图像分析与应用 project

叠加文字的检测与分割

小组成员：

雍佳伟（组长）：201528015126035

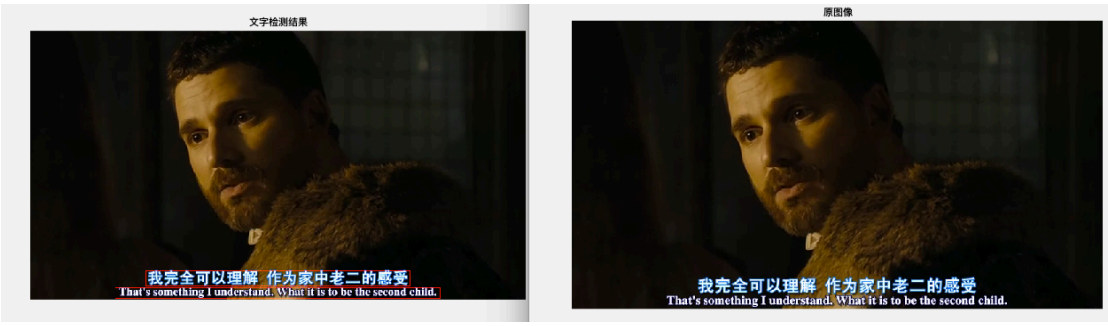
郭晓锋：2015E8015161053

熊安：2015E8007361073

目录

一 程序运行说明

1. 首先解压文件，找到“叠加文字的检测与分割”目录下的“代码部分”文件夹
2. 找到 text_detect.m 文件，这个文件可以进行单张图片检测，更改图片所在路径（为方便老师进行测试，我们在所给 12 个数据集里面各随机选了 1 张，共 12 张图片，作为测试图片）点击运行，即可输出检测结果, 如下所示：



这个函数的返回值是文字区域的二值图像，具体后面关于算法讨论部分在详细陈述。

注意：

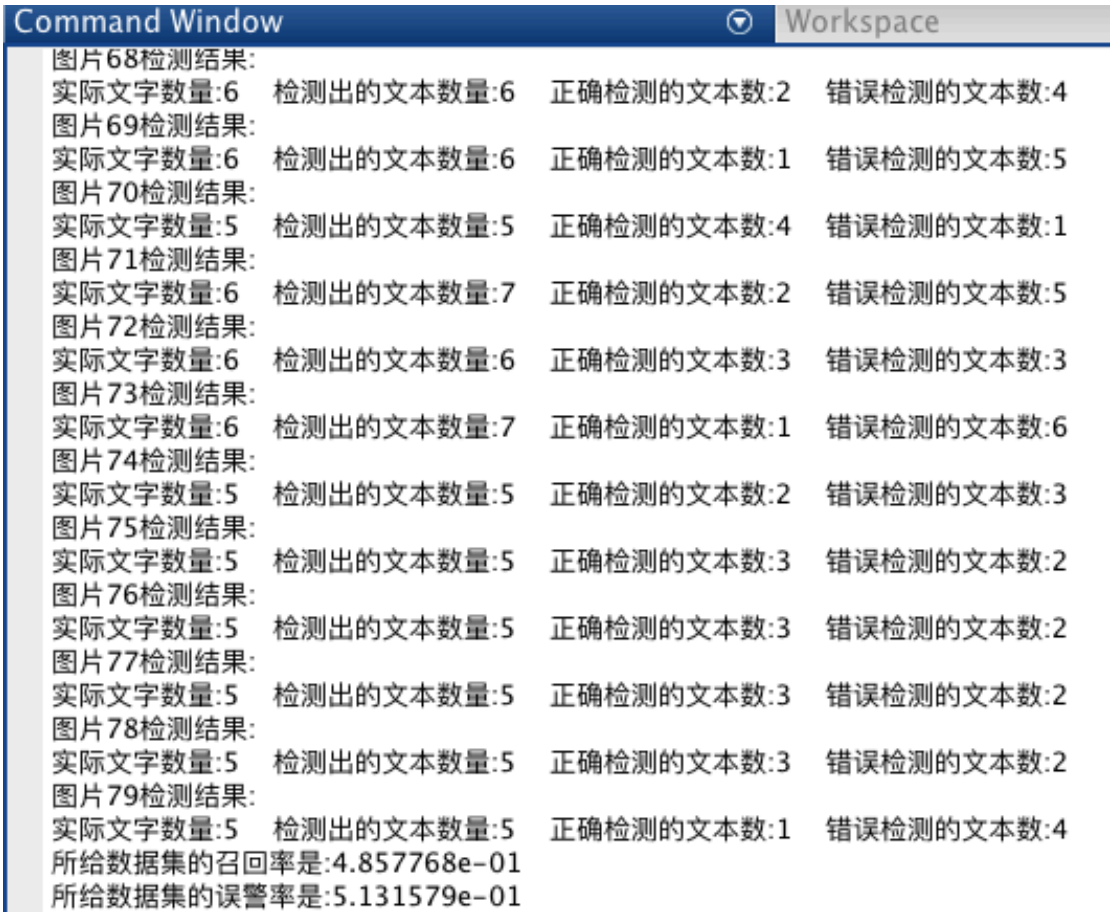
文件 batch_text_detect.m 是批量处理图片的程序。可以自动读取文件夹下的所有图片，然后进行文字检测，并且和标定好的结果进行比对，然后输出召回率和误警率。具体运行方式为，点开这个 m 文件。

