图像基础

图像相关名词概述

1 图像相关名词概述

任务 01 位图

任务 02 通道

任务 03 颜色模式

任务 04 布局

学习目标

- 了解图像的基本构成
- 了解图像的通道含义
- 了解图像中不同的颜色模式
- 了解图像的布局方式

1

位图

1 位图

位图图像（bitmap），亦称为点阵图像或栅格图像，是由称作像素（图片元素）的单个点组成的。这些点可以进行不同的排列和染色以构成图样。当放大位图时，可以看见赖以构成整个图像的无数单个方块。

位图的特点是可以表现色彩的变化和颜色的细微过渡，产生逼真的效果，缺点是在保存时需要记录每一个像素的位置和颜色值，占用较大的存储空间。

1 位图

图像中像素点占的bit位数，就是图像的深度。比如：

单色位图：每个像素最多可以表示2种颜色，只需要使用长度为1的二进制位来表示，因此每个像素占1/8 B。

16色位图：每个像素最多可以表示16种颜色，所以只需要长度为4的二进制表示，因此每个像素占1/2 B。

256色位图：每个像素最多可以表示256种颜色，所以只需要长度为8的二进制表示，因此每个像素占1B。

依次轮推，通常将计算机中存储单个像素点所用的 bit 位称作图像的深度。BMP图像大小计算公式：大小=分辨率×位深度/8。

2

通道

2 通道

图像通常分为单通道，三通道，四通道。

单通道：就是通常所说的灰度图，每个像素点只有一个值表示，如果图像的深度为8，那么图像像素值在0(黑)~255(白)之间。

三通道：就是通常所说的彩色图，每个像素点由三个值表示，如果图像深度为8，那么图像像素值由红(0~255)、绿(0~255)、蓝(0~255)叠加表示，色彩更加艳丽。

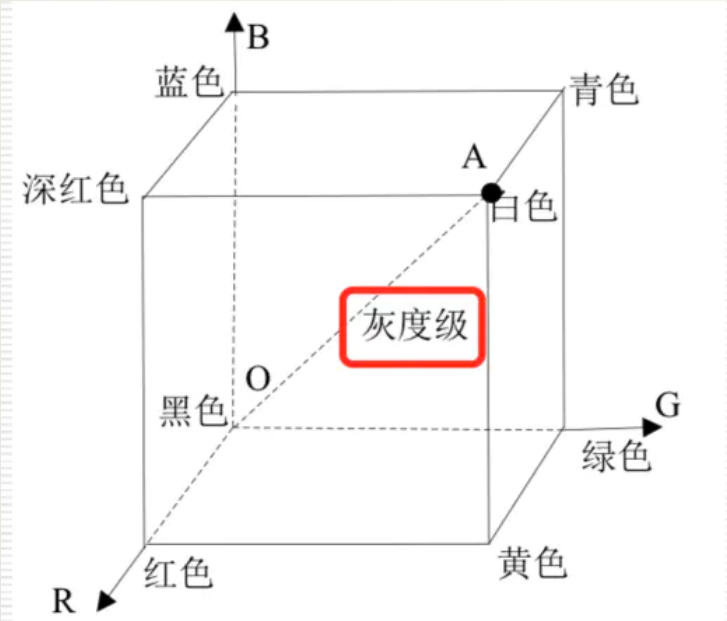
四通道：也就是在三通道图像基础上加上透明程度，0是完全透明，255是完全不透明。

3

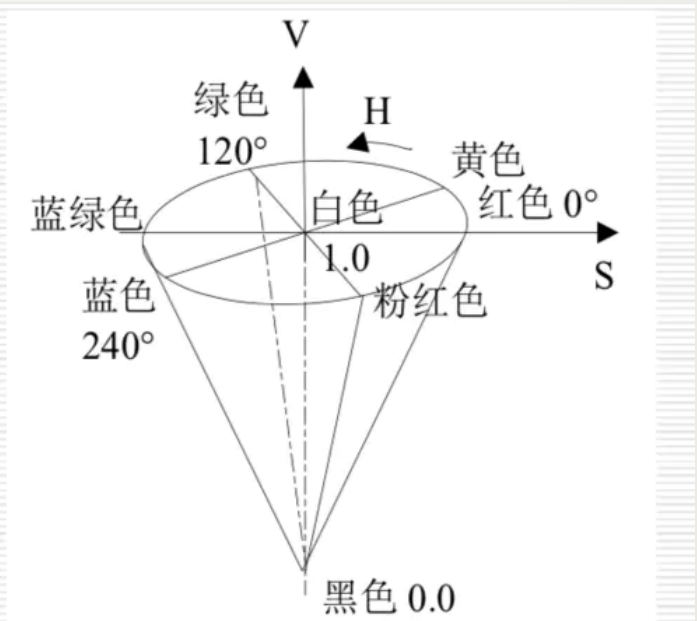
颜色模式

3 颜色模式

通过赋予C的不同维度不同的含义，可以用来描述不同的颜色空间。颜色模式，是将某种颜色表现为数字形式的模型，或者说是一种记录图像颜色的方式。本单元主要讲述两个常用的颜色模式：RGB，HSV。



