Please visit website: http://cxyroad.com

看完稳赚不亏，计算机视觉的基础概念与入门

欢迎点赞收藏哟

快进来看爆款文章

一. 前言

之前学习了一下 Python 环境下计算机视觉方面的一些应用（主要是 OpenCV）.

但是对于计算机视觉方面的种种概念都是一笔带过，计算机视觉是一个很大的领域，`在深入它之前 ，有必要对其中的一些基础概念有一个宏观的理解`。

二. 要学清楚哪些东西?

* ： 一个图像在计算机里面存储的时候 ， 涉及到了哪些概念?
* : 图像在计算机里面存储的实质是什么样的?
* : 那么最终在应用中如何产生作用?

三. 学一学那些生涩又难懂的术语

3.1 非黑即白 ： 计算机图形的存储基础

上学的时候或多或少都学过 ， 计算机底层的存储方式是二进制。**不论什么数据 ， 在最底层都是 0-1 的关系，图像同样如此。**

* 以 《**计算机视觉40例从入门到深度学习**》 这里面的一个案例来说明 ：

![image.png](https://p6-xtjj-sign.byteimg.com/tos-cn-i-

73owjymdk6/c20444e222ed427cbf5aed0ad6ecb41a~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expires=1722163286&x-signature=v0%2BXtmha6zsmbDoWGfIdJXi3LBE%3D)

但为了表示一张复杂的图形 ， 远远不是 0-1 这么简单 ， 由此出现了一个概念 ： **像素**。

* 像素（Pixel）是计算机图像的基本单位，`用于表示图像中的一个点`。每个像素都包含`颜色`和`亮度`信息，组合在一起形成整个图像。
* 如果抽象从黑白的角度来说 ， 一个像素点就是一个字节 ， 有八位Bit 表示 0-255 这个区间

> 分辨率是什么?

分辨率是指图像或显示器上像素的密度，通常以水平像素数和垂直像素数来表示，如 1920x1080。它决定了图像或显示器可以显示的细节和清晰度

* 一个分辨率为 1920x1080 的图像意味着图像有 1920 个像素宽度和 1080 个像素高度
* 分辨率越高，图像可以显示的细节和清晰度就越高 ， 可以展示更多的细节（`但是不等于图像就清晰了，它只是可以`）

3.2 颜色模型 ： 将颜色进行量化

之前说到了 ， 一个像素点里面是包含颜色信息的 ， 这些颜色信息通常称为 ： **颜色模型**

* **RGB（Red, Green, Blue）模型** ： 通过`红绿蓝` 3色组合形成的颜色模型
+ 表示方式：每个颜色分量通常用8位表示，范围是0到255
- `红色`：R = 255, G = 0, B = 0
- `白色`：R = 255, G = 255, B = 255
- `灰色`：R = 128, G = 128, B = 128
* **RGBA（Red, Green, Blue, Alpha）模型** : 在上述的基础上增加了透明度的概念
+ 表示方式：Alpha通道也用8位表示
- `不透明红色`：R = 255, G = 0, B = 0, A = 255
- `半透明红色`：R = 255, G = 0, B = 0, A = 128
* **CMYK（Cyan, Magenta, Yellow, Black）模型** : 由`青色、品红、黄色

和黑色`四种颜色
+ 表示方式：每个颜色分量的范围是0到100%
– 青色：C = 100%, M = 0%, Y = 0%, K = 0%
* **HSB/HSV（Hue, Saturation, Brightness/Value）模型**：用于描述颜色的感知特性
+ 表示方式：通过色相，饱和度，亮度来全方位的表示
– `色相（Hue）`：表示颜色的类型，范围是0到360度。
– `饱和度（Saturation）`：表示颜色的纯度，范围是0%到100%。
– `亮度/值（Brightness/Value）`：表示颜色的明暗程度，范围是0%到100%。
– `纯绿色`：H = 120°, S = 100%, B/V = 100%
* **HSL（Hue, Saturation, Lightness）模型**： 类似HSV模型，但使用光度来表示颜色的明暗程度
+ 表示方式：色相（Hue） + 饱和度（Saturation） + 光度（Lightness）
– 纯蓝色：H = 240°, S = 100%, L = 50%
* **Lab（CIELAB）模型** : Lab模型是基于人眼感知的颜色模型
+ 表示方式： L (明度) + a (绿色到红色的颜色分量) + b (蓝色到黄色的颜色分量)
* **YUV/YCbCr 模型**： Y表示亮度，U和V（或Cb和Cr）表示色度

> 为什么需要这么多的模型?

