天津大学

机器学习大作业课程报告

基于聚类算法的面向二次元风格图片的色块化色彩处理技术

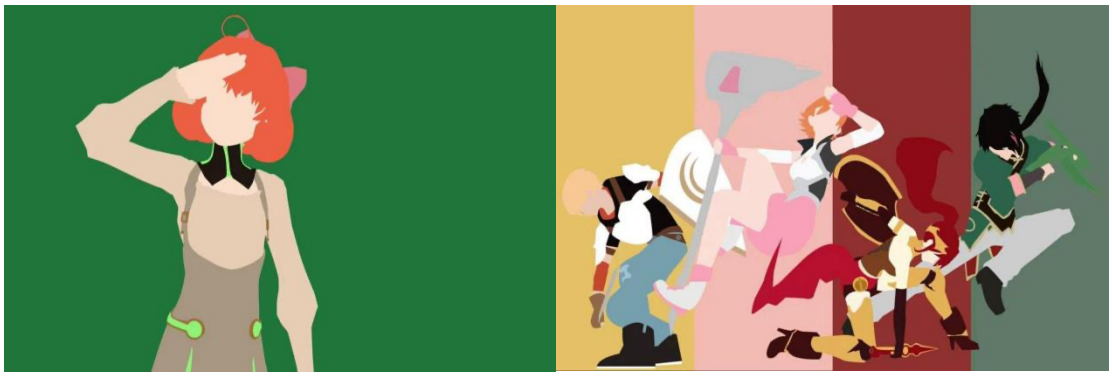
别问我为什么没有代码，因为不给你抄

哈哈哈哈哈哈哈哈哈哈

一，项目主题和动机

本项目的主题是基于聚类算法的面向二次元风格图片的色块化色彩处理技术。

本项目的动机是个人非常喜欢色块化风格的图片，但一般情况下画师不会主动的制作这类型的图片，而大多数的二创作品都是基于 ps 的手动抠图和填色制作的。



色块画

我第一次对色块画感兴趣是源自一个动画片叫 RWBY，这是一个以童话为蓝本的魔改的动画作品，其中人物众多，我搜索人物的背景故事时，有一个 UP 的视频封面就是这个，我就深深的被色块画吸引，她在直播中制作色块画的方式就是用 PS 抠图，然后指定颜色填色。

左图为动画截图，右图为视频封面

但抠图时间非常长，费时费力。后来接触**聚类算法**后，我发现相同颜色之间，即使有细微差距，也可以依靠聚类的方式兼并，从而起到**色块画**的效果，所以我立刻选择了这个题目为大作业的主题。

