第 3 章 色彩空间

3.1 色彩类型

图像有多种色彩空间类型，包括 RGB 色彩空间、GRAY 色彩空间、XYZ 色彩空间、YCrCb 色彩空间、HSV 色彩空间、HLS 色彩空间、Bayer 色彩空间等。本章主要介绍几类常用的色彩空间。

3.1.1 RGB 色彩空间

RGB(红、绿、蓝)是构成图像的三原色，即所有颜色都可以通过 RGB 这 3 种颜色混合而成，每个通道的取值范围为 0～255，如图 3-1 所示，蓝色、绿色、红色分别用(0,0,255)、(0,255,0)、(255,0,0)表示，红绿蓝三色混合成白色(255,255,255)，红色和绿色混合成黄色(0,255,255)，蓝色和绿色混合成青色(255,255,0)，红色和蓝色混合成品红(255,0,255)。每种颜色用 3 个通道表示，每个通道有 256 种取值，因此 3 个通道不同取值的组合可以表示 256×256×256 种颜色。

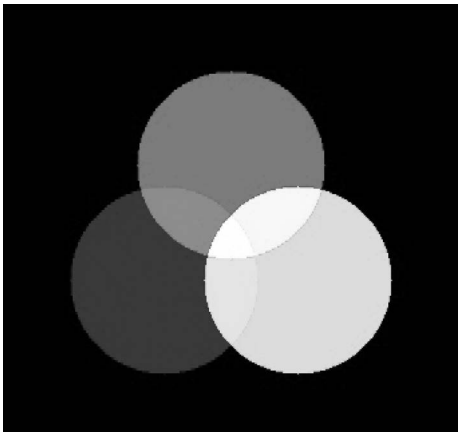


图 3-1 彩色图

【例 3-1】 查看彩色图像颜色通道。

彩色图像每种颜色由 3 个数值表示，通过查看彩色图像的颜色通道，了解各个通道的特性。

解：

(1) 读入彩色图像。OpenCV 以 BGR 的格式读入图像，若要正常显示图像，需要把 BGR 格式转换为 RGB 格式。

(2) 获取图像的各个通道。分别获取每个通道的颜色，将其他通道数值设置为 0。例如

当彩色图像只有 B 通道有数值且其他两个通道的数值为 0 时，整个图像呈蓝色。

（3）显示图像，代码如下：

```
#chapter3_1.py 显示彩色图像的 3 个通道
import cv2
import numpy as np
import matplotlib.pyplot as plt

#1. 读入彩色图像 img
img = cv2.imread('pictures/L1.png', 1)
#2. 获取图像的各个通道 img[...,[b,g,r]]
#获取彩色图像的蓝色通道，并存放在图像 b 中
b = img.copy()
b[..., [1, 2]] = 0
#获取彩色图像的绿色通道，并存放在图像 g 中
g = img.copy()
g[..., [0, 2]] = 0
#获取彩色图像的红色通道，并存放在图像 r 中
r = img.copy()
r[..., [0, 1]] = 0
#3. 显示图像
re = np.hstack([img, b, g, r])
plt.imshow(re[..., ::-1])
cv2.imwrite('pictures/p3_2.jpg',re)
```

