2 OpenCV图像基础

图像是计算机视觉项目关键组成部分，在许多情况下图像是计算机视觉的输入。本章介绍图像概念，这是计算机视觉项目所需的基本知识。也介绍OpenCV库的一些特性，如坐标系统或BGR顺序(而不是RGB)。
开始编写第一个程序。

本章将讨论以下主题

- 图像基础理论介绍
- 颜色、通道、图像和颜色空间的概念

- OpenCV中的坐标系
- 在OpenCV中访问和操作不同颜色空间中的像素(获取和设置)
- OpenCV中的BGR顺序，不是RGB

图像基本理论介绍

本节主要介绍一些图像基础理论，下一节作较详细解释。简要介绍一下在计算机视觉项目中开发图像处理集时遇到的一些困难，结合图像说明一些简单的公式。

图像处理的主要问题

图像有关概念

图像可看成三维世界的二维(2D)投影。数字图像是二维图像的数字表示，通常是二进制的。一组图像数值，称为像素。像素概念在“像素、颜色、通道、图像和颜色空间的概念”一节中详细解释。计算机视觉的目标是将这些二维数据转换成以下形式

- 新表示(例如，新图像)
- 决策(例如，执行一个具体的任务)
- 新的结果(例如，图像的正确分类)
- 有用信息提取(例如，对象检测)

在处理图像处理技术时，计算机视觉可以处理常见的问题，或者说困难：

- 图像模糊，因为受到透视的影响，透视会使图像的视觉外观发生变化。例如，从不同的角度看同一个物体会产生不同的图像。
- 图像常受到许多因素影响，如光照、天气、反射和运动。
  -图像中的物体可能被其他物体遮挡。遮挡的物体很难检测及分类。根据遮挡的程度，视觉项目的任务将图像分成预定义类别等，可能非常具有挑战性。
  如开发人脸识别系统，问题有：系统应该足够健壮，能应对光照或天气条件的变化，应能解决头部的运动，可以处理摄像头跟用户远近不同的情况，应能够检测用户头部在各旋转轴(偏航，横滚和俯仰)的旋转程度。例如，许多人脸检测算法在头部靠近前额时表现出良好的性能，但脸不是正面的，侧面图中的脸，就无法识别。此外，即使用户戴着眼镜或太阳眼镜，也可能想要识别面部，这会导致眼睛区域出现遮挡。在开发计算机视觉项目时，必须考虑所有这些因素。较好的方法是使用数量更的的测试图像，由综合性问题验证算法。还可以根据测试图像的难度进行分类，以较易检测算法的弱点。

图像处理步骤

图像处理包括以下三个步骤

- 1. 使用图像。这个过程通常涉及一些功能，以便您可以从不同的来源(相机、视频流、磁盘、在线资源)读取图像

- 2. 通过应用图像处理技术处理图像，实现所需的功能，例如，在图像中检测一只猫。
- 3. 显示处理步骤的结果，例如，在图像中绘制边界框，保存到磁盘。

步骤2可以分为三个处理级别

- Low-level process
- Mid-level process
- High-level process

低层处理通常以图像作为输入，输出图像。此步骤的例程包括以下内容

- 噪声消除
- 图像锐化
- 光照归一化
- 透视校正
  结合人脸检测实例，输出图像可以是光照归一化图像，以处理太阳反射引起的变化。

中间层处理采用预处理后的图像来输出图像的某种表示形式。将其视为数字的集合(例如，一个包含100个数字的向量)，它汇总了用于进一步处理的图像的主要信息。在人脸检测的例子中，输出可以是一个矩形，由一个点(x,y)、宽度和高度定义，包含被检测的脸。

高级层获取数字向量，通常称为属性，并输出最终结果。例如，输入是检测到的人脸，输出可如下所示

- 人脸识别
- 表情识别
- 睡意和注意力分散
- 从面部远程测量心率

图像表示

图像可以描述为二维函数$f(x,y)$，其中(x,y)是空间坐标。f在任意点(x,y)的值与图像的亮度或灰度成比例。当f的(x,y)和亮度值都是有限的离散量时，图像称为数字图像。f(x,y)取以下值

- $x \in [0, h-1]$，h是图像高度
- $y \in [0, w-1]$，w是图像宽度
- $f(x,y) \in [0, L-1]$ L=256(对8位图像)

彩色图像表示方式可相同，但需要定义三个函数分别表示红色、绿色和蓝色值。这三个单独函数都与灰度图像定义的$f(x,y)$函数相同。这三个函数分别索引R、G、B，各表示为
$fR(x,y)$、$fG(x,y)$和$fB(x,y)$。

黑白图像用相同方法表示图像，但只需一个函数，关键是f(x,y)只能取两个值。通常这些值是0(黑色)和255(白色)。

> 这三种类型的图像常用于计算机视觉，记住其格式

下面的截图显示了三种类型图像，彩色图像、灰度图像和黑白图像

> 数字图像可看作真实场景的近似值，因为f(x,y)值是有限的离散量。无论是灰度图像还是黑白图像，每点只有一个样本(只需要一个函数)，彩色图像每点有三个样本(需要三个函数分别对应图像的红、绿、蓝三个分量)。