二，项目要求和设计方案

项目的要求就是生成令人满意的色块画，但实际上很难界定“满意”，所以只能把这个“NPH”问题化成几个简单的“NPC”问题来解决，其中包括：

1，保留主要轮廓，去除不必要的细节
2，用主色彩填充轮廓

初步的设计方案如此，但实际操作中遇到的问题相当多：

比如，色彩之间的细微差距会导致类似“**噪音**”的效果，影响聚类算法的判断，如图：

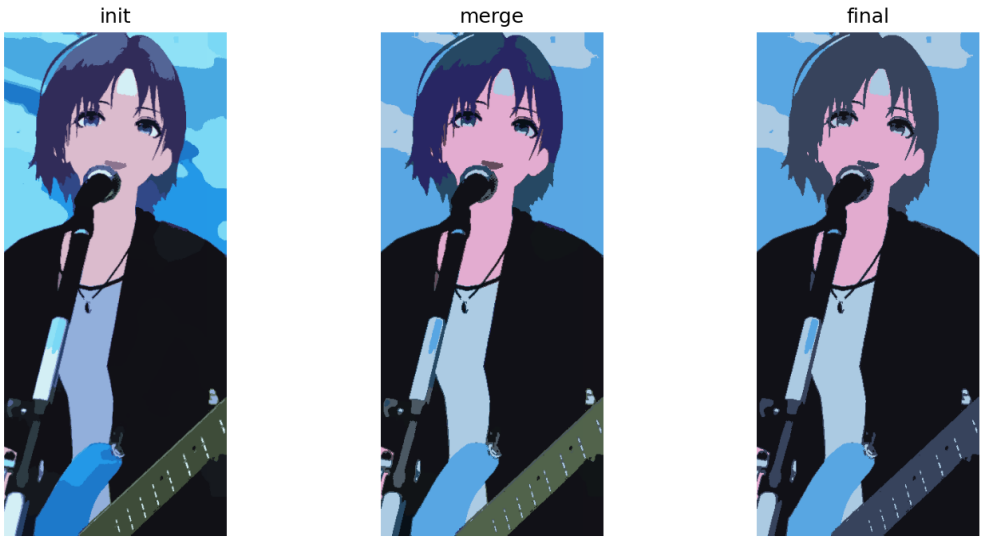
左图是预处理的照片，右图是聚类后的照片

由图可以看出，用聚类的方式直接处理色彩可以做到基本的颜色兼并，但任何算法在面对数据噪声的时候都会有漏洞。

左图的人脸是看不出右图的黑印的，因为黑印部分属于“脸红”，但聚类算法严格限制了颜色种类，而这部分的颜色数值又比较偏向灰色（HSV 色域的奇妙），所以这部分在聚类后就显现为不和谐的灰色。

为了解决这个问题，我尝试了许多滤波的方法，比如高斯滤波，中值滤波和傅里叶变换，它们导致的结果基本无一例外的是导致图像的轮廓模糊，而只有 MeanShiftFiltering 均值漂移滤波算法，可以维持轮廓的情况下滤波。

此外，不同色彩之间存在**光照的差距**，而光照的差距在计算机的眼中就会体现为 RGB 数值的差距，这时就需要一种方式“忽略”光照带来的色彩区别，而传统的计算机视觉光流处理是基于动态视频的，它需要前后文的数据才能做到消除光线影响，但我这个项目的数据只有单独的一张图片，无法依靠前后文对照的方式消除光照。

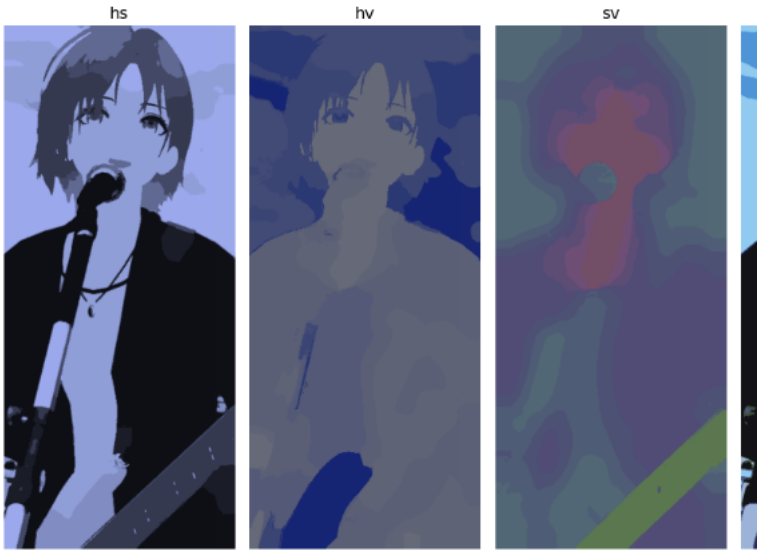


上图可明显看到，人物的额头和头发在阳光的影响下会造成聚类算法判断的错误，这是不能依靠修改聚类算法的超参数来解决的。

具体的解决方案是，我在了解色域的时候看到计算机处理色彩图像除了 RGB 还有一种 HSV 的方式，在 HSV 体系下，人物额头的光照强度往往体现在 H，S，V 三个域的某两个或某一个域中，这为解决光照问题提供了可能

从左到右分别为 V 通道，S 通道和 H 通道

故我可以在这三个色域中单独使用聚类算法，利用算法的简并性，去除光照在单色域下的存在，然后把三个通道合并，这是某两个或一个色域（不一定是 HSV 的哪个）是没有光照影响的，如图：