运行结果如图 3-2 所示。

(a) 原图　(b) 蓝色通道　(c) 绿色通道　(d) 红色通道

图 3-2　彩色图像颜色通道

【例 3-2】 图像色彩变换。

使用不用的权重对各个通道加权，使图像展现不同的色彩。

解：

（1）图像预处理。把图像像素设置在 0～1 内。

（2）通道加权。设置各通道的权重，根据权重对通道加权，实现图像颜色变换的目的。

（3）显示图像。把原图和变换后的图像水平放置，并显示、保存图像，代码如下：

```
#chapter3_2.py
import cv2
```

```
import numpy as np
import matplotlib.pyplot as plt

#1. 图像预处理
img0 = cv2.imread('huaduo.jpeg', 1)
#归一化
img = img0.copy() / 255.0
#2. 通道加权
#拆分通道
b, g, r = cv2.split(img)
#权重系数
wb1 = 0.5
wg1 = 0.5
wr1 = 0.55
wb2 = 0.25
wg2 = 0.25
wr2 = 0.75
#通道处理
r1 = wr2 *(wb1 *g + (1 - wb1) *b) + (1 - wr2) *r
g1 = wg2 *(wr1 *b + (1 - wr1) *r) + (1 - wg2) *g
b1 = wb2 *(wg1 *r + (1 - wg1) *g) + (1 - wb2) *b
#通道融合
img1 = (cv2.merge([r1, g1, b1]) *255).astype(np.uint8)
#3. 显示图像
re = np.hstack([img0[..., ::-1], img1])
plt.axis('off')
plt.imshow(re)
#cv2.imwrite('imgs_re/chapter_03/p3_3.jpeg', re)
```

运行结果如图 3-3 所示。

(a) 原图　　(b) 通道加权

图 3-3　图像色彩变换

【例 3-3】 生成老照片。

按照固定权重加权图像的各个通道，使图像看起来像老照片。

解：

(1) 图像预处理。把图像像素设置在 0～1 内。

(2) 通道加权。根据固定权重对通道进行加权，再进行通道融合。

(3) 显示图像，代码如下：

```
#chapter3_3.py
import cv2
import numpy as np
import matplotlib.pyplot as plt

#1. 图像预处理
img = cv2.imread('pictures/L3.png', 1)
#归一化
img1 = img.copy() / 255.0
img0 = img1.copy()
#2. 通道加权
#拆分通道
b, g, r = cv2.split(img1)
r1 = 0.393 *r + 0.769 *g + 0.189 *b
g1 = 0.349 *r + 0.686 *g + 0.168 *b
b1 = 0.272 *r + 0.534 *g + 0.131 *b
#合并通道
img1 = cv2.merge([r1, g1, b1])
#像素归一化为 0~1
img1 = (img1 - img1.min()) / (img1.max() - img1.min())
#3. 显示图像
re = np.hstack([img0[..., ::-1], img1])
re = (re *255).astype('uint8')
plt.imshow(re)
cv2.imwrite('pictures/p3_4.jpeg', re[..., ::-1])
```

运行结果如图 3-4 所示。

(a) 原图　　(b) 老照片

图 3-4　生成老照片

3.1.2　GRAY 色彩空间

GRAY(灰度图像)通常指 8 位灰度图,只有一个通道,每个像素有 256 个灰度级,像素的范围是[0,255]。在 OpenCV 中,彩色图像通过对各个通道加权得到灰度图像,处理方式如下:

$$\text{Gray} = 0.114 \times \text{B} + 0.587 \times \text{G} + 0.299 \times \text{R} \tag{3-1}$$

其中，B、G、R 分别为彩色图像的蓝、绿、红颜色通道。彩色图像灰度化也可以对 3 个通道求平均值：

$$\text{Gray} = (\text{B} + \text{G} + \text{R})/3 \tag{3-2}$$

当将灰度图像转换为彩色图时，直接把灰度图像的通道复制 3 遍，堆叠在一起。生成的彩色图像与灰度图像视觉上无异，但是彩色和灰色图像的通道数不一样。

$$\begin{aligned} \text{B} &= \text{Gray} \\ \text{G} &= \text{Gray} \\ \text{R} &= \text{Gray} \end{aligned} \tag{3-3}$$

【例 3-4】 彩色图像转灰度图。

使用两种不同的方式，把彩色图转换成灰度图。

解：

（1）读取图像。读入彩色图像和灰度图像，对彩色图进行通道拆分。

（2）彩色图像转灰度图像。分别用平均法和加权法生成灰度图像。

（3）显示图像，代码如下：

```
#chapter3_4.py
import cv2
import numpy as np
import matplotlib.pyplot as plt

#1. 读取图像
img = cv2.imread('pictures/L1.png', 1)          #读取彩色图像
img_gray = cv2.imread('pictures/L1.png', 0)     #读取灰度图像
#拆分通道
b, g, r = cv2.split(img)
#2. 彩色图像转灰度图像
#平均法，每个通道的 b、g、r 乘以 1.0，把数据类型由 uint8 转换为浮点型，防止图像失真
img_gray_avg = (b *1.0 + g *1.0 + r *1.0) / 3
#加权法
img_gray_weg = 0.114 *b + 0.587 *g + 0.299 *r
#3. 显示图像
re = np.hstack([img_gray, img_gray_avg, img_gray_weg])
plt.imshow(re, 'gray', vmin=0, vmax=255)
cv2.imwrite('pictures/p3_5.jpeg', re)
```

