清华大学《模式识别》

# 数字字符识别：特征设计

郭一隆

April 7, 2018

## 1 问题描述

- 在第一次作业的基础上，尝试设计不同特征用于模板匹配，比较不同特征情况下对模板匹配性能的影响。
- 提供数据：同第一次作业，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。

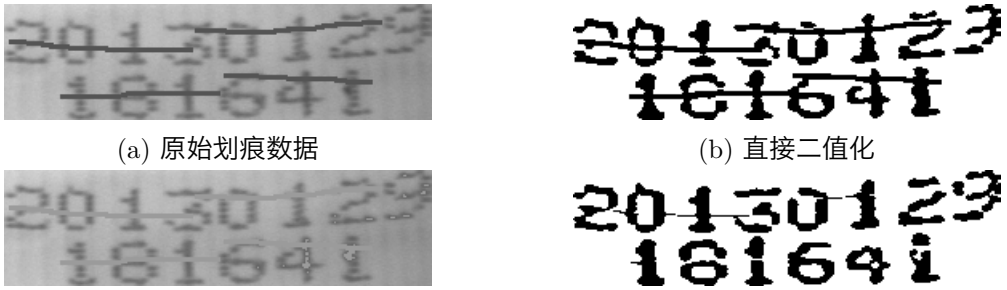
## 2 问题分析及实现思路

仍采用基于图像分割的方法，预处理、图像分割等流程与第一次作业基本一致。

## 2.1 预处理（同第一次作业）

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

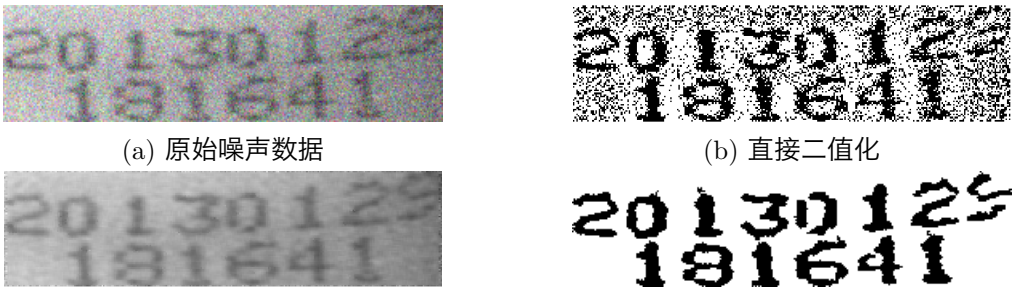
(c) 削弱划痕　(d) 削弱划痕后二值化

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（同第一次作业）

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）。

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

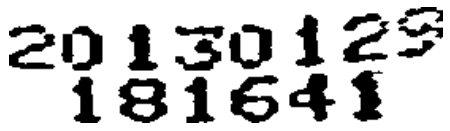
通过调整行切割、列切割的参数（极值特征判定）可将测试图像大体切割成若干块，每个块**至多包含一个数字字符**。

