HOMEWORK9 REPORT

1700012803 徐思睿

任务要求

编程实现一个图像分割的算法。算法类型不限，可以是基于阈值的（graythresh()）、基于聚类的、基于分水岭的(watershed())、基于图论的、基于概率论的，及其他。鼓励多做实验测试。

总述

这次作业我基于图论与并查集，实现了可参数调节的图像分割。

本次作业也是之后大作业的准备工作，这次的工作可以通过参数调节生成细粒度的超像素分割，作为之后聚类等方法等基础。

实现算法

算法思路参考了论文"Efficient Graph-Based Image Segmentation"，IJCV 2004。所有的代码均自己实现。

- 基本概念
    - 并查集：只有查找、合并两个操作，两个操作的时间复杂度均为O(1)。需要每个元素保存所属集合的size，也可增加rank项使得查找变得更加快捷。
    - 不相似度：两个像素点在颜色空间上的距离。可以考虑利用LUV空间，也可加上图像坐标的差距。在本次实现中，仍采用了RGB空间。通过参数ϕ来调节颜色空间的距离与图像坐标距离之和。
    - 自不相似度：指某集合内边权值（不相似度）最小的结果
    - 自不相似度阈值：自不相似度加上一个常数除以该集合的size，这是为了防止在集合size较小（比如集合只有一个元素）时自不相似度过高导致合并操作过少。同样在集合的size较大时能够不受这一常数项的影响。
    - 4-邻域、8-邻域：分别对应上下左右四个像素点、以该点为中心的3*3正方形内的点。
    - 高斯模糊：通过gaussian kernel进行卷积操作对噪声点过滤，通过ksize与sigma调节kernel的大小与高斯函数的方差。
- 算法步骤
    1. 对图像进行高斯模糊。
    2. 以每个像素点为顶点，与其4-邻域或8-邻域的连边为变，每条边的权重为该像素点与其4-邻域或8-邻域的不相似度。将每条边按照不相似度进行从小到大的排序。
    3. 从小到大遍历每一条边，如果此时该边的两个顶点分属不同集合，并且两个顶点所属集合

的自不相似度阈值大于该边的不相似度，则将这两个集合合并。这里面的查找与合并利用了并查集的操作来加速。

4. 如果合并发生，则更新合并集合的自不相似度阈值，因为并查集只有父节点是有效的，因此只需更新父节点的阈值，此时阈值将等于当前边的权值加上新的常数项（详见基本概念中的自不相似度阈值）。
5. 遍历结束后，重新从小到大遍历每一条边，如果该边两个顶点中的任意一个顶点所属集合所包含的元素较少，则合并这两个顶点所在的集合。

- 算法理解：

  我将按照算法的步骤进行一一解释。

  1. 高斯模糊是为了对噪声点进行过滤，但实际上算法会对分割出的小区域进行筛除，因此这一步的意义更多在于像素之间的不相似度会变得更小。
  2. 我们目前只考虑了4-邻域或8-邻域，之后在大作业中会引入nearest neighbor，并且考虑提取像素的特征与位置信息，对k近邻的元素进行连边。
  3. 从小到大遍历是为了先合并不相似度小的连边，类似于最小生成树算法的思想。
  4. 我们合并的原则是边权值（元素之间的不相似度）小于两个自不相似度（类内最小的边权值、不相似度），每次阈值都会更新为该集合中最小的不相似度加上（常数项/size）。
  5. 因为可能会得到一些非常小的分割块影响效果，我们最后还需要筛出一些比较小的分割块，并和其最相似的集合合并，这只需要再一次从小到大遍历每条边即可。

对比实验

在该算法中比较重要的几个参数有，由于篇幅限制，在报告中仅呈现一部分结果：

1. Neighbor：4或8，4-近邻或8-近邻。更多结果详见"./asset/concate_images_neighbor=*"系列文件夹。
2. K：阈值中的常数项，会除以集合的大小。
3. min_comp_percent：原算法使用的直接是最小的集合的像素数量，现改为占整张图片的比例。
4. phi：调节位置差异与颜色差异作为边权值的比例。更多结果详见"./asset/concate_images_phi=**"系列文件夹。