RGB模型



HSV模型

3 颜色模式

RGB模式是工业界的一种颜色标准，通过对红(R)、绿(G)、蓝(B)三个颜色通道的变化以及通道之间相互的叠加来得到各式各样的颜色。R、G、B分别代表红、绿、蓝三个通道的颜色，这个标准几乎囊括了人类视力所能感知的所有颜色，是目前运用最广的颜色系统之一。

HSV(Hue, Saturation, Value)是由A. R. Smith在1978年创建的一种根据颜色直观特性（色调H，饱和度S，明度V）得到的颜色空间。在 HSV 颜色空间下，相比于 RGB 更容易跟踪某种颜色的物体，常用于分割指定颜色的物体。

4

布局

4 向图片添加文字

图像布局一般分为CHW和HWC两种布局方式。其中：C：channel，图像的通道数；H：height，图像的高度；W：width，图像的宽度。

在实际运用中经常会用到这两种布局的转换，转换方式也非常简单，只需使用数组的transpose((2,0,1))方法即可。

Thank YOU!

图像基础

1 图像基础

任务 01 概述

任务 02 图像的读写与保存

任务 03 图像绘制

任务 04 向图片添加文字

学习目标

- 了解OpenCV
- 掌握图像的读写与保存
- 学会绘制线段，矩阵，圆，椭圆，矩阵，多边形等。
- 向图片添加文字

1

概述

1 概述

在学习图像操作前，我们先了解一下OpenCV。为什么呢？

OpenCV是一个基于BSD许可（开源）发行的跨平台计算机视觉和机器学习软件库，实现了图像处理和计算机视觉方面的很多通用算法。OpenCV用C++语言编写，它具有C ++，Python，Java和MATLAB接口，并支持Windows，Linux，Android和Mac OS。

在以后的学习中，关于图像操作部分（主要是数据预处理环节），我们大部分使用OpenCV实现。

OpenCV的安装命令“pip install opencv-python”，使用时通过“import cv2”导入就行了。

2

图像的读写与保存

2 图像的读写与保存

图像是由众多的像素值构成的，我们如何去操作图像呢？

答案就是将图像转化为数组。

OpenCV提供了这样的方法。

我们使用cv2.imread()方法读取图片，返回数组格式。

2 图像的读写与保存

对于cv2.imread(filename, flags)函数参数如下：

参数filename：图片的路径和文件名。如果图片放在当前文件夹下，直接写文件名就行了，如“lena.jpg”，否则需要给出绝对路径或相对路径，如“img/lena.jpg”。

参数flags：图片的读取方式，省略则为默认值。读取方式有三种，分别为：

cv2.IMREAD_COLOR：彩色图(1)，默认值；

cv2.IMREAD_GRAYSCALE：灰度图(0)；

cv2.IMREAD_UNCHANGED：包含透明通道的彩色图(-1)。

2 图像的读写与保存

在实际应用中，通常对图像进行一系列操作后要显示一下处理后的结果。首先需要新建一个空白窗口用作图像显示，再调用图片显示命令在窗口中显示出图片。cv2.namedWindow(window_name, 默认参数) 的功能就是新建一个显示窗口，可以指定窗口的类型。具体的参数为：

参数window_name是窗口的名字；

默认参数为cv2.WINDOW_AUTOSIZE，表示窗口大小自适应图片，也可以设置为cv2.WINDOW_NORMAL，表示窗口大小可调整。图片比较大的时候，可以考虑用后者。在一般使用的时候，可以跳过此步，直接使用imshow()方法。

2 图像的读写与保存

OpenCV中可以使用cv2.imshow()方法显示图片，同时窗口会自适应图片大小。imshow(window_name, image)方法也有两个参数，第一个参数window_name是窗口的名字，第二个参数image是要显示的图片内容的数组形式。该方法执行后会弹出一个窗口，窗口的名字就是上面定义的window_name。

如果想要设置窗口的显示时间，则需要使用waitKey()方法，参数为设置的毫秒数，0代表永久显示（除非手动关掉窗口）。显示完成后，还需要释放窗口占用的资源，这里使用cv2.destroyAllWindows()方法，该方法会释放所有窗口占用的资源，如果要释放指定窗口的资源，可以使用cv2.destroyWindow(window_name)方法，参数window_name为要释放的窗口的名字。

2 图像的读写与保存

如果想把处理后的图片结果保存到本地，可以使用cv2.imwrite(filename, img [, paras])方法，参数filename是保存的路径，参数img是保存的图片内容，paras表示不同编码格式的参数，一般为nparray 多维数组形式。

3

图形绘制

3 图形绘制

在实际运用中，我们会在图片上添加一些图形，比如目标检测时在物体周围画个矩形框，人脸识别中将人脸的关键点用点（圆形）标出来。

OpenCV常用的形状绘制方法：

线段的绘制是使用cv2.line(img, pt1, pt2, color[, thickness[, lineType[, shift]]])方法，参数img为输入的图像（绘制图像的每个方法的第一个参数都是输入的图像），参数pt1、pt2、color、thickness、lineType（可省略）依次为起点的坐标、终点的坐标、颜色、线条的粗细和线条的类型。参数shift代表坐标精确到小数点后第几位。

