**1. Mat结构**

Mat img;

img = imread(“../data/image.jpg”);

高，行数

int rows = img.rows;

宽，列数

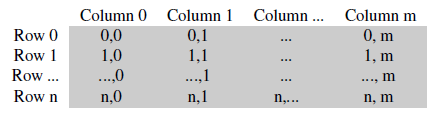
int cols = img.cols;//对于多通道图像，则会将多个通道看做一列

某一点值

double val = img.at<double>(i, j);

灰度图像某一点像素值，元素类型为uchar

int gray\_value = (int)img.at<uchar>(i, j);



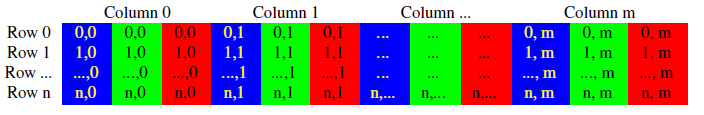
彩色图像某一点像素值，元素类型为Vec3b

cv::Vec3b vec = img.at<cv::Vec3b>(i, j);

int blue\_val = vec[0];

int green\_val = vec[1];

int red\_val = vec[2];



最高效的彩色图像遍历，使用指针

int channels = img.channels();//图像的通道数

int rows\_of\_img = img.rows;

int cols\_of\_img = img.cols \* channels;//对于3通道图像，每个元素占用3个通道即3列

if (img.isContinuous())//当图像的数据连续存储在内容中时，可按一行直接来进行遍历

{

cols\_of\_img \*= rows\_of\_img;

rows\_of\_img = 1;

}

uchar\* p;

for (int i = 0; i < n\_rows; i++)

{

p = img.ptr<uchar>(i);

for (int j = 0; j < n\_cols; j += 3)

{

int blue\_val = p[j];

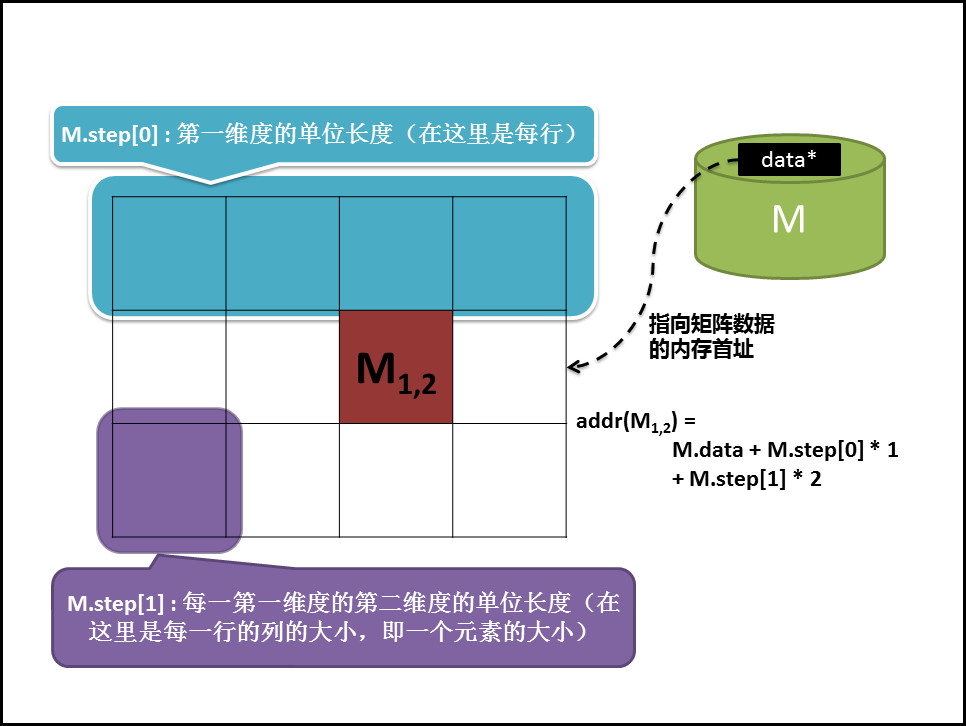
int green\_val = p[j + 1];

int red\_val = p[j + 2];

