# 程序具体实现

概要

这个程序主要是将姚迪图像优化和兴华的图像识别部分进行整合，而整合过程中我们需要添加一些额外的步骤，比如图片格式转换和噪点的剔除（其实就是对字符串进行处理），然后将结果文件输出到用户指定的目录。

## 以下是具体步骤：

1. 处理输入参数**(这步已经做完)**

程序一共有三个参数

RunOCR.exe <图片路径> <结果输出路径> <-r> （不区分大小写）

图片路径：图片路径可以是一张图片，也可以是一个文件夹，但是只能有一个，不能传入两个图片的路径或者既传入单个图片路径又传入一个文件夹，这样算非法的

结果输出路径：第二个参数是可选的，结果输出路径是将生成的结果文件(.txt)放到这个文件夹中。

-R或者-r: 用户既可以输入-R也可以输入-r,都是一个意思，代表如果第一个参数如果给的是文件夹，那我们会递归查找这个文件夹中的所有子文件夹中所有图片文件，如果不带这个参数，那我只检查当前文件夹；如果第一个参数是单个文件的路径，但它也给第三个参数的话，那么第三个参数忽略。

1. 第二步是姚迪的去噪和优化步骤，这步具体要做的事情，会由姚迪来详细说明
2. 第三步转换格式，将姚迪去噪处理后的.jpg图片转成.tif文件，我们在网上找到一个工具<https://www.cnblogs.com/kekec/archive/2011/12/21/2295154.html>， 部署时我们会把它放到tesseract-ocr的环境变量中，这样我们就可以直接调用它。

具体用法：

C:\Users\Administrator\Desktop>PictureConverter.exe /?

用法示例: PictureConverter d:\picFiles png bmp

参数1:待转图片目录

参数2:源格式

参数3:目标格式

-----------------------

注意: 1.转换后的图片将和原图片放在同一目录下!

2.递归子目录进行图片转换!

那么在我们的程序中，它的第一个参数应该是我们的第一个参数给出的路径，第二个参数应该是jpg, 第三个路径是tif

1. 将第三步转换完成的.tif文件，调用tesseract命令来进行OCR解析**(这步已经做完，但需要更加详细的测试，只做了最基本的测试，能够正常生成.box文件)**

具体的命令为：

Tesseract <tif文件的路径> <tif文件的路径（去掉后缀名）> -l num\_All10000\_Review+num\_25+num\_All10000\_Updated batch.nochop makebox。最后它会在图片的当前路径生成一个.box格式的同名文件，这个文件可以用notepad打开。

例如：

有一张tif图片它的路径是："C:\ 2017112801102909.Tif"，那么这条命令的格式就是：

Tesseract C:\2017112801102909.Tif C:\2017112801102909 -l num\_All10000\_Review+num\_25+num\_All10000\_Updated batch.nochop makebox

注意：每一张tif图片都需要运行一次这个命令

1. 处理第四步生成的.box文件中的数据

如果一个.box文件中的数据如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 50 | 23 | 81 | 81 | 0 |
| - | 87 | 65 | 91 | 67 | 0 |
| 8 | 94 | 27 | 149 | 85 | 0 |
| 0 | 144 | 28 | 193 | 75 | 0 |
| - | 310 | 48 | 341 | 52 | 0 |
| 3 | 381 | 17 | 422 | 80 | 0 |
| 7 | 439 | 21 | 449 | 74 | 0 |
| 4 | 469 | 21 | 520 | 75 | 0 |
| 0 | 536 | 35 | 558 | 72 | 0 |
| - | 647 | 48 | 678 | 52 | 0 |
| 6 | 736 | 24 | 768 | 80 | 0 |
| 0 | 772 | 22 | 810 | 63 | 0 |
| 6 | 827 | 18 | 867 | 83 | 0 |
| 5 | 873 | 17 | 934 | 80 | 0 |

