

第四次作业 Bonus 部分报告

PB21010362 汪兆辰

2023 年 4 月 7 日

1 算法介绍

该算法的功能是通过对图像 context 的检测来标记图像中视觉显著区域, 从而生成 saliency map. 对于图像中的每个像素, 需要考察它周围的一个邻域来获得其 context 的信息, 为了方便计算我们取其邻域中所有点的均值. 当某个像素点是一个显著点时, 它应该与图中大部分的其他点在颜色上不同, 但是由于我们不应考虑孤立的像素点, 我们应该允许当其他像素点距离其较近时颜色与其差异不大. 基于以上两点我们可以定义度量:

$$d(p_i, p_j) = \frac{d_{color}(p_i, p_j)}{1 + c \cdot d_{position}(p_i, p_j)} \quad (1)$$

其中 d_{color} 为 p_i, p_j 在 CIL 颜色空间上的距离, $d_{position}$ 为两点的欧氏距离. d 的值越大, 说明两个点的差异越大.

但事实上我们并不用对每个点都遍历图像上的所有点, 仅用所有点中与给定点差异最小的 K 个点来衡量该点的显著程度就足够了, 这是因为如果对于差异最小的一系列点来说给定点已经是显著的, 那么它对于整张图来说都是显著的. 因而对于每个点可以用:

$$S_i^r = 1 - \exp\left(-\frac{1}{K} \sum_{k=1}^K d(p_i^r, q_k^r)\right) \quad (2)$$

来衡量其显著程度, 其中 r 为邻域的尺寸.

为了让结果更加精细化, 我们可以考虑取一系列尺寸 r , 对每种尺寸下的 S_i^r 取平均, 从而可以增大显著点与其他点的差异, 这时显著程度的度量表示为:

$$\bar{S}_i = \frac{1}{M} \sum_{r \in R} S_i^r \quad (3)$$

其中 R 为所有尺寸 r 组成的集合.

为了满足论文中提出的第四条 principle, 我们考虑对足够显著的视觉焦点再进行操作. 将最显著的点定义为焦点并归一化, 在图像所有点中筛选 $\bar{S}_i > 0.8$ 的点集, 对这些点再次定义度量:

$$\hat{S}_i = \bar{S}_i(1 - d_{foci}(i)d) \quad (4)$$

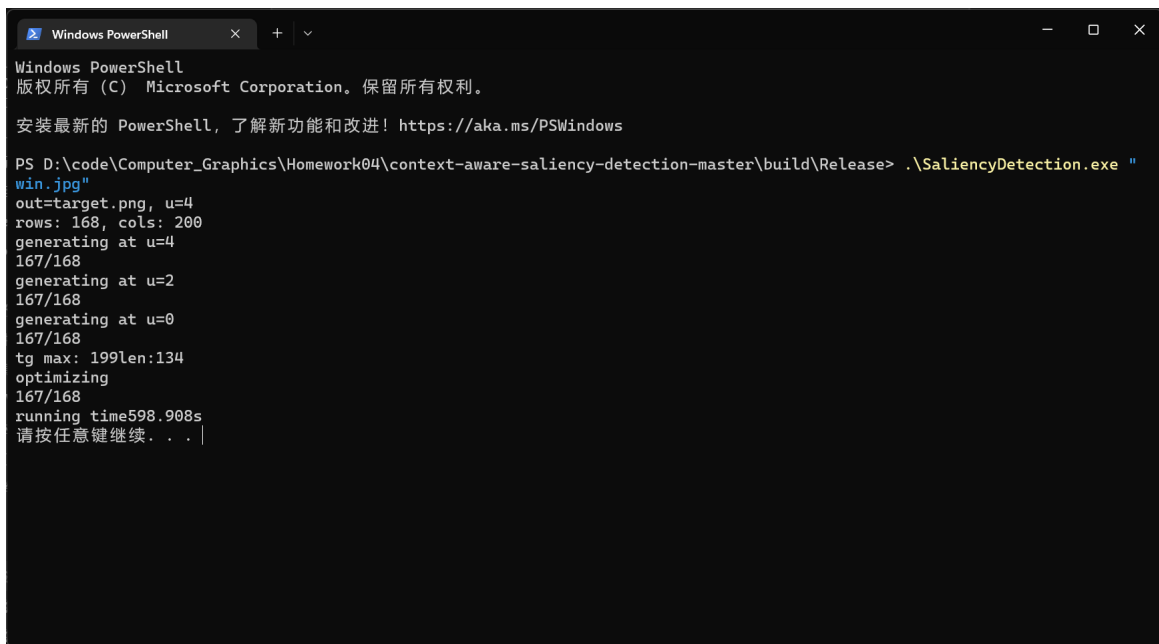
这样我们就得到了最终的 saliency map.