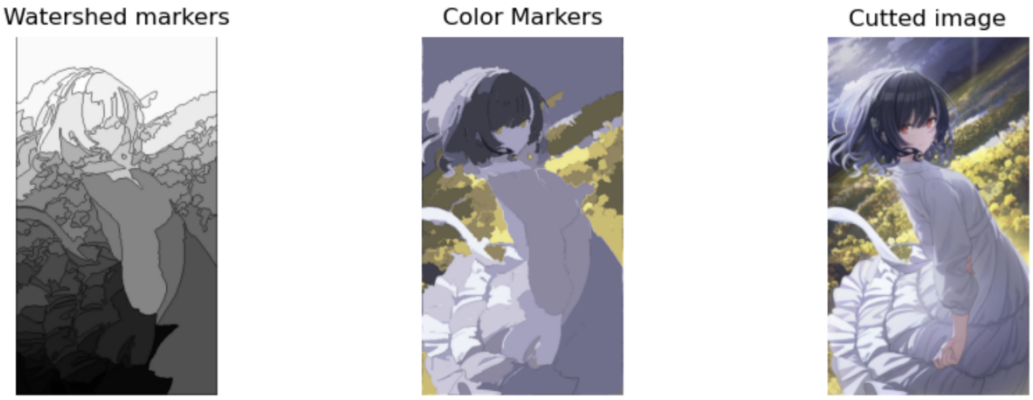


经过聚类后的 V 通道，S 通道和 H 通道

我们可以清楚的看到，在 S 通道和 H 通道中几乎看不到光照带来的影响，而即使 V 通道还保留一部分光照，我们在通道兼并后进行二次聚类，这种影响就会被抵消，从而做到消除光照的效果。

为了展示 HSV 和 RGB 的对比，我展示一张测试图，体验一下：

除此之外，我还尝试使用了计算机图像处理的**轮廓检测，分水岭算法**，实际上这占据了我**六成**的开发时间，以至于最后**放弃**这个方向的时候非常**心痛**，但事实上，轮廓检测算法如果没有深度学习模型的加成，几乎不可能做到合理的分割，要不就是色块侵蚀，要不就是颜色混乱，下图是跑的最成功一次：



左图为分水岭的掩膜，中图为掩膜的颜色填充图，右图为原图

上图使用了 canny 算法求解轮廓信息，分水岭算法划分色块，制作掩膜，然后依靠掩膜中的色彩均值来填充图像，这种算法看似非常通用，但实际上它直接放弃了保留轮廓，在复杂的图像中，即使是裙子的褶皱，人物的发丝，楼房的窗户，天空的云，都会导致色彩的严重混乱和侵蚀。

而且**高斯滤波和傅里叶滤波的滤波方式实在是太过抽象**，没有很好的可解释性和可视化方法，以至于开发难度极大，尤其是我这种没有系统了解的同学。

所以最终放弃了这个方向。

最终，我选择了多次变换色域+多次聚类来完成这个项目。

PS：本来想用密度聚类 DBSCAN 试一下，发现这个算法的标准库难用得很，要自己复

现的话也很难，所以放弃了。

三，项目实现简要流程

大致流程是：

1，导入图像，转换为 HSV

2，图像金字塔，降分辨率

3，MeanShiftFiltering 滤波

4，使用 cv 的 kmeans 进行预聚类

5，拆分通道，分为 hsv 三个通道

6，对每个通道使用 sklearn 的 kmeans 聚类

7，合并通道

8，图像转化为 RGB 格式

9，使用 cv 的 kmeans 进行最终聚类，确定总颜色数量

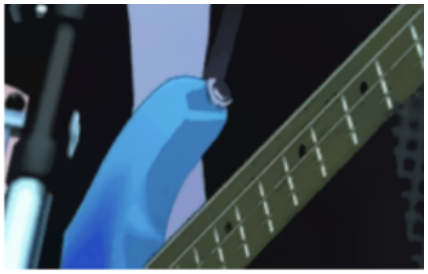
10，显示图像（可选保存图像）

首先使用 opencv 的库函数导入图像，这比 pillow 的库函数要方便且直观，没有多余的类包装，直接就是多重数组。

然后使用从 cv2.pyrDown 函数使图像分辨率降低，因为在我的实际测试中，即使是运用了 cuda 异构计算的 cv 库函数，在面对大于 400000 数据的图像信息时，依然会完美导致电脑崩溃，这个 400000 是经验数据，没有科学依据。

