图形学小作业报告

计科60 王蕴韵 2016011346

我选择了基于优化给图片上色的小作业。

在描述这个算法之前，我想先表达一下我对给的图片的不满。在里面有这么一张黑白的中国地图轮廓线，下面写着昵图网的水印。来看看这个算法的原理，Our method is based on a simple premise: neighboring pixels in space-time that have similar intensities should have similar colors. 是利用照片中相邻像素颜色差距不大这个特性，是真实照片才有这种特性，整个优化的方程设置都是建立在这个条件之上的。给一张明显不符合前提的图片，尤其是里面还有文字，会严重破坏像素颜色的连续性的，并不能测试出算法的优劣。其次，给了压缩严重的 jpg 图片，这种上色是精确针对每个像素点求值的，在原论文网页上和附的代码上也有说明建议使用无压缩的 bmp 图片获得好的效果。我把 jpg 转换成 bmp 后图片质量也不会得到提高，大片的相近颜色的色块丢失精度被模糊，造成解方程的时候算邻居之间权值时失去了很多信息。希望以后可以改进一下给的图片。

算法的流程很简单，先把图片的颜色修改为YUV格式，分别表示 intensity，和二维色彩。利用 intensity 相近的邻居像素来约束此格像素，根据 intensity 的接近程度给邻居像素赋上权值，再求解这个优化问题。求解可以使用解方程。

$$J(U)=\sum_r\left(U(r)-\sum_{s\in N(r)}w_{rs}U(s)\right)$$

让$A=I-W, W=\{w_{rs}\}$, 并解方程$Ax=U,Ax=V$可以获得近似解。

邻居点是这样算的：如果是一张图，那么曼哈顿距离小于 W 的是邻居。如果是视频，那么对这张图，除了这张图的邻居，还有和前一张的邻居。

前一张的邻居定义是，在(x,y)点的速度是(v_x,v_y)，$||(x+v_x,y+v_y)-(x',y')||<T$，则$(x,y)$和前一张的$(x',y')$是邻居。速度可以用 Lucas-Kanade 法计算。利用一个小框内移动速度近似一致的特性，解方程：

$$A=\begin{bmatrix}I_x(q_1) & I_y(q_1)\\ I_x(q_2) & I_y(q_2)\\ \vdots & \vdots\\ I_x(q_n) & I_y(q_n)\end{bmatrix}\qquad v=\begin{bmatrix}V_x\\ V_y\end{bmatrix}\qquad b=\begin{bmatrix}-I_t(q_1)\\ -I_t(q_2)\\ \vdots\\ -I_t(q_n)\end{bmatrix}$$

$$A^T Av = A^T b$$

在连接好邻居后，根据论文中的权重公式分配权值。矩阵比较大，使用 Python中的scipy.sparse来记录稀疏矩阵并解方程。

我又根据课上老师提到过的思路进行了加速优化。对图像进行插值，可以把长宽都减少一半，大大加速了计算。因为 intensity 已经知道，我们分配的仅仅是颜色，所以基本不会影响图片质量。这样最终跑起来很快，由于 Python 本身比较慢，优化后的速度大致和论文作者给的 MATLAB 代码相当。

我发现这个方法在遇到大片全白色块的时候解方程的答案容易出现大偏差，原因是邻居之前 intensity 相差较小，算出来的权值偏小，解方程的时候答案容易误差大，一误差白色就跑偏成别的颜色了，因为白色 intensity 小，看起来还很扎眼，所以就对白色进行了单独处理。保证了最后结果中白色还是白色。注意到论文作者的代码实现中并没有考虑这一点，它跑出来的白色也会发生大偏移。改进后的图片看起来比较正常了。

附录图片在 pics 目录下，flower 是我自己使用的图片，实现了一个换色。_m的后缀是标注过的，大部分我都使用了随机撒点的形式（因为懒得自己手标），随机撒10%的大小为5*5的点。flower 是我自己手标的，因为换色没有参考颜色。

视频用 FFmpeg 把原视频拆开，图片弄成黑白，后面每20张标注一次，剩下的使用前一张的渲染结果利用上述对视频的邻居构建计算。最后用 FFmpeg 再合成视频。

附录视频video 目录下，提供了两个，一个是反面例子，因为像素移动过快并且一直在旋转，出现了新颜色来不及填上，导致有时画面颜色丢失，也说明了这个方法的一定局限性，就是视频画面移动速度要慢、平稳才适用，并且最好画面不要不停出现新物体。另一个例子是画面内容相对固定，移动相对幅度小，出来的颜色就比较稳定。