不同的颜色模型有着各自的适用的场景，RGB 比较适用于电子显示设备，而 CMYK 主要用于打印和出版， 而 YUV 则可以用于视频压缩和传输。

> 文件格式和颜色模型的关系？

* **不同的文件格式支持的颜色模型是不一样的**：例如 JPEG 通常支持 RGC 模型，但不支持 RGBA.

> 关于颜色模型的深入文章：

* 通过图表直观的感受颜色模型： @ [深入理解color model(颜色模型)](http://cxyroad.com/ ”https://www.jianshu.com/p/f03e9ac9c9ef”)

四. 快速入门

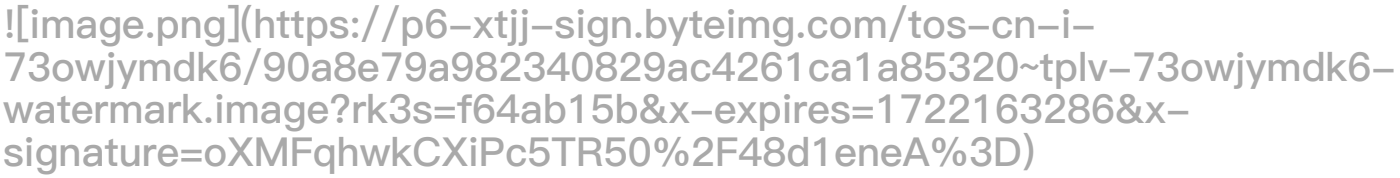
4.1 从单色调开始： 黑白是如何存储的?

在上面我们学习了图像是以像素点的形式存在，又弄明白了颜色是怎么进行融入的，但是始终有一个问题： 数据存储的格式最终是什么样的?

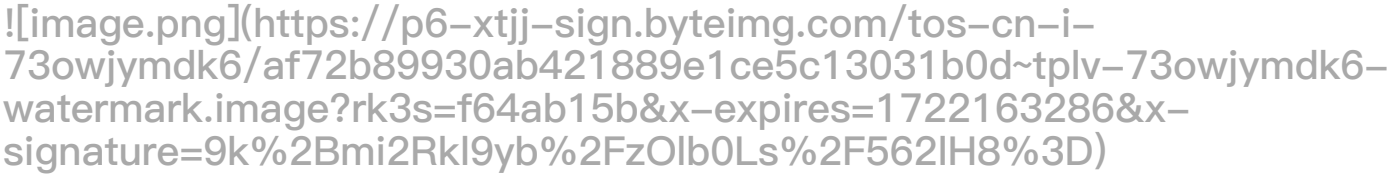
* 通常情况下，采用一个字节来描述灰度图像中一个像素点的像素值。
* 一个字节表示的范围是`[0,255]`

---

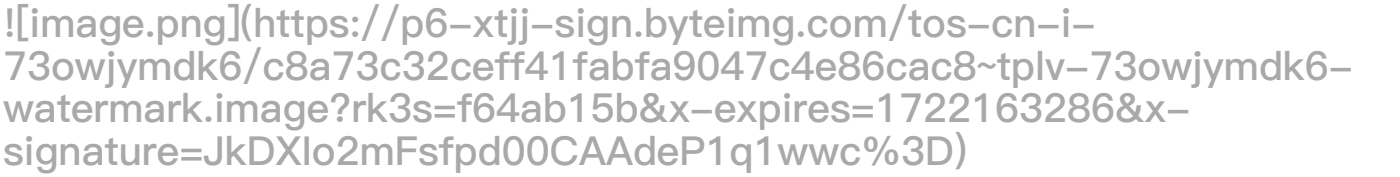
* 首先： 我们假设一张图片 `看起来` 只有黑白二色，当他放大的时候，那么图像应该是这样的



