

综合作业三 实验报告

2017011010 杜澍滢 自 71

实验要求

- (1) 实现 SLIC 超像素分割算法和交互式分割 GUI
- (2) 显示 SLIC 算法的中间过程（聚类中心、边界等）
- (3) 尝试不同的超像素数，分析其对对象分割性能和速度的影响
- (4) 尝试改进对象分割性能

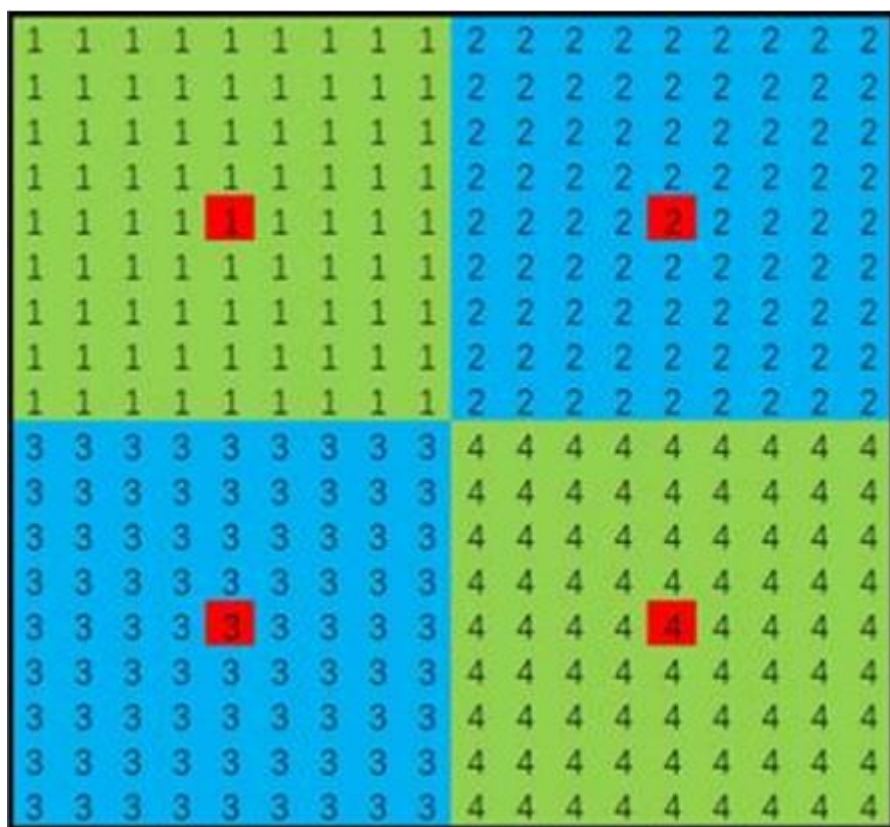
基本原理及代码实现

SLIC 算法的基本思路是将输入图像从 RGB 空间转到 CIE-lab 空间，对每个像素生成一个五维向量 $C[L, a, b, x, y]$ ，其中 (L, a, b) 为其在 CIE-lab 空间的颜色值， (x, y) 则为其坐标。用两个像素的向量距离来度量它们的相似性，距离越大向量越小。实现时先生成 K 个种子点，在每个点周围一个小范围空间内搜索距其最近的若干像素，将他们与该种子点归为一类，然后再分别计算各个超像素的聚类中心，找新的中心周围距离最近的像素群，再次更新聚类和聚类中心，依此迭代直到达到迭代次数为止。下面介绍具体步骤和代码实现方法：

(1) 初始化聚类中心

设待处理图总像素为 N ，事先希望得到的超像素数为 k ，首先得到步距为 $S =$

$\sqrt{\frac{N}{k}}$, 再根据原图尺寸对此 S 值进行一点微调, 以最终为整数的 S 值划分超像素
并将它们的中心位置置为初始的聚类中心。



(2) 迭代更新距离和中心

用二维矩阵 C 来存储上文提到的五项参数, 在计算距离时要同时考虑图像的颜色信息和空间信息, 公式如下图所示:

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2},$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2},$$