a. 更改想要检测图片所在的文件夹
b. 建立一个目录用于存储检测结果的图片
c. 使用 importdata 导入 txt 文件，这里面存储了我们标定好了的数据
d. 设置目录将输出结果存储到刚才建立的目录中。

注意：点击运行时，务必打开 text_detect.m 文件，看看第 16 行读取图片这行代码是否被注释，如果没有注释，务必注释。不然每次从文件夹读取的图片会被这个覆盖。

经过一段时间后（具体时间根据数据集的大小来定，图片越多，自然检测时间越长），程序执行完毕，在命令行输出类似下面的结果（这是第五个文件夹，火线战将的部分输出结果）：

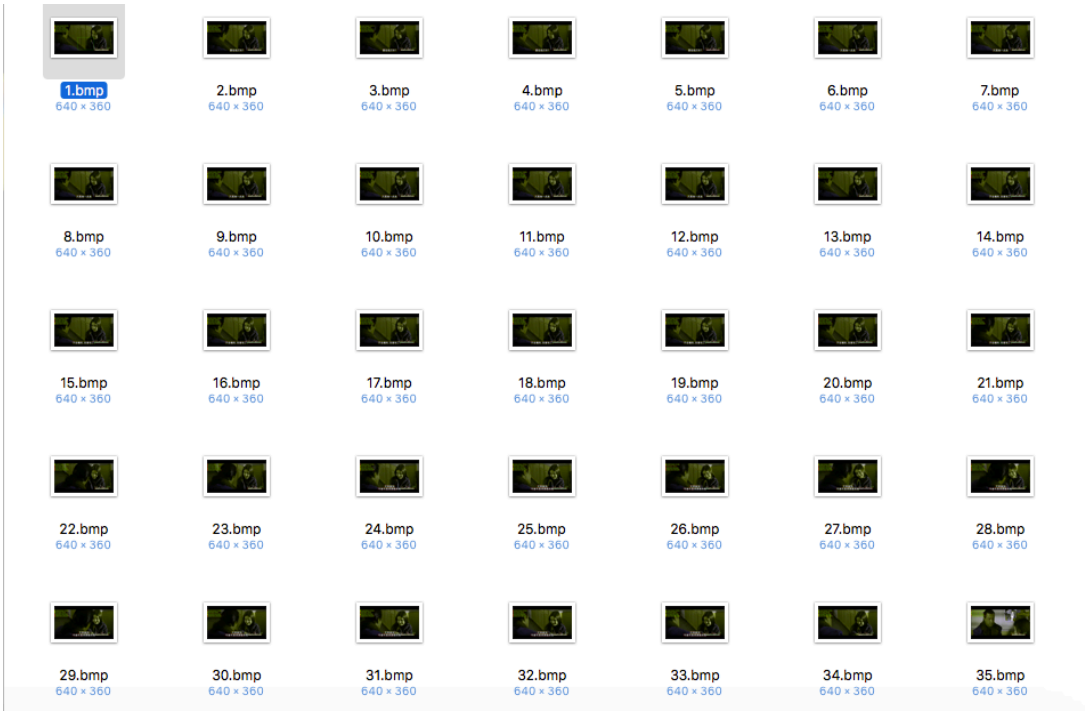

Command Window
Workspace
>> batch_text_detect
Warning: Directory already exists.
> In batch_text_detect110291 (line 5)
图片1检测结果:
实际文字数量:6 检测出的文本数量:5 正确检测的文本数:1 错误检测的文本数:4
图片2检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:2 错误检测的文本数:4
图片3检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:3 错误检测的文本数:3
图片4检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:4 错误检测的文本数:2
图片5检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:4 错误检测的文本数:2
图片6检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:3 错误检测的文本数:3
图片7检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:4 错误检测的文本数:2
图片8检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:5 错误检测的文本数:1
图片9检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:5 错误检测的文本数:1
图片10检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:4 错误检测的文本数:2
图片11检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:5 错误检测的文本数:1