2.6 Concepts of pixels, colors, channels, images, and color spaces

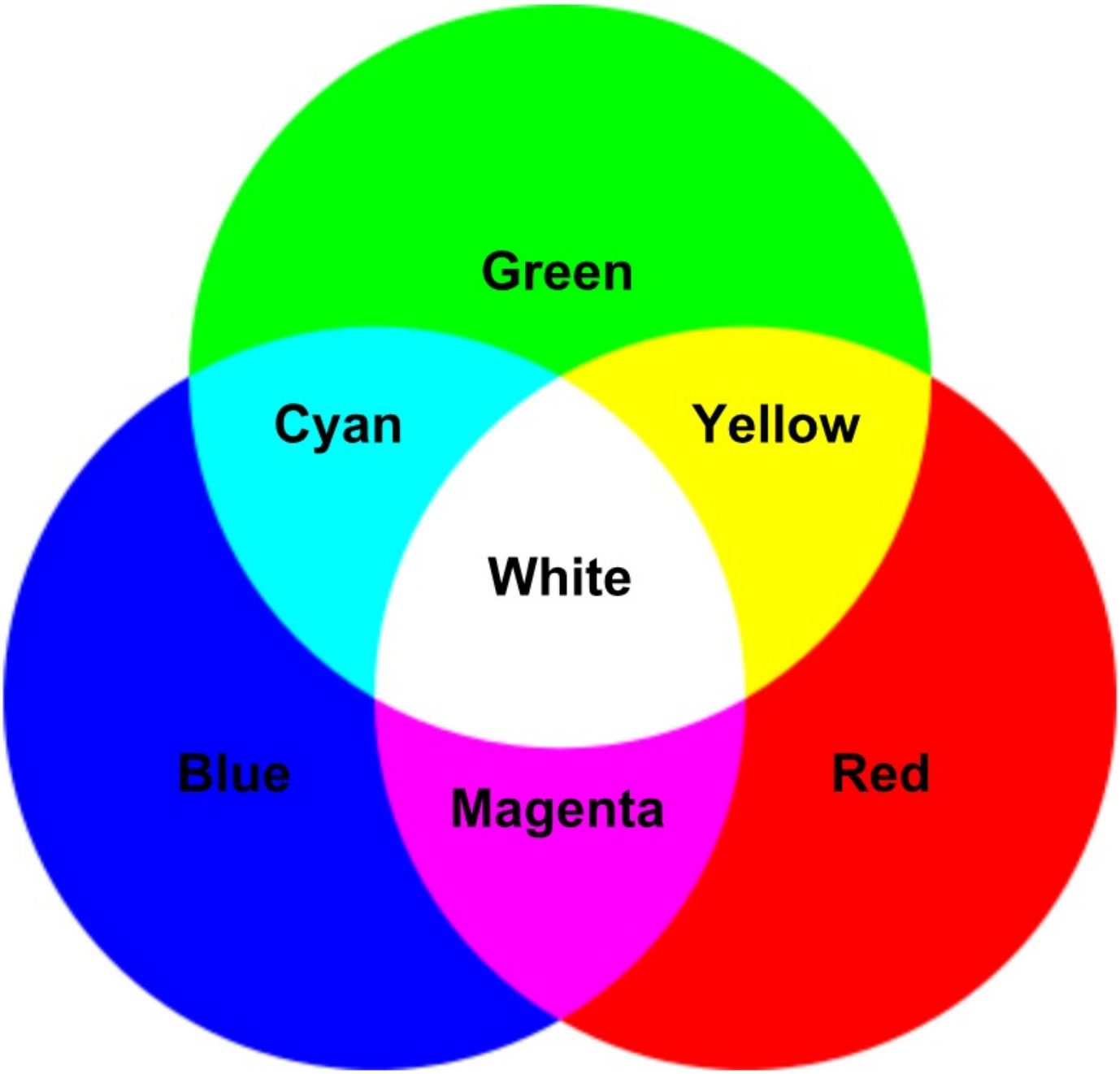
像素颜色通道图像和颜色空间的概念

颜色模型有多种，最常见的是红、绿、蓝(RGB)模型。RGB模型可用于解释数字图像有关的一些关键概念。

> 第5章图像处理技术，将充分解释主要颜色模型，也称颜色空间

RGB模型是一种加色模型，原色(R、G、B)混合在一起，产生较大范围的颜色。在RGB模型中，原色是红色、绿色和蓝色.

每个原色(R, G, B)常称为一个通道，表示为[0,255]之间的整数值。每个通道有256个离散值，对应于颜色通道值$(2^8 = 256)$的总比特数。由于有三个不同的通道，每个通道8位，称为24位颜色深度。



上图中，可看到RGB颜色空间的附加颜色属性

- 绿色加红色等于黄色
- 蓝色中加入红色会产生洋红色

- 绿色加到蓝色会产生青色
- 三种原色加在一起就成了白色

如前所述，在RGB颜色模型中，颜色由红色、绿色和蓝色值表示，像素值表示为RGB三元组(r，g，b)。一般图形软件中典型的RGB颜色选择器如下图所示。每个滑块的范围是从0到255

可看到纯红色加到纯蓝色产生完美的洋红色。

https://www.rapidtables.com/web/color/RGB_Color.html 可查看RGB颜色图表。

分辨率为800×1200的图像为800列和1200行的网格，800×1200 = 96万像素。一幅图像中有多少像素并不表示它的物理尺寸，一个像素并

不等于一毫米，像素大小即图像有多大取决于图像的每英寸像素(PPI)，一般情况下PPI在[200-400]范围内。

> 计算PPI的基本公式如下：
> PPI =宽度(像素)/图像宽度(英寸)
> PPI =高度(像素)/图像高度(英寸)
> 例如，如果想打印一幅ParseError: KaTeX parse error: Undefined control sequence: \tims at position 2: 4\tims 6英寸的图像，图像是800×1200，那么PPI将是200。

We will now look into file extensions.
现在我们来看看文件扩展名

文件扩展

在OpenCV中操作的图像可以看作是RGB三色的矩形阵列(对于RGB图像)，但不一定是以这种格式创建、存储或传输的。因此，如GIF、PNG、bitmaps或JPEG等文件格式，使用不同形式的压缩(无损或有损)更有效地表示图像。

为完整起见，简要介绍一下这些图像文件格式，重点介绍OpenCV支持的文件格式。OpenCV支持以下文件格式(相关文件扩展名)

