清华大学《模式识别》

数字字符识别：模板匹配

郭一隆

April 7, 2018

1 问题描述

- 提供数据：本次作业提供两组数据，分别存放在 `train` 文件夹和 `test` 文件夹中。
  - `train` 文件夹中有已单独分割出来的 `0-9` 数字图像模板。如图1所示，模板已统一到相同的尺度，每个模板为对应数字的外切分割结果。

Figure 1: 数字图像模板

  - `test` 文件夹中有 8 张用于测试的图像，其中 6 张正常尺度（与模板尺度相同），1 张存在划痕，1 张有噪声。
- 请利用给出的图像模板对测试图像进行数字字符识别。
- 补充题：
  - `test` 文件夹中还有 2 张图像进行过放缩处理（尺度与模板不一致），有兴趣的同学可以思考如何对这 2 张图像进行数字字符识别。

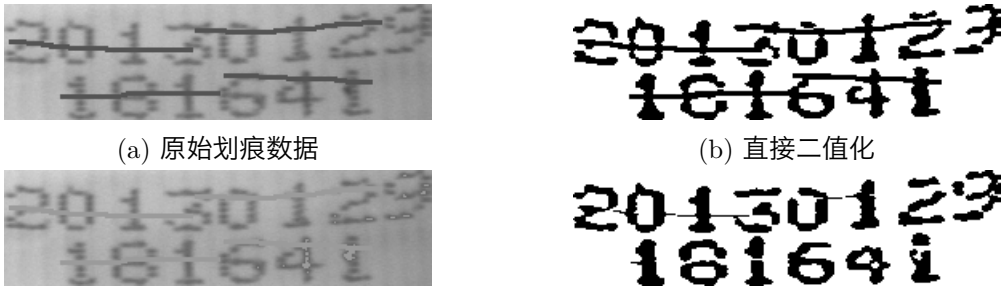
2 问题分析及实现思路

给定的测试数据集包括了**正常数据**、**尺度放缩数据**、**划痕数据**以及**富噪声数据**，欲搭建一个能较为通用地处理以上四种数据的模式识别系统。

2.1 预处理

给定的数字图像模板是灰度图像，测试图像是 RGB 图像，可将模板和测试图像均使用二值化处理，以获得更清晰的特征。严格来说，二值化处理获得二值化图像实际上是提取了一种特征，而二值化特征对于字符识别非常有效。

- 划痕数据的处理：直接进行二值化操作很容易将划痕也作为字符的一部分，二值化为黑色像素，如图2(a)(b)，对后续识别的准确度会产生较大影响。



(a) 原始划痕数据　(b) 直接二值化

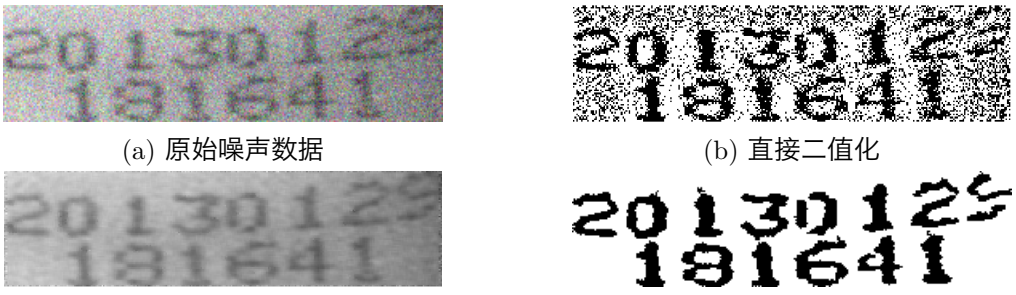
(c) 削弱划痕　(d) 削弱划痕后二值化

Figure 2: 划痕数据的预处理

仔细观察原始划痕数据可发现，图像中的划痕人为制造痕迹非常明显：划痕区域的灰度值基本是完全一致的。进一步观察可以发现：**背景最亮，字符略暗，划痕最暗。**因此通过灰度阈值即可确定划痕的位置，替换成背景的平均灰度后如图2(c) 所示。

