CodingProject1 Report

张景浩 PB20010399

2023.5.8

1 问题描述

对比度对视觉效果的影响非常关键，一般来说对比度越大，图像越清晰醒目，色彩也越鲜明艳丽；而对比度小，则会让整个画面显得模糊的。高对比度对于图像的清晰度、细节表现、灰度层次表现都有很大帮助。在数字图像中，对比度就是图像最大灰度值和最小灰度值的落差。为了将模糊的图像变得清晰，我们要将图像中不同物体边界处的对比度增强，也就是说，要增大边界处的像素梯度值，分别完成对比度的增强，这就是局部对比度增强。

为了实现这一点，我们参考文章 Greedy Algorithm for Local Contrast Enhancement of Images [1] 中的算法，利用贪心算法编程实现图像的局部对比度增强。对输入的图片在不同的参数下进行局部对比度增强，并对比得到的结果。

2 算法原理

2.1 灰度图的对比度增强

灰度图的对比度感知与图像的局部梯度直接相关，我们的目标是在严格的约束下增强图像的局部梯度，防止图像过饱和/不饱和以及灰度值的无限增强。因此，将最大化目标函数

$$f(\Omega)=\frac{1}{4\Omega}\sum_{p\in\Omega}\sum_{q\in N_4(p)}\frac{I'(p)-I'(q)}{I(p)-I(q)}$$

同时满足约束

$$\begin{cases}1\le\dfrac{I'(p)-I'(q)}{I(p)-I(q)}\le 1+\delta\\ L\le I'(p)\le U\end{cases}$$

其中，Ω 是像素集，$N_4(p)$ 是 p 的 4 个邻居，$I(p)$ 和 $I'(p)$ 分别表示输入和输出图像在 p 处的灰度值，L 和 U 分别是像素集灰度值的下界和上界，δ 是控制增强量的单一参数。这个约束保证了增强后的局部梯度不会缩小且增加上限确定，同时保证了增强后的灰度值不会超出像素灰度值的上界和下界。

我们假设两个相邻像素的灰度值 r 和 s($r \neq s$)，使用 $1+\delta$ 对它们进行放大，直到结果 r' 和 s' 满足

$$\frac{r'-s'}{r-s} \leq (1+\delta)$$

为了避免一点的饱和度过强，我们使用贪心算法逐步迭代。

我们把 I 看作是 Ω 上的一个 $m \times n$ 高度场，每一点高度为 $I(p)$ $p \in \Omega$ $I(p) \in [L,U]$。每一次迭代，我们选择一个阈值 $b \in [L,U]$，定义矩阵 $R \in \mathbb{R}^{m \times n}$ 满足

$$R(i,j) = \begin{cases} 1 & I(i,j) > b \\ 0 & I(i,j) \leq b \end{cases}$$

我们称 R 中的非零连通分量为一个小丘，记为 h_i^b。将每一个 h_i^b 中的灰度值增大，但增强因子不超过 $(1+\delta)$ 且增强后灰度值不超过 U。

我们从 L 开始向 U 迭代扫描阈值平面，并在每次扫描的阈值下贪婪上升。注意到，当我们扫描连续的平面时，小丘 h_i^b 只有三种可能的情况：消失不见、保持不变或分裂为 h_j^{b+1} 和 h_k^{b+1}。这是因为当阈值从小到大迭代时，本次扫描得到像素是上一次扫描得到像素的子集。因此我们对整个算法给出两个优化：一，每轮扫描只在上一轮得到的小丘中进行，进而得到本轮的小丘；二，我们储存每一个小丘的已增强因子，通过确保其不超过 $(1+\delta)$ 来避免在每次迭代中显式计算梯度约束。

当 b 较小时，每一个小丘的尺寸都较大，其峰值接近 U 的可能性就比较大，考虑到灰度值约束，此时在小丘上的增强因子可能小于 $(1+\delta)$。而当 b 不断增大，较大小丘被细分后，被细分得到的小丘才能够得到更大的增强。基于这一点我们的算法自底向上扫描，每次都只增强 I 局部的灰度值，最后得到图像 I_1。为了进一步的增强图像的对比度，我们可以选择对山谷部分进行同样的操作，将扫描过程作用于 I_1 的补图 $I_2 = U - I_1$，得到图像 I_3，I_3 的补图 $I' = U - I_3$ 就是局部对比度增强后的输出图像。

Algorithm 1 Sweepandpush($I,\ S,\ \delta,\ U$)

Input: Input image I, List od gray values S, Control parameter δ, Upper bounds U

Output: Output image I'

$I' \leftarrow I$
for each $s \in S$ **do**
    obtain boolean matrix B with $B_{ij} = 1$ iff $I_{ij} \geq s$
    identify set of hillocks H in B
    for each $H_i \in H$ **do**
        find p_{max} with $I(p_{max}) \geq I(p)$,$\forall p, p_{max} \in H_i$
        $\delta_{max} = \min(\delta, (U - s)/(I(p_{max}) - s) - 1)$
        for each $p \in H_i$ **do**
            $\delta_{apply} = \delta_{max}$
            lookup $\delta_h(p)$,the net scaling factor in the history for p
            if $\delta_{apply} + \delta_h > \delta$ **then**
                $\delta_{apply} = \delta - \delta_h$
            end if
            $I'(p) = \max(1 + \delta_{apply})(I(p) - s) + s,\ I'(p))$
            update $\delta_h = \delta_h + \delta_{apply}$
        end for
    end for
end for
Return I'

Algorithm 2 Enhance($\delta,\ I,\ L,\ U$)

Input: Control parameter δ, Input image I, Lower and upper bounds L and U

Output: Enhanced image I'