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2}.$$

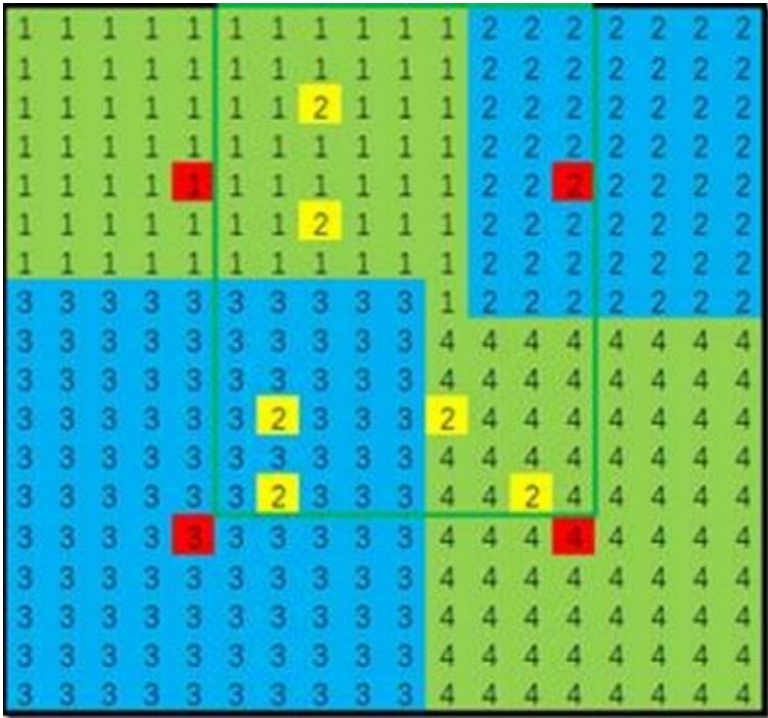
其中 m 表示空间和颜色相对重要性的度量, 当 m 大时, 空间邻近性更重要, 得到的超像素更紧凑; 当 m 小时得到的超像素更紧密地粘附到图像边界,

但具有较小的规则尺寸和形状，建议的 m 范围为[1, 40]。

设定一个距离矩阵 d 来存储各个像素与其对应聚类中心的距离 (d 初始化为所有元素均为无穷大)；设定一个标记矩阵 L 来存储各个像素点对应的聚类标记 (L 初始化为所有元素均为-1)。更新矩阵时仅在原聚类中心周围 $2S \times 2S$ 的像素范围内,按照上面的公式计算每个点的距离值,如果比先前更小则更新为此小值,然后将其标记为属于此聚类 (这里的(L, a, b)使用的是均值)，这样完成一次迭代后，通过计算均值得到新的聚类中心坐标(x, y)用以进行下一次迭代。

后续处理

论文中提到的“orphaned pixels”即下图中黄色部分这种在一个与自己标记不同的聚类中孤立存在的小区域。



这样的结点会影响分割效果，需要通过形态学处理去除。这一部分的实现步骤如下：

- (1) 给定一个半径值，然后利用`strel('disk')`算出一个中间中间参数`se`（对于面积小于此值所确定面积的分区，它们的标记值最终都将变为最近分区的标记值，这个参数用于之后的形态学处理）
- (2) 用一个`list`来存储所有不相邻（用的是4近邻）的区域组合，例如，若上面算法得到的标记图`L`为：

1	1	1	4	4	4	2	2	2
1	3	1	4	4	2	2	5	2
1	1	1	4	4	4	2	2	2

那么`list`的内容就应该为`{1, 2}`和`{3, 4, 5}`

- (3) 对`list`中的每一项做二值化，以`{3, 4, 5}`为例，应该得到如下所示的中间结果：

0	0	0	1	1	1	0	0	0
0	1	0	1	1	0	0	1	0
0	0	0	1	1	1	0	0	0

- (4) 利用前面得到的`se`对此结果做`imopen`，以上图为例，效果应为将两个孤立的1变为0，然后用上图结果减去本步结果得到：

0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0

- (5) 通过（3）（4）两步最终能得到一张和`L`大小相同，孤立区域为1其他部分为0的`mask`，对此`mask`利用`bwdist()`将孤立区域的标记值全部置为与其最近的区域的标记值
- (6) 更新`L`（以上操作改变了总标记数）

