1. OpenCV—矩阵数据类型转换 cv::convertTo

函数
void convertTo(OutputArray m, int rtype, double alpha=1, double beta=0) const;

m　　– 目标矩阵。如果 m 在运算前没有合适的尺寸或类型，将被重新分配。
rtype – 目标矩阵的类型。因为目标矩阵的通道数与源矩阵一样，所以 rtype 也可以看做是目标矩阵的位深度。如果 rtype 为负值，目标矩阵和源矩阵将使用同样的类型。
alpha – 尺度变换因子（可选）。
beta　– 附加到尺度变换后的值上的偏移量（可选）。

描述
m(x,y) = saturate_cast<rType>(alpha(*this)(x,y)+beta)

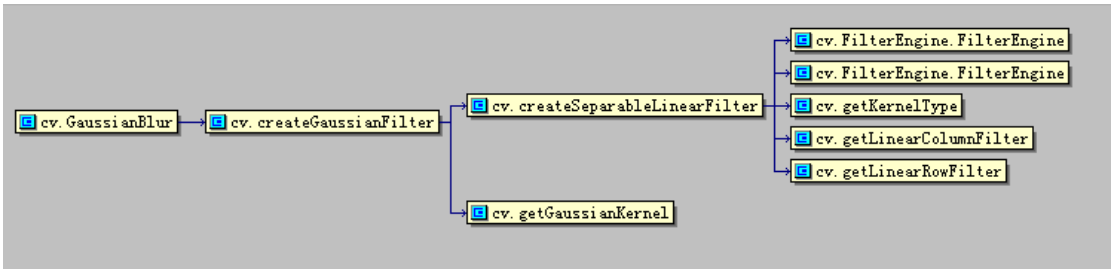
markers.convertTo(tmp,CV_8U,255,255);
将矩阵 markers 转换为 CV_8U 类型的矩阵 tmp：tmp(x,y)= markers(x,y)*255+255.这样，将图像做线性变换，使值为-1 的像素变为 0（-1*255+255=0）。值大于 255 的像素将赋值为 255，这是因为 CV32S 转换为无符号 CV_8U 时，应用了饱和度运算。具体应用参看分水岭算法的相关博文。

2. cv::GaussianBlur
高斯滤波 GaussianBlur()

其函数声明为：
void GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT) ;
功能：对输入的图像 src 进行高斯滤波后用 dst 输出。
参数：src 和 dst 当然分别是输入图像和输出图像。Ksize 为高斯滤波器模板大小，sigmaX 和 sigmaY 分别为高斯滤波在横线和竖向的滤波系数。borderType 为边缘扩展点插值类型。

其函数声明为：
void GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT) ;
功能：对输入的图像 src 进行高斯滤波后用 dst 输出。
参数：src 和 dst 当然分别是输入图像和输出图像。Ksize 为高斯滤波器模板大小，sigmaX 和 sigmaY 分别为高斯滤波在横线和竖向的滤波系数。borderType 为边缘扩展点插值类型。

该函数内部调用了很多其他的函数，其调用的函数层次结构如下图所示：



这里分析源代码不需要深入到最底层，只需分析到函数 createSeparableLinearFilter 和 getGaussianKernel 即可。

要分析函数 GaussianBlur，必须先分析其调用过的内部函数。因此首先分析函数 getGaussianKernel。
功能：返回一个 ksize*1 的数组，数组元素满足高斯公式：

$$G_i = \alpha * e^{-(i-(ksize-1)/2)^2/2*sigma^2}$$

cv::GaussianBlur(imgF, blurred, cv::Size(5, 5), sigma);