1: $Find\ P = \{b \mid b = I(p)\}, \forall p\ at\ minimas/saddle\ points$

2: $Sort\ P\ into\ list\ P'$

3: $I' \leftarrow I$

4: $I' \leftarrow Sweepandpush(I',\ P',\ \delta)$

5: $I' \leftarrow U - I'$

6: $Find\ Q = \{b \mid b = I'(p)\}, \forall p\ at\ minimas/saddle\ points$

7: $Sort\ Q\ into\ list\ Q'$

8: $I' \leftarrow Sweepandpush(I',\ Q',\ \delta)$

9: $I' \leftarrow U - I'$

10: $Return\ I'$

2.2 彩色图的对比度增强

因为彩色图一般有三个颜色通道，所以我们将 RGB 值线性转换到 CIE XYZ 空间得到亮度 (Y) 以及色度坐标 $x = \frac{X}{X+Y+Z}$ 和 $y = \frac{Y}{X+Y+Z}$。我们将 2.1 所述算法中的灰度值 I 替换为亮度 Y，同时修正约束防止亮度越界。然后在保证色度坐标 x 和 y 不变的情况下将图像转换回 RGB 空间。

3 代码实现

在算法的具体实现中，根据遇到的各种情况并结合自己的理解，在代码中做了一些修改如下：

1. 在 Algorithm2 的第 14 行，原算法中并没有取最大值这个操作。但是在实验中发现当 $\delta_h = \delta$ 后，δ_{apply} 会变为 0，这样一来 $I'(p)$ 就会等于 I(p)，相当于没有增强效果。于是做了如上修改，使得增强的结果能够得到保留。

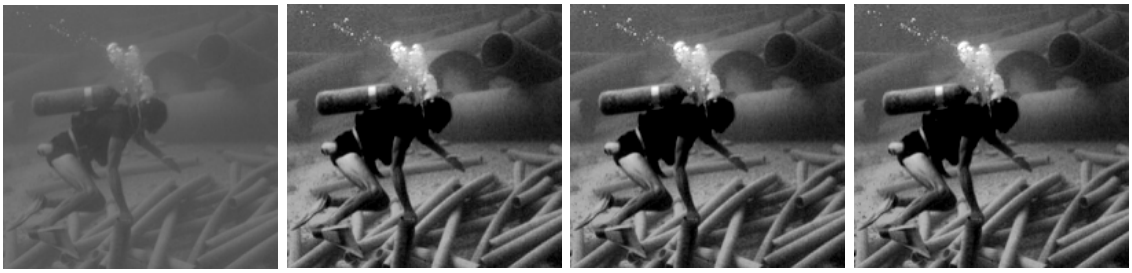
2. 在具体实现中，Algorithm1 的第 1、6 行选择了局部极小点。

3. 在处理彩色图像中，由于将 RGB 转化为 CIE XYZ 并对亮度 Y 进行处理，导致在 Algorithm1 的第 1、6 行中得到了巨量的不等值局部极小点，并且两个相邻的局部极小值之间相差在小数点后 4、5 位左右，这是因为亮度 Y 的取值范围时 $[0, 1]$。因为每次的增强值往往大于这个差值，这使得代码在运行过程中出现了大量的无用重复操作，所以我人

为的选取了两个局部最小值之间最小落差，大大减少了运行时间的同时保证了算法的效果。经过多次调试，这个最小落差选择为 0.01，同时，由于落差小于 1，所以这个操作不会影响对于灰度图的处理。

4. 在文件 Local_Contrast_Enhancement.m 中，我用注释标注了处理彩色图和灰色图的不同算法，可以人为进行切换。

4 测试结果



原图像　(a) $\delta = 2$　(b) $\delta = 4$　(c) $\delta = 6$

图 1: 这组图片对比了灰度图片在不同增强因子 δ 下的结果，程序运行时间为 1.5s 左右。可以看出，2.1 中的算法对于灰度图局部对比度提升效果显著。在 $\delta = 2$ 时就得到了非常好的增强效果，并且随着 δ 的增大，得到的结果并没有明显变化。

原图像　(a) $\delta = 2$　(b) $\delta = 4$　(c) $\delta = 6$

原图像　(d) $\delta = 2$　(e) $\delta = 4$　(f) $\delta = 6$

图 2: 这组图片对比了彩色图片在不同的算法和不同的增强因子 δ 下的结果。(a)(b)(c) 使用了 2.2 中的方法完成了图像的局部对比度增强，程序运行时间为 4~6s，(d)(e)(f) 使用了对 RGB 三个颜色通道分别使用 2.1 中的算法完成了图像的局部对比度增强，程序运行时间为 37~39s。可以看出，增强因子 δ 越大，图像的局部对比度越明显，但是当 δ 过大时，增强后图像会失真并且出现黑色噪点。对比两种不同的增强方法，对三个颜色通道分别处理得到的结果，其色调会与原图有明显的差异，从暖色调变成了冷色调，而对亮度进行处理并保持色度坐标不变的方法则很好的在保持原图色调的基础上完成了局部对比度增强。

5 总结

文章 [1] 算法在处理灰度图像的局部对比度增强问题时有着非常好的效果，同时程序的运行速度也非常快；而在处理彩色图像的局部对比度增强问题时，根据处理方法的不同得到的结果也不同，主观上来说通过亮度进行处理得到的结果更符合预期的目标，在保留了原有色调的情况下完成了局部的对比度增强。同时我们发现，对于彩色图增强因子不能过大，否则会出现

因增强过度而产生的噪声点，而对于灰度图来说对于增强因子则没有这么敏感。

参考文献

[1] K. Subr, A. Majumder, and S. Irani, "Greedy algorithm for local contrast enhancement of images," in *Image Analysis and Processing – ICIAP 2005*, F. Roli and S. Vitulano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 171–179.