图片12检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:4 错误检测的文本数:2


Command Window
Workspace
图片68检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:2 错误检测的文本数:4
图片69检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:1 错误检测的文本数:5
图片70检测结果:
实际文字数量:5 检测出的文本数量:5 正确检测的文本数:4 错误检测的文本数:1
图片71检测结果:
实际文字数量:6 检测出的文本数量:7 正确检测的文本数:2 错误检测的文本数:5
图片72检测结果:
实际文字数量:6 检测出的文本数量:6 正确检测的文本数:3 错误检测的文本数:3
图片73检测结果:
实际文字数量:6 检测出的文本数量:7 正确检测的文本数:1 错误检测的文本数:6
图片74检测结果:
实际文字数量:5 检测出的文本数量:5 正确检测的文本数:2 错误检测的文本数:3
图片75检测结果:
实际文字数量:5 检测出的文本数量:5 正确检测的文本数:3 错误检测的文本数:2
图片76检测结果:
实际文字数量:5 检测出的文本数量:5 正确检测的文本数:3 错误检测的文本数:2
图片77检测结果:
实际文字数量:5 检测出的文本数量:5 正确检测的文本数:3 错误检测的文本数:2
图片78检测结果:
实际文字数量:5 检测出的文本数量:5 正确检测的文本数:3 错误检测的文本数:2
图片79检测结果:
实际文字数量:5 检测出的文本数量:5 正确检测的文本数:1 错误检测的文本数:4
所给数据集的召回率是:4.857768e-01
所给数据集的误警率是:5.131579e-01

除了在命令行输出如上结果外，检测后的图片也被存入刚才设置的文件夹下，下面这张是图片 77（上面命令行输出结果对应的图片）的检测结果：

图片 77 的检测结果，红色框是检测的结果，蓝色框是标定的结果，图中有部分黄色是因为标定的结果和检测的结果重合在一起，这样就显示出了黄色

文件夹下的其他图片如下所示：



这里面只列出了一部分，详细的情况可以进入文件夹进行查看。

二 程序的算法说明

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。具体出现了哪些问题我们会在下一个部分，数据统计和结果分析部分进行分析。

这些问题给我们小组的成员造成了一种骑虎难下的情况，如果我们舍弃作者的方法重新采用别的方法，时间是上似乎来不及。如果不舍弃，误警率又很高。最终我们小组讨论之后，加了一些去噪方法，去除一些非文字的区域。稍微降低了一下误警率。