对比实验如下所示：

1. K的对比实验：

   Gaussian blur kernel size: 5

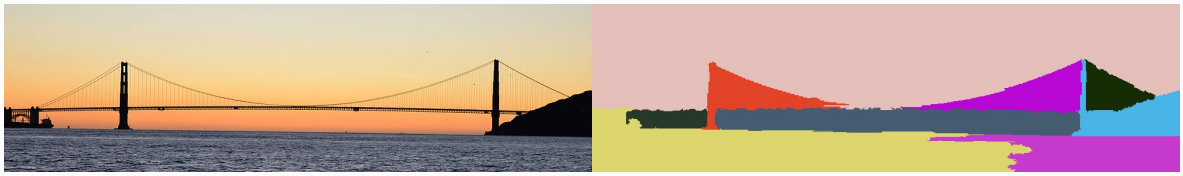
   Gaussian blur kernel variance: 1.0

   The number of edges for a pixel: 4

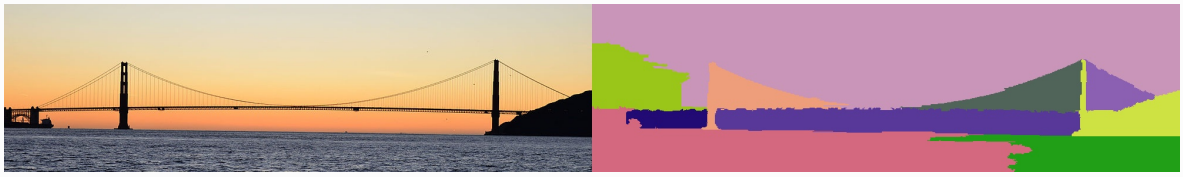
   Phi = 0

   fewerest percentage of one component: 0.01

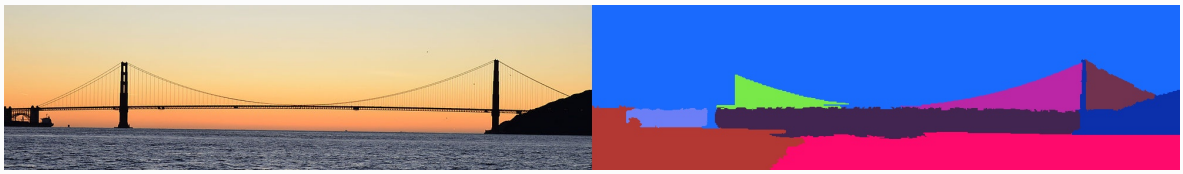
   K=1.0：

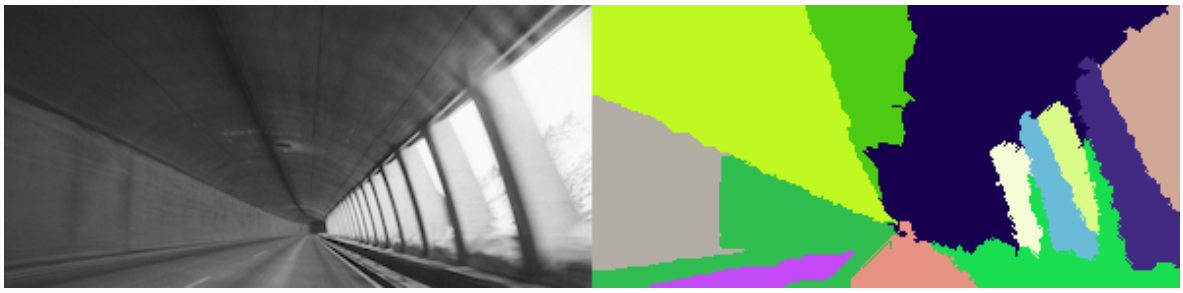