尝试改进

Adaptive-SLIC, or ASLIC, adapts the color and spatial normalizations within each cluster. As described in Section III-B, S and m in Eq. 3 are the assumed maximum spatial and color distances within a cluster. These constant values are used to normalize color and spatial proximity so they can be combined into a single distance measure for clustering. Instead of using constant values, ASLIC dynamically normalizes the proximities for each cluster using its maximum observed spatial and color distances (m_s, m_c) from the previous iteration. Thus, the distance measure becomes

$$D = \sqrt{\left(\frac{d_c}{m_c}\right)^2 + \left(\frac{d_s}{m_s}\right)^2}. \quad (6)$$

As before, constant normalization factors are used for the first iteration, but the algorithm subsequently keeps track of the maximum distances for each cluster. The advantage of this approach is that the superpixel compactness is more consistent, and it is never necessary to set m . This comes at the price of reduced boundary recall performance, as shown in Fig. 4(a).

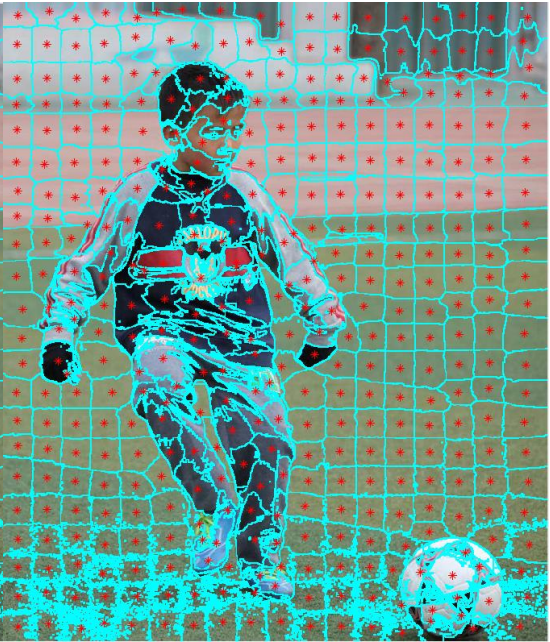
改进部分我将计算距离的方法改为如上图所示，即在第一次迭代时仍然使用常量的 m_c 和 m_s ，在后续的迭代中则使用前次迭代得到的最大颜色距离和最大空间距离，结果对比在下面说明。