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：

(a) Input

(b) Fourier-Laplacian filtered

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：

MD map of Fig. 4b

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：

Text cluster after *opening*

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。

2. 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。

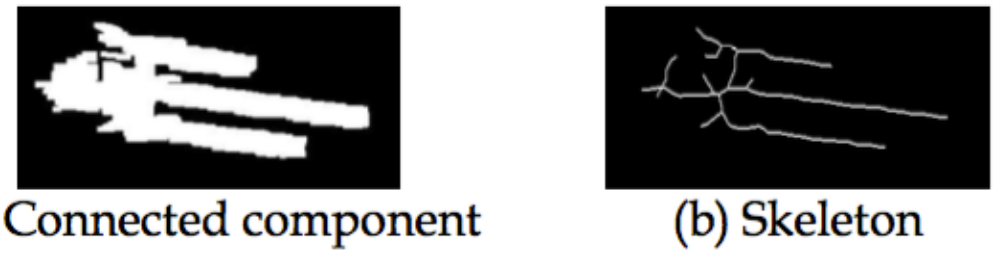
个人觉得这是本文的一个唯一亮点吧，抛开作者采用的方法的检测效果不谈。作者的这种使用连通分量来展示文字行的方法是少数可以用于多方向检测的方法，传统上的方法大都是基于水平，竖直方向，或者斜 45° 方向检测。但作者这个方法就突破了这个限制。下面就具体思路进行说明：

作者认为，连通分量分成简单的和复杂两类：

A simple CC is either a single text string or a false positive

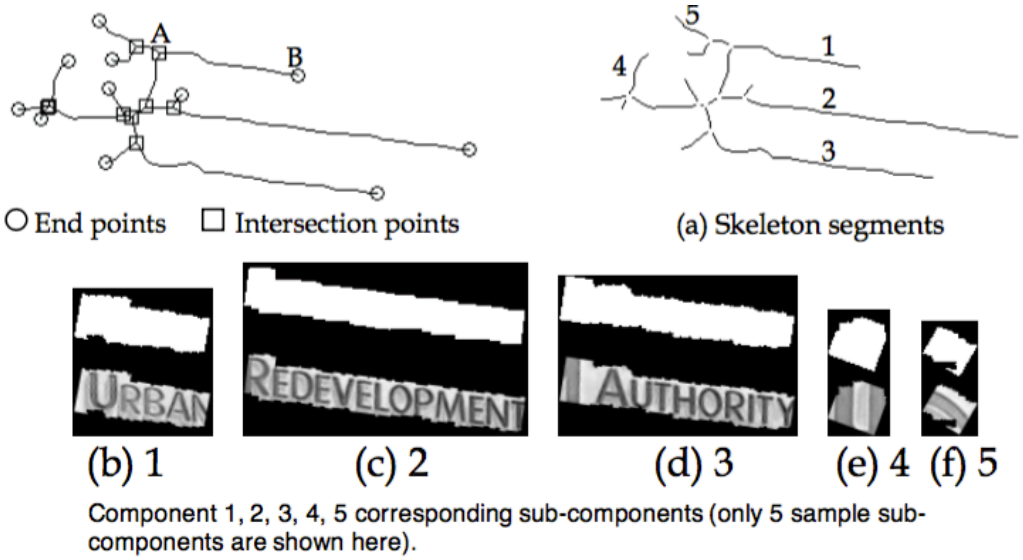
A complex CC contains multiple text strings which are connected to each other and to false positives in the background.

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：



3. 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：



Component 1, 2, 3, 4, 5 corresponding sub-components (only 5 sample sub-components are shown here).

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。

4. 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：

$$s_i = Skeleton(b_i)$$

$$Straightness(b_i) = \frac{Length(s_i)}{End\_Distance(s_i)}$$

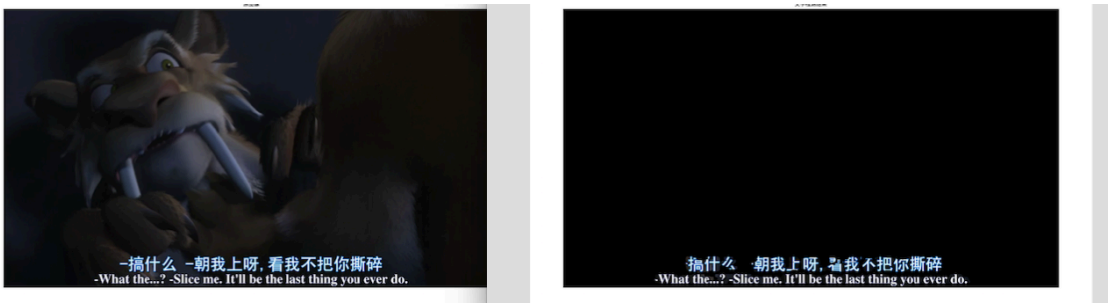
2. 边缘密度，计算公式如下：

$$Edge\_Density(b_i) \geq T_2$$

b_i 表示简单的连通分量，根据上述公式，进行计算，阈值 T_1 ，T_2 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。

