# **算法说明**

我们参考的论文是2011年发表在PAMI上的一片论文：A Laplacian Approach to Multi-Oriented Text Detection in video. 作者这篇论文引用比较高，从作者的测得的数据来看也确实不错。

但是当我们按作者的算法一步步实现起来的时候才发现，作者隐藏了太多的细节。单纯的把论文里所说的步骤重现远远达不到作者所说的那么好的结果，尤其是极高的误警率。

下面是算法的详细步骤说明，可以结合text\_detect.m中的代码注释来看（整个过程在灰度图像上进行处理）：

1. **文字检测**

a.采用理想低通滤波器进行平滑噪声，模糊范围是填充后宽度的0.8倍。

b.然后在频域使用拉普拉斯，这里使用拉普拉斯是因为它作为一个二阶微分算子，相比于一阶微分算子对细节更加敏感，这样可以检测到许多低对比度的文字。但这个也是一把双刃剑，就是因为拉普拉斯的过于敏感，使得检测结果有超高的误警率，不过高误警率除了这部分造成的原因外，后面的骨架分割也是一个重要原因。

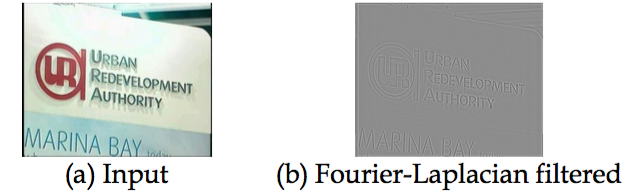
MATLAB没有自带的频域拉普拉斯滤波函数(只有空域)，所以我们参考《数字图像处理MATLAB版》自己写了一个。这里我们使用的函数是：paddesize.m，，dftuv.m，lpfilter.m， dftfilt.m。

paddesize.m接受图像大小参数，进行图像填充。

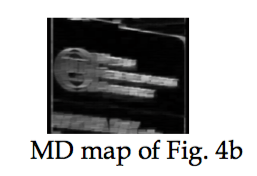
dftuv.m接受填充后图像的大小作为参数，产生离散U，V的网格数据。

lpfilter.m产生低通滤波算子，可以选择理想，巴特沃斯，高斯低通滤波。作者采用理想低通滤波的理由是：参数比较简单。但实际上个人觉得采用别的滤波参数也没有更复杂。至于实现效果，我们测试了一几幅图片，看不出有什么差距，不过这么小的测试量也许不能代表什么。所以我们还是选择了作者的给出的方法。

dftfilt.m进行频域滤波，这个函数接受两个参数，一个是图片，一个是滤波算子，将通过lpfilter.m得到的低通滤波算子和拉普拉斯算子进行乘积作为第二个参数进行滤波，就可以得到滤波后的图像。如下所示：



c.在上面结果的基础上，计算MD图，所谓MD图，就是对于图像中的每一个像素取一个1×N(作者取N=21)的小窗口，求这个窗口的像素灰度的最大值和最小值，然后作差，就得到了该像素的MD值，对每一个像素进行遍历，就得到了MD图。如下所示：



这个过程在函数text\_detect.m里面进行

e.使用K-means进行分类，基于MD值得欧氏距离把像素分成两类，均值大的就认为是文字类，均值小的就认为是非文字类。在使用K-means之前，作者做了一步开操作，而且K-means之后，作者就给出了如下的结果图片，这显然是一副二值图片：



但是作者却没有提及如何就变成了二值图片，而且在灰度图像上进行开操作去除毛刺效果会不会不如在二值图像上进行开操作？这里是作者隐藏的第一个细节，这里我们实际编程的时候时将K-means分类后均值大的那一类置为1，均值小的置为0，并且按作者所说的，在分类之前进行开操作。总之我们想尽量还原作者原来的算法，希望得到较好的结果。