我们注意到，小图放大到像素点级别的时候，总会有边界模糊的地方。在忽略这些地方的情况下，黑色和白色就可以分别通过 0 和 255 进行表示了：



* 然后，可以观察到，在边界处数值是在 0-255 之间变化的：

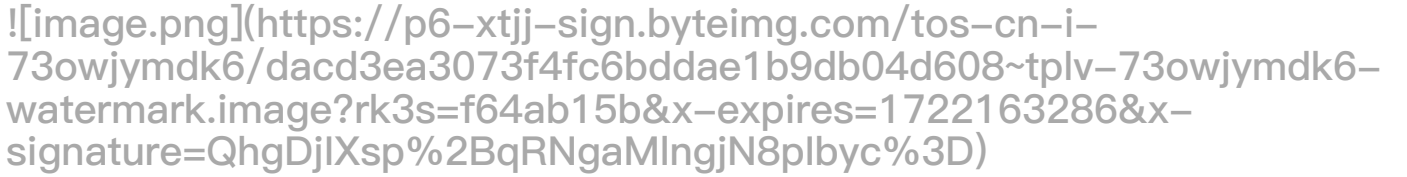


* 总结： `在由黑到白的过程中，我们可以通过一个字节表示其黑白的强度，通过 0-255 的转换，得到一个基础的黑白图像。`

4.2 量化： 灰度表示五彩斑斓的黑

由此我们可以联想到计算机视觉里面的一个常见的操作： `灰度化。`

* `灰度的目的是把有颜色的图像转换成黑白的图像，避免其他的干扰因素。`

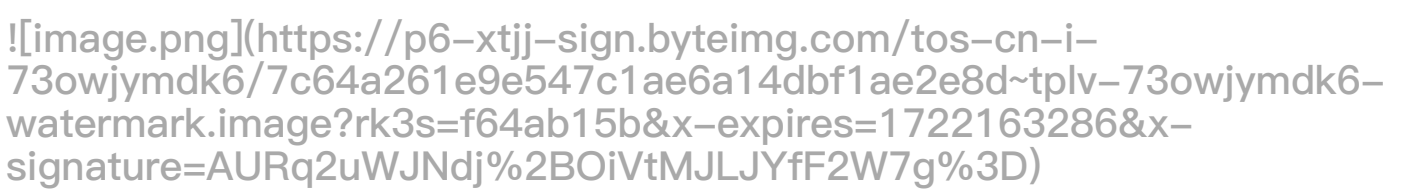


* 总结：`同样的，我们把一个彩色图像按照灰度化的方式，就可以得到一个五彩斑斓的黑白图像了。其中同样通过 0-255 的范围来表示黑白的边界。`

4.3 颜色表示起来也很简单了

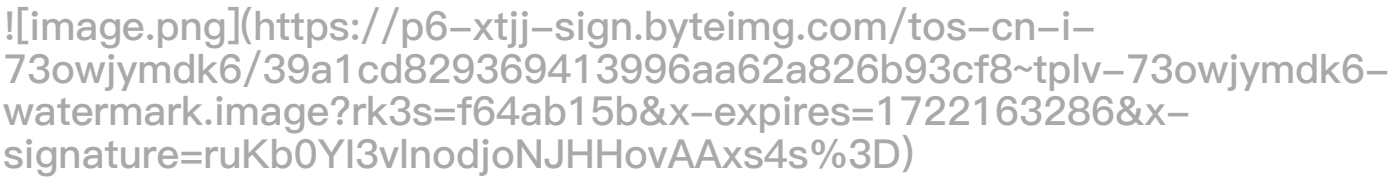
那么我们再进一步，颜色该怎么表示呢？我们以 RGC 颜色图像为例：

* 每个像素点的颜色用`红色、绿色和蓝色`三个通道的值来表示。因此，整个图像可以用`三个二维矩阵`来表示，每个矩阵对应一个颜色通道。



* 如上图： 一个图片的每个像素点，都可以看作三个矩阵的组合：`R(A,B,C)`

* 而每个矩阵中只需要`存储对应的颜色值`的值即可：



五. 扩展用法

5.1 如何对图片进行索引？或者说如何操作图像？

> 查询整个图像中暗色的区域并且修改其色度

```
import cv2
import numpy as np

# 读取图像
image_path = "C:\\Users\\zzg\\Desktop\\test003.png"
image = cv2.imread(image_path)

# 检查图像是否正确读取
if image is None:
    print("Error: Could not read image.")
    exit()

# 将图像转换为灰度图像（可选）
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 定义暗色像素的阈值（这里假设灰度值低于100的像素为暗色像素）
dark_threshold = 100

# 复制图像以便后续处理
brightened_image = np.copy(image)

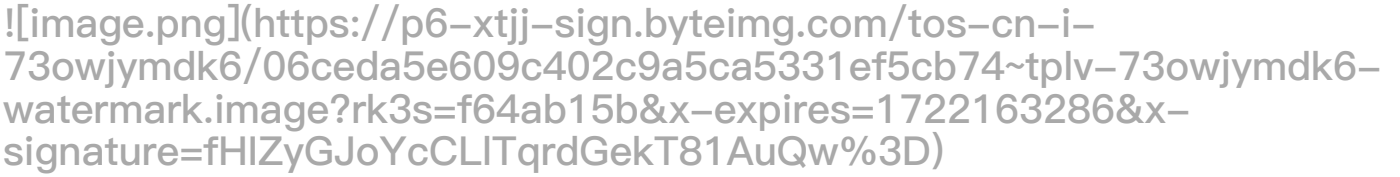
# 提高暗色像素的亮度
indices = np.where(gray_image < dark_threshold)
brightened_image[indices] += 100  # 增加像素的亮度值（可以根据需要调整）

# 显示处理后的图像
```

```
cv2.imshow(”Original Image”, image)
cv2.imshow(”Brightened Image”, brightened_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# 保存处理后的图像（可选）
cv2.imwrite(”brightened_image.jpg”, brightened_image)
```

```



* 可以看到，天空中暗色的区域被我们直接改成了白色，图像中暗色的区域也被替换了
* 只不过由于我们`替换的幅度太大`，导致图像变化的`太明显`
* 我们所知的修图， 美颜 主要是在这个基础上用了`更复杂的算法`

### 5.2 计算机视觉处理图像的原理

还是以 OpenCV 为例，我们可以很轻易的拿到图像的像素点位，然后获取到图像的颜色模型，然后展示它们：

```
import cv2
import numpy as np