5. 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。

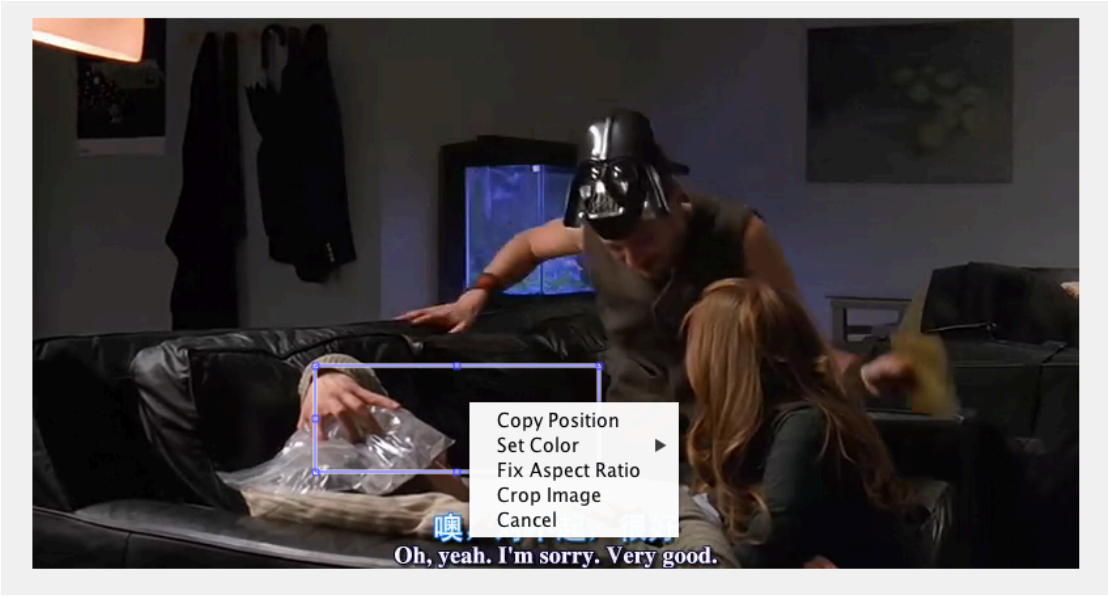
上述过程在 Eliminate_false 中执行。

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。

三 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。

2. 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：

这里记 g 为 Ground truth 矩形框面积，d 为检测结果矩形框面积，o 为两者重合部分的面积

$$\delta(d,g,o)=\begin{cases}1 & if\frac{o}{g}>0.95\ and\frac{o}{d}>0.75\\ 0 & else\end{cases}$$

这里$\delta(d,g,o)=1$，表示一个文本行检测正确，$\delta(d,g,o)=0$， 表示检测错误.

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

a. 首先产生两个和图像一样大小的零矩阵
b. 将标定的文字区域以及检测到的文字区域置 1
c. 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：

$$\text{Recall}=\frac{Number\ of\ correctly\ detected\ text}{Number\ of\ text}$$

$$\text{False alarm rate}=\frac{Number\ of\ falsely\ detected\ text}{Number\ of\ detected\ text}$$

以上过程在 batch_text_detect.m 文件中执行

四 结果统计以及分析

1.单幅图片结果统计分析

对所给的 12 个文件夹里面的数据集分别进行检测。命令行将输出结果，并且在相应目录生成标定好的图片，下面是文件夹 8 其中一部分图片，绿巨人大战雷神托尔在命令行的输出结果（限于篇幅，这里我们这里列出一部分，别的点开结果统计文件夹即可查看）：

图片1检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片2检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片3检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片4检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片5检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片6检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片7检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片8检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片9检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片10检测结果:

实际文字数量:0　　检测出的文本数量:0　　正确检测的文本数:0　　错误检测的文本数:0

图片11检测结果:

实际文字数量:0　　检测出的文本数量:1　　正确检测的文本数:0　　错误检测的文本数:1

图片12检测结果:

实际文字数量:2　　检测出的文本数量:3　　正确检测的文本数:2　　错误检测的文本数:1

图片13检测结果:

实际文字数量:2　　检测出的文本数量:7　　正确检测的文本数:1　　错误检测的文本数:6

图片14检测结果:

实际文字数量:2　　检测出的文本数量:5　　正确检测的文本数:2　　错误检测的文本数:3

图片15检测结果:

实际文字数量:2　　检测出的文本数量:3　　正确检测的文本数:2　　错误检测的文本数:1

……

图片77检测结果:

实际文字数量:2　　检测出的文本数量:2　　正确检测的文本数:0　　错误检测的文本数:2

图片78检测结果:

实际文字数量:2　　检测出的文本数量:2　　正确检测的文本数:0　　错误检测的文本数:2

所给数据集的召回率是:4.298246e-01

所给数据集的误警率是:7.151163e-01

这里的图片 1.2.3…77.78 对于的图片在文件夹 8_result 中，由于总共 12 个数据集，原图片以及结果，加起来约 2G，所有我们只上传了部分图片。

打开文件夹 8_result 中的图片 78.bmp,我们可以看到如下这张图片：

图片中红色框是我们程序的检测结果，绿色框是手工标出的结果。图中的出现了黄色是因为红色和绿色框重合了，即标定结果与检测结果重合了。

根据命令行的结果，我们可以知道：

图片78检测结果:

实际文字数量:2　　检测出的文本数量:2　　正确检测的文本数:0　　错误检测的文本数:2

我们的检测结果（红色框）的左右边界偏宽，但是上下边界很紧凑，很好的贴近了文字，甚至比标定的结果还要好。但这里计算了重合面积，并利用老师说给的判断检测成功的规则进行了判定之后。程序判定上述图片文字检测失败。究其原因，是因为判定公式：

$$\delta(d,g,o)=\begin{cases}1 & if\frac{o}{g}>0.95\ and\frac{o}{d}>0.75\\ 0 & else\end{cases}$$

上述判定公式中的第二项比较好满足，但是第一项：重合面积占标记文字区域的面积大于 0.95 这个要求相对较为严格。也就是说检测的文本区域可以宜大不宜小，大一点点没关系，大一点可以确保检测的文本区域包括标定文字区域，这样就更容易满足第一个条件，因为第二个条件更加宽松，所以就算偏大一点也没关系，第二个条件没那么容易不满足。

基于此，如果我们组对检测后的图像做一次膨胀的话，也就是把检测的框画大一点，是可以提高正确率的。从另一个角度看，如果我们标记文字数据的时候把框尽量画紧凑一点，偏小一点，那么上述图片的检测结果就会被判定为检测成功，这样也可以提高正确率。

不过上述提高正确率的方法都不是真正有意义的，有理由，事实去支撑的“改进”。所以我们并没有去做上述尝试，而是把真是结果呈现出来！

2.所有图片的结果统计分析

下面给出各个文件夹的统计结果分析：

图片集 1-12 分别代表如下文件夹的结果（下面这些文件夹名字前的数字 1-12 是我们为了方便标上去的，数字后面的代表老师说给数据集）：

1_冰河世纪 1　　2_非常人贩 2A　　3_孤岛惊魂　　4_怪物史瑞克

5_火线战将　　6_见龙卸甲　　7_紧急迫降　　8_绿巨人大战雷神托尔

9_马达加斯加 2_逃往非洲　　10_潘神的迷宫-cd1

11_王室双姝12_先婚后友

将\结果统计 文件夹下的文件中的结果做一个统计，得到下表：

参数 图片集	召回率(Recall)	误警率(False alarm rate)
1	1.896552e-01	8.562092e-01
2	3.113208e-01	7.105263e-01
3	1.578947e-01	9.333333e-01
4	3.797468e-01	7.887324e-01
5	4.857768e-01	5.131579e-01
6	9.090909e-02	9.672131e-01
7	2.162162e-01	7.500000e-01
8	4.298246e-01	7.151163e-01

9	5.882353e-02	9.502762e-01
10	1.400000e-01	9.606742e-01
11	2.380952e-02	9.821429e-01
12	2.317073e-01	8.318584e-01

标记为蓝色的第 5 个文件夹表示检测效果最好的情况，Recall 为 48.6%.