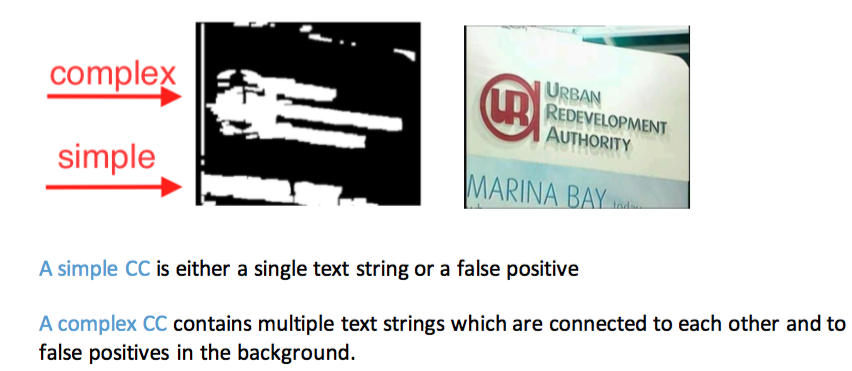
这个过程在函数text\_detect.m里面进行。

1. **连通分量分类**

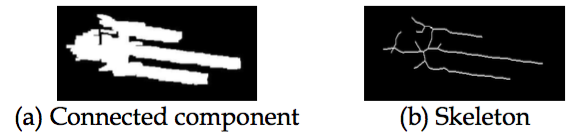
作者在这里使用了一种独一无二的方法：用连通分量来展示文字行，而不是用传统的bounding box。而且进一步提出了使用骨架来把连通分量分割成不同的文字行。

个人觉得这是本文的一个唯一亮点吧，抛开作者采用的方法的检测效果不谈。作者的这种使用连通分量来展示文字行的方法是少数可以用于多方向检测的方法，传统上的方法大都是基于水平，竖直方向，或者斜45°方向检测。但作者这个方法就突破了这个限制。下面就具体思路进行说明：

作者认为，连通分量分成简单的和复杂两类：

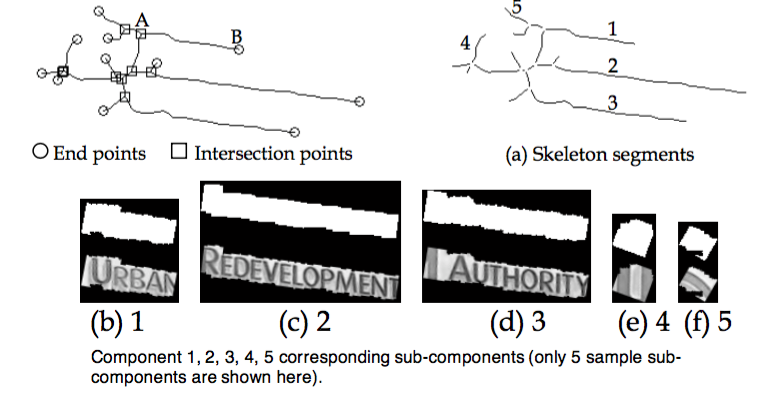


具体来说，就是对连通分量求骨架，如果骨架有相交的地方，那么就认为是复杂的连通分量，否则就是简单的连通分量。如下面这个就是复杂的连通分量：



1. **连通分量分割**

对于上面提及的复杂的连通分量，我们求它的骨架，然后对骨架进行分割，分割的方法就是断开骨架的交点，断开后就可以得到很多简单的骨架。没有交点的骨架所对应的连通分量就是简单的连通分量。如下图所示：



第一行的左侧那幅图就是复杂连通分量所对应的骨架，第一行右侧就表示骨架分割后的情况，可以看到，骨架进行分割后，骨架变成了一个个单一的，简单的形式。上图的第二行表示各个骨架（第一行右侧的骨架1，2，3，4，5）所对应的连通分量。