# 读取图像
image_path = ”C:\\Users\\test.png”
image = cv2.imread(image_path)

获取图像的高度和宽度
height, width, _ = image.shape
scale_factor = 100 # 放大倍数
new_height = height * scale_factor
new_width = width * scale_factor

# 放大图像
resized_image = cv2.resize(image, (new_width, new_height),
interpolation=cv2.INTER_NEAREST)
```
```

```
# 在放大后的图像上标注RGB值
for i in range(height):
    for j in range(width):
        b, g, r = image[i, j]
        text = f"({r},{g},{b})"
        # 在放大后的像素区域的中心位置绘制RGB值
        x = j * scale_factor + scale_factor // 4
        y = i * scale_factor + scale_factor // 2
        # 使用黑色或白色文本颜色，取决于像素的亮度
        brightness = (int(r) + int(g) + int(b)) / 3
        text_color = (0, 0, 0) if brightness > 127 else (255, 255, 255)
        cv2.putText(resized_image, text, (x, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.3, text_color, 1, cv2.LINE_AA)

# 显示新图像
cv2.imshow("Image with Pixel Values", resized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# 保存新图像
cv2.imwrite("image_with_pixel_values.png", resized_image)


```

* 如果我们需要对图像进行处理 ， 我们也可以对颜色模型进行修改 ：

```
# 指定要修改的像素位置
x, y = 100, 100  # 例如修改位置 (100, 100) 的像素

# 指定新的 RGB 值
new_color = [0, 255, 0]  # 例如修改为绿色 (B=0, G=255, R=0)

# 修改像素点的 RGB 值
image[y, x] = new_color

# 显示修改后的图像
cv2.imshow("Modified Image", image)

# 保存修改后的图像
cv2.imwrite("modified_image.jpg", image)

```

总结

学完了图像的基础概念，后续就是 OpenCV 的常见用法了。

`在这个逻辑上，我们可以想到很多图片的处理方向：`

* 通过图像的处理实现美颜的功能，整体风格的转变
* 基于像素点的模型，来搜索相似或者相同的图片
* 对图像特定的区域添加蒙版或者遮挡，对图片进行裁剪等
* 。。。。。。太多太多了

参考文档

`@ 部分图片使用以下参考数据，感谢以下大佬`

* 《计算机视觉40例从入门到深度学习》
* ChatGPT / Bard
* [图像如何存储在计算机中？](http://cxyroad.com/”https://zhuanlan.zhihu.com/p/367823319”)

原文链接: https://juejin.cn/post/7391065678545584137