我们重点关注检测最差的那个文件夹 11，正确率只有 2%，打开检测结果所在的文件夹 2_result。下面是其中一幅图片：

和前面提及的情况一样，如果我们标记的时候，标记的文字尽量标记的紧凑一些，或者对检测的结果适当往外延宽是完全可以提高正确率的。这个文件夹中的其余图片大都如此。所以这个检测结果也在情理之中。只要这个数据集文字表定稍微注意一下技巧，正确率完全可以达到其他文件夹的水平。

其他文件夹也存在这个问题，这里就不在赘述。至于误警率很高的原因，前面也提及多次，不再赘述。

3. 检测时间

在Intel Core i7，主频 2.9 GHz的macbook pro上测试:
1024×576（取自冰河世纪）大小的图片运行约45秒
640×360（取自非常人贩2A）大小的图片运行约12秒
这个和作者在论文中声称的使用Core 2 Duo 2.0 GHz机器的处理256×256时间为7.8秒相差不大。批量处理图片的时间根据文件夹里面图片数量的不等，时间也不尽相同。

五 总结及感悟，体会

1.算法总结

整个过程没有什么很高深的算法，但是作者隐藏了大量细节，有时甚至让我们怀疑其论文中数据的真实性。虽然我们已经努力还原了作者的算法，但检测结果依旧不理想。总结起来有如下原因：

a. 作者算法本身的缺陷,二阶微分算子对细节响应很强，必然导致高的误警率。另一方面，求连通分量的骨架的时候，必然导致有很多毛刺。这些问题的处理作者只字未提
b. 文字标记的时候，稍微一不小心标记宽了，就会被判断为检测失败。
c. 作者很多实现细节的隐藏

2.未来展望

关于这个 project，我们小组认为一个更加可行的方案是广读文献，开拓思路。因为作者这个方法本身是有些固有的缺陷的，但是其用连通分量来展示文字区域，并且提出的多方向文字检测的思路是值得借鉴的。所以我们考虑的改进思路是加入一些去噪的方法，去完善这个工作。

3.感悟及体会

由于我们小组的三名成员都是工科物理背景，没有一个学计算机的，大一学了 C 语言之后基本就告别了编程。所以在作业完成起来还是花了一些时间和走了一些弯路的。除了编程方面有一些小小的提升之外，我想我们收获更多的是一种做事情方法，还有一种批判和质疑的精神。这种质疑精神也是在上课讨论中也感受颇深。

第一，关于科研中做事情的习惯。在以后的科研工作中，要时刻警醒自己要保持严谨的态度，时刻用数据去说话。从算法的提出，执行，验证，得到结果后在分析，再度改进，这是一个不断前进的过程。

第二，保持质疑精神，论文中的方法很多时候只是一家之言，仅供参考。一定要经过自己的思考，过滤。要广度文章，拓展自己的视野，才不会局限于一个思路之中。

六 小组成员分工

小组成员分工如下：

程序编写：郭晓锋(60%)，雍佳伟(20%),熊安(20%)

算法讨论及分析：郭晓锋(40%)，雍佳伟(30%)，熊安(30%)

数据标定：雍佳伟(40%)，熊安(40%)，郭晓锋(20%)

程序相关文档，说明整理：郭晓锋(60%)，雍佳伟(20%)，熊安(20%)

七 参考文献

1. Shivakumara P, Quyphan T, Limtan C. “A Laplacian Approach to Multi-Oriented Text Detection in Video”[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(2): 412-419.

2. T.Q. Phan, P. Shivakumara and C.L. Tan, “A Laplacian Method for Video Text Detection”, In Proc. ICDAR 2009, pp. 66-70.

3. Kyushu University, Seiichi Uchida, “Text Localization and Recognition in Images and Video”

4. 付立波,复杂背景图像中的叠加文字提取技术研究

5. 詹耀文,图像和视频中叠加文字提取算法的研究与应用

6. Rafael C. Gonzalez “Digital Image Processing Using MATLAB” 2004.5

7. Rafael C. Gonzalez “Digital Image Processing, Third Edition” 2011.6