵，调用`fit_transform()`方法得到降维后的数据。

`make_blobs()`函数最终返回经过降维后的特征矩阵以及标签向量。

2.2.3 分类

本题使用了scikit-learn库中的`RandomForestClassifier`进行分类。在获取了训练集数据和测试集数据后，使用训练集数据训练随机森林分类器，然后再在两个数据集上测试分类器的分类准确率。

2.3 结果



题目3

3.1 题目要求

对每张测试图提取 Sift 特征点及 Sift 特征，采用 PCA 算法将特征降维至 10 维，以每个 sift 特征点为中心截取 16 * 16 的 patch（超出边界的 patch 可以舍去），计算归一化颜色直方图（4 * 4 * 4=64 维），将两个特征拼接（sift 特征和颜色直方图特征），这样每个 sift 特征点表示为 74 维向量，采用 k-means 聚类算法将该图像所有 sift 特征点聚为 3 类，并依次将各聚簇中的 patch 组合形成一张展示图（例如总共有 N 个 sift 点，对应 N 个 patch， 展示图中需要将同一聚簇的patch 按顺序粘贴，不同聚簇用分割线分开）。要求每张测试图生成一张可视化聚类展示图。

3.2 实现原理

3.2.1 SIFT特征计算

SIFT特征计算利用了opencv库实现的SIFT特征检测。首先将输入图像转换为灰度图像，随后利用`sift=cv2.xfeatures2d.SIFT_create()`，`sift.detectAndCompute(gray, None)`得到SIFT特征点列表`kps`和SIFT特征矩阵`des`。假定检测得到n个SIFT特征点，`des`的大小为(n, 128)，每个SIFT特征点对应有一个128维的特征向量。

3.2.2 RGB直方图特征计算

遍历检测得到的所有SIFT特征点，将以其为中心点得到的patch不完全在图像边界内的SIFT特征点除去，对剩余的每个特征点截取一个以其为中心点的16 * 16的patch（对patch的截取实现于`main.py`的`calcPatch`函数中）。随后对每个patch计算RGB直方图特征向量，将所有patch的特征向量组合为图像的RGB直方图特征矩阵。

3.2.3 特征矩阵

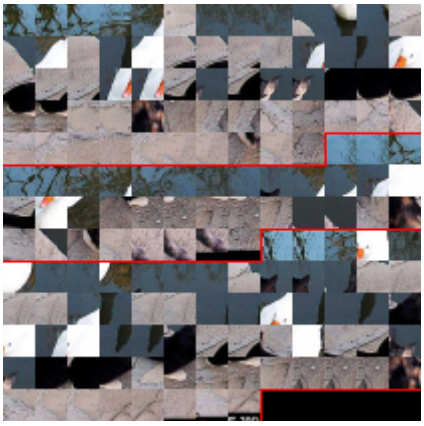
将得到的SIFT特征矩阵使用PCA降维至10维，随后将RGB直方图特征矩阵和SIFT特征矩阵按列拼接，得到n * 74 的特征矩阵

3.2.4 KMeans聚类

本题利用了scikit-learn库的KMeans实现，将上述所得的特征矩阵使用KMeans算法聚为3类。最后将同一聚簇的patch 按顺序粘贴，不同聚簇用分割线分开，生成一张聚类展示图。

3.3 结果

聚类展示图的结果存储于`data/test/cluster_fig`中，以57.png为例：



最后的几个纯黑色patch是将图像补足为矩形所用。