OPENCV 和 SKLEARN 都拥有自己的 kmeans 算法，前者的算法包括了对像素点位置信息的计算，后者完全依赖于输入数据的通道数，两者都有各自重要的作用。

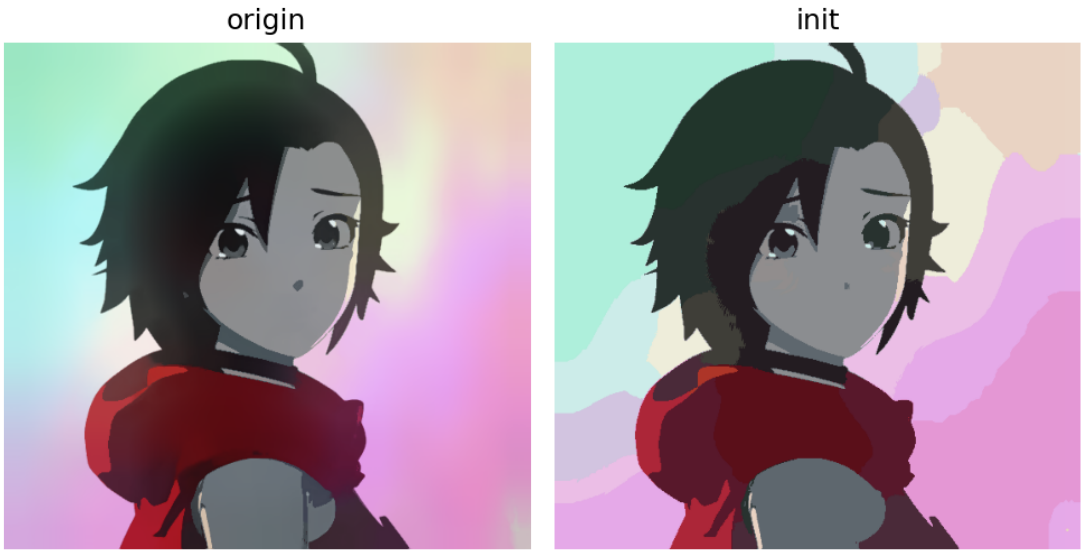
OPENCV 的 kmeans 函数能够有效利用图像的像素点位置信息来进行聚类算法，可以更有效的吞并细节信息，比如人物的饰品，吉他的品丝，衣服的扣子等。并且 cv 的 k 均值实现了 kmeans++的初始化方法，测试中发现，这个方法可以有效提高稳定性和速度。



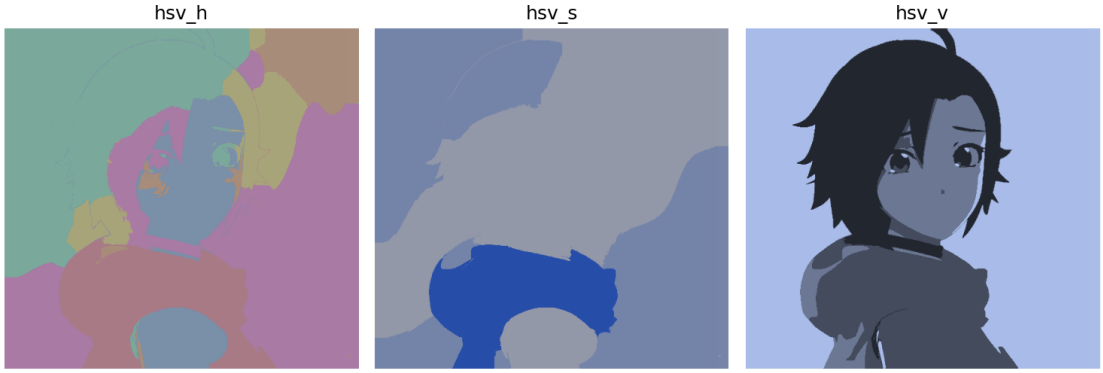
SKLEARN 的 kmeans 函数只依靠通道信息进行聚类，可以有效地保护色彩不被周边的大块颜色吞并，比如人物头发的颜色和衣服相近，这时候使用 cvkmeans 就会导致颜色兼并以至于人物的头发和衣服变成一个颜色。