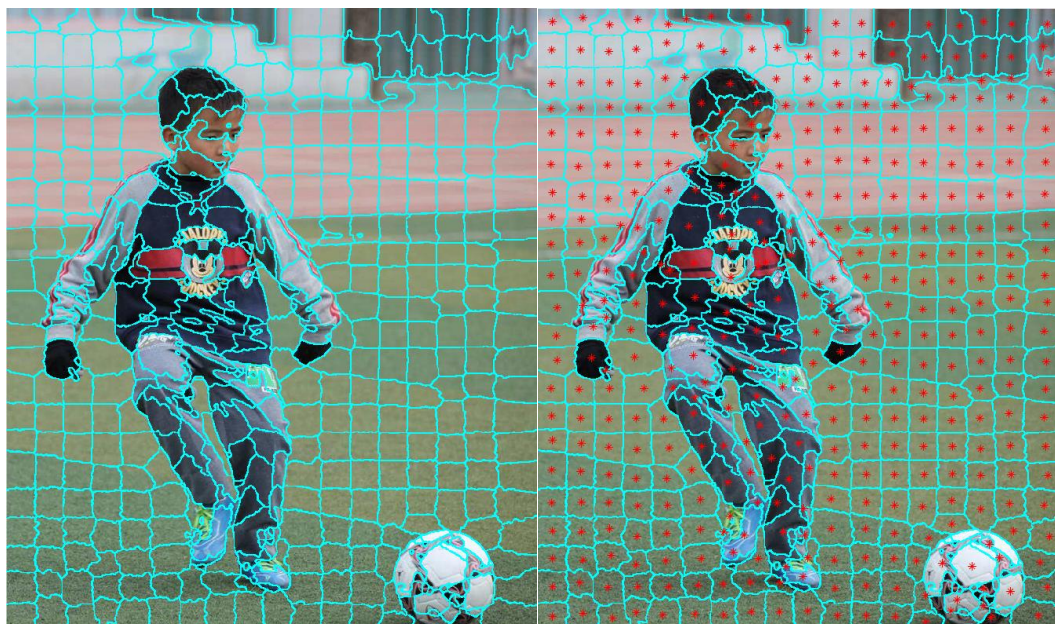
实验结果

(1) 显示 SLIC 算法的中间结果

这一部分可以通过交互界面显示及调整参数，下面贴一个例子。

读图 ‘1.jpg’， $k=400$, $m=10$ ，半径=2，迭代次数=10，在迭代 2 次、5 次和迭代完成后的边界图及标注了聚类中心的边界图如下(运行 processing_result.m 可以看到)：





(2) 实现交互式分割 GUI

GUI 界面如下图所示：



可以选择的参数有 k , m 和迭代次数，半径设定为 2.生成时按下前景键开

始用 `ginput()`取点，取完后按下回车，利用 `impoly()`函数将折线显示在图上：



然后再选择背景：



按下生成键，同时显示 SLIC 结果和最终结果，注意这里如果是某张图片以某组参数第一次生成，因为要进行 SLIC，速度会比较慢，但如果后续不改变图片和参数而只是改画前背景的话则不用再进行 SLIC，结果很快就出来了。



这里抠图效果不好, 需要在前背景上加取一些点然后再次生成, 过程如下:





可以看到效果好了一些，若还要取点，重复前面的步骤即可。如果某一步选点选错想要重画则点击恢复键，这会将原来的标记清除，但保留先前的 SLIC 结果，重画后可以很快出结果：



(3) 尝试不同的超像素数，分析其对对象分割性能和速度的影响

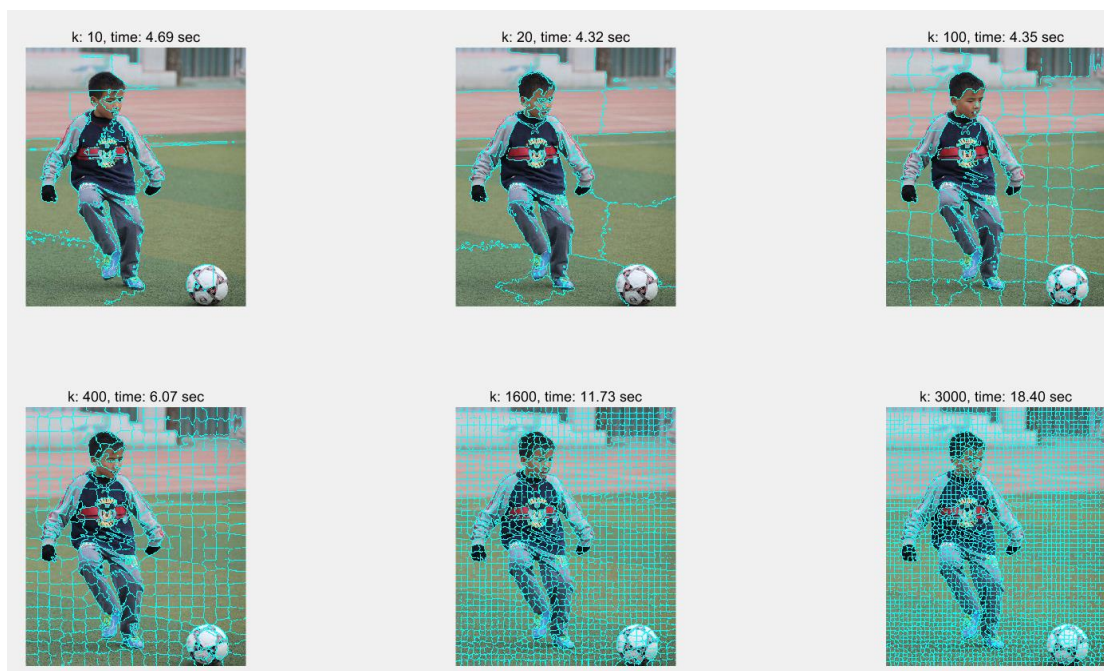
分割性能可用 GUI 体现，下面对比在同样的前背景标注下使用不同 k 值的结果（从上至下依次为 1000, 3000, 6000）：