尽管人眼仍能感觉到明显的划痕痕迹，但由于预处理后的划痕颜色更接近背景颜色，二值化可以较好地消除划痕，如图2(d)。

- 噪声数据的处理：噪声数据中的噪声是较简单的白噪声，先进行简单的噪声滤波（采用了 `wiener2` 滤波器）即可，效果如图3。



(a) 原始噪声数据　(b) 直接二值化

(c) wiener 滤波去噪　(d) 去噪后二值化

Figure 3: 噪声数据的预处理

- 尺度放缩数据暂不进行特殊预处理。

2.2 图像分割

测试数据中大部分图像中字符的尺寸与模板尺寸是一致的，因此直观的思路是直接拿模板在整个测试图像上做全局滑动相关，设定阈值寻找匹配的窗口位置。但考虑到测试数据中存在尺寸放缩的数据，而且在真正实际应用中测试数据也不可能在尺寸上完美贴合模板尺寸，所以应该用不同尺度的模板进行全局滑动相关。这样一来不仅计算量较大，而且也不能保证较好的识别率。

考虑到识别任务（数字字符识别）的一个重要特点：图像中各数字字符在空间上的排列具有较强的**规则性**和**周期性**。因此可以先将测试图像分割至一个个数字字符的子图，再与各模板进行匹配。如果切割效果好，那么匹配准确度会大幅上升。进一步地，应该考虑将每个子图切割至与数字字符基本相切（模板具有相切性）。

采用切割方法的另一个优点是能够处理各尺度的测试图像：在切割为相切子图的情形下，再将子图和模板 `resize` 到同一尺寸，基本可以保证两图中的数字字符大小相差无几。

当然这一切的前提是切割能达到很好的相切性，以下介绍优化切割过程的思路。

2.2.1 行切割

按照正常的书写习惯逻辑（逐行书写），应首先将待测数据切割成若干行。而分行依据的基本特征为：

- 空行前一行黑色像素较多，空行黑色像素骤降；
- 空行后一行黑色像素较多。

综合来看，可以依据每一行黑白像素比例的极值点进行行切割。

2.2.2 列切割

对于切割好的每一行，再按照同样的原理（黑白像素比例的极值点）来进行列切割。

2.2.3 块修剪

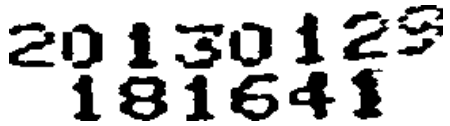
通过调整行切割、列切割的参数（极值特征判定）可将测试图像大体切割成若干块，每个块**至多包含一个数字字符**。

但粗分割远远达不到用于与模板进行匹配的程度，需进行进一步块修剪（细分割）来达到块中**有且只有一个相切数字字符的效果**。

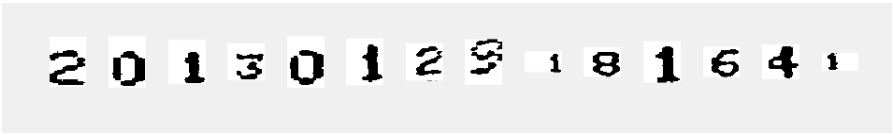
从以下几个方面进行块修剪：

- 舍弃过小的块：宽度过小或高度过小的块，包含的可识别信息很少，设定宽度、高度阈值直接舍弃；
- 过滤块中小连通分量：这部分小连通分量可能是（1）噪点（2）相邻数字字符的一小部分（粗切割不准确），应该去除以免影响细切割的相切效果；
- 去除白边以达到相切。

经过粗分割、细分割，测试图像应被分割为若干相切字符的小块，典型结果如图4、图5所示。



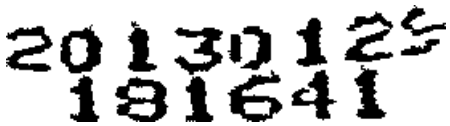
(a) 二值化 1.bmp



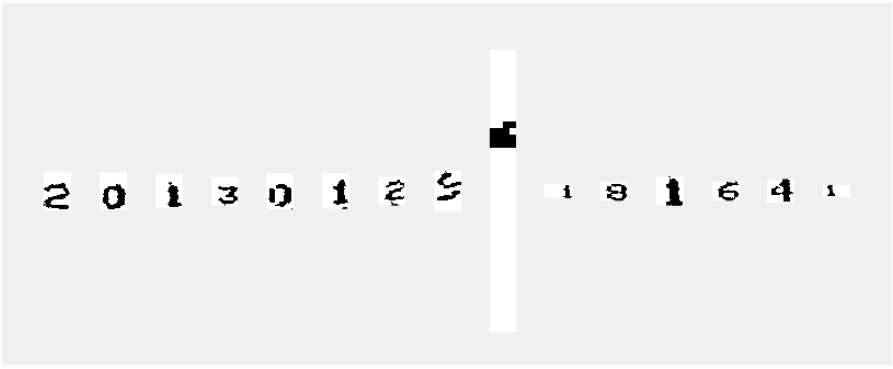
(b) 1.bmp 粗分割

(c) 1.bmp 细分割

Figure 4: 1.bmp 切割过程



(a) 二值化 噪声.bmp



(b) 噪声.bmp 粗分割

(c) 噪声.bmp 细分割

Figure 5: 噪声.bmp 切割过程

2.3 模板匹配

经过细切割得到的图像块与模板具有相似的相切性，但两者的尺寸仍然不尽相同。只需将两者归一化 `resize` 到同一尺寸，由相切性即可保证其包含的数字字符大小相差无几。