在这里，作者又隐藏了两个极为关键的细节：

1.对于连通分量求骨架后，必然会有许多毛刺。但是如何去除毛刺，作者完全没有提及。这给我们造成了不少困扰，也是误警率比较高的有一个原因。我们尝试过检测骨架的端点，然后对骨架端点进行“擦除”，发现效果一般，而且有的时候甚至破坏了文字区域。

2.求得图像的骨架后，然后断开骨架，怎么从骨架恢复成连通分量。

这又是一个很大的问题。作者在这里也没有提及。这个也困惑了我们很久，在请教了马竞学长，以及别的小组的同学之后。我们使用MATLAB自带的bwdist函数求骨架到连通分量边界的距离，基于这个距离，对骨架中的每一个像素使用ploy2mask函数做一个类似膨胀运算（实际上不是）的操作。这样就完成了从骨架恢复成连通分量的过程。

这里我们还使用了MATLAB自带的bwlabel函数，对于断开交点后的骨架，检测出所有8连通的骨架，然后对每一个骨架进行上述所提及的骨架恢复成连通分量的操作。

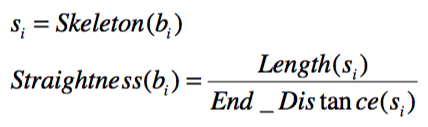
连通分量分割以及骨架恢复成连通分量，在个人看来是极为关键的一步。作者却寥寥数语，没提及具体实施细节。使得我们采用了自己主观猜想的一些方法去实现，这也导致了我们算法的执行时间有点慢，大约80%的时间都耗费在这里，我们小组也尝试过给作者发邮件请教这个问题，但作者没有回复。

上述这个过程在skel2cc.m文件中执行。

1. **错误正样本剔除**

通过上一步，我们有很多简单的连通分量（复杂的连通分量以及被分割为简单的了）根据两个准侧：

1. 直线度，计算公式如下：



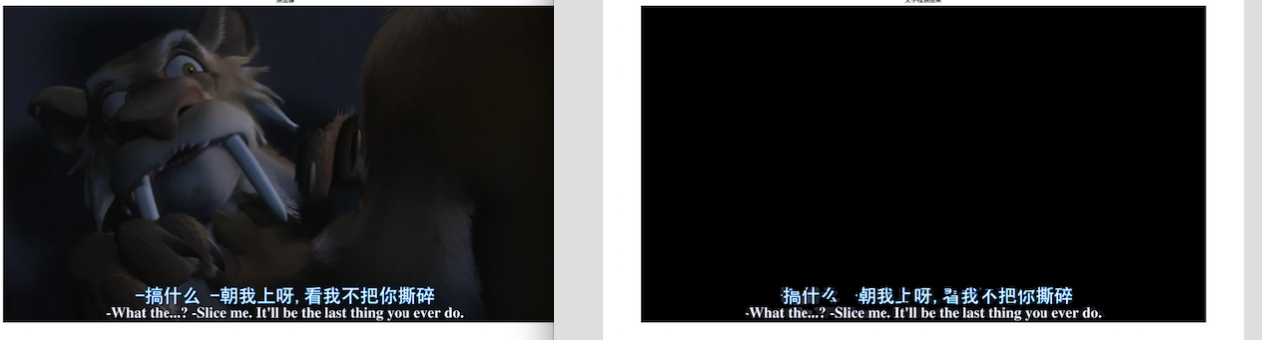
1. 边缘密度，计算公式如下：



bi表示简单的连通分量，根据上述公式，进行计算，阈值T1 ，T2为1.2和0.1。只要两个条件满足一个，就认为是文字块。

上述这个过程在skel2cc.m文件中执行。

到这里，作者论文里所提及的算法就结束了，我们按上面的步骤实现了算法之后，对于其中一张图片，得到了如下结果：



正如前面所提及的，作者采用了一中独一无二的展示文字区域的方式——基于连通分量的方式。而非传统的bounding box。

这样就使得传统的评测手段不太适合作者的方法，于是作者提出了自己的评价方法。这里我们不列出具体评价方法（具体方法可以参考论文，或者论文对应的PPT-第13张），因为我们最后采用的方法还是传统的评价方法，即根据重合区域的面积来判断是否检测正确。

