# 一 opencv数据存储结构

## 何为Mat?

计算机视觉中，图像的读取是图像处理的基础，图像就是一系列像素值，OpenCV使用数据结构cv::Mat来存储图像**。cv::Mat是一个矩阵类**，矩阵中**每一个元素都代表一个像素**，对于灰度图像，像素用8位无符号数，0表示黑色，255表示白色。**对于彩色像素而言，每个像素需要三位这样的8位无符号数来表示**，即三个通道（R,G,B），**矩阵则依次存储一个像素的三个通道的值，然后再存储下一个像素点**。

Mat的可用信息

cv::Mat中，

|  |
| --- |
| cols代表图像的宽度（图像的列数），  rows代表图像的高度（图像的行数），  step代表以字节为单位的图像的有效宽度，  step[0]表示一行所占的字节数(如果有间隔,也会算上间隔,这样可以保证拿到第i行)  step[1]表示一个数值(**像素?**)所占字节数.  data是uchar类型.  方位一个int类型的Mat的第i行第j列的值如下:  ***\*( (int\*)(data+i\*m.step[0]+j\*m.step[1]))***  elemSize返回像素的大小，  channels（）方法返回图像的通道数，  total函数返回图像的像素数。  像素的大小 = 颜色大小（字节）\*通道数， |

比如：

三通道short型矩阵（CV\_16SC3）的大小为2\*3 = 6，

三通道Byte型矩阵（CV\_8UC3）的大小为1\*3= 3，像素的channels方法返回图像的通道数，total函数返回图像的像素数。

## 1.2 访问Mat图像数据的方法及实验

**题目(目标)**: RGB图像的颜色数目是256\*256\*256，对图像进行量化，缩减颜色数目到256的1/8（即32\*32\*32）.

如下使用三种方式访问Mat图像数据.

