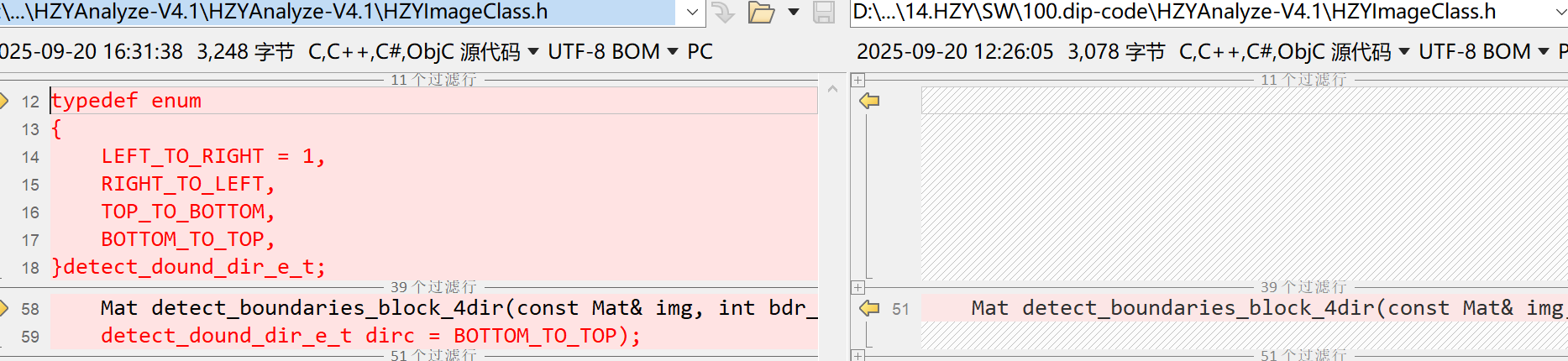
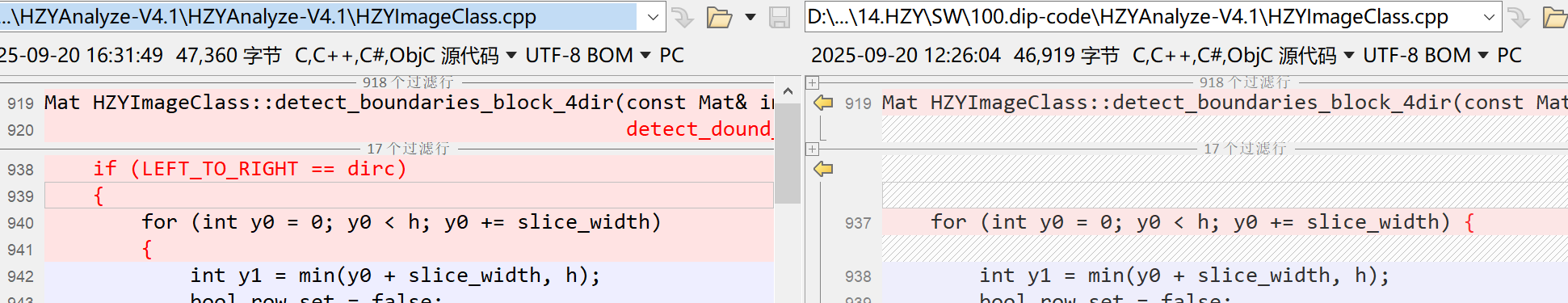
修改点：

1. detect\_boundaries\_block\_4dir函数增加了一个方向参数dirc，类型为自定义的detect\_dound\_dir\_e\_t，指示提取的时候从哪个方向开始。

