**Mat数据结构**

Mat类型是opencv2.0后的类型，使用此类型无需进行内存管理，即无需手动分配内存，在不需要时自动释放内存，但因目前的许多嵌入式系统仅支持c语言，故而除了在某些嵌入式系统中使用cvMat或IplImage, 基本使用Mat类型。

注：后续的matrix均表示**存储图像像素值**的**矩阵** 。

Mat包含2个数据部分：

**the matrix header**--包含matrix大小，存储方法，matrix存储地址等，matrix header内存大小固定。

**指向像素matrix的指针**。

OpenCV使用的是引用计数系统：每个Mat对象都有着自己的header，但matrix可在两个实例中通过指向同一个matrix首地址的指针来共享数据。

拷贝构造和赋值均只拷贝其header和指向matrix的指针，而非像素数据的本身。当拷贝一个Mat对象的header时，关于matrix的计数器(**refcount**)数值增加，一旦header被清理了，counter减少，当counter减少到0的时候，matrix的内存空间会被释放。

OpenCV有深复制和浅复制两种:

* 浅复制是指针操作，会修改原图像矩阵数据。
* 深复制是复制一个数据拷贝，修改和原图像数据无关。类似Linux的软连接和硬链接。可创建关于全部数据的子部分的header。

例如，可通过创建一个有着新边界的header来创建ROI（感兴趣区域）。对应代码如下：

Mat D (A, Rect(10, 10, 100, 100) ); //using a rectangle

Mat E = A(Range::all(), Range(1,3)); //using row and column boundaries

OpenCV提供了**clone（）**和**copyto（）**两函数来拷贝matrix数据本身，示例如下:

Mat F = A.clone();

Mat G;A.copyTo(G);

像上述示例，修改F或G不会影响A中header所指matrix。（应该是深复制）。

**存储方法**：颜色空间，数据类型。

颜色空间：RGB，HSV，HLS，YCrCb（JPEG格式图片常用），CIE。每一种颜色空间的成分都有其有效范围，从而决定了村塾的数据类型。显示创建Mat对象：Mat类型不仅可以存储图像，还可以通过构造函数创建一个Mat对象。如：

Mat M(2,2, CV\_8UC3, Scalar(0,0,255));

cout << "M = " << endl << " " << M << endl << endl;

<<操作符可以输出Mat里的真实数据值，但仅限于二维矩阵。

构造函数的参数含义：行大小，列大小，数据类型及每个矩阵元素的通道数（具体形式

CV\_[The number of bits per item][Signed or Unsigned][Type Prefix]C[The channel number]），数据。

**已经存在的IplImage类型转换成Mat类型：**

IplImage\* img = cvLoadImage("greatwave.png", 1);

Mat mtx(img); // convert IplImage\* -> Mat

**加载，转换颜色空间，保存图像文件**

Mat image;

image = imread( imageName, 1 );//读取图像文件Mat gray\_image;

cvtColor( image, gray\_image, COLOR\_BGR2GRAY );//转换颜色空间

imwrite( "../../images/Gray\_Image.jpg", gray\_image );//保存图像文件