3 图形绘制

矩形的绘制是使用cv2.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]])方法。画矩形的方法可分为两种，一种是确定四个顶点的坐标，另一种只确定两个对角顶点的坐标。rectangle()方法基于后者。参数img为输入的图像，参数pt1、pt2为绘画矩形的两个对角顶点坐标，参数color、thickness、lineType（可省略）依次为颜色、线条的粗细和线条的类型。

圆形的绘制使用cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]])方法，参数img为输入的图像，参数center、radius、color、thickness、lineType（可省略）依次为圆形的原点、圆形的半径、颜色、线条的粗细和线条的类型。如果thickness变量的值为负，则代表要画一个实心圆。

3 图形绘制

椭圆的绘制与圆形绘制相似，利用cv2.ellipse(image, centerCoordinates, axesLength, angle, startAngle, endAngle, color [, thickness[, lineType[, shift]]])方法实现椭圆的绘制。其中参数img为输入的图像，centerCoordinates、axesLength、angle、startAngle、endAngle、color、thickness、lineType分别为椭圆的中心坐标、元组（椭圆的长轴长度，短轴长度）、旋转角度、椭圆弧的起始角度、椭圆弧的终止角度、边界线的颜色、线条的粗细和线条的类型。

对于多边形的绘制，需要先创建包含顶点坐标的数组，再调用cv2.polylines(img, pts, isClosed, color[, thickness[, lineType[, shift]]])进行绘制。其中参数img为输入的图像，参数pts、isClosed、color、thickness、lineType分别为多边形上点的数组、标志、多边形颜色、多边形线的粗细、多边形线的类型。标志代表绘制的多边形是否闭合，若为True， 则画若干个闭合多边形， 若为False， 则画一条连接所有点的折线。

4

向图片添加文字

4 向图片添加文字

OpenCV中的cv2.putText(img, text, org, fontFace, fontScale, color, thickness=None, lineType)方法实现了添加文字的功能，参数img、text、org、fontFace、 fontScale、color、thickness、lineType分别为添加文字的图片、添加的文字、左上角坐标、字体、字体大小、颜色、字体粗细和线条类型。

Thank YOU!

图像几何变换

任务01 图像缩放

任务02 图像翻转

任务03 图像平移及旋转

学习目标

- 掌握图像的缩放
- 掌握图像的翻转
- 掌握图像的平移及旋转方法

1

图像缩放

1 图像缩放

图像缩放，顾名思义，就是对图像进行整体放大或缩小的操作。图像缩放在数据预处理时经常会用作规范图像的大小（宽高），从而便于后面神经网络的处理。OpenCV中是利用cv2.resize(src,dsize,dst=None,fx=None,fy=None,interpolation=None)方法来对图像进行缩放操作，该方法可以按照指定的宽度，高度缩放图片，也可以按照比例缩放图片。其中参数src、dsize、dst、fx、fy、interpolation分别为原图片、输出图像尺寸、目标图像、沿水平轴的比例因子、沿垂直轴的比例因子、插值方法。

1 图像缩放

按照比例缩放：如果想让图片的宽和高均放大了一倍，则使用这种方式，需指定第二个参数dsize为None，接着指定fx和fy的值，表示要将宽和高放大或缩小的倍数。interpolation参数代表插值方式，默认为INTER_LINEAR双线性插值方式，通常可以不指定。

2

图像翻转

2 图像翻转

图像翻转，即沿着某条线对图像进行翻转操作。图像翻转在数据集偏少的时候经常用来扩充数据集，从而增加拟合性。OpenCV中使用cv2.flip(src, flipCode, dst=None)方法实现图像的翻转，其中参数src为要翻转的图片，参数flipCode的值为0则垂直翻转(沿x轴)；大于>0则水平翻转(沿y轴)；小于0则代表水平垂直翻转，参数dst为目标图片。

3

图像平移及旋转

3 图像平移及旋转

图像平移，即让图片沿着x轴和y轴方向进行平移操作。首先需要具体了解一下图片中x轴和y轴是怎么确定的。整个坐标系是以图像的左上角为原点，向右为x轴，向下为y轴。

OpenCV中利用cv2.warpAffine(src, M, dsize[, flags[, borderMode[, borderValue]]]) 方法来实现图像平移。首先需要定义平移矩阵M，矩阵M中是一个2行3列的放射变换矩阵，定义了x轴和y轴的平移量。参数src、dsize、flags、borderMode、borderValue分别为输入图像、输出图像的大小、插值法INTE_LINEAR(默认)、填充模式（当borderMode=BORDER_CONSTANT时为值填充）。

3 图像平移及旋转

图像旋转使用的也是OpenCV中的cv2.warpAffine()方法，不同的是还需要使用其中的cv2.getRotationMatrix2D(center, angle, scale)方法来计算旋转矩阵。与图像翻转的区别是图像旋转是在同一水平面进行的，其中center为旋转中心，angle为旋转角度，负数表示顺时针旋转，scale为缩放比例。

Thank YOU!