2 算法实现

代码参考自 [github\[2\]](#). 在添加注释的同时对原代码也进行了一些改正与优化.

对代码的详细说明见代码注释, 这里只作简略介绍. 代码的主要函数在 `SaliencyDetect.cpp`, 其中 `color_distance`, `distance` 和 `salient` 三个函数分别返回两点在颜色空间的距离、两点的欧氏距离及每个点的显著度量. `saliencyMatrix` 函数的功能是返回原图在给定尺度 `u` 下的显著度矩阵; 函数 `exec` 通过调用 `saliencyMatrix` 按照算法的要求得到相应的显著图.

但是注意到原代码的运行时间相当长, 对于分辨率 200×168 的图像, 在 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz 3.11 GHz 上的运行时间为 598.908s. 因此考虑优化代码来加速程序运行.



```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell, 了解新功能和改进! https://aka.ms/PSWindows

PS D:\code\Computer_Graphics\Homework04\context-aware-saliency-detection-master\build\Release> .\SaliencyDetection.exe "
win.jpg"
out=target.png, u=4
rows: 168, cols: 200
generating at u=4
167/168
generating at u=2
167/168
generating at u=0
167/168
tg max: 199len:134
optimizing
167/168
running time598.908s
请按任意键继续. . .
```

由于代码在数据处理中使用的都是 `for` 循环, 因此考虑对部分不会引起数据冲突的循环采用并行计算, 调用 `cv` 库中的 `parallel_for_` 函数, 例如:

```
parallel_for_(Range(0, rows), [&](const Range& r){
    for (int row = 0; row < rows; ++row) {
        for (int col = 0; col < cols; ++col) {
            auto avg = (uchar)(
                ((int)tg4.at<uchar>(row, col)+
                 (int)tg2.at<uchar>(row, col)+
                 (int)tg0.at<uchar>(row, col))/3);
            tg.at<uchar>(row, col) = avg;
            if (avg > _max) _max = avg;
        }
    }
});
```

另外原代码中不必要的矩阵复制等操作也被修改，从而在不对算法改动的前提下最大限度提高速度，运行结果如下：

```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS D:\code\Computer_Graphics\Homework04\context-aware-saliency-detection-master\build\Release> .\SaliencyDetection.exe "
win.jpg"
out=target.png, u=4
rows: 168, cols: 200
generating at u=4
167/168
generating at u=2
167/168
generating at u=0
167/168
tg max: 199len:134
optimizing
167/168
running time: 353.67s
请按任意键继续. . . |
```

对于同一张图在相同的条件下，运行时间减少到 353.67s，有明显提升. 进一步的速度提升可能需要修改算法，如对三张不同的尺寸下显著值相差不大的点无需重复计算，而经过筛选直接使用一次计算的结果；或者考虑采用硬件加速，例如使用 opencv 的 gpu 模块，但我不会.

3 运行效果

其他一些图像的测试结果如下：



(a) Row



(b) Saliency



(c) Row



(d) Saliency



(e) Row



(f) Saliency

参考文献

- [1] Stas Goferman, Lihi Zelnik-manor and Ayellet Tal. Context-Aware Saliency Detection. CVPR2010.
- [2] <https://github.com/Somefive/context-aware-saliency-detection>