|  |
| --- |
| 117实验室 |
| 推荐算法文档 |
| 陈婕 |

|  |
| --- |
| cj silence  2017-7-20 |

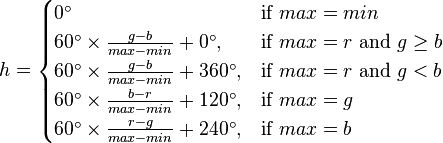
修改记录

|  |  |  |
| --- | --- | --- |
| 修改日期 | 修改人 | 修改摘要 |
| 2017.07.20 | 陈婕 | Initial commit |
| 2017.08.01 | 陈婕 | 完善色系分类及伪代码 |
| 2017.08.27 | 陈婕 | 扩充色盘分类及伪代码 |

# 0 概述

## 0.1 RGB到HSV的转换

设 (*r*, *g*, *b*) 分别是一个颜色的红、绿和蓝坐标，它们的值是在 0 到 1 之间的实数。设 *max* 等价于 *r*, *g* 和 *b* 中的最大者。设 *min* 等于这些值中的最小者。



s = \begin{cases}0, & \mbox{if } max = 0 \\\frac{max - min}{max} = 1 - \frac{min}{max}, & \mbox{otherwise}\end{cases}

v = max \,

## 0.2 HSV简介

HSV(Hue, Saturation, Value)是根据颜色的直观特性由A. R. Smith在1978年创建的一种颜色空间, 也称六角锥体模型(Hexcone Model)。这个模型中颜色的参数分别是：色调（H），饱和度（S），明度（V）。HSV模型的三维表示从RGB立方体演化而来。设想从RGB沿立方体对角线的白色顶点向黑色顶点观察，就可以看到立方体的六边形外形。六边形边界表示色彩，水平轴表示纯度，明度沿垂直轴测量。

1）色调H

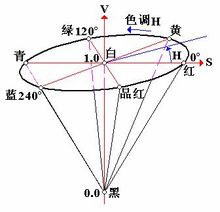
用角度度量，取值范围为0°～360°，从红色开始按逆时针方向计算，红色为0°，绿色为120°,蓝色为240°。它们的补色是：黄色为60°，青色为180°,品红为300°；

2）饱和度S

饱和度S表示颜色接近光谱色的程度。一种颜色，可以看成是某种光谱色与白色混合的结果。其中光谱色所占的比例愈大，颜色接近光谱色的程度就愈高，颜色的饱和度也就愈高。饱和度高，颜色则深而艳。光谱色的白光成分为0，饱和度达到最高。通常取值范围为0%～100%，值越大，颜色越饱和。

3）明度V

明度表示颜色明亮的程度，对于光源色，明度值与发光体的光亮度有关；对于物体色，此值和物体的透射比或反射比有关。通常取值范围为0%（黑）到100%（白）。



## 0.3 欧几里得距离

欧几里得度量（euclidean metric）（也称欧氏距离）是一个通常采用的距离定义，指在m维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

对于RGB或HSV空间的色彩而言，欧几里得距离具体指的是在三维向量空间中，每个两个颜色各自对应的点在三维空间中的真实距离。公式如下：

其中) ,)

HSV同理。

# 1 扩充肤色的色例

## 1.1肤色色例扩充流程图



图1-1 扩充肤色的色例流程图

## 1.2 肤色的色例扩充伪代码

|  |
| --- |
| 输入：色例对应的图片：X.jpg，色例对应的色系：colorType |
| 输出：色例扩充结果：colorList |
| 01：调用函数get\_dominant\_color获取x.jpg对应的rgb |
| 02：for r in (0,256,5) |
| 03： for g in (0,256,5) |
| 04： for b in (0,256,5) |
| 05 将色例的rgb转为(H1，S1，V1) |
| 06： 将(r,g,b) 转为(H2，S2，V2) |
| 07： 获取该色系限制条件 |
| 08： if (r,g,b)满足色系限制条件 |
| 09： 将(r,g,b) 以及它与色例的距离加入最终结果集 |
| 10：对最终结果集按照距离降序排序 |
| 11：选择前N个结果作为输出集合colorList |

