模式识别第二次作业

超像素分割

姓名： 晋一卓
学号:22336106

一. 实验目的

1. 熟悉 Graph based Segmentation 和 SLIC 算法
2. 熟悉两种方法的应用场景

二. 实验内容

1. 编写 Graph based Segmentation 和 SLIC 两种算法的代码
2. 使用两种方法的代码对图片进行处理，并比较和分析参数不同所带来的不同结果
3. 分析两种方法的优缺点

三. Graph based Segmentation

（一）实验原理

“高效图像分割”算法（GBS)的核心原理如下：

1. **图建模**：将图像中的每个像素视作一个图节点；
2. **最小生成森林**：将所有边按照颜色差异从小到大进行排序；依次考察每一条边，若该边连接的两个像素属于不同的连通分量，且边权小于各自的内部差异则将两个分量进行合并，构成一棵“最小生成森林”；
3. **小区域后处理**：对于初步结果中像素数< min_size 的小组件，强制其与至少一个临界组件进行合并，以消除过小的孤立区域；
4. **可视化**：直观展示结果。

（二）关键函数分析

1. **并查集结构**

```
class UnionFind:
    def __init__(self, size):···
    def find(self, u):···
    def union(self, u, v, weight):···
```

• def find：通过路径压缩将整条链“扁平化”，加速后续查找

• def union：按尺寸将小树合并到大树，更新内部最大差；异只有节点属于不同集合时才真正合并，合并时边权作为新的内部差异阈值

2. **def build_edge**

```
    for i in range(height):
        for j in range(width):
            if j < width-1:  # 右…
            if i < height-1:  # 下…
            if i < height-1 and j < width-1:  # 右下…
            if i < height-1 and j > 0:  # 左下…
    edges.sort(key=lambda x: x[0])
    return edges
```

- 用 8 邻域将图像转成加权无向图，每条边的权重就是相邻像素的颜色距离
- 只枚举“右、下、右下、左下”四个方向，避免重复添加边
- 所有边按照颜色差异进行升序排序，便于后续的贪心合并

3. def GBS

- step1：高斯平滑：sigma 控制滤波强度，去除噪声，弱化细节

```
#高斯滤波
smoothed = cv2.GaussianBlur(image, (0,0), sigmaX=sigma, sigmaY=sigma)
smoothed = img_as_float(smoothed)
height, width, _ = smoothed.shape
num_pixels = height * width
```

- step2：核心合并：遍历排序后的所有边，根据（原理中讲解的）阈值规则判断合并

```
if root_u != root_v:
    threshold = min(uf.int_diff[root_u] + k/uf.size[root_u],
                    uf.int_diff[root_v] + k/uf.size[root_v])
    if weight <= threshold:
        uf.union(u, v, weight)
```

- step3：小区域后处理

```
if 0<=nx<height and 0<=ny<width and mask[nx,ny] != label:
    uf.union(pt[0]*width+pt[1], nx*width+ny, 0)
    mask = np.where(mask == label, mask[nx,ny], mask)
```

- step4：relabel：确保输出掩码中每个像素都被标注为组件根节点

```
mask = np.zeros((height, width), dtype=np.int32)
mask[i,j] = uf.find(i*width + j)
```

（三）实验结果分析

为方便分析不同参数带来的不同结果，在生成分割图像的同时额外生成 可以输出“分割数目”和“平均超像素面积”的 stats.csv，程序运行得到的数据如下：

sigma	k	min_size	分割数目	平均面积（像素）
0.3	200	50	4	135606
0.3	200	100	3	180808
0.3	500	50	1	542424
0.3	500	100	1	542424
0.3	1000	50	1	542424
0.3	1000	100	1	542424
0.5	200	50	5	108484.8
0.5	200	100	3	180808
0.5	500	50	1	542424
0.5	500	100	1	542424
0.5	1000	50	1	542424

0.5	1000	100	1	542424
0.8	200	50	6	90404
0.8	200	100	3	180808
0.8	500	50	1	542424
0.8	500	100	1	542424
0.8	1000	50	1	542424
0.8	1000	100	1	542424

对以上数据进行分析可知：

1. k 越大，越容易合并，分割块数急剧下降：当 k 从 200 增加到 500 时，阈值便宽松，几乎所有边都满足合并条件，最终只剩一个区域；

2. min_size 增加，提出更大的“小区域”，导致块数减小，块平均面积增加：在 k=200，σ 固定时，min_size 从 50 增加到 100，超像素数量大约减半，平均面积相应增大；