1. Data和step, elemSize1的组合方法.
2. 使用img.at方法.
3. 使用img.ptr方法.

|  |
| --- |
| #include <iostream>  #include < opencv.hpp>  **using** **namespace** cv**;**  **using** **namespace** std**;**    int main**()**  **{**  //新建一个uchar类型的3通道矩阵  Mat img**(**5**,** 3**,** CV\_8UC3**,** Scalar**(**50**,**50**,**50**));**  cout **<<** img**.**rows **<<** endl**;** //5    cout **<<** img**.**cols **<<** endl**;** //3    cout **<<** img**.**channels**()** **<<** endl**;** //3    cout **<<** img**.**depth**()** **<<** endl**;** //CV\_8U 0    cout **<<** img**.**dims **<<** endl**;** //2    cout **<<** img**.**elemSize**()** **<<** endl**;** //1 \* 3,一个位置，三个通道的CV\_8U  cout **<<** img**.**elemSize1**()** **<<** endl**;** //1    cout **<<** img**.**size**[**0**]** **<<** endl**;** //5  cout **<<** img**.**size**[**1**]** **<<** endl**;** //3    cout **<<** img**.**step**[**0**]** **<<** endl**;** //3 \* ( 1 \* 3 )  cout **<<** img**.**step**[**1**]** **<<** endl**;** //1 \* 3    cout **<<** img**.**step1**(**0**)** **<<** endl**;** //3 \* 3  cout **<<** img**.**step1**(**1**)** **<<** endl**;** //3    cout **<<** img**.**total**()** **<<** endl**;** //3\*5    //-------------------------------------- 地址运算 --------------------------------//  **for** **(**int row **=** 0**;** row **<** img**.**rows**;** row**++)**  **{**  **for** **(**int col **=** 0**;** col **<** img**.**cols**;** col**++)**  **{**  //[row, col]像素的第 1 通道地址被 \* 解析(blue通道)   1. **使用data和step成员.** 2. **Data是uchar类型, 操作单元是字节.** 3. **Step[0]描述一行(如果有间隔,也算间隔)的字节数.** 4. **Step[1]描述的是bpp.** 5. **elemSize1()通道间隔** 6. **B通道 elemSize1()\*0** 7. **G通道 elemSize1()\*1** 8. **R通道 elemSize1()\*2**   **\*(**img**.**data **+** img**.**step**[**0**]** **\*** row **+** img**.**step**[**1**]** **\*** col**)** **+=** 15**;**    //[row, col]像素的第 2 通道地址被 \* 解析(green通道)  **\*(**img**.**data **+** img**.**step**[**0**]** **\*** row **+** img**.**step**[**1**]** **\*** col **+** img**.**elemSize1**())** **+=** 15**;**    //[row, col]像素的第 3 通道地址被 \* 解析(red通道)  **\*(**img**.**data **+** img**.**step**[**0**]** **\*** row **+** img**.**step**[**1**]** **\*** col **+** img**.**elemSize1**()** **\*** 2**)** **+=** 15**;**  **}**  **}**  cout **<<** img **<<** endl**;**  //-------------------------------------- Mat的成员函数at<>( ) --------------------------------//  **for** **(**int row **=** 0**;** row **<** img**.**rows**;** row**++)**  **{**  **for** **(**int col **=** 0**;** col **<** img**.**cols**;** col**++)**  **{**  img**.**at**<**Vec3b**>(**row**,** col**)** **=** Vec3b**(**0**,** 0**,** 0**);**  **}**  **}**  cout **<<** img **<<** endl**;**  //-------------------------------------- 使用Mat的成员函数ptr<>() --------------------------------//  **for** **(**int row **=** 0**;** row **<** img**.**rows**;** row**++)**  **{**  // data 是 uchar\* 类型的, m.ptr(row) 返回第 row 行数据的首地址  // 需要注意的是该行数据是按顺序存放的,也就是对于一个 3 通道的 Mat, 一个像素3个通道值, [B,G,R][B,G,R][B,G,R]...  // 所以一行长度为:sizeof(uchar) \* m.cols \* m.channels() 个字节  uchar**\*** data **=** img**.**ptr**(**row**);**  **for** **(**int col **=** 0**;** col **<** img**.**cols**;** col**++)**  **{**  data**[**col **\*** 3**]** **=** 50**;** //第row行的第col个像素点的第一个通道值 Blue    data**[**col **\*** 3 **+** 1**]** **=** 50**;** // Green    data**[**col **\*** 3 **+** 2**]** **=** 50**;** // Red  **}**  **}**  cout **<<** img **<<** endl**;**    Vec3b **\***pix**(NULL);**  **for** **(**int r **=** 0**;** r **<** img**.**rows**;** r**++)**  **{**  pix **=** img**.**ptr**<**Vec3b**>(**r**);**  **for** **(**int c **=** 0**;** c **<** img**.**cols**;** c**++)**  **{**  pix**[**c**]** **=** pix**[**c**]** **\*** 2**;**  **}**  **}**  cout **<<** img **<<** endl**;**  //-------------------------------------- 使用Mat的成员函数ptr<>() --------------------------------//  MatIterator\_**<**Vec3b**>** it\_im**,** itEnd\_im**;**  it\_im **=** img**.**begin**<**Vec3b**>();**  itEnd\_im **=** img**.**end**<**Vec3b**>();**  **for(;** it\_im **!=** itEnd\_im**;** it\_im**++)**  **{**  **\***it\_im **=** **(\***it\_im**)** **\*** 2**;**  **}**  cout **<<** img **<<** endl**;**    cvWaitKey**();**  **return** 0**;**    **}** |

# 二 卷积

## 2.1 何为卷积

图像I, 卷积核k.

1. 先把卷积核旋转180度. 表现上是对角替换.
2. I和K做卷积. 有三种类型.

## 2.2 Full类型卷积

I的类型为H1,W1, K的类型为 H2,W2.

生成的Cfull类型为 H1+H2-1,W1+W2-1.

## 2.3 valid类型

Cvalid类型 H2-1:H1 , W2-1:W1

## 2.4 same类型

Csame类型为 H1,W1, 和原图I的类型一致.

三 Smooth