## 1.3 色系限制条件与分类

|  |  |  |
| --- | --- | --- |
| 色系名 | 文件夹名称 | 限制条件 |
| 红色系 | Red | max(r,g,b) == r |
| 紫色系 | Purple | 无限制 |
| 绿色系 | Green | 无限制 |
| 蓝色系 | Blue | max(r,g,b) == b |
| 橙色系 | Orange | 无限制 |
| 黄色系 | Yellow | 在g！=0时，r/g大于0.9小于1.1 |
| 棕色系 | Brown | 无限制 |
| 白色系 | White | 无限制 |
| 黑色系 | Black | R<70 and g<70 and b<70 |

# 2 扩充色盘的色例

本文认为色盘上的任何一个颜色都可以被映射到一个基础颜色分类。

## 2.1 关于色盘量化存储

如图2-1所示，对于每个色盘，从红色箭头标注扇区开始，每个扇区对应一个元组，整个色盘存储为一个一维数组，由于一个扇区为15°，故整个色盘共有24个扇区，因此整个每个色盘对应于一个含有24个元组的一维数组。例如图2-1对应的一维数组为：

color\_one = [("null","null","null"),("152,158,35","126,130,57","null"),

("171,164,66","153,147,63","133,127,49"),("194,165,49","181,153,40","null"),……,("null","null","null")]

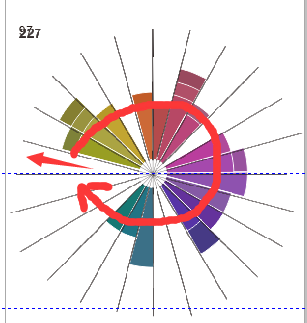


图2-1 色盘举例

## 2.2 色系限制条件与分类

基础颜色分类包括黑、白、灰、红、橙、黄、绿、青、蓝、紫。但是在对这些色系的色例进行扩充时，并不是所有的色系在扩充时都有限制条件。故只列出有限制条件的色系，以及在进行筛选时在HSV三个维度的限制条件。

|  |  |  |  |
| --- | --- | --- | --- |
| 颜色 | H的范围 | S的范围 | V的范围 |
| Red |  |  |  |
| Yellow |  |  |  |
| Black |  |  |  |
| Blue |  |  |  |
| Default | 以上H的范围都不满足 | 以上S的范围都不满足 | 以上V的范围都不满足 |

## 2.3 扩充色盘的流程图



## 2.4 色盘扩充流程伪代码

|  |
| --- |
| 输入：每一扇区的色例元祖colorTuple，如("152,158,35","126,130,57","null") |
| 输出：色例扩充结果：colorList |
| 01：for color\_str in colorTuple |
| 02：if color\_str !=”null”: |
| 03： 将color\_str转为三个项的元祖color:  04: colorType = judugeColorType(color)  05: 初始化结果集result=[]  06： for r in (0,256,5)  07： for g in (0,256,5)  08： for b in (0,256,5)  09： 判断(r,g,b)的颜色类型与色例的颜色类型colorType是否一致 |
| 05 将色例的rgb转为(H1，S1，V1) |
| 06： 将(r,g,b) 转为(H2，S2，V2) |
| 07： 获取该色系限制条件 |
| 08： if (r,g,b)满足色系限制条件 |
| 09： 将(r,g,b) 以及它与色例的距离加入最终结果集 |
| 10：对最终结果集按照距离降序排序 |
| 11：选择前N个结果作为输出集合colorList |

## 2.5 效果展示

原色例；

扩充后：



# 3 关于颜色扩充的进一步改进

第二章阐述的伪代码中，第6~8行是用于组成一个(R,G,B)候选三元组，转换成HSV后，与色例进行HSV空间上的距离运算，然后选出距离最小的前N个。这样做可以保证每个候选元组都被考虑到，但是会造成计算量过大。对每个色例都要进行256\*256\*256次距离计算，在进行排序的时候要对这么多个结果进行排序，这样做会导致计算时间过长，内存占用过大。故想到了如下优化的方法：

在对一个色例（r，g，b）进行颜色扩充时，在原有基础上，对区间进行缩减。原本R,G,B的定义域在[0,255]的范围内，现在在R,G,B三个维度上缩小到，，的范围，大大减少了不必要的计算量。

# 参考文档：

http://blog.csdn.net/taily\_duan/article/details/51506776