

# OPENCV4

---

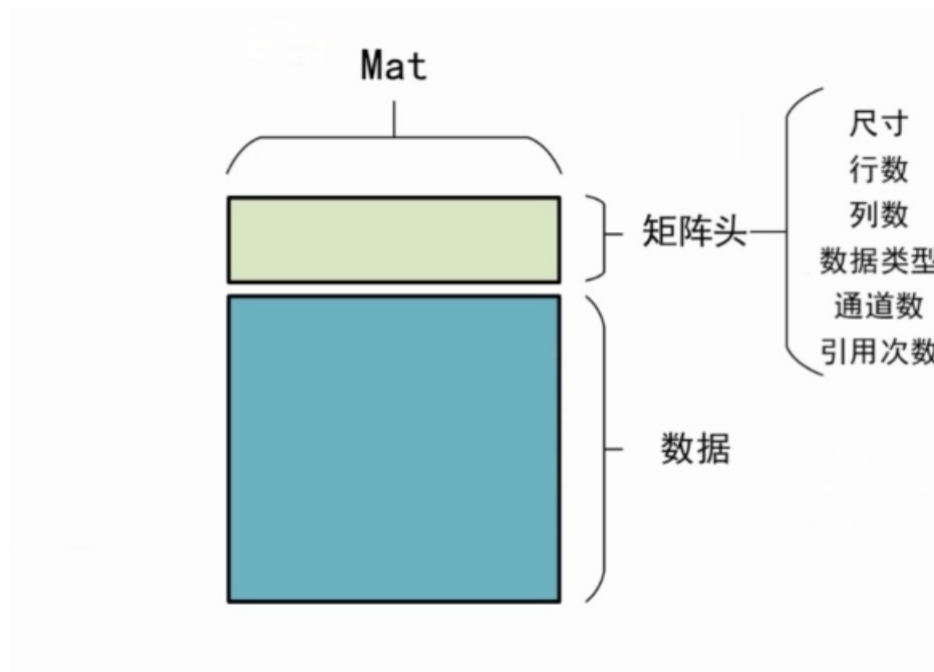
## 1. 认识Mat

---

### 1.1 Mat类的基本概念

Mat类是类似于int, double的一种数据类型，以矩阵的形式存储数据，可以有很多维，不一定是二维。

Mat类的结构如下：

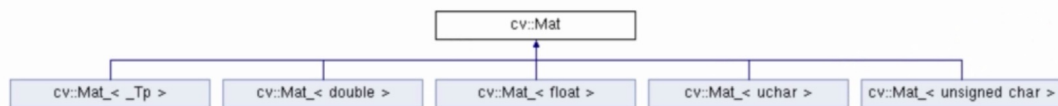


Mat类存储的数据有好几种类型，但是因为不同平台对于int, float规定可能有变，所以做出如下对于数据类型的统一规定：

- OpenCV 中规定的数据类型

数据类型	具体类型	取值范围
CV_8U	8位无符号整数	0—255
CV_8S	8位符号整数	-128—127
CV_16U	16位无符号整数	0-65535
CV_16S	16位符号整数	-32768—32767
CV_32S	32位符号整数	-2147483648—2147483647
CV_32F	32位浮点整数	-FLT_MAX—FLT_MAX, INF, NAN
CV_64F	64位浮点整数	-DBL_MAX—DBL_MAX, INF, NAN

## Mat能存储的数据



## 1.2 Mat类的创建与赋值

- 利用矩阵宽，高和类型参数创建Mat对象

```
cv::Mat::Mat (int rows, int cols, int type)
```

其中，前两个参数代表行数和列数，第三个代表数据类型，如：**CV\_8UC1**，重要的是，C1代表这是一个单通道矩阵，我们可以在上面一小节中的数据类型后加**C(n)**表示这个对象有几个通道。

- 利用Size结构创建

```
cv::Mat::Mat (Size size, int type)
```

```
//如: Mat b(Size(3,3), CV_8UC1) ;
```

- 利用已有的Mat对象创建

```
//假如我们已经有了一个 a 矩阵
```

```
//接着创建c
```

```
Mat c(a, Range(2,5), Range(2,5));
```

