图像处理和opencv总结

待解决问题：

1. 自己总结
2. 图像的等号是相当于指针赋值，指向相同的内存，不是拷贝。
3. 掩膜（mask）：0-黑，255-白

<https://www.cnblogs.com/skyfsm/p/6894685.html>

1. 常用函数：其中页码为《opencv3 编程入门》对于页数

* 创建滚动条函数（P73）：第三个参数是滑动条当前值

int createTrackbar(const String& trackbarname, const String& winname,

int\* value, int count,

TrackbarCallback onChange = 0,

void\* userdata = 0);

* Rect(起点横坐标, 起点纵坐标, 宽（列，x方向）, 高（行，y方向）)
* namedWindow("原图", 0); //命名窗口和显示图像，有时直接用imshow窗口不能变化

imshow("原图", src\_img);

* 方法reference at(size\_type \_Pos)：可用来存取元素，但是必须在编译器知道图像的数据类型，其本身不会对任何数据类型进行转换（P114）
* 取感兴趣部分代码（P115），注意理解掩膜mask和语句构成

// 【2】定义一个Mat类型并给其设定ROI区域

Mat imageROI= srcImage1(Rect(200,250,logoImage.cols,logoImage.rows));

// 【3】加载掩模（必须是灰度图）

Mat mask= imread("dota\_logo.jpg",0);

//【4】将掩膜拷贝到ROI，imageROI = logoImage & mask（详情见1）

logoImage.copyTo(imageROI,mask);

* addWeighted(InputArray src1, double alpha, InputArray src2,\

double beta, double gamma, OutputArray dst, int dtype = -1):计算数组加权和，gamma相当于线性函数的常数项（P117）

* 多通道图像混合（P125）,注意split（）、merge（）、at（）用法

//【2】将一个三通道图像转换成三个单通道图像，BGR顺序：字母排序

split(srcImage,channels);//分离色彩通道，将一个多通道组分成几个单通道组

//【3】将原图的绿色通道的引用返回给imageBlueChannel，注意是引用，相当于两者等价，修改其中一个另一个跟着变

imageGreenChannel= channels.at(1);

//【4】将原图的绿色通道的（500,250）坐标处右下方的一块区域和logo图进行加权操作，将得到的混合结果存到imageGreenChannel中

//注意加权1.0和0.5，这里主要是为了突出绿色通道 addWeighted(imageGreenChannel(Rect(500,250,logoImage.cols,logoImage.rows)),1.0,

logoImage,0.5,0.,imageGreenChannel(Rect(500,250,logoImage.cols,logoImage.rows)));

//【5】将三个独立的单通道重新合并成一个三通道

merge(channels,srcImage);

* 对比度亮度调整（P133）

// g\_nContrastValue变化，变化对比度（可用方差度量），为GBR每种颜色通道乘个系数，称为增益（gain）

// g\_nBrightValue变化，只是对每种颜色通道增大或减小一个值，即偏置（bias）

// saturate\_cast<type>：溢出保护,其中uchar范围为0-255

g\_dstImage.at<Vec3b>(y,x)[c] = saturate\_cast<uchar>( (g\_nContrastValue\*0.01)\*\

( g\_srcImage.at<Vec3b>(y,x)[c] ) + g\_nBrightValue );

* 输入和输出XML和YAML文件（P144）, FileStorage是opencv中XML和YAML文件的存储类，对于vector结构（一堆数），要注意前后分别加[];对于map结构（相似于python字典）操作，前后加{}。

FileStorage fs("test.yaml", FileStorage::WRITE);

//开始文件写入

// 变量名 变量

fs << "name" << "zhangyilu";

fs << "frameCount" << 5;

//获取当前时间，https://zhidao.baidu.com/question/1817949634825883388.html

time\_t rawtime;

time(&rawtime);

fs << "calibrationDate" << asctime(localtime(&rawtime));

//初始化

FileStorage fs2("test.yaml", FileStorage::READ);

std::string name;

// 第一种方法，对FileNode操作

name = fs2["name"];

// 第二种方法，使用FileNode运算符> >

fs2["name"] >> name;

//注意FileNode，FileNodeIterator的使用方法

FileNode features = fs2["features"];

FileNodeIterator it = features.begin(), it\_end = features.end();

int idx = 0;

std::vector<uchar> lbpval;

//使用FileNodeIterator遍历序列

for( ; it != it\_end; ++it, idx++ )

{

cout << "feature #" << idx << ": ";

cout << "x=" << (int)(\*it)["x"] << ", y=" << (int)(\*it)["y"] << ", lbp: (";

// 我们也可以使用使用filenode > > std::vector操作符很容易的读数值阵列

(\*it)["lbp"] >> lbpval;

for( int i = 0; i < (int)lbpval.size(); i++ )

cout << " " << (int)lbpval[i];

cout << ")" << endl;

}

1. 常见错误

* Printf打成print：“初始化”: 无法从“const char \*”转换为“cv::Formatted \*”

1. 重要公式

1、部分简单公式：线性混合（P116）、图像亮度和对比度调整（P131）、

1. 专业名词

* 亮度（brightness）和对比度（contrast）调整、颜色校正（colorcorrection）、变换（transformation）
* 傅里叶变换（discrete Fourier transform）
* 2种文件格式：可扩展标识语言XML（eXtensible Markup Language）、YANL(Yet Another Markup Language)