- Windows bitmaps: *.bmp and *.dib JPEG files: *.jpeg, *.jpg, and *.jpe JPEG 2000 files: *.jp2
- Portable Network Graphics: *.png
- Portable image format: *.pbm, *.pgm, and *.ppm

- TIFF files: *.tiff and *.tif

位图图像文件(BMP)或设备无关的位图(DIB)文件格式是用于存储位图数字图像的光栅图像文件格式。BMP文件格式可以处理各种颜色深度的2D数字图像，也可以处理数据压缩、alpha通道或颜色配置文件。

JPEG（Joint Photographic Experts Group）是一种光栅图像文件格式，用于存储已压缩的图像，在小文件中存储大量信息。

JPEG2000是一个图像压缩标准和编码系统，使用基于小波的压缩技术，提供高水平的可扩展性和可访问性。通过这种方式，jpeg2000压缩的图像比常规的JPEG包含更少的人工信息。

PNG（Portable Network Graphics）是一种压缩的光栅图形文件格式，于1994年引入，作为图形交换格式(GIF)的改进替代。

PPM（Portable Pixmap）、PBM（Portable Bitmap）和
PGM（Portable Graymap）指定了交换图形文件的规则。一些应用程序将这些文件格式统称为可移植的PNM格式(Portable nymap)。这些文件是方便、简单的保存图像数据的方法。此外，它们很容易阅读。在这个意义上，PPM、PBM和PGM格式都被设计得尽可能简单。

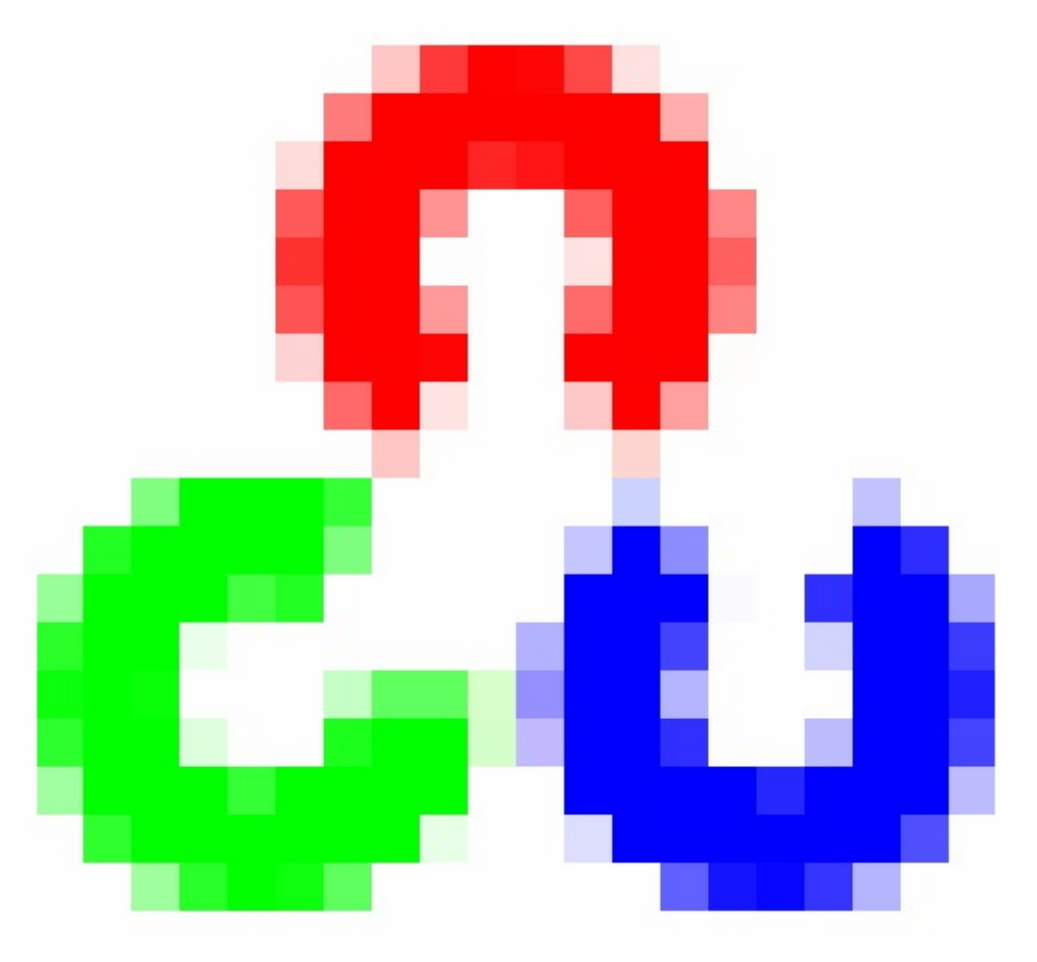
TIFF(Tagged Image File)是单个文件中处理图像和数据的可调的文件格式。

无损和有损类型的压缩算法应用于图像，导致图像比未压缩的图像更小。一方面，在无损压缩算法中，生成的图像与原始图像是等价的，这意味着在反转压缩过程之后，生成的图像与原始图像是等价的。另一方面，在有损压缩算法中，生成的图像并不等同于原始图像，这意