3. σ 越大，平滑效果越强，小噪声边界减少，更多块进行合并，块数减小：然而，可能出现：平滑后，各像素之间的颜色差异变得更加均匀，触发更多的“小区域”，min_size 又将其合并为更大的区域，导致 σ 增加，块数反而增加（如 σ 从 0.3 增加到 0.8，而 k=200 时块数从 4 降到 6）。

四. SLIC

（一）实验原理

“SLIC”的核心原理如下：

1. 网格初始化：将图像像素等分为超像素，并计算步长 S,在每个 S*S 的网格中心直接取点作为初始聚类中心，记录 lab 颜色；

2. 迭代聚类：计算每个聚类中心到边长为 2S 的窗口内像素的距离，比较并更新想读标签 labels 与对应的最小距离，根据分配的标签，累加每个簇内的所有像素坐标与 lab 值，除以像素数后得到新的中心位置与颜色。

3. 可视化输出分割结果

（二）关键函数分析

1. def initialize_clusters:中心初始化

```
def initialize_clusters(self):
    centers = []
    offset = self.step // 2
    for i in range(offset, self.height, self.step):
        for j in range(offset, self.width, self.step):
            L, a, b = self.lab[i, j]
            centers.append([i, j, L, a, b])
    self.clusters = np.array(centers, dtype=float)
```

2. def assign_labels:像素分配

```
dc = np.sqrt((sub_lab[:,:,0] - Lc)**2 +
             (sub_lab[:,:,1] - ac)**2 +
             (sub_lab[:,:,2] - bc)**2)
ds = np.sqrt((xs - ci)**2 + (ys - cj)**2)
D = np.sqrt(dc**2 + (sc * ds)**2)
```

```
mask = D < np.inf
```

3. def update_clusters:中心更新

```
self.clusters[mask, :2] = new_centers[mask, :2] / counts[mask, None]
self.clusters[mask, 2:] = new_centers[mask, 2:] / counts[mask, None]
```

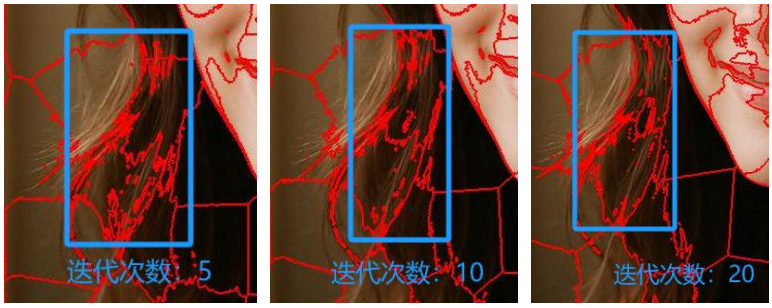
（三）实验结果分析

改变“迭代次数”与“超像素数”，分别对其进行比较，可以得到：

1. 迭代次数（5、10、20）

为方便分析，我们重点关注不同迭代次数下蓝框标注出来的区域：

对局部进行放大，可以得到 5、10、20 次迭代下的细节对比图：



• 对比发现，随着迭代次数的提升，超像素边界越平滑，簇中心迁移，分割更加精细，边界也更加自然（5→10→20）。

• 当迭代书超过一定阈值后，效果趋于稳定，继续迭代的提升效果不明显（5→10 的迭代效果明显比 10→20 要好）

2. 超像素数（100、200、400）

对比发现：超像素数越少，区域的面积越大，数量越少。随着超像素数增加，区域数量越多，捕捉到的细节也越多。

五. 实验方法比较

（一）Graph based Segmentation(GBS)

1. 优点：

• 算法直接给予像素之间的颜色差异构建图，可以自适应地发现任意形状地区域边界，不受格网的约束；

• 通过阈值参数 k 控制合并松紧，相较于 SLIC 省去事先规定超像素划分个数的步骤；
• 细节保留较好。

2. 缺点：

• 需要构建边并对所有边进行排序，并需要并查集合并，计算复杂度高，速度慢，在验证此算法时能明显体会到运行时间之长；
• 对参数敏感，需要对 k, σ，min_size 多个参数同时调试来得到满意结果；
• 分割结果形状不均匀，不利于特征提取。

（二）SLIC

1. 优点：

• 效率高，迭代次数少，算法复杂度也较低；
• 分割结果规则紧凑，利于后期特征提取；
• 参数直观且易于调试（超像素数，紧凑度）

2. 缺点：

• 迭代次数影响分割效果：迭代次数不足时边界不够贴合，迭代过多会造成额外开销，且效果不一定好。
• 需要预先确定分割的超像素数量，不适合动态场景等，分割精度较低。