我们以第一行为例，设a=0, b=50, c=23, d=81, e=81, f=0, 那么有以下几种情况我们需要删除某条数据：

1. (d - b) \* (e - c) < 65
2. (d - b) > 300
3. d == 988
4. b == 0

如果有一条数据满足这四个条件中的一种，那么我们就需要输出这条数据，在上面的例子中， 第二条数据就需要删掉。

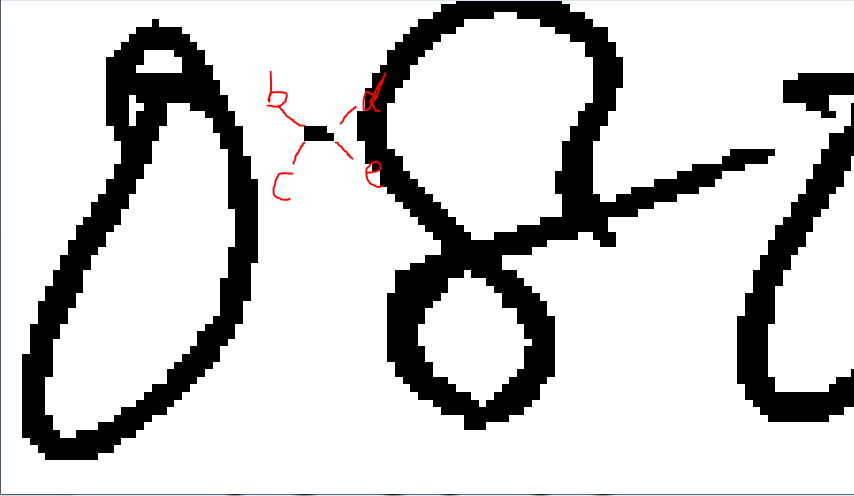
Notes:

以上数据中第一列a代表识别出的字符；第二列代表这个字符左上x轴的坐标，第三列代表左下y轴的坐标，第四列代表右上x轴的坐标，第五列代表右下y轴的坐标，第六列代表页码(可以忽略)。

例如下面这张tif图片是这样的：

C:\Users\Administrator\Desktop\Test_Image - 副本\2017112801102909.Tif

放大后：



1. 对第五步处理过的数据进行再处理：

通过每条数据的第二列，也就是b列对数据进行排序

例如：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 50 | 23 | 81 | 81 | 0 |
| - | 97 | 65 | 91 | 67 | 0 |
| 8 | 94 | 27 | 149 | 85 | 0 |
| 0 | 144 | 28 | 193 | 75 | 0 |
| - | 310 | 48 | 341 | 52 | 0 |
| 3 | 878 | 17 | 422 | 80 | 0 |
| 7 | 900 | 21 | 449 | 74 | 0 |
| 4 | 921 | 21 | 520 | 75 | 0 |
| 0 | 945 | 35 | 558 | 72 | 0 |
| - | 647 | 48 | 678 | 52 | 0 |
| 6 | 736 | 24 | 768 | 80 | 0 |
| 0 | 772 | 22 | 810 | 63 | 0 |
| 6 | 827 | 18 | 867 | 83 | 0 |
| 5 | 873 | 17 | 934 | 80 | 0 |

上面这个数据，第六条到第九条数据中的第二列数值是要比第十条到第十四条数据大，那么排序后的结果应该是：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 50 | 23 | 81 | 81 | 0 |
| - | 97 | 65 | 91 | 67 | 0 |
| 8 | 94 | 27 | 149 | 85 | 0 |
| 0 | 144 | 28 | 193 | 75 | 0 |
| - | 310 | 48 | 341 | 52 | 0 |
| - | 647 | 48 | 678 | 52 | 0 |
| 6 | 736 | 24 | 768 | 80 | 0 |
| 0 | 772 | 22 | 810 | 63 | 0 |
| 6 | 827 | 18 | 867 | 83 | 0 |
| 5 | 873 | 17 | 934 | 80 | 0 |
| 3 | 878 | 17 | 422 | 80 | 0 |
| 7 | 900 | 21 | 449 | 74 | 0 |
| 4 | 921 | 21 | 520 | 75 | 0 |
| 0 | 945 | 35 | 558 | 72 | 0 |

1. 第六步排序后，我们只取这块数据的第一列写入到用户指定文件夹中，文件名与图片名同名，后缀名为.txt。

例如：如果一个图片名为2017112809100002.Tif，那么这个结果文件名字就是2017112809100002.txt

1. 删除用户给定的第一个参数目录中我们产生的中间过程文件，比如.box文件和.tif文件。

程序的注释中也是分为8步，和本文档中这八个步骤是一样的，这篇文档是程序注释步骤中的扩展和详细描述。