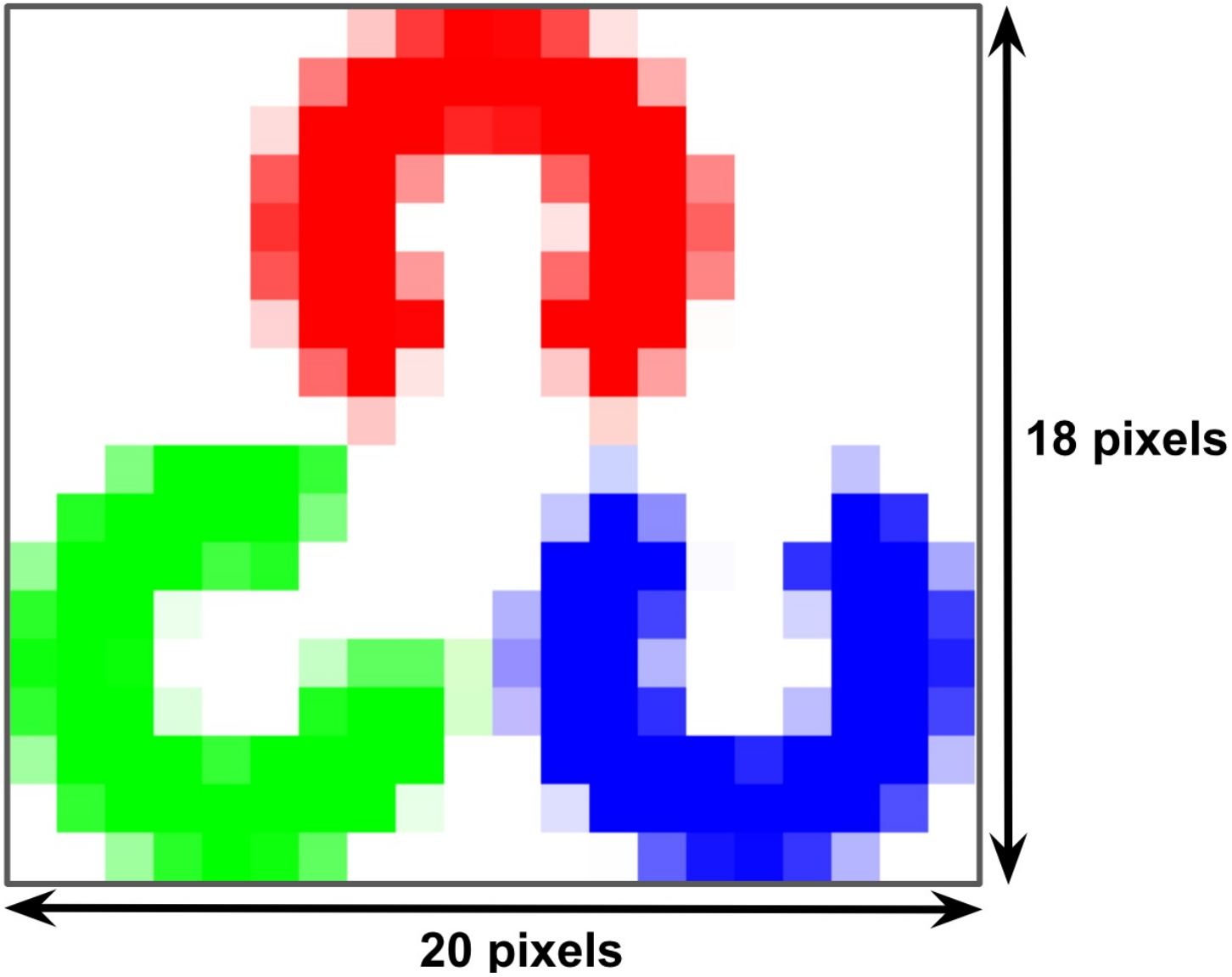
味着图像中的一些细节丢失了。从这个意义上说，在许多有损压缩算法中，压缩级别是可以调整的。

OpenCV中的坐标系

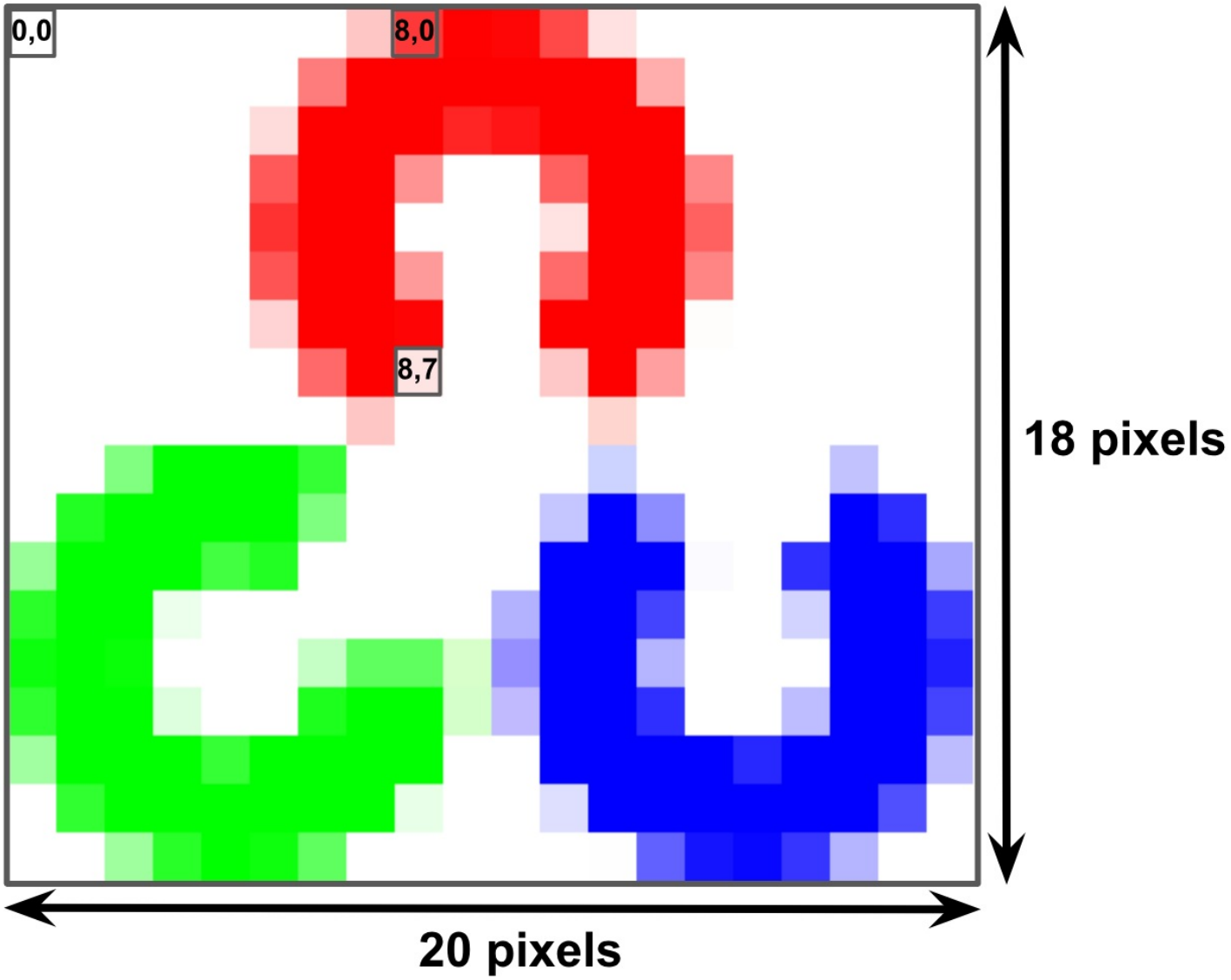
为演示OpenCV中的坐标系统以及如何访问单个像素，展示OpenCV徽标低分辨率图像