1. **文字边界的确定**

我们检测每一个连通分量检测左右上下边界，将其作为定义为文字区域，并把文字区域的像素标记为红色。利用bounding box作为检测手段，导致了我们的文字检测只能基于检测水平方向的文字了，其实这也有些失去了作者论文的意义了。因为作者这篇论文的目的就是检测多方向的文字。

这个过程在text\_detect.m文件中执行。框定文字区域之后，我们测算了一些图片，发现检测效果很差，误警率很高，至于原因，我们在前面也提及了：

一方面是因为拉普拉斯作为二阶微分算子对细节相应敏感，这样可以检测出低对比度的文字，但这也使得误警率很高。

第二个原因是骨架分割产生很多的毛刺。同时骨架恢复成连通分量的具体算法作者也没有提及。这也造成了误警率高的情况

于是我们小组经过讨论，提出了一些去除错误文字的方法:

1.根据文字块的宽高比信息，宽高比小于2就剔除，这个方法很直观，对于一个文字块，也就是一句话，通常可以认为宽高比要大于2

2.对于连通分量像素个数太少的团块，我们也做了去除。这个方法也很直观。

上述过程在Eliminate\_false中执行。

经过上述过程，代码使用的算法以及全部结束。但是检测效果依然不好，但是由于时间关系，很多工作我们只能放弃了。其实我们加的这些工作就有些杯水车薪的感觉，作者提出的这个方法本身的特性就无法避免程序的高误警率，这个作者在论文中也承认了这个事实。

# **数据的标定方法以及结果评价方法**

1. **数据的标定方法**

我们使用imcrop函数进行文字标定，这个函数接受图片作为参数。返回截取图像的左上角的坐标，宽度，高度。设置想要标定的数据集，并且设置想要保存的文件名。运行程序text\_label.m，即可进行标定，如下所示：

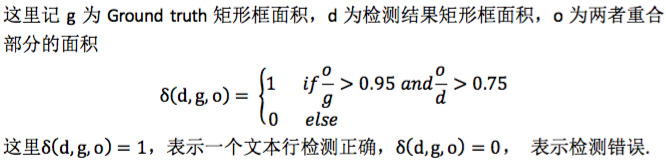


拖动鼠标，调整方框的位置，选取合适的位置后点击crop Image,即完成一个文字块的标定。之后再次拖动鼠标就可以标定下一个文字块，在这里，为了加快标记效率，由于大部分图片只有两个文字块，所以对于每张图片，标记两次之后，自动跳到下一张图片的标记（对于有些文件夹文字块比较多的情况下，我们只要适当更改一下参数就行了）。如果点击cancel，则认为图片中没有文字。

1. **结果评价方法**

由于作者使用连通分量来展示文字块，所以作者并没有采用传统的视频文字检测方法，而是自己定义了一系列相关参数。不过在这里，我们依然使用通常用的视频文字检测评价方法，这个方法虽然对于作者这篇文章有失公允，但也不是不能作为评价手段。

文字正确与否的评定方法如下：

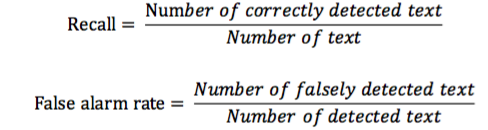


在算法具体实现时，我们采用如下的方法计算重合面积：

标定的文字区域 检测到的文字区域

1. 首先产生两个和图像一样大小的零矩阵
2. 将标定的文字区域以及检测到的文字区域置1
3. 两个矩阵求逻辑与运算，其中矩阵里面非0元素的个数就代表重合面积

经过上述步骤，我们可以求得重合面积，基于重合面积，可以求文字的召回率以及误警率，计算公式如下：



以上过程在batch\_text\_detect.m文件中执行