可以看到 3000 的效果明显比 1000 好，3000 和 6000 差别不大。

为了更直观地对比速度的不同,我在 my_slic_result.m 中将不同 k 值下 SLIC 的运行时间显示在一起 (因为 SLIC 是最主要的耗时,比较其用时即可),结果如

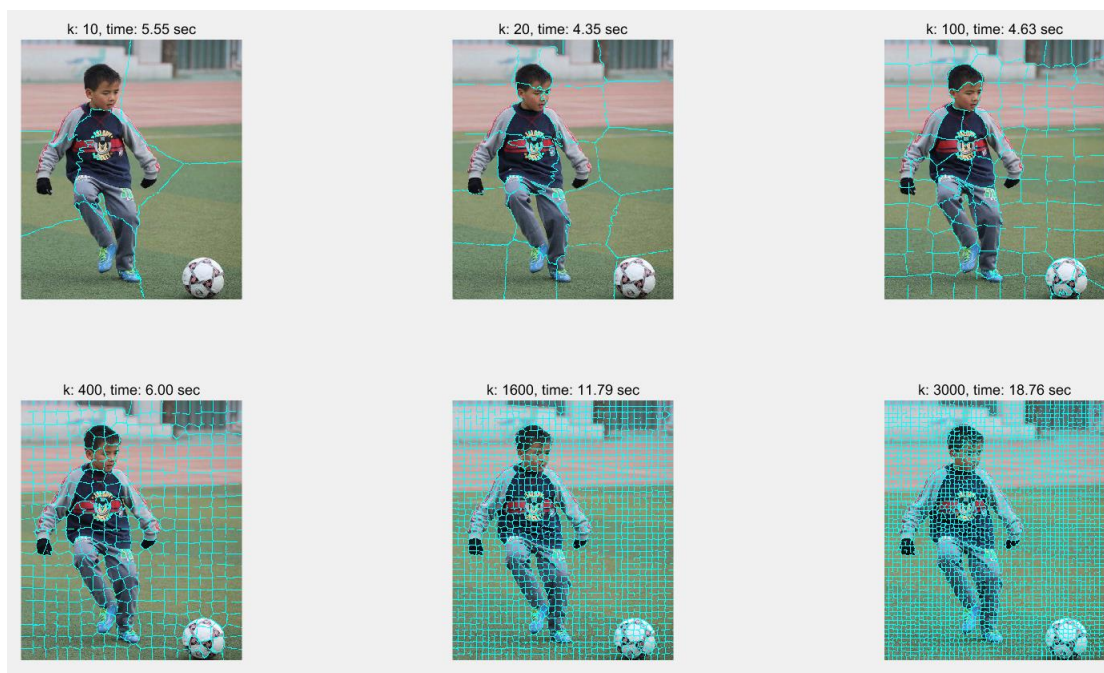
下:



(4) 尝试改进对象分割性能

这里主要是对比 SLIC 和 ASLIC 的结果, 同样在 my_slic_result.m 中将不同

k 值下 ASLIC 的运行结果显示在一起:



将其和上面的组图作对比, 大致可以感受到在 k 值较小的时候有一定的改

善，仅显示单张图片对比（左为 SLIC 右为 ASLIC）：








对比结果正如论文中所说，ASLIC 算法使各个超像素的形状尺寸更规则，但在边界粘附性上不如 SLIC 算法，在 k 值比较大时优势并不明显，交互式 GUI 中仍然是使用 SLIC 进行分割。

其他说明

通过完成本次大作业，我不仅实践了新学的关于超像素的知识，还综合应用

了前面学过的形态学处理的相关知识,对 SLIC 算法的原理有了比较清楚的理解。

我的 GUI 中有一个不足之处即如果选取点时鼠标处出现了这一图标仍然在该处取点会导致程序卡死,我想这应该是我使用 `ginput()`和 `impoly()`这两个函数的方法导致的,避免的做法即在出现该图标时挪动鼠标到其他不会出现这个图标的位置,这一提示信息我在界面上注明了。另外上面的各个部分均是用‘1.jpg’举例,另两张图的处理是完全相同的,运行程序即可检查结果。