Opencv图像叠加

两种方法：roi区直接叠加，线性叠加（roi区线性叠加）

Roi区直接叠加：

生成roi矩阵，在pic()中用rect()函数选定roi区域，注意rect(pic中x, pic中y, 需要叠加的图像x, 需要叠加的图像y) 生成pic的灰度掩膜mask，使用logo.copyto()函数把mask加载到roi上，最后显示pic（为什么mask加载到roi上最后不显示roi，这步意义？=为引用对ROI的操作就是对pic的操作）

roi法能够直接选定区域叠加，不用保证图像大小一致，因为在rect中已经规定，缺点是直接叠加，没有透明效果。

线性叠加：

读取后使用addWeighted()函数直接进行线性叠加，addWeighted(第一张图片，第一张图片加权，第二张图片，第二张图片加权，整体加权，生成mat名)最后imshow（生成文件名）

线性叠加法能够直接对两张图片进行加权叠加，达到有透明度效果。缺点是需要保证两张图片大小格式一致，不然会报错。

Roi区线性叠加：

生成roi矩阵，在pic中使用rect()选定roi区域，后使用addWeighted()进行线性加权，注意addWeighted()最后生成矩阵还是ROI矩阵，输出imshow()时使用原图函数名即pic（？=为引用，对ROI矩阵的改变就是对pic的改变）

Roi区线性叠加即使图像大小不一致依然能够线性叠加，达到透明度效果。

Opencv split

split()函数将图像的蓝绿红三通道分离，需要在一开始定义向量vector<Mat> channels，后使用split(图像mat，channels)把三通道分离到channels中，调用时，用B=channels.at(0)调用，表明调用蓝色通道。在单通道合并图像时，需要合并的图像应转化为灰度形式，因为原图像是单通道，如果不加以转化，logo图像还是三通道，合并会出现错误。最后使用merge()函数将通道合并，merge(channels, pic)后可用imshow()读取。

Opencv trackbar control contrast&bright

使用trackbar控制对比度和亮度时，需要新建dst矩阵并初始化，使其与pic矩阵大小类型一致。使用dst=Mat::zeros(pic.size(), pic.type())语句初始化预分配。后用createTrackbar(“进度条名字”, “所在窗口名”, &控制的量, 最大值, 回调函数)创建trackbar。再创建回调函数，ContrastAndBright(int, void\*)一个整形变量为当前值，一个指针（可能是createTrackbar传递？），在回调函数中，将contrast和bright带入公式逐帧计算，row->col->BGR逐个计算存入dst中，读取使用dst.at<Vec3b>(y,x)[c]，最后imshow，进度条改变后数据自动传入回调函数，重新计算输出。

Opencv 滤波

3种线性滤波：方框boxFilter()、均值blur()、高斯滤波GaussianBlur()。2种非线性滤波：中值medianBlur()、双边滤波bilateralFilter()

线性滤波使用线性邻域滤波算子，使用不同权重去结合小邻域内的像素，得到处理结果。

方框滤波为全1矩阵，当归一化后，变为均值滤波。图像的高斯模糊过程就是图像与正态分布做卷积。三类滤波函数的使用：boxFilter(原图，目标图, 深度(-1), 内核大小(Size(a,b)))

Blur(src, dst, Size(a,b)) GaussianFilter(src, dst, Size(a,b), sigmaX, sigmaY)

中值滤波使用该点邻域中的中值代替该点，让像素接近真实值，对斑点噪声特别适用。中值滤波的结果几乎不受噪声的影响，但是花费时间是均值滤波的5倍以上，并且对细节多的图片，特别是线，尖顶，不宜使用。

medianBlur(src,dst, 孔径线性尺寸，>1的奇数)

bilateralFilter(src, dst, 像素领域直径, 颜色sigma, 坐标sigma)

opencv形态学操作

形态学操作就是基于形状的一系列图像处理操作。

一、膨胀与腐蚀（Dilation and Erosion）膨胀与腐蚀是对高亮部分而言的，不是黑色部分。

Dilation是求局部最大值的操作，核与图形卷积，把最大值赋给参考点指定的像素，使高亮区域逐渐增长

Erosion是求局部最小值的操作。

Dilate(src, dst, element);erode(src, dst, element); 在执行语句前需要获取核的大小，使用getStructingElement(方式，Size())得到

二、开闭运算(opening/closing operation)

开运算其实就是先腐蚀后膨胀的过程，即dilate(erode(src, element))，可用来消除小物体，在纤细点处分离物体，平滑较大物体的边界但不明显改变其面积

闭运算其实就是先膨胀后腐蚀的过程，即erode(dilate(src, element))，可用来排除小型黑色区域

三、形态学梯度(morphological gradient)

形态学梯度为膨胀图和腐蚀图之差，即dilate(src, element)-dilate(src, element)，可用来保留物体的边缘轮廓

四、顶黑帽(top/black hat)

顶帽为原图和开运算之差，即src-open(src, element) 开运算放大了裂缝或局部亮度较低的区域，从原图中减去后得到了比原图轮廓周围的区域更亮的区域。可用来分离比临近点亮的一些斑块。

黑帽为闭运算和原图之差，即close(src, element)-src 闭运算排除了小型黑洞区域，使物体主体保留，再减去原图可得到比原图轮廓周围更暗的区域，可用来分离比临近点暗一些的斑块

以上操作都可以再函数morphologyex()中改变第三个参数得到。

MORPH\_OPEN/MORPH\_CLOSE/MORPH\_GRADIENT/MORPH\_TOPHAT/MORPH\_BLACKHAT

同样，内核需要使用getStructingElement()得到。

Opencv边缘检测

边缘检测一般步骤：1）滤波：高斯滤波2）增强：梯度幅值3）检测：阈值化方法

1. canny算子

1.滤波：高斯滤波 2.计算梯度幅值和方向 3.非极大值抑制 4.滞后阈值：高于高阈值被保留，低于低阈值被排除，在中间仅在高阈值旁边时被保留

Canny(src, dst, 3, 9, 3)其中第三第四项为低高阈值设置，推荐在2:1-3:1

二、sobel算子

结合了高斯平滑和微分求导，用来计算灰度函数的近似梯度。可得到梯度矢量或法矢量。1.分别在x，y方向求导：使用内核进行卷积 2.每一点求出近似梯度：x，y方向梯度平方和的根号

三、laplace算子

拉普拉斯算子就是对x，y偏导数的和，简化了求导过程，其内部调用sobel算子。让一副图像减去他的laplacian可以增强对比度。