# Colorization using Optimization

王世因 2016011246

## 目录

# 1 算法简介

在 YIQ 和 YUV 图片格式中，Y 代表图片的亮度，含有各种阴影细节，单独输出后就可以得到黑白照片。

# 2 图片上色

图 2: gray

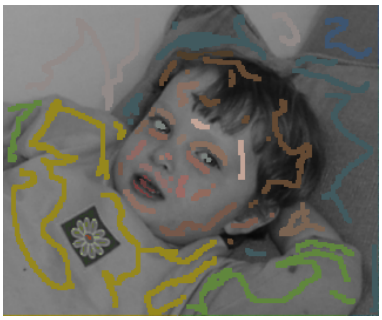

图 3: sketch

图 4: result

图 5: gray

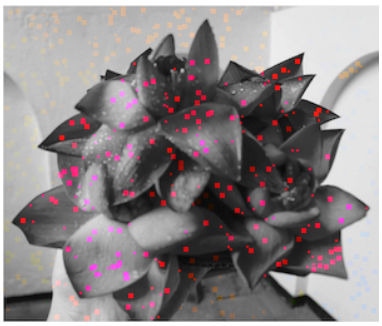

图 6: sketch

图 7: result

# 3 从静态图片扩展到视频

## 3.1 准静态图片的像素采样法

这部分算法对应代码文件 *video_color.py*，我通过在上一帧图片中获得一些采样点来，作为本帧的标注色彩。因为我采用的是 30 帧一秒的视频质量，视频中相邻两帧间的变化不大，所以可以直接进行一定的色彩继承分析。因为优化算法最小化的是总体的灰度匹配概率，因为部位微小位移造成的标注误差可以在最优化求解的时候被补偿。相比下面的这种方法，本方法因为没有增加权重矩阵的大小，所以计算起来更快，占用的计算资源也更小。


图 7: 相邻帧之间采样标注色彩

### 3.2 位移捕捉和临点检测

这是原文中提到的方法，通过算法 Lucas-Kanade 计算各个像素点的运动情况，找到相邻两帧中对应的点，拓展静态图片中像素点的临点，再进行求解。

$$Y_t(p) + \nabla Y_{(x,y)} v = 0 \tag{1}$$

$$\begin{bmatrix} Y_x(p_1) & Y_y(p_1) \\ Y_x(p_2) & Y_y(p_2) \\ Y_x(p_3) & Y_y(p_3) \\ \cdots & \cdots \\ Y_x(p_{25}) & Y_y(p_{25}) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = - \begin{bmatrix} Y_t(p_1) \\ Y_t(p_2) \\ Y_t(p_3) \\ \cdots \\ Y_t(p_{25}) \end{bmatrix} \tag{2}$$

$$||(x_{t+1} - v_x, y_{t+1} - v_y) - (x_t, v_t)|| < T \tag{3}$$

在具体的实现上，我根据上面的公式生成一个包含两帧的权重矩阵，重新跑一边静态图片的上色算法，得到后一帧的色彩。因为权重矩阵是一个很大的稀疏矩阵，这个做法的权重矩阵是静态图片权重矩阵的四倍，显著地增加了耗时。对比来看，这个方法适用于移动迅速的图片和每秒帧数少的图片。相应的代码实现在 *video_dynamic.py* 中，对应 $frame.DynamicFrame$ 类。

## 4 算法的不足