logo的尺寸是20×18像素，即图片尺寸是360像素。可在每个轴上添加像素计数，如下图所示：



看一下(x,y)格式像素的索引。像素zero-indexed，左上角是(0,0)，而不是(1,1)。下面的图像，索引了三个单独的像素。图像的左上角是原点的坐标。且y坐标越往下越大：



可从图像中提取单个像素的信息，方法与Python中引用数组的单个元素相同。在下一节中介绍如何做到这一点。

在OpenCV中访问和操作像素

本节介绍使用OpenCV访问和读取像素值，以及如何修改它们，如何访问图像属性。如果一次处理多个像素，需创建图像区域(ROI)，也介绍如何做到这一点。如何分割和合并图像。

> 在Python中，图像是用NumPy数组表示的。因此，这些示例中多数操作都与NumPy相关，要理解这些示例中的代码并用OpenCV编写优化代码，需要对NumPy有很好的理解。

使用BGR图像访问和操作OpenCV中的像素

如何在OpenCV中使用BGR图像。OpenCV加载彩色图像，蓝色通道是第一个，绿色通道是第二个，红色通道是第三个。请参阅OpenCV灰度图像访问和操作像素一节，以充分理解这个概念。

首先，使用cv2.imread()函数读取图像。图像应在工作目录中，或图像完整路径。本例读取logo.png图像并将其存储在img变量中

```
# The function cv2.imread() is used to read an image from the tl
# Alternatively, you should provide a full path of the image:
# Load OpenCV logo image (in this case from the working directo
img = cv2.imread('logo.png')
```

在img中加载图像后，可访问图像属性。从加载图像中提取的第一个属性是shape，行、列和通道的数量(如果图像是彩色的)。在dimensions变量中存储这些信息，以备使用

```
# To get the dimensions of the image use img.shape
# img.shape returns a tuple of number of rows, columns and chan
# If image is grayscale, img.shape returns a tuple of number of
# So,it can be used to check if loaded image is grayscale or co
# Get the shape of the image:
dimensions = img.shape
```

另一个属性是图像的大小(img.size等于通道数×高度×宽度)

```
# Total number of elements is obtained by img.size:
total_number_of_elements=  img.size
```

属性图像数据类型由img.dtype获得。在本例中，图像数据类型为uint8(unsigned char)，取值范围[0 - 255]

```
# Image datatype is obtained by img.dtype.图像数据类型由img.dtype
# img.dtype is very important because a large number of errors
# Get the image datatype:获取图像数据类型
image_dtype = img.dtype
```

To display an image, we will use the cv2.imshow() function to show an image in a window. The window automatically fits to the image size. The first argument to this function is the window name and the second one is the image to be displayed. In this
case, since the loaded image has been stored in the img variable, we will use this variable as the second argument:

为显示图像，用cv2.imshow()函数在窗口中显示图像。窗口自动适应图像大小。此函数的第一个参数是窗口名，第二个参数是要显示的图像。在本例中，由于加载的图像已经存储在img变量中，所以我们将使用这个变量作为第二个参数

```
# The function cv2.imshow() is used to display an image in a wi
# The first argument of this function is the window name
# The second argument of this function is the image to be shown
# Each created window should have different window names.
# Show original image
cv2.imshow("original  image",  img)
```

显示图像之后，键盘绑定函数cv2.waitKey()为键盘事件等待指定的时间，毫秒，参数是时间(以毫秒为单位)。如果此时按下任何键，程序将继续。如果毫秒数为0 (cv2.waitKey(0))，它将无限期等待击键。因此，函数将允许我们看到显示的窗口，等待击键

```
# The function cv2.waitKey(), which is a keyboard binding funct
#  This  function  waits  the  value  indicated  by  the  argum
# If any keyboard event is produced in this period of time, the
# If the value of the argument is 0, the program waits indefini
cv2.waitKey(0)
```

访问(读取)像素值，需要将像素的行和列提供给img变量，该变量包含已加载的图像。例如要获得像素(x=40, y=6)的值，代码如下

```
# A pixel value can be accessed by row and column coordinates.
# In case  of BGR  image, it returns an array of (Blue, Green, R
# Get the value of the pixel (x=40, y=6):
(b, g, r) = img[6, 40]
```