1. 《opencv3 编程入门》第5章14种遍历像素方法运行时间

【方法一】利用.ptr 和 []的方法所用时间为 10.4476ms

【方法二】利用 .ptr 和 \* ++ 的方法所用时间为10.4114ms

【方法三】利用.ptr 和 \* ++ 以及模操作的方法所用时间为13.2222ms

【方法四】利用.ptr 和 \* ++ 以及位操作的方法所用时间为8.52616ms

【方法五】利用指针算术运算的方法所用时间为8.76294ms

【方法六】利用 .ptr 和 \* ++以及位运算、channels()的方法所用时间为69.9129ms

【方法七】利用.ptr 和 \* ++ 以及位运算(continuous)的方法所用时间为8.98351ms

【方法八】利用 .ptr 和 \* ++ 以及位运算 (continuous+channels)的方法所用时间为7.47544ms

【方法九】利用Mat\_ iterator 的方法所用时间为364.827ms

【方法十】利用Mat\_ iterator以及位运算的方法所用时间为366.975ms

【方法十一】利用Mat Iterator\_的方法所用时间为510.019ms

【方法十二】利用动态地址计算配合at 的方法所用时间为790.24ms

【方法十三】利用图像的输入与输出的方法所用时间为9.11233ms

【方法十四】利用操作符重载的方法所用时间为6.45561ms

1. 矩阵行是图像高度，矩阵列是图像宽度。原点在左上角，x轴向右，y轴向下
2. 图像处理简介：

<https://blog.csdn.net/jialibang/article/details/78181410>

重点提取：

* 如果想做图像处理方面的研究，Matlab是必须掌握的，而且是熟练掌握，推荐一本相关的书籍《MATLAB图像处理实例详解（附光盘）》，刚上研究生时就靠两本书入门的，一是冈萨雷斯的《数字图像处理》，二是这本《MATLAB图像处理实例详解》。千万不要试图去记忆所有的工具函数，这种做法是十分愚蠢的。正确的做法是根据自己的情况快速翻阅这类工具书，可以找出里面的有实际意义的源码来敲一敲练练手感，至于具体的工具函数，只需要知道Matlab提供了这方面的功能就行了，以后用到了再回来查，或者谷歌百度。
* 由于Opencv是开源的，那么如果你在项目中直接调用了它的API，那就意味着你的项目也必须开源。因此在真正的产品开发过程中，往往需要从Opencv库里面挖代码，而不是直接调用，幸好Intel公司允许我们看源码，自己编译一把就可以了。

这里同样对推荐两本关于Opencv方面的教程。一本是CSDN博客大牛毛星云写的《OpenCV3编程入门》，不要试图记忆API函数。重要的工具用多了自然会记住，不重要的工具记住了也没用。第二本书是《图像识别与项目实践――VC++、MATLAB技术实现》。

说道C++和Opencv，有一个问题不得不提，那就是深度学习领域大名鼎鼎的Caffe框架。

* Python，Numpy是Python的线性代数库，Theano是Python的机器学习库人觉得单从图像处理的角度评价的话，Python并没有前面两个工具（Matlab和OpenCv）应用广泛，小甲鱼的《零基础入门学习Python》视频教程
* 图像处理研究主要可以分为三个部分：基础概念、基本思想、算法研究。

a.基础概念：冈萨雷斯编写的、阮秋琦翻译的《数字图像处理》。这本书已经作为图像处理领域的经典教材使用了三十多年，我自己也把这本书看了好几遍，每一遍都会有新的体会。

b.基本思想：刚开始想把这部分内容命名为“基本算法”，意在介绍图像处理中的一些基本算法，后来仔细想想决定不这么写，因为图像处理是一个非常大的概念，图像处理不等于人脸识别，也不等于模式识别，直接介绍诸如图像处理基本算法之类的内容很容易写成空话，没有什么实际意义。读者有兴趣的话可以直接谷歌百度“图像处理十大经典算法”

http://www.360doc.com/content/18/0816/18/15930282\_778796343.shtml

在模式识别方向判断一个学生是否入门有一个非常简单的方法，就是“如果你能把图像很自然的想象成高维空间中的一个点”，那就说明在模式识别方面入门了，可以对图像进行分类了。

c.算法研究：算法研究应该是图像处理的核心工作，尤其是各大高校的博士硕士。这里我并不想谈那些高大上的算法，我更想说的是一些算法研究的一些基础的东西，比如说一些基础课程，比如说矩阵运算。

建议图像处理方面的硕士一定要上两门课：《泛函分析》以及《最优化算法》，有的学校已经将这两门课列为了研究生阶段的必修课程。这两门可可以说是图像处理（至少是模式识别）的基础。我当初没上过最优化算法，但后来也自己补上了，不然真的是寸步难行。至于泛函我当时听课的时候也不是很懂，但是在之后的研究过程中发现很多图像处理的基本知识基本理论都和泛函分析中枯燥的定理如出一辙，没办法，有的东西本身就是枯燥的干货，学着费力，缺它不行。

忠告：不仅仅针对图像处理，对于其他新技术的入门学习也是一样，尽快迈出第一步，尽快去建立自信和成就感，让自己有勇气走下去，然后缺什么补什么就行了。我觉得真正让人望而却步的往往不是技术本身，而是我们对自身的不自信。唯有果断开工，才能战胜心魔。