   K=10：



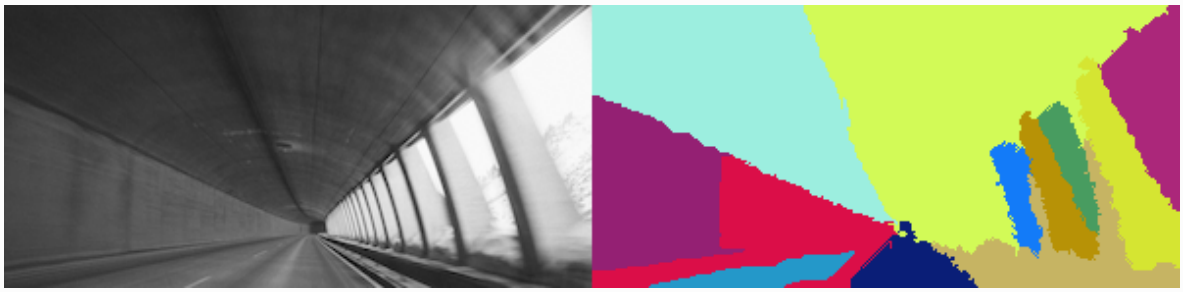
K=100：



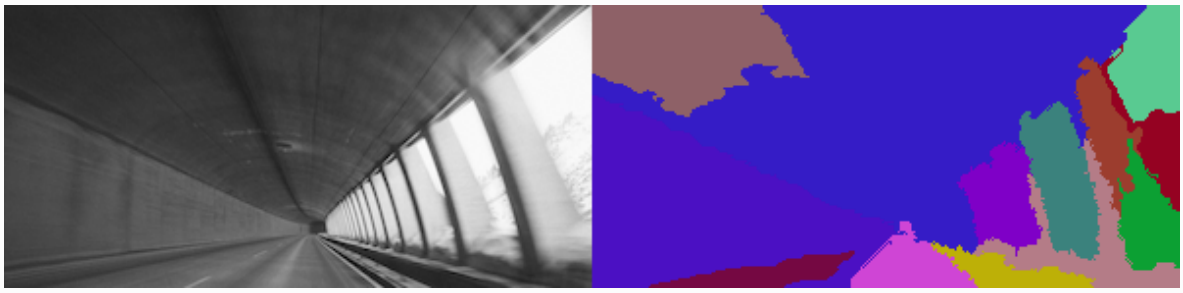
K=1.0：



K=10.0:

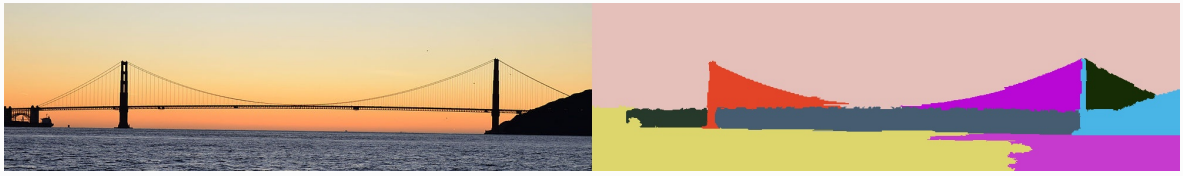


K=100.0：

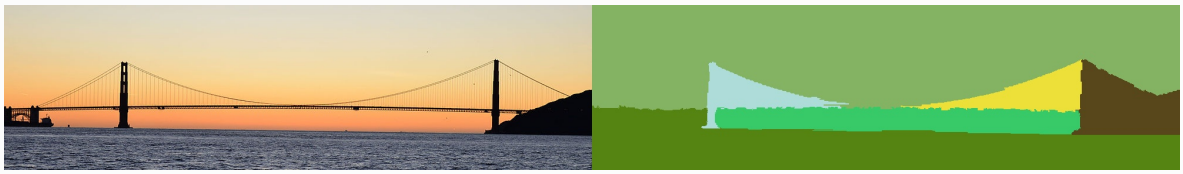


3. Gaussian blur kernel size: 5
   Gaussian blur kernel variance: 1.0
   The number of edges for a pixel: 8
   K = 1 Phi = 0