三个变量(b, g, r)加载三个像素值。可看到OpenCV对彩色图像使用BGR格式。可以一次访问一个通道。在本例中，用通道的行、列和索引进行索引。例如，只获取像素的蓝色值(x=40, y=6)，代码如下

```
# We can only   access one channel at a time.
# In this case, we will use row, column and the index of the de
# Get only blue value of the pixel (x=40, y=6):
b = img[6, 40, 0]
```

用同样的方法可修改像素值。记住(b, g, r)格式。例如要将像素(x=40, y=6)设置为红色，操作如下

```
# The pixel values can be also modified in the same way - (b, g,
img[6, 40] = (0, 0, 255)
```

不是处理单个像素，而是处理特定区域的时候，应获取一定范围的值，而不是单个值。例如，图像左上角，代码如下

```
# In this case, we get the top left corner of the image:

top_left_corner = img[0:50, 0:50]
```

左上角变量是另一个图像(比img小)，也可用同样方法处理

在OpenCV中使用灰度图像访问和操作像素

灰度图像只有一个通道。在处理这些图像时引入了一些差异。我们将在这里强调这些差异

同样用cv2.imread()函数来读取图像。在这种情况下，第二个参数是必需的，因为要加载灰度图像。第二个参数是标志指定图像读取方式。加载灰度图像所需的值是cv2.IMREAD_GRAYSCALE

```
# The function cv2.imshow() is used to display an image in a wi
# The first argument of this function is the window name
# The second argument of this function is the image to be shown
# In this case, the second argument is needed because we want t
# Second argument is a flag specifying the way the image should
#  load  OpenCV  logo  image:
gray_img  =  cv2.imread('logo.png',  cv2.IMREAD_GRAYSCALE)
```

在本例中，图像存储在gray_img变量中。用gray_img.shape获得图像的尺寸，行和列是两个值。灰度图像中，不提供通道信息

```
#  To  get  the  dimensions  of  the  image  use  img.shape
# If color image, img.shape returns returns a tuple of number o
# If grayscale, returns a tuple of number of rows and columns.
# So, it can be used to check if the loaded image is grayscale
# Get the shape of the image (in this case only two components!

 dimensions = gray_img.shape
```

img.shape 将返回一个元组中图像的维数，就像(99,82)。

像素值可通过行和列坐标访问。在灰度图像中，只获得一个值,通常称为像素强度。例如，要获取(x=40, y=6)的像素强度，代码如下

```
# You can access a pixel value by row and column coordinates.
# For BGR  image, it returns an array of (Blue, Green, Red)  va
# Get the value of the pixel (x=40, y=6):
i = gray_img[6, 40]
```

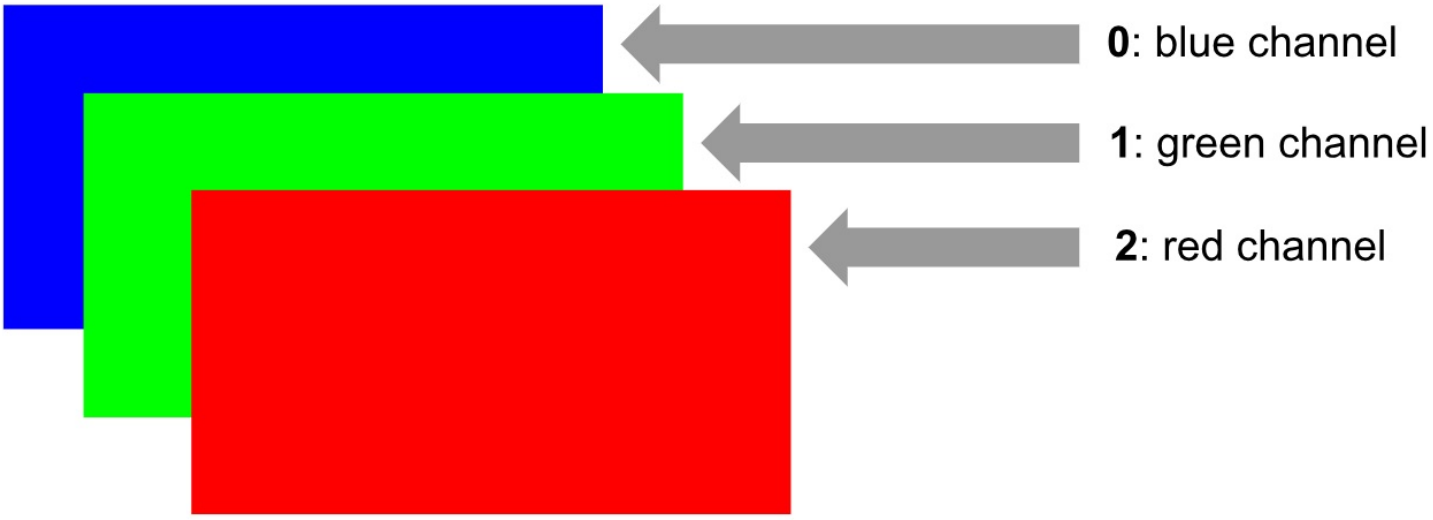
图像的像素值也可以用同样的方法修改。例如要将像素值(x=40, y=6)更改为黑色(强度等于0)，代码如下

```
# You can modify the pixel values of the image in the same way.
#  Set  the  pixel  to  black:设置像素为黑色
 gray_img[6, 40] = 0
```