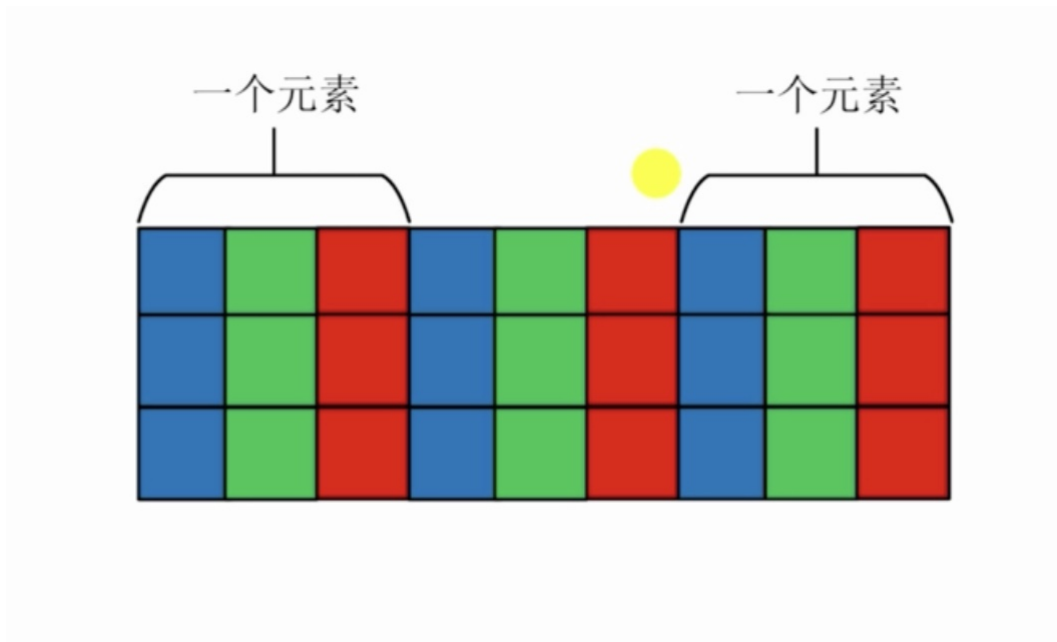
```
//注意2-5不包括2, 5 且用这种方法创建数据时a中必须含有数据
```

- 创建时赋值：直接在type参数后面加Scalar参数，有几个通道就给几个数。
- 利用类方法赋值：用类方法创建特殊矩阵，如对角矩阵
- 用枚举法创建：上面两种都不能对每一个像素点分别赋值，所以我们可以使用枚举法：

```
cv::Mat a = (cv::Mat_<int>(3,3)<<1, 2, 3, 4, 5, 6, 7, 8, 9);  
//对一个3*3的矩阵分别赋值，注意尖括号里面填的是double等数据类型，不是CV_8U
```

## 1.3 Mat 读取

- Mat类在数据中的存储形式如下：



- Mat类对象的常用属性（可以用对象直接调用）

属性	作用
cols	矩阵的列数
rows	矩阵的行数
step	以字节为单位的矩阵的有效宽度
elemSize()	每个元素的字节数
total()	矩阵中元素的个数
channels()	矩阵的通道数

- Mat元素的读取：

使用`at`方法读取Mat矩阵元素， `at (int row, int col)`

```
//单通道
int value = (int)a.at<uchar>(0,0)

//多通道，因为多通道元素的每个元素有多个值，所以不能用int存， 所幸OPENCV已经帮我们定义了一些类型
cv::Vec3b vc3 = b.at<cv::Vec3b>(0,0);
//Vec3b代表一个元素有三个通道 每个值是uchar类型
int first = (int)vc3.val[0]; //读取第一个通道的值
```

也可以使用统一转化的方法，这样就不需要知道对象中元素的类型。

补充：

---

## 1.4 Mat的运算

- 四则运算
  - 符号运算

这种方法我们可以直接使用 `+` `-` `*` `/` 进行运算。做运算时两个Mat内的数据类型要一致。
  - 两个矩阵相乘

可以普通矩阵乘法，满足一般规律。

可以点积，要求两个矩阵中元素个数相同，维度则不做要求。

也可以对应位元素进行乘积，矩阵维度必须一样。
  - 也可以利用OPENCV自带函数。

---

## 2. 图像的加载与保存

### 2.1 图像读取

- `imread`
  - 第一个参数为**filename** 写文件名称及其路径
  - 第二个参数为**flags**,为读取图像形式的方法，默认为**彩色读取**

### 2.2 图像显示

- `namedWindow`
  - 第一个参数为winname，可以设置窗口的名称。
  - 第二个参数为flags，设置窗口属性。
- `imshow`

- 第一个参数为winname, 即上面函数的第一个参数
- 第二个参数为mat, 即要显示的图像对象

## 2.3 图像保存

- **imwrite**

- 第一个参数为filename, 为保存图像的地址和格式。
  - 第二个参数为img, 为要保存的图像对象。
  - 第三个参数为params, 对保存图像的属性处理, 如要不要压缩。
- 

## 3. 视频加载与摄像头的使用

---

### 3.1 视频加载与摄像头调用

- **VideoCapture**

- 第一个参数为filename, 为读取视频的名称。若调用摄像头为int类的参数, 为摄像头在电脑内的id。
- 第二个参数为apiPreference, 为读取数据时的要设置的属性。一般不写。

**P.S.** VideoCapture是一个类, 他允许用户先定义一个空的对象, 后续用open()函数再设置参数。可以用isOpened()来判断是否读取成功。也可以用get()函数来获取属性, 如下表:

### 3.2 视频保存

- **VideoWriter**

- 第一个参数为filename, 为保存视频的地址和文件名。
  - 第二个参数为fourcc, 编解码器参数(大概), 一般会选一些比较默认的格式。
  - 第三个参数为fps, 为保存视频的帧率。
  - 第四个参数为frameSize, 为保存视频的尺寸, 要和视频大小一样, 不能随便填。
  - 第五个参数为isColor, 决定是否保存为彩色视频, 默认为true。
- 