其中只有系数 alpha 和参数 sigma 未知，sigma 的求法为：
如果输入 sigma 为非正，则计算公式为：sigma = 0.3*((ksize-1)*0.5 - 1) + 0.8 .
如果输入 sigma 为正，则就用该输入参数 sigma。
最后 alpha 为归一化系数，即计算出的 ksize 个数之和必须为 1，所以后面只需求 ksize 个数，计算其和并求倒即可。

$$e^{-(i-(ksize-1)/2)^2/2*sigma^2}$$

```
cv::Ptr<cv::FilterEngine> cv::createGaussianFilter( int type, Size ksize,
                                                    double sigma1, double sigma2,
                                                    int borderType )
{
    int depth = CV_MAT_DEPTH(type);//取数组元素的深度
    if( sigma2 <= 0 )
        sigma2 = sigma1;//当第 3 个参数为非正时，取其与第二个参数相同的值

    // automatic detection of kernel size from sigma
    /*一般情况下满足 sigma1>0*/
    if( ksize.width <= 0 && sigma1 > 0 )//当滤波器核的宽非正时，其宽要重新经过计算
    /*根据 CV_8U 来计算，核宽为接近 7*sigma1 或者 9*sigma1*/
        ksize.width = cvRound(sigma1*(depth == CV_8U ? 3 : 4)*2 + 1)|1;
    if( ksize.height <= 0 && sigma2 > 0 )
        /*同理，核高根据 CV_8U 来计算，为接近 7*sigma2 或者 9*sigma2*/
        ksize.height = cvRound(sigma2*(depth == CV_8U ? 3 : 4)*2 + 1)|1;

    CV_Assert( ksize.width > 0 && ksize.width % 2 == 1 &&
        ksize.height > 0 && ksize.height % 2 == 1 );//确保核宽和核高为正奇数

    sigma1 = std::max( sigma1, 0. );//sigma 最小为 0
    sigma2 = std::max( sigma2, 0. );

    Mat kx = getGaussianKernel( ksize.width, sigma1, std::max(depth, CV_32F) );//得到 x 方
向一维高斯核
    Mat ky;
    if( ksize.height == ksize.width && std::abs(sigma1 - sigma2) < DBL_EPSILON )
        ky = kx;//如果核宽和核高相等，且两个 sigma 相差很小的情况下，y 方向的高斯
核去与 x 方向一样，减少计算量
    else
        ky = getGaussianKernel( ksize.height, sigma2, std::max(depth, CV_32F) );//否则计算
y 方向的高斯核系数

    return createSeparableLinearFilter( type, type, kx, ky, Point(-1,-1), 0, borderType );//返回
2 维图像滤波引擎
}
```

最后来看真正的高斯滤波函数 GaussianBlur：
功能：对输入图像_src 进行滤波得到输出图像_dst,滤波核大小为 ksize，滤波参数由 sigma1 和 sigma2 计算出，边缘扩展模式为 borderType.
其源代码和注释如下：

```
void cv::GaussianBlur( InputArray _src, OutputArray _dst, Size ksize,
                       double sigma1, double sigma2,
                       int borderType )
{
    Mat src = _src.getMat();//创建一个矩阵 src，利用_src 的矩阵头信息
    _dst.create( src.size(), src.type() );//构造与输入矩阵同大小的目标矩阵
    Mat dst = _dst.getMat();//创建一个目标矩阵

    if( ksize.width == 1 && ksize.height == 1 )
    {
        src.copyTo(dst);//如果滤波器核的大小为 1 的话，则说明根本就不用滤波，输出矩阵与输入矩阵完全相同
        return;
    }

    if( borderType != BORDER_CONSTANT )//当边缘扩展不是常数扩展时
    {
        if( src.rows == 1 )
            ksize.height = 1;//如果输入矩阵是一个行向量，则滤波核的高强制为 1
        if( src.cols == 1 )
            ksize.width = 1;//如果输入矩阵是一个列向量，则滤波核的宽强制为 1
    }

    /*生成一个高斯滤波器引擎 f*/
    Ptr<FilterEngine> f = createGaussianFilter( src.type(), ksize, sigma1, sigma2, borderType );
    f->apply( src, dst );//调用引擎函数，完成将输入矩阵 src 高斯滤波为输出矩阵 dst
}
```