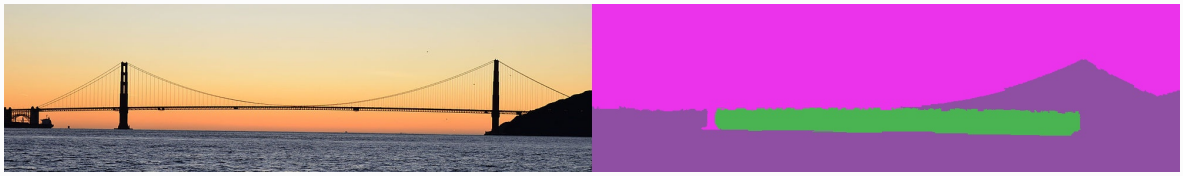
   min_comp_percent=0.01



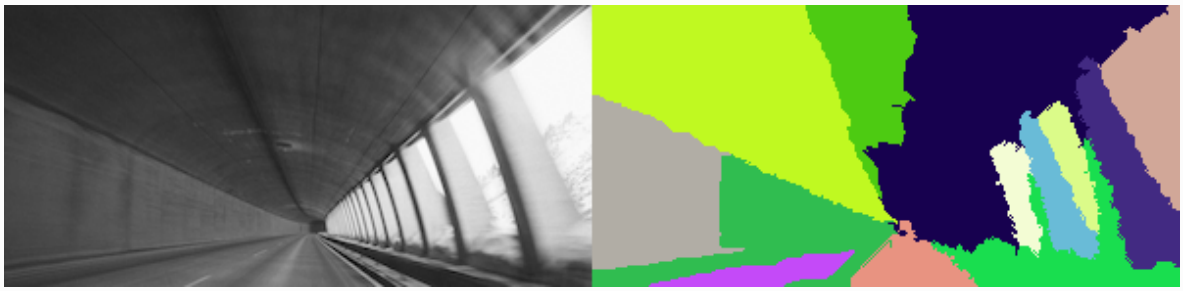
min_comp_percent=0.02



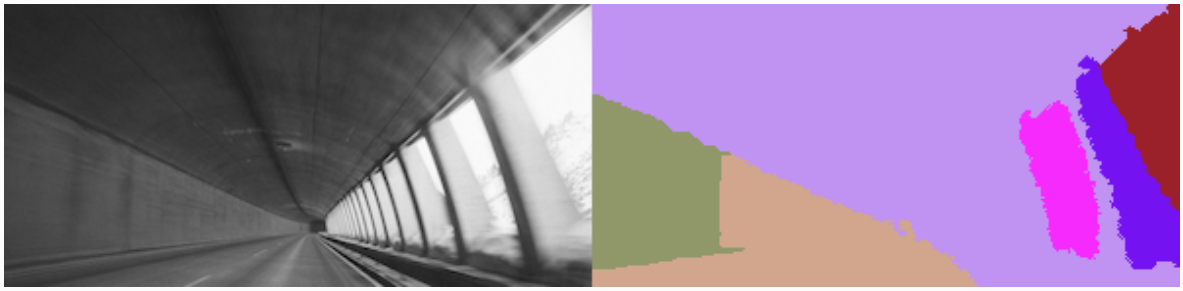
min_comp_percent=0.05



min_comp_percent=0.01



min_comp_percent=0.02



min_comp_percent=0.05

对于K而言，K越大，合并的可能性越大，分割出来的块数可能就越少；对于min_comp_percent，很明显当其提升时，分割出来的块数就越少。

为了得到超像素，则需要更多细碎的分割块。如果只是为了得到最后的分割的结果，则当min_comp_percent=0.02是一个比较好的参数选择。

更多的实验结果详见./assets/

可能的改进

由于这一算法仅考虑了相邻区域的颜色关系，可以设置参数获取细粒度的划分，得到类似超像素的分割。如果将这一步骤作为聚类算法、图网络流算法的前置步骤，可以极大地减小计算代价。因此，我考虑将这个模型结合以特征为基础的聚类算法，再加上后续的fine-tune算法，最后得到带有语义的分割。

这些改进将在大作业中尝试。

调试命令

测试命令: python3 ./test.py

各项参数可通过: python3 ./test.py --help 查看。