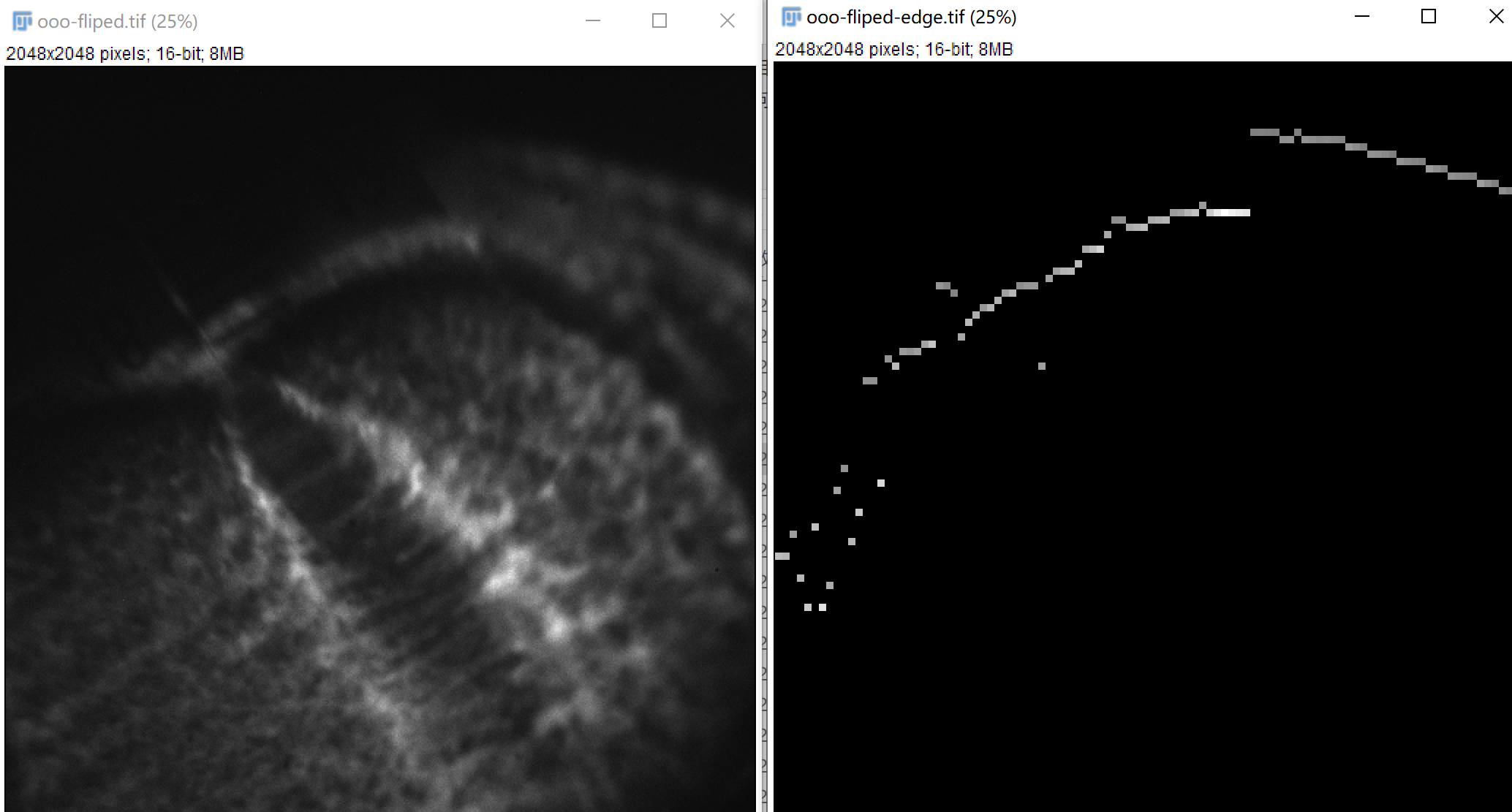


1. detect\_boundaries\_block\_4dir实现中，根据参数dirc的值，只进行一个方向的处理：

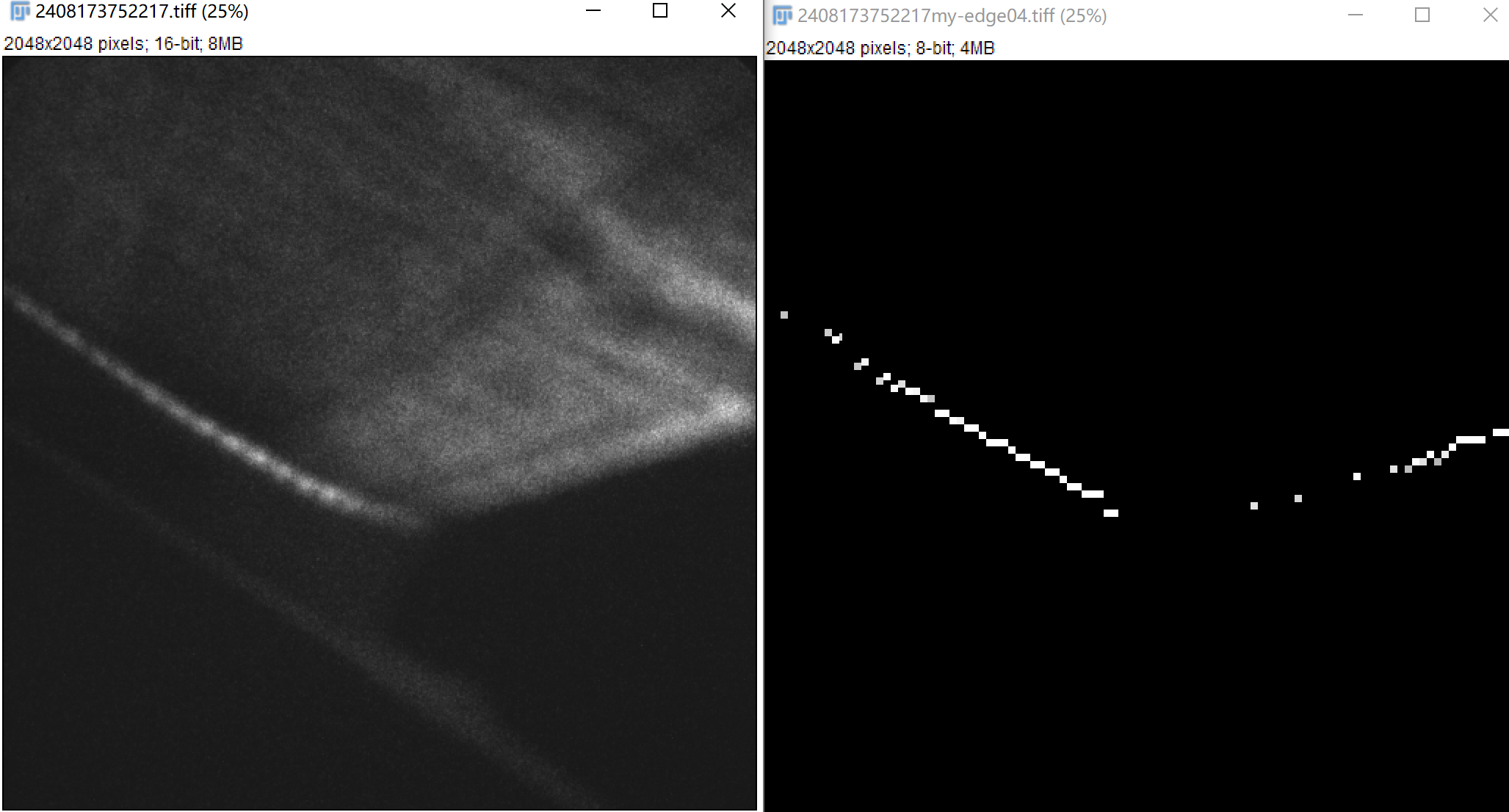


1. 调用detect\_boundaries\_block\_4dir时指定方向。这个方向作为配置参数在GUI上配置（和现在的其它三个参数一样）。在我们目前进行了图像上下翻转的情况下，GUI的默认值可以设置为TOP\_TO\_BOTTOM。从目前我们收到的的图看，这个方向都可以处理。
2. 一个小修改，实际影响不大：detect\_boundaries\_block\_4dir函数中从上到下和从下到上的处理部分交换了一下，更符合逻辑。

上述修改后的处理效果如下：



detect\_boundaries\_block\_4dir 函数的效果如下（左边是原图，右边是提取出来的效果。原图是16bit的，可以下载Fiji工具打开：https://imagej.net/software/fiji/downloads）：



右图图中提取出来的边界就是波阵面。

算法的原理：

注意到这些图像都是由波阵面分开的两部分，一边基本上是全黑，另一半是密布的灰色点。图像的特征是，从波阵面两侧到波阵面，都有一个亮度突变。所以处理思路是找这种突变，具体实现是：