（强行复现 bug）

使用 cv.kmeans 函数预聚类，预聚类可以有效降低颜色数量和颜色敏感度，因为过于突出的颜色很可能，直接聚类成少量的色彩时，会导致它“带偏”整幅图，所以预聚类的作用就是让这种特立独行的颜色先“安分下来”，具体的预聚类数目我选择的是 20，并且这个数字**不算入超参数**，因为没有必要，在测试中发现，任何图片无论复杂与否，这个数字都可以很好地驾驭，没必要增加超参数。



使用 cv.split 函数拆分通道，并进行二次聚类，使用 sklearnkmeans 聚类 hsv 三个通道。



然后合并，并进行最后一次 cv.kmeans 聚类，这次的聚类严格执行色彩数量约束。
封装函数如下，超参数只有四个，不多不少，任何图片都可以做到较好的效果。

```
def process(path, Color=8, H=8, S=2, V=2, save=False, show=True):
    # path 图片路径
    # Color 最终颜色聚类簇数
    # H（色调、色相）
    # S（饱和度、色彩纯净度）
    # V（明度）
    # save 是否保存
    # show 是否展示
```

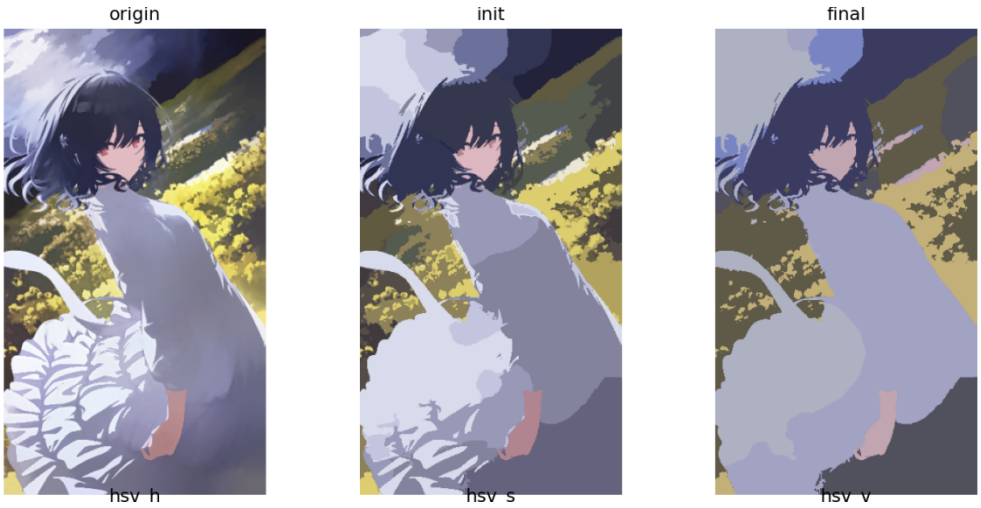
四，项目结果和分析

项目结果保存在文件夹中，共有三组实例图片，对应的函数和参数卸载 main .py 的主函数中了，接下来展示所有图片的输出结果，相对而言令人满意。

因为没有什么**准确度**的问题，所以没有相关的数据输出，即使是聚类算法，在高次数的迭代下，这种纯色的色块画也没什么可波动的，所以一个比较高的迭代值就可以解决问题，故也不测了。

Pic1

Pic2



Pic3

五、还需改进的地方

本项目的漏洞较大，比如背景和人物是分不开的，会抢颜色，后期需要引入**注意力机制**和**深度学习**的算法，保证项目拥有更好的发挥。

PS：我使用 SEGMENT ANYTHING 试了一下，它根本不能生成理想的遮罩，一旦颜色复杂起来，它也不能解决，所以事实上这个项目的前景很好。

（二次元是检验图像技术的唯一标准；）