

# 《数字图像处理》作业 1

学号: SA21229075 姓名: 陈文博

2020 年 9 月 19 日

## 1 作业要求:

实现如下算法之一 (或两个都实现)

- **Color Transfer:** E. Reinhard et al. Color Transfer between Images. IEEE CG&A, 2001.
- **Colorization:** T. Welsh, et al. Transferring Color To Greyscale Images. Siggraph 2002.

## 2 开发环境

**IDE:** Microsoft Visual Studio 2019 Community

**CMake:** 3.20.0

相关依赖库: std\_image, spdlog, imgui, glm, glfw, glad, entt, OpenCV 等

### 3 交互界面



图 1: 交互界面说明

### 4 算法原理

#### 4.1 Color Transfer

##### 4.1.1 问题描述



图 2: 源图像



图 3: 目标图像

如上两幅图，我们希望通过设计一个颜色转换算法，使得源图像具有目标图像的颜色风格。

#### 4.1.2 算法描述

衡量一幅图像的颜色分布最基础的统计特征就是均值和标准差，文献 [1] 中便通过这两个特征对图像进行简单的变换并取得不错的效果。首先，需要将图像的颜色空间从  $RGB$  空间变换到  $l\alpha\beta$  空间：

- $RGB$  空间  $\rightarrow XYZ$  空间

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.5141 & 0.3239 & 0.1604 \\ 0.2651 & 0.6702 & 0.0641 \\ 0.0241 & 0.1228 & 0.8444 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (1)$$

- $XYZ$  空间  $\rightarrow LMS$  空间

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = \begin{pmatrix} 0.3897 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (2)$$

- 变换到对数空间

$$L = \log L$$

$$M = \log M \quad (3)$$

$$S = \log S$$

- $LMS$  空间  $\rightarrow l\alpha\beta$  空间

$$\begin{pmatrix} l \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} L \\ M \\ S \end{pmatrix} \quad (4)$$

同理，将图像的颜色空间从  $l\alpha\beta$  空间变换到  $RGB$  空间只需将相应的变换矩阵依次求逆即可。

设变换到  $l\alpha\beta$  颜色空间的源图像的均值和标准差分别为  $\mu_s$ 、 $\sigma_s$ ，目标图像的均值和标准差分别为  $\mu_t$ 、 $\sigma_t$ ，只需通过如下简单处理，即可将源图像的颜色分布变换到接近与目标图像

$$\begin{aligned} l' &= \frac{\sigma_t^l}{\sigma_s^l}(l - \mu_s^l) + \mu_t^l \\ \alpha' &= \frac{\sigma_t^\alpha}{\sigma_s^\alpha}(\alpha - \mu_s^\alpha) + \mu_t^\alpha \\ \beta' &= \frac{\sigma_t^\beta}{\sigma_s^\beta}(\beta - \mu_s^\beta) + \mu_t^\beta \end{aligned} \quad (5)$$

并将图像  $(l', \alpha', \beta')$  变换回  $RGB$  空间即可得到处理后的结果，前述两图的计算结果为：



图 4: ColorTransfer 结果

更多实验结果：

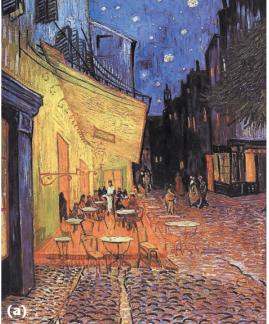
源图像	目标图像	结果图像
 (b)	 (a)	 (b)
 (a)	 (b)	 (a)
 (a)	 (b)	 (a)
 (b)	 (a)	 (b)
 (a)	 (b)	 (a)

表 1: Color Transfer 实验结果

## 4.2 Colorization

### 4.2.1 问题描述



图 5: 源图像



图 6: 目标图像

如上两幅图，我们希望通过设计一个灰度图着色算法，使得灰度目标图像能够利用彩色源图像的颜色信息进行合理的上色

### 4.2.2 算法描述

实验参考了论文 [2]，并使用全局匹配的方法进行求解。和 Color Transfer 类似地，我们在图像的  $l\alpha\beta$  颜色空间上进行求解，这是由于  $l\alpha\beta$  颜色空间的  $l$  即代表着亮度，通过查找灰度源图像上的像素与彩色目标图像的  $l$  分量最佳匹配像素，再将目标图像对应匹配像素的  $\alpha$  和  $\beta$  分量赋予源图像即可进行灰度图像上色。算法流程如下：