运行结果如图 3-5 所示。

3.1.3　HSV 色彩空间

HSV 色彩空间是从颜色色度、亮度、饱和度 3 个方面描述色彩空间变化。色调（Hue）是指光的颜色，例如赤橙黄绿青蓝紫。饱和度（Saturation）是指色彩的深浅程度，饱和度越高，颜色越浓；反之，颜色越淡。亮度（Value）指颜色的明暗程度，亮度越高，说明颜色越亮；反之，越暗。

(a) 灰度图像　　(b) 通道平均　　(c) 通道加权

图 3-5　彩色图像转灰度图像

色度的取值范围为[0,360],饱和的取值范围为[0,1],亮度的取值范围为[0,1]。把彩色图像映射到[0,1]区间上,再通过以下方式可以转换成 HSV 色彩空间:

$$V=\max(\mathrm{R},\mathrm{G},\mathrm{B}) \tag{3-4}$$

$$S=\begin{cases}\dfrac{V-\min(\mathrm{R},\mathrm{G},\mathrm{B})}{V}, & V\neq 0\\ 0, & \text{其他}\end{cases} \tag{3-5}$$

$$H=\begin{cases}\dfrac{60-(\mathrm{G}-\mathrm{B})}{V-\min(\mathrm{R},\mathrm{G},\mathrm{B})}, & V=\mathrm{R}\\ 120+\dfrac{60-(\mathrm{B}-\mathrm{R})}{V-\min(\mathrm{R},\mathrm{G},\mathrm{B})}, & V=\mathrm{G}\\ 240+\dfrac{60-(\mathrm{R}-\mathrm{G})}{V-\min(\mathrm{R},\mathrm{G},\mathrm{B})}, & V=\mathrm{B}\end{cases} \tag{3-6}$$

如果计算得到 $H<0$,则需要对 H 进一步地进行处理:

$$H=\begin{cases}H+360, & H<0\\ H, & \text{其他}\end{cases} \tag{3-7}$$

在图像处理中,把图像从 BGR 格式转换到 HSV 格式,然后确定特定颜色的取值范围,获取特定颜色的掩模,再根据掩模和原图做逻辑运算就可以获取原图上指定颜色区域。常见色彩的 HSV 值见表 3-1。

表 3-1　HSV 色彩区间

HSV	白	红	黑	灰	橙	黄	绿	青	蓝	紫
h_min	0	0|156	0	0	11	26	35	78	100	125
h_max	180	10|180	180	180	25	34	77	99	124	155
s_min	0	43	0	0	43	43	43	43	43	43
s_max	30	255	255	43	255	255	255	255	255	255
v_min	221	46	0	46	46	46	46	46	46	46
v_max	255	255	46	220	255	255	255	255	255	255

3.2　色彩空间转换

色彩空间类型转换是指将图像从一个色彩空间转换到另外一个色彩空间。通过色彩空间转换，可以方便用户根据需求对图像进行操作。例如先把彩色图像转换为灰度图，然后对灰度图阈值化处理使其变为二值图，再根据二值图寻找图像轮廓。

在 OpenCV 内，使用 cv2.cvtColor()函数实现色彩空间变换，其语法格式如下：

dst＝cv2.cvtColor(scr,code[,dstCn])

(1) dst：输出图像。

(2) scr：输入图像。

(3) code：色彩空间转换码。色彩空间转换码有多种类型，常用类型有 cv2.COLOR_RGB2BGR、cv2.COLOR_BGR2RGB、cv2.COLOR_BGR2HSV、cv2.COLOR_RGB2HSV、cv2.COLOR_BGR2GRAY、cv2.COLOR_RGB2GRAY、cv2.COLOR_HSV2BGR、cv2.COLOR_HSV2RGB。

