Color2Gray

计 42 殷达 2014011320

January 18, 2018

原始论文：Gooch, A. A., Olsen, S. C., Tumblin, J., & Gooch, B. (2005). Color2gray: salience-preserving color removal. ACM Transactions on Graphics, 24(3), 634-639.

1 算法原理及实现细节

这篇论文主要是在考虑邻域差别的情况下进行色彩空间上的降维（RGB 空间到灰度空间）。由于一般的灰度化会采用亮度值而压缩颜色值，但是有时候临近色彩不同但亮度相近的时候，灰度化就会抹除这这之间的差异。而 Color2Gray 这篇论文则是为了针对这个问题进行了优化，使得经过灰度化的图像能够较好地保留原来的差异性。

首先将图像从 RGB 颜色空间转换到 CIE 的 L*a*b 色彩空间上。其中 L 为亮度通道，a 和 b 分别是另外两个色彩通道。该算法将原来的转换问题变成一个二次优化问题，其优化的目标函数为

$$min \sum_i \sum_{j \in N_\mu(i)} ((g_i - g_j) - \delta_{ij})^2$$

其中 g_i, g_j 是目标灰度图的灰度值，$N_\mu(i)$ 为像素点 i 的邻域，邻域 μ 可以是以该像素点为中心的一个 Patch 或者是全局，δ_{ij} 是用来控制目标色彩差异的值，具体的公式为

$$\delta_{ij} = \begin{cases} \Delta L_{ij} & if \ \ | \Delta L_{ij} | > crunch(\| \overrightarrow{\Delta C_{ij}} \|) \\ crunch(\| \overrightarrow{\Delta C_{ij}} \|) & if \ \overrightarrow{\Delta C_{ij}} \cdot \overrightarrow{v_\theta} \geq 0 \\ crunch(- \| \overrightarrow{\Delta C_{ij}} \|) & otherwise \end{cases}$$

$$crunch(x) = \alpha \times tanh(\frac{x}{\alpha})$$

$$\overrightarrow{v_\theta} = (cos\theta, sin\theta)$$

其中 ΔL_{ij} 为像素点 i, j 在亮度上的距离。$\Delta C_{ij} = (\Delta A_{ij}, \Delta B_{ij})$ 为在色彩上的距离。

可以看到，α 起到了控制色彩差距相对于亮度差距所应该占的比重，α 越大，色彩上的差距越重要。θ 则是起到了控制颜色差距与灰度上对应的差距方向。可以看到如果选取 $\theta_1, \theta_2 \ \ st. \ theta_1 = \theta_2 + \pi$ 其对应的优化方向正好相反，在图 1 中像素点 i, j 的灰度优化目标差为 δ，那么在图 2 中恰好为 $-\delta$。

在优化该二次函数的时候，可以先令其偏导为零得到相应的优化目标方程组。

$$\forall i \ st. \ g_i = \frac{1}{N_{\mu_i}} \sum_{j \in \mu_i} (g_j + \delta_{ij})$$

由于 N 为像素点数量，往往较大，可以采用迭代优化的方法。在原论文以及我的实现中都采用了较为简单的 Gauss-Seidel（高斯赛德尔）迭代法求解。一般迭代 3 ～ 5 轮就可以基本收敛。

需要注意到的一个细节就是，原始目标函数只能保证像素之间相对差值逐渐朝目标值变化，并不能保证整张图的平均亮度，因此在每次迭代更新后，需要将整体的平均亮度向原始图像的平均亮度调节，否则在较多的迭代过后，图像会明显变暗。

2 时间复杂度分析

由于 δ_{ij} 在每轮迭代后都不变，所以可以预先用 $O(N^2)$ 的时间将其预先算好。之后每轮迭代使用 $O(N\mu^2)$ 的时间来计算。因此最后的时间复杂度是 $O(N^2 + RN\mu^2)$，其中 R 是迭代的次数。

3 结果

如下样例是源图像大小为 $538px \times 384px$，在 Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 的结果。其中基准图像的参数是 iter=5,alpha=100,theta=$\frac{1}{4}\pi$,u=5。

Figure 1: 原始图像

Figure 2: 基准图像：3.3s

Figure 3: alpha=1：3.6s

Figure 4: alpha=10000：3.6s

Figure 5: iter=45：6.6s

Figure 6: theta=$\frac{1}{5}\pi$：3.6s

Figure 7: u=2：0.98s

以下是另外两个在标准参数下的例子