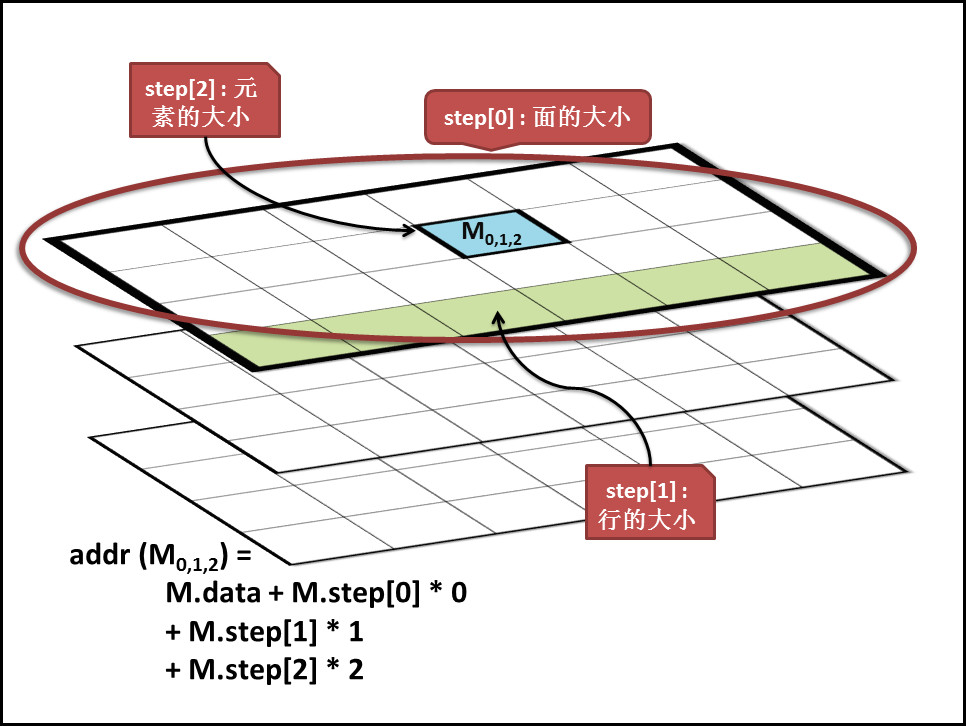
}

}

二维，按行存储：



三维，按面存储：



矩阵 (M) 中数据元素的地址计算公式：

addr(Mi0,i1,…im-1) = M.data + M.step[0] \* i0 + M.step[1] \* i1 + … + M.step[m-1] \* im-1 (其中 m = M.dims M的维度)

* data：Mat对象中的一个指针，指向内存中存放矩阵数据的一块内存 (uchar\* data)
* dims：Mat所代表的矩阵的维度，如 3 \* 4 的矩阵为 2 维， 3 \* 4 \* 5 的为3维
* channels：通道，矩阵中的每一个矩阵元素拥有的值的个数，比如说 3 \* 4 矩阵中一共 12 个元素，如果每个元素有三个值，那么就说这个矩阵是 3 通道的，即 channels = 3。常见的是一张彩色图片有红、绿、蓝三个通道。
* depth：深度，即每一个像素的位数(bits)，在opencv的Mat.depth()中得到的是一个 0 – 6 的数字，分别代表不同的位数：enum { CV\_8U=0, CV\_8S=1, CV\_16U=2, CV\_16S=3, CV\_32S=4, CV\_32F=5, CV\_64F=6 }; 可见 0和1都代表8位， 2和3都代表16位，4和5代表32位，6代表64位；
* step：是一个数组，定义了矩阵的布局，具体见下面图片分析，另外注意 step1 (step / elemSize1)，M.step[m-1] 总是等于 elemSize，M.step1(m-1)总是等于 channels；
* elemSize : 矩阵中每一个元素的数据大小，如果Mat中的数据的数据类型是 CV\_8U 那么 elemSize = 1，CV\_8UC3 那么 elemSize = 3，CV\_16UC2 那么 elemSize = 4；记住另外有个 elemSize1 表示的是矩阵中数据类型的大小，即 elemSize / channels 的大小。

很多OpenCV的函数支持的数据深度只有8位和32位的，所以要少使用double和CV\_64F。

有一个小问题：vs的编译器会把float数据自动变成double型。

**2. 读取视频**

读取摄像头

VideoCapture video;

video.open(0);

读取视频

video.open(“../data/video.wmv”);

得到帧数

int count\_of\_frames = video.get(CV\_CAP\_PROP\_FRAME\_COUNT);

**3. 显示图像**

imshow(“window\_name”, img);

**4. 保存图像为文件**

char file\_path[50];

sprintf(file\_path, "../data/frame\_database/%d.png", j);

imwrite(file\_path, img);