(4) dstCn：目标图像通道数。

【例 3-5】 颜色分割。

使用 kmeans 算法对图像像素进行分类。

解：

(1) 图像处理。先读入彩色图像，对图像做高斯模糊去除噪声，再把图像展开成一维，每个像素为一个单元。

(2) 颜色分类。设置分类类别数和分类标准，用 kmeans 算法把像素分为两类，得到类的中心点坐标和每个像素的类别。根据每个像素的类别，把每个像素用所属类别值替代，从而得到分割数据，最后把分割数据缩放到原图尺寸。

(3) 显示图像，代码如下：

```
#chapter3_5.py
import cv2
import numpy as np
import matplotlib.pyplot as plt

#1. 图像处理
img0 = cv2.imread('pictures/knn.png', 1)
img0 = cv2.cvtColor(img0, cv2.COLOR_BGR2RGB)
#高斯模糊
img_g = cv2.GaussianBlur(img0, (13, 13), 10, 10)
h, w, c = img_g.shape #(110, 283, 3)
img_blur = img_g.reshape([-1, 3]).astype('float32') #(31130, 3)
#2. 颜色分类
#分类标准
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
#分类类别数
num_clusters = 2
```

```
#kmean 分类,把像素分为两类,center_color 存储两类颜色的均值
#label 存储每个像素属于哪个类别
_, label, center_color = cv2.kmeans(img_blur, num_clusters,
                                    None, criteria,
                                    num_clusters,
                                    cv2.KMEANS_RANDOM_CENTERS)
#把类别中心转换成 uint8
center_color = center_color.astype(np.uint8)
#img_blur[398396,3],label[398396,1],center[2,3]
#把每个像素用对应类别的中心点像素替代,res[398396,3]
res = center_color[label.ravel()]
#res[398396,3]--> [548,727,3]
res = res.reshape([h, w, c])
#3. 显示图像
re = np.hstack([img0, img_g, res])
plt.imshow(re)
#cv2.imwrite('pictures/p3_6.jpeg', re[..., ::-1])
```

运行结果如图 3-6 所示。

(a) 原图　　(b)高斯模糊图　　(c) 颜色分割图

图 3-6　图像颜色分割

【例 3-6】 色彩抠图。

抠取图像指定颜色区域。

解：

(1) 图像处理。将图像 img 转换到 HSV 空间。

(2) HSV 色彩抠图。先获取指定颜色像素,把指定颜色转换到 HSV 空间。再根据 HSV 设定指定颜色的阈值。例如要获取图像中的黄色区域,先获取图中任意黄色点的像素。打开并显示彩色图像,将鼠标移动到图像黄色区域,获取黄色点的坐标[y,x]。假设根据坐标获取 img[y,x]的黄色像素为[246,199,67],把黄色像素由 BRG 转换为 HSV 为[98,186,246],然后根据[H－100,100,100]和[H＋100,255,255]设定黄色的取值范围。最后根据阈值构建掩模,通过原图与掩模的逻辑与运算获取指定颜色区域。

(3) 显示图像,代码如下：

```
#chapter3_6.py
import cv2
import numpy as np
import matplotlib.pyplot as plt

#1. 图像处理。读取图像并转换成 HSV
img = cv2.imread('pictures/knn.png', 1)
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