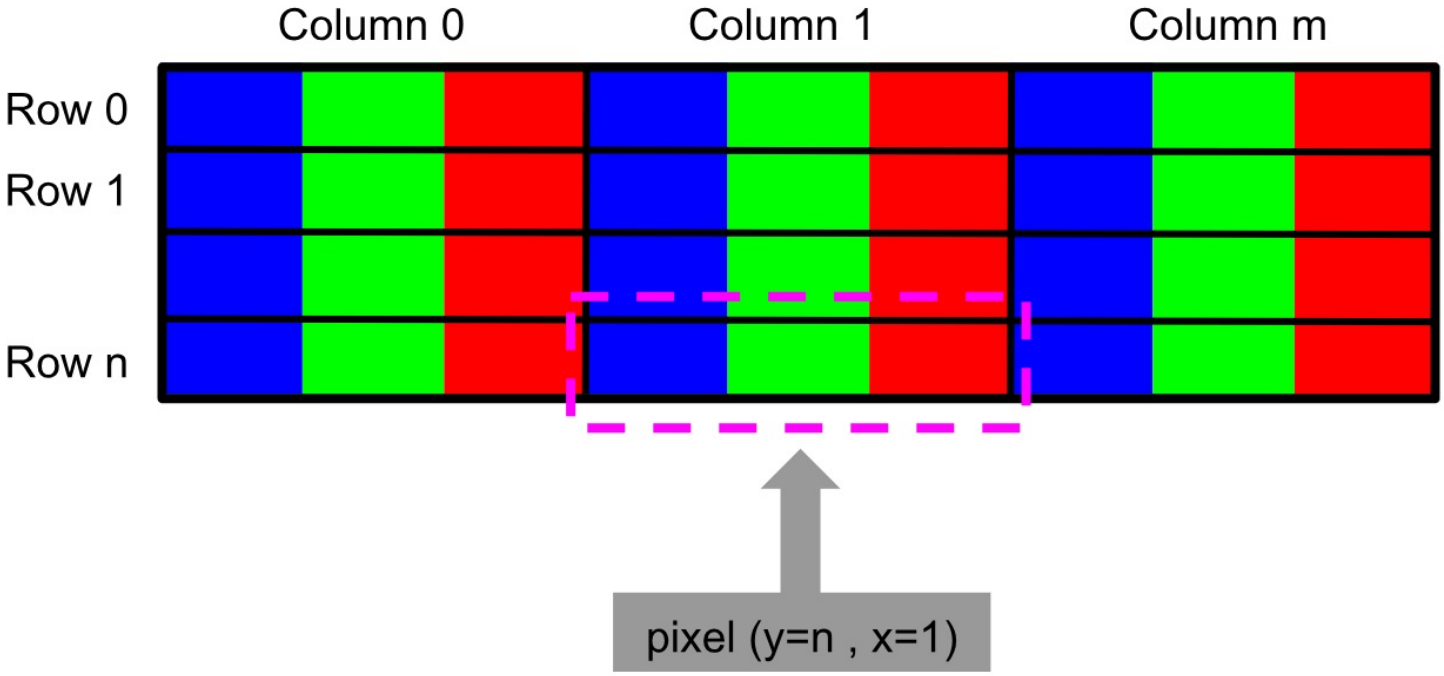
2.12 BGR order in OpenCV

OpenCV中的BGR顺序

我们已经提到OpenCV使用BGR颜色格式而不是RGB格式。下图列出了三个通道的顺序



BGR图像的像素结构如下图所示。特别地，为了澄清起见，详细介绍如何访问像素(y=n，x=1)



OpenCV的最初开发人员选择了BGR颜色格式而不是RGB格式，因为当时BGR颜色格式在软件供应商和相机制造商中非常流行。例如，在Windows中，当使用COLORREF指定颜色值时，为GR格式0x00bbggrr (https://docs.microsoft.com/es-es/windows/desktop /gdi/COLORREF)。选择BGR出于历史原因。

另外，其他Python包使用RGB颜色格式。需要知道如何将图像从一种格式转换成另一种格式。例如，Matplotlib使用RGB颜色格式。Matplotlib (https://matplotlib.org/)是最流行的2D Python绘图库，提供了各种绘图函数。可以与绘制图像交互，例如，放大图像、保存图像。Matplotlib既可以在Python脚本中使用，也可以在Jupyter笔记本中使用。Matplotlib文档详见(https://matplotlib.org/contents.html)。

因此对python项目,显示图像用Matplotlib包，比OpenCV提供功能更好。看看如何处理这两个库中的颜色格式。首先用cv2.imread()函数加载图像

```
# Load image using cv2.imread:
img_OpenCV = cv2.imread('logo.png')
```

cv2.imread()函数按BGR顺序加载图像，图像存储在img_OpenCV变量中。用cv2.split()函数将加载的图像分割为三个通道(b、g、r)。函数的参数是我们要分割的图像

```
# Split the loaded image into its three channels (b, g, r):
b, g, r = cv2.split(img_OpenCV)
```

下一步是再次合并通道，以便根据通道提供的信息构建顺序不同的新图像。遵循Matplotlib的RGB格式，更改b和r通道的顺序

```
# Merge again the three channels but in the RGB format:
img_matplotlib = cv2.merge([r, g, b])
```