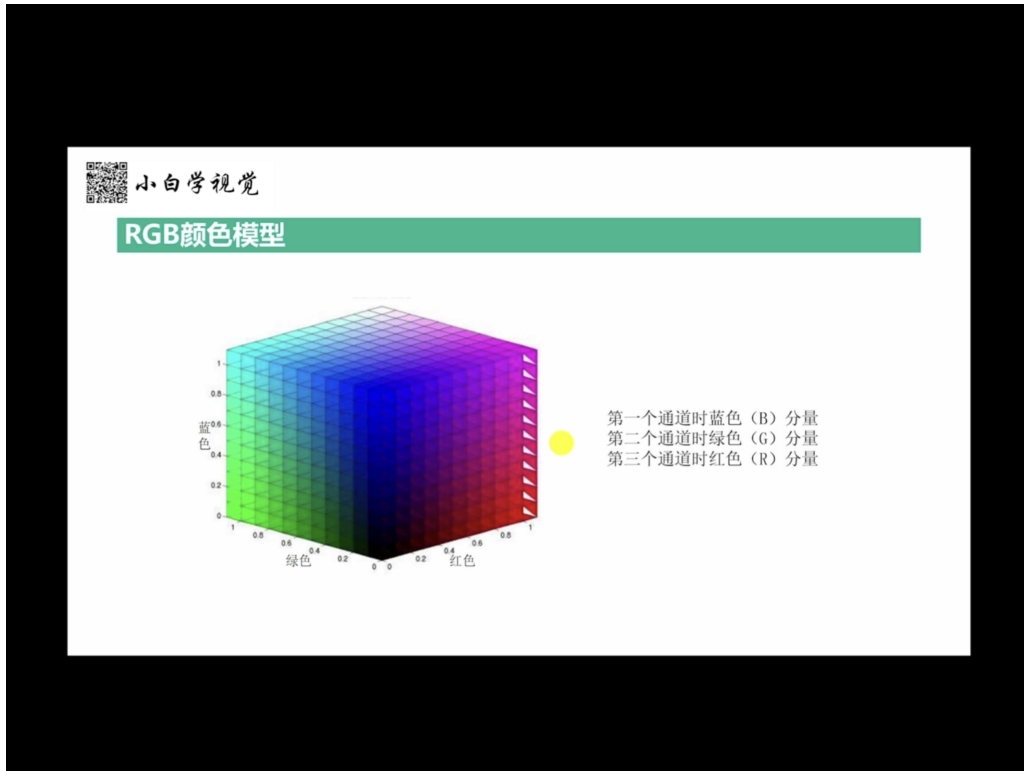
## 4. 图像颜色空间变换

---

## 4.1 图像颜色空间介绍

- RGB模型

为什么0-1，因为有可能出现小数点数据，所以可以把8U(0-255) 变为 float(0-1) 和 double(0-1),



- 图像数据类型的相互转换（不是色彩空间）

- convertTo()

- 第一个参数为m, 指输出的图像。
    - 第二个参数为rtype,为转换后的数据类型。
    - 第三个参数为alpha,为缩放系数。
    - 第四个参数为beta,为平移系数。

//如:

```
a.convertTo(b, CV_32F, 1 / 255.0, 0)
```

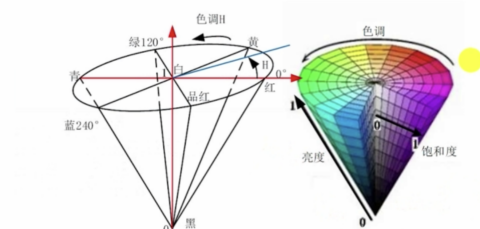
//将a转换为b, 从0-255变成0-1, 类型从uchar变为float.

- HSV颜色模型



小白学视觉

## HSV颜色模型



H是色度 (Hue) —— 颜色  
S是饱和度 (Saturation) —— 深浅  
V是亮度 (Value) —— 亮暗

- GRAY颜色模型

RGB可以推出GRAY，而反过来不能。



小白学视觉

## GRAY颜色模型



RGB模型转灰度图关系：  
$$\text{Gray} = R * 0.3 + G * 0.59 + B * 0.11$$

## 4.2 颜色转换

- **cvtColor()**

- 第一个参数为src, 为待转换的原始图像。
- dst, 转换后的目标图像。
- code, 要从什么空间转到什么空间。
- dstCn, 目标图像中的通道数, 如果参数为零, 则从src和代码中自动导出通道数。

bgr 到 hsv 如果不先把0-255转化到0-1, 结果会不一样。

bgr 到 gray 和 rgb 到 gray, 因为公式, 效果会不一样。