```
#2. HSV 色彩抠图
#获取指定颜色
yellow = np.uint8([[[246, 199, 67]]])
#把指定颜色转换到 HSV
hsv_yellow = cv2.cvtColor(yellow, cv2.COLOR_BGR2HSV)
print(hsv_yellow) #[[[ 98, 186, 246]]]
#设定指定颜色的阈值 [H-100,100,100] 和 [H+100,255,255]
lower_yellow = np.array([0, 100, 100])
upper_yellow = np.array([198, 255, 255])
#根据阈值构建掩模
mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
#对原图像和掩模进行位运算,从而得到指定颜色
img1 = cv2.bitwise_and(img, img, mask=mask)
#3. 显示图像
re = np.hstack([img, img1])
plt.imshow(re[..., ::-1])
cv2.imwrite('pictures/p3_7.jpeg', re)
```

运行结果如图 3-7 所示。

(a) 原图　　(b) 抠图

图 3-7　色彩抠图

【例 3-7】 根据 HSV 定位车牌。

解：

(1) 图像处理。

(2) HSV 色彩抠图。将图像转换到 HSV 空间。车牌由绿色和白色组成，获取绿色、白色像素并转换到 HSV 空间。根据 HSV 设定指定颜色的取值范围。再根据阈值构建掩模，最后通过原图与掩模的逻辑与运算获取指定颜色区域。

(3) 显示图像，代码如下：

```
#chapter3_7.py 车牌定位
import cv2
import numpy as np
import matplotlib.pyplot as plt

#1. 读取图像
img = cv2.imread("pictures/che.png", 1)
#2. HSV 色彩抠图
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
#设定指定颜色的阈值 [H-100, 100, 100] 和 [H+100, 255, 255]
#绿色取值范围,green[70, 149, 217]
lower_green = np.array([0, 100, 100])
upper_green = np.array([170, 255, 255])
#白色取值范围,white[58, 18, 230]
```

```
lower_white = np.array([0, 100, 100])
upper_white = np.array([158, 255, 255])
#根据阈值构建掩模
mask1 = cv2.inRange(hsv, lower_green, upper_green)
mask2 = cv2.inRange(hsv, lower_white, upper_white)
mask = mask1 & mask2
#对原图像和掩模进行位运算,从而得到指定颜色
img1 = cv2.bitwise_and(img, img, mask=mask)
#3. 显示图像
re = np.hstack([img, img1])
plt.imshow(re[..., ::-1])
cv2.imwrite('pictures/p3_8.jpeg', re)
```

运行结果如图 3-8 所示。

(a) 原图　　(b) 车牌

图 3-8　车牌定位

【例 3-8】 去除背景噪声。

解：

(1) 读取图像。

(2) 图像处理。调整图像亮度，设置和原图一样大、像素为 70 的模板 tmp，然后把 tmp 与原图像相加，从而得到图像 out，使原图背景中的黑色阴影淡化。如果想得到更好的效果，则可把图像 out 中的设计草图分割出来，从而得到掩码 m，根据掩码 m 对图像 out 做与运算，从而得到最终图像 out_1。

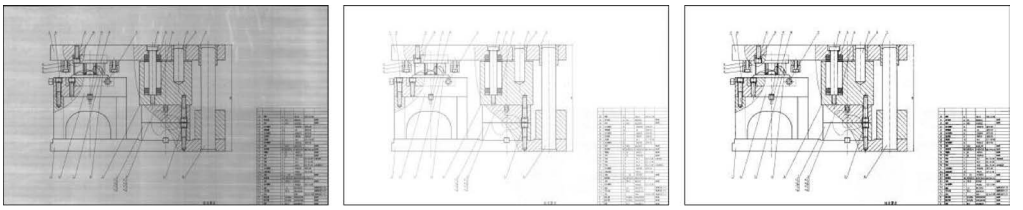
(3) 显示图像，代码如下：

```
#chapter3_8.py
import cv2
import numpy as np
import matplotlib.pyplot as plt

#1. 读取图像
img = cv2.imread("pictures/tu.jpeg", 1)
#2. 图像处理
#调整图像亮度
tmp = np.ones_like(img) *70
#给原图像的每个像素加 70,从而提升图像亮度
```

```
out = cv2.add(img, tmp)
#分割出图像草图
m = np.array(out[..., 0] > 190).astype(np.int8)
#按位与运算
out1 = cv2.bitwise_and(out, out, mask=m)
#3. 显示图像
re = np.hstack([img, out, out1])
plt.imshow(re)
cv2.imwrite('pictures/p3_9.jpeg', re)
```

运行结果如图 3-9 所示。



(a) 原图　(b) 图像out　(c) 图像out_1

图 3-9　去除背景噪声