此时有img_OpenCV和img_matplotlib两个图像，分别用OpenCV和Matplotlib绘制。首先用Matplotlib显示这两个图像。

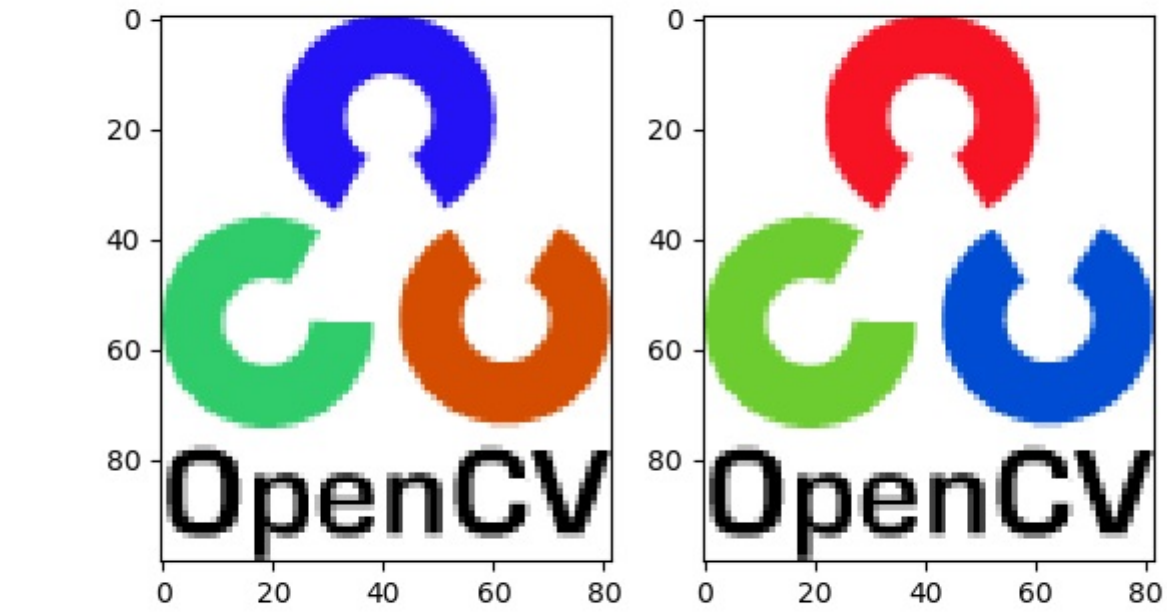
用subplot在Matplotlib同一个窗口中显示两个图像，subplot将多个图像放置在同一个窗口中。subplot有三个参数，如subplot(m,n,p)。在本例中，subplotm x n网格绘图中，m确定行数，n确定列数，p确定网格绘图位置。Matplotlib显示图像用imshow。

在本例中，水平显示两幅图像，m = 1和n = 2。对第一个子图使用p=1（img_OpenCV），对第二个子图使用p = 2（img_matplotlib）

```
# Show both images (img_OpenCV and img_matplotlib) using matplo
# This will show the image in wrong color:
plt.subplot(1 2 1)
plt.imshow(img_OpenCV)
# This will show the image in true color:
plt.subplot(1 2 2) plt.imshow(img_matplotlib)
plt.show()
```

Therefore, the output you will get should be very similar to the output that's shown in the following diagram:

输出应与下图所示的输出非常相似



可以看到，第一个子图以错误的颜色(BGR顺序)显示图像，而第二个子图以真实的颜色(RGB顺序)显示图像。用cv2.imshow()以同样的方式显示这两个图像

```
# Show both images (img_OpenCV and img_matplotlib) using cv2.ims
# This will show the image in true color:
cv2.imshow('bgr image', img_OpenCV)
# This will show the image in wrong color:
cv2.imshow('rgb image', img_matplotlib)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

下面的屏幕截图显示了执行代码的结果

正如所料，截图显示的图像是真实的颜色，而第二个图显示的图像是错误的颜色。

另外，如果想在同一个窗口中显示两个图像，我们可以构建一个包含两个图像的完整图像，将它们水平连接起来。为此，用NumPy的concatenate()方法。该方法的参数两个图像连接和轴。在本例中，axis = 1(水平堆积)

```
# To stack horizontally (img_OpenCV to the left of img_matplotli
img_concats = np.concatenate((img_OpenCV, img_matplotlib), axis=
# Now, we show the concatenated image:
cv2.imshow('bgr image and rgb image', img_concats)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

下面屏幕截图为连接的图像:

需要考虑的一个因素是cv2.split()是一个耗时的操作。可根据需要考虑使用NumPy索引。例如，想获得图像通道，可使NumPy索引，而不是用cv2.split()。请参阅下面的示例，用NumPy索引获取通道

```
# Using numpy capabilities to get the channels and to build the
# Get the three channels (instead of using cv2.split):

B = img_OpenCV[:, :, 0]
G = img_OpenCV[:, :, 1]
R = img_OpenCV[:, :, 2]
```

另一个需要考虑的问题是，可用NumPy在一条指令中将映像从BGR转换为RGB

```
# Transform the image BGR to RGB using Numpy capabilities:
img_matplotlib = img_OpenCV[:, :, ::-1]
```

2.13 Summary

本章介绍了与图像相关的关键概念。图像包含计算机视觉项目所需的丰富信息。OpenCV使用BGR颜色格式而不是RGB，Python包如Matplotlib使用RGB格式。讨论了如何将图像从一种颜色格式转换成另一种颜色格式。

还总结了处理图像的主要功能和选项

cv2.imread(), cv2.split(), cv2.merge(), cv2.imshow(), cv2.waitKey(), and cv2.destroyAllWindows()
How to get and set image pixels in both BGR and grayscale images

- 访问图像属性
- 一些OpenCV功能,如cv2.imread(), cv2.split(), cv2.merge(), cv2.imshow(), cv2.waitKey(), and cv2.destroyAllWindows()

- 如何获取和设置BGR、灰度图像中的像素

最后，介绍了两个notebooks，操作这些概念。notebooks可通过按Shift + Enter逐步运行它，或通过单击单元| run All菜单全部运行。

下一章学习如何处理文件和图像，这是编写计算机视觉应用程序所必需的。

问题

1. 图像处理的主要步骤是什么?
2. 图像三个处理级别是什么?
3. 灰度图像与黑白图像的区别是什么?
4. 像素是什么?
5. 图像分辨率是什么?
6. 使用什么OpenCV函数来执行以下操作?

- 加载(读取)一个图像
- 显示图像
- 等待按键
- 拆分通道
- 合并通道

7. 用什么指令来运行Jupyter Notebook?
8. 由下列三原色得到什么颜色?

- B = 0, G = 255, R = 255
- B = 255, G = 255, R = 0
- B = 255, G = 0, R = 255
- B = 255, G = 255, R = 255

9. 假设已经在img中加载图像。如何检查img是彩色的还是灰度的