尽管都具有相切性，但数字字符与边框的切点仍很难保证一致（切割损失、测试图像的略微旋转形变），因此利用滑动求相关（`xcorr2`）来对两图像块进行匹配，以提高鲁棒性（相对于直接做内积）。

对于每个细切割图像块，与所有模板进行匹配，将相关系数最高的模板作为该图像块的匹配结果。

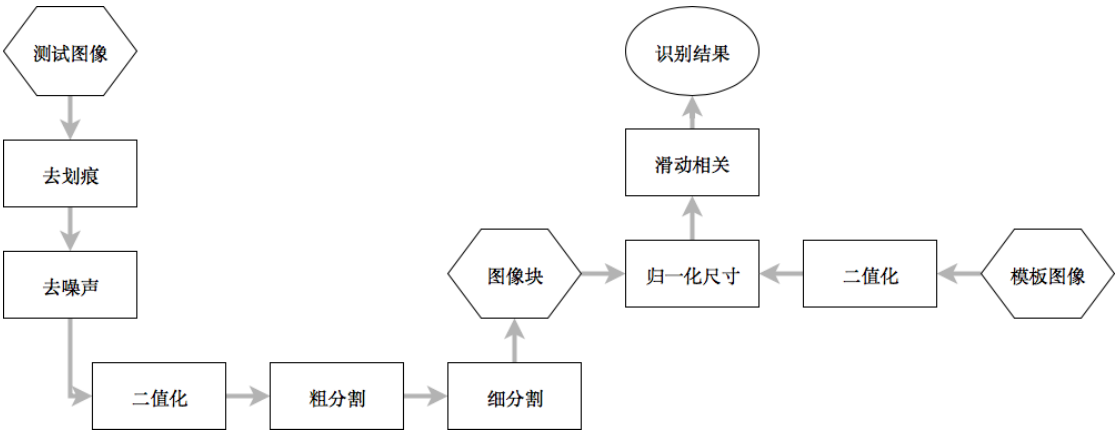


Figure 6: 识别过程流程图

3 实验结果

- 源码仓库链接:

  https://github.com/Nuullll/assignment-simple-ocr

- 运行环境:

  ```
  MATLAB 版本: 9.1.0.441655 (R2016b)
  MATLAB 许可证编号: 1114839
  操作系统: Mac OS X Version: 10.13.4 Build: 17E199
  Java 版本: Java 1.7.0_75-b13 with Oracle Corporation Java
      HotSpot(TM) 64-Bit Server VM mixed mode
  ```

- 运行方法:

  直接运行 `run.m` 即可。

结果如图7。

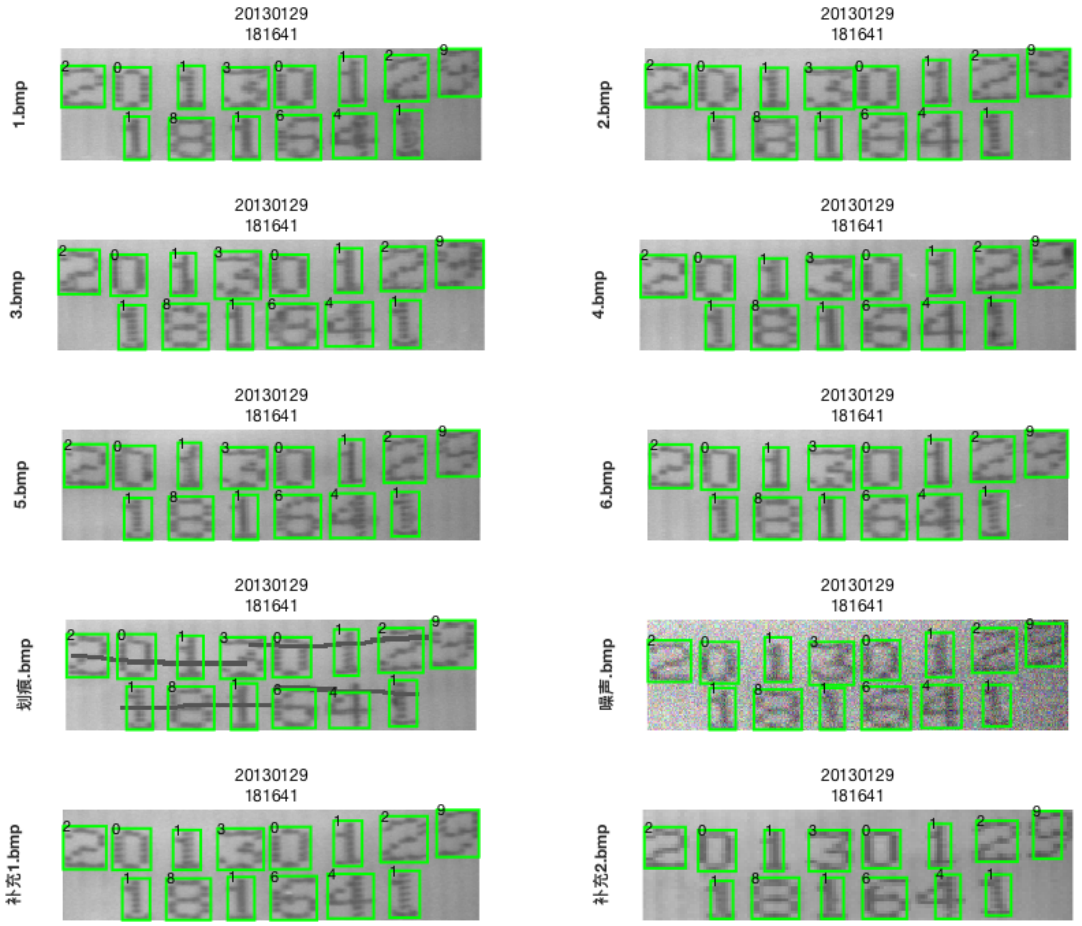


Figure 7: 识别结果

$$TN = TP = 1$$

$$FN = FP = 0$$

4 结果分析及总结

4.1 结果分析

基于图像分割的方法对于本作业所给的数字字符测试图像较为有效，达到了全部正确的识别效果，并且能够直接识别出各数字字符的空间位置逻辑关系，进一步得到全图的识别字符串 `20130129 181641`。

4.2 总结

尽管最终达到了准确无误的识别效果，但调试过程并不是一帆风顺的，调试过程还是收获挺多的。

- 一开始并没有直接处理划痕的问题，因为直观感觉划痕对识别结果应该影响不大，在未去除划痕的条件下，准确率下降至 $137/140 \approx 97.9\%$。如果不处理划痕，那么二值化后划痕与数字字符的像素变得不可区分，使得细切割最终相切于划痕，而不是数字字符。这直接导致了 `resize` 后切割图像块中的数字字符比模板中偏小，导致了误识。幸运的是，观察原图发现划痕的特殊性太明显了，简单的像素值替换就可以极大地削弱划痕。在实际应用中，类似于划痕的干扰因素应该也很多，显然不能简单地用本报告提到的方法进行处理，则需要更精细、更能区分人为划痕与数字字符的特征。

- `wiener` 滤波器对本测试图像中的白噪声具有很好的去除作用，但注意本身每个字符的尺寸都不大，`wiener` 滤波的窗口也不能太大，否则会破坏数字字符的结构。`wiener` 滤波一定程度上会加粗原本的数字字符区域，这对于识别来说可能是有益的。

- 在对噪声图像的预处理过程中，实际上还加入了**去除小连通分量**的步骤，能够更彻底地去除噪点。但对于**小连通分量**的界定也不容易，(1) 不能用**固定的**面积值判定，因为有不同尺寸的测试图像，要用**归一化的**面积值；(2) 阈值设过高可能会把数字字符中因种种原因（如去除划痕）而断开的连通分量去除，导致识别率下降。

- 本报告中所实现的粗切割其实还非常不成熟，在原本给定测试集全部正确识别后，我曾尝试自己手写另外的测试图像，但并不能良好地识别。原因是所实现的粗切割算法鲁棒性不足，对空行的判定受测试图像数字字符的**工整度**影响较大，如果测试数据中行列对齐地不好，很有可能在粗切割就出现问题，可能出现的最大问题即把两个甚至更多字符切割至一个子块内，后续对此子块的识别基本是无用功了。基于切割的方法对切割效果依赖性太强，而缩放模板全局滑动在这方面则略有优势。

- 在实现细切割的**去除白边**环节时，最终采取了较**贪婪**的白边检测：即把含有少量黑色像素的行（列）也判定为白边。尽管可能在切割时丢掉一点数字字符的**本体**，但对于**毛刺的去除**非常有效。