1. 将源图像  $img_{src}$  变换到  $l\alpha\beta$  颜色空间得到  $img_{src}^{(l\alpha\beta)}$ ，同时将源图像  $img_{src}$  转换为灰度图  $img_{src}^{(grey)}$
2. 利用此前 Color Transfer 的计算方法<sup>5</sup>，将  $img_{src}^{(l\alpha\beta)}$  的  $l$  通道和  $img_{src}^{(grey)}$  映射到具有灰度目标图像  $img_{tar}$  的像素值分布，得到  $luminance_{src}$  和  $luminance_{src}^{grey}$

3. 近邻域标准差计算。利用一个  $n \times n$  (本实验中取  $n = 5$ ) 滑动窗口遍历图像 (类似于卷积操作, 边界用 0 填充), 对窗口内的所有像素值计算它们的标准差并赋值到一幅标准差图像上。对  $luminance_{src}^{grey}$  和  $img_{tar}$  进行该计算, 得到  $stddev_{src}^{grey}$  和  $stddev_{tar}$
4. 最佳匹配查找。遍历  $img_{tar}$ , 已知  $img_{tar}$  上的某一像素值  $p_{tar}$  和其标准差  $stddev_{tar}$  对应的值  $\sigma_{tar}$ , 在源图像上寻找最佳匹配的像素点, 这里简单地采用加权的  $\mathcal{L}_2$  范数平方去算:

$$\arg \min_{x,y} \left( w_1 * \|luminance_{src}(x,y) - p_{tar}\|^2 + w_2 * \|stddev_{src}^{grey}(x,y) - \sigma_{tar}\|^2 \right) \quad (6)$$

其中  $w_1$  和  $w_2$  分别表示亮度值和标准差的对匹配结果的贡献, 这里简单地取  $w_1 = w_2 = 0.5$

5. 灰度图上色。将查找到的最佳匹配像素  $(\tilde{x}, \tilde{y})$  在  $img_{src}^{(l\alpha\beta)}$  所在像素值的  $\alpha$  和  $\beta$  分量赋值给源图像作为其  $\alpha$  和  $\beta$  通道的值, 而原来的灰度值则作为  $l$  分量继续使用, 并将源图像重新变换回  $RGB$  颜色空间得到最终的结果

前述两图的计算结果为:



图 7: Colorization 结果

更多实验结果：

源图像	目标图像	结果图像
		
		
		
		
		

表 2: Colorization 实验结果

## 5 总结

本次实验主要是利用图像像素点的统计特性进行图像颜色的变换，采用自搭 C++ 渲染框架进行扩展开发，并且充分利用 OpenCV 提供的内置算法以提高开发和程序执行的效率，在像素遍历操作方面使用 OpenCV 的并行访问加速方法，实测对性能提升有很大帮助

## A 附录

项目源代码、Demo 视频下载方法：

- 通过链接<https://rec.ustc.edu.cn/share/ca1e2da0-19c1-11ec-816d-0da21db7cb06>下载整个工程源代码以及编译好的可执行文件、视频 Demo
- 通过 Github 访问源代码<https://github.com/Chaphlagical/Chaf-Engine/tree/DIP>

如需从源码构建，需要配置好 OpenCV 的环境路径：

1. 下载 OpenCV 最新版，并安装在不含中文的路径上
2. 将 `opencv-path/build/x64/vc15/bin` 添加至系统变量 `Path` 中
3. 新建系统变量：变量名为 `OpenCV_DIR`，变量值为 `opencv-path/build`

然后使用 CMake 构建生成，使用 VS2019 进行编译和运行

## 参考文献

- [1] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

- [2] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 277–280, 2002.