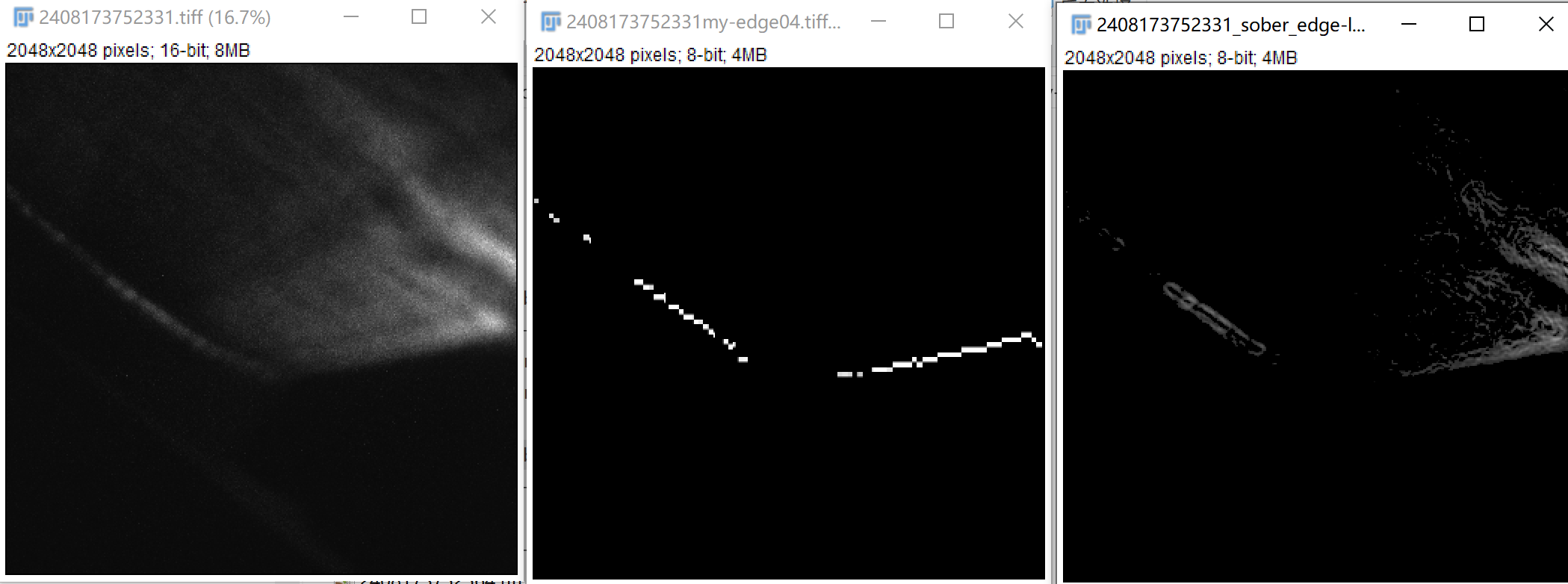
1. 从左到右：图像被分割为高度slice的横向条纹。对每个条纹，从左到右分割为width宽的小方块，然后依次检查每个小方块C和它左侧小方块L的像素平均值center\_mean和left\_mean，如果left\_mean/center\_men < border\_diff\_r（目前是用的0.7），则认为是一个亮度突变，小方块C处于边界上，将它内部像素值都设置为center\_mean（也可以保持不变），然后将L清0。这样可以得到一条边界。如下图：



1. 从右往左、从上往下、从下往上，都这样做一个类似的操作。最后再把四个方向的边界内外全部清0。

四个方向都做这种操作是因为不知道波阵面的方向。如果事先知道波阵面是水平方向，那么只需要做从上到下和从下到上两个方向即可。

这种“突变”的寻找就是找梯度的变化，所以也可以考虑用opencv的sobel来计算。我有一个python脚本sober-level.py是这样实现的。但它处理有些图像效果比较好，但有些图像会留存一些灰色区域处理不掉。比如2408173752331.tiff，原图、detect\_boundaries\_block\_4dir和sober-level.py处理后的效果如下：



Sober-level.py脚本如下：



用chatgpt转的c++代码：