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**。

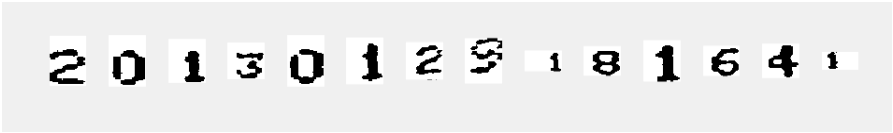
从以下几个方面进行块修剪：

- 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。



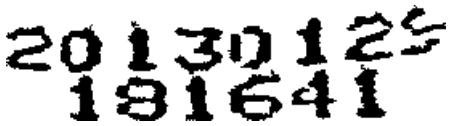
(a) 二值化 1.bmp



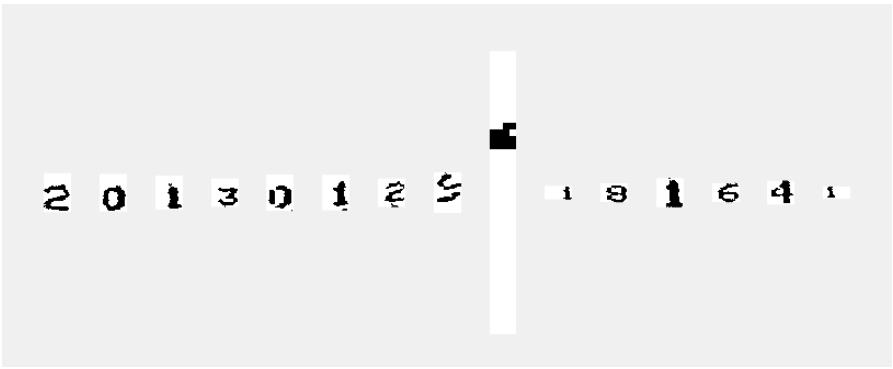
(b) 1.bmp 粗分割

(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 特征设计

严格来说，去除划痕、去除噪声、二值化以及图像分割都可视为预处理。经过预处理，待测图像和模板图像均为包含相切数字字符的小图像块，但尺寸不一定相同。以下

列出针对这种情况设计的若干简单特征。

#### 2.3.1 局部黑（白）像素计数

- 将图像块划分为若干子块（子块之间允许重叠），分别计算黑（白）像素的数量；
- 将子块的局部特征汇总作为特征向量；
- 归一化有两种方式：
  1. 将整个图像块 `resize` 至固定大小（如 32x32），再进行分块；
  2. 按照长宽比例划分子块（如每个子块的长宽分别 $=0.1$ 原图长宽），无需对整个图像块进行 `resize`。

  特征函数列表：

- `blockwise_counter`
- `blockwise_white_counter`
- `noresize_blockwise_counter`
- `noresize_blockwise_white_counter`

#### 2.3.2 按列（行）黑（白）色像素比例直方图

以按列黑色像素比例直方图特征为例进行说明。

- 计算**每一列**黑色像素的占比 $\alpha \in [0,1]$；
- 用归一化的宽度把整个图像分为若干个纵向的 `bin`，比如 `bin[0,0.2]` 表示 `image(:,1:0.2*width_of_image)`；
- 根据 $\alpha$ 所在的列位置将 $\alpha$ 的值**叠加**至相应的 `bin` 中；
- 每个 `bin` 的值排列成为特征向量。

特征函数列表：

- `colwise_hist`
- `colwise_white_hist`
- `rowwise_hist`
- `rowwise_white_hist`

#### 2.3.3 固定尺寸列（行）黑色像素比例排序

- 将图像块 `resize` 至某固定尺寸；
- 计算**每一列**黑色像素的占比 $\alpha \in [0,1]$；
- 将 $\alpha_i$ 排序，获得排序后对应的原列索引顺序作为特征向量。

特征函数列表：

- `colwise_sort`
- `rowwise_sort`

#### 2.3.4 梯度直方图（HOG）

- 将图像块 `resize` 至某固定尺寸；
- 按一定的 `cell`、`block` 规格参数提取 `HOG` 特征向量。

特征函数列表：

- `hog`

### 2.4 多种特征综合

我们可以从各个角度对数据提取出多种特征。某些特征可能整体上具有优秀的识别性能，但不能很好地区分 `0` 和 `8`；而一些其他特征可能对于 `0` 和 `8` 的区分非常有效。这就需要我们综合多种特征来提高准确度。

直接拼接各特征的特征向量是意义不大的，存在归一化、权重等问题。

对于单个特征的模板匹配，通过归一化内积的方式来求模板与目标之间的**相关系数** $\in [0,1]$。综合多个特征较简单的方法之一是**将各特征所得相关系数相乘**，得到的乘积作为总相关系数。

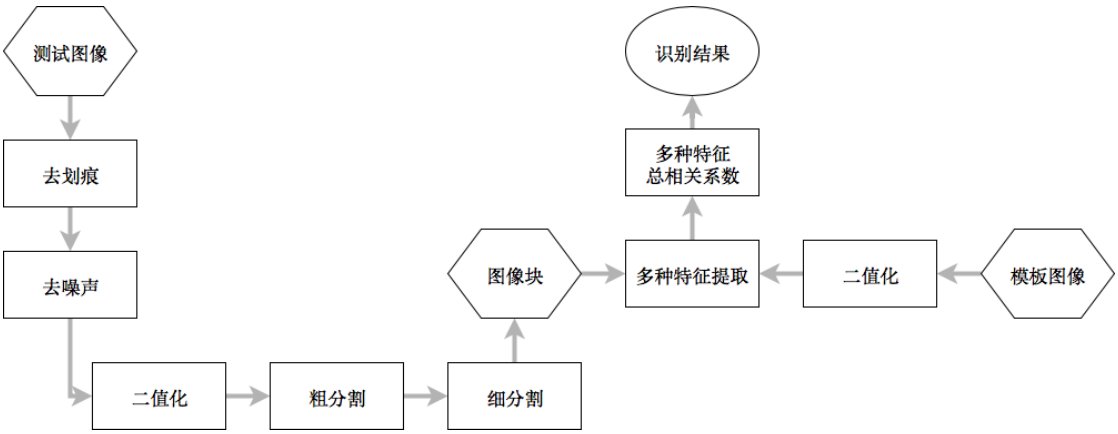



Figure 6: 识别过程流程图

# 3 实验结果

- 源码仓库链接:

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

- 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。

  若要调整所要提取的特征，增减 `run.m` 中 `extractor_list` 定义的**函数句柄**列表即可。

最终选取了三种特征进行综合：局部黑色像素计数、局部白色像素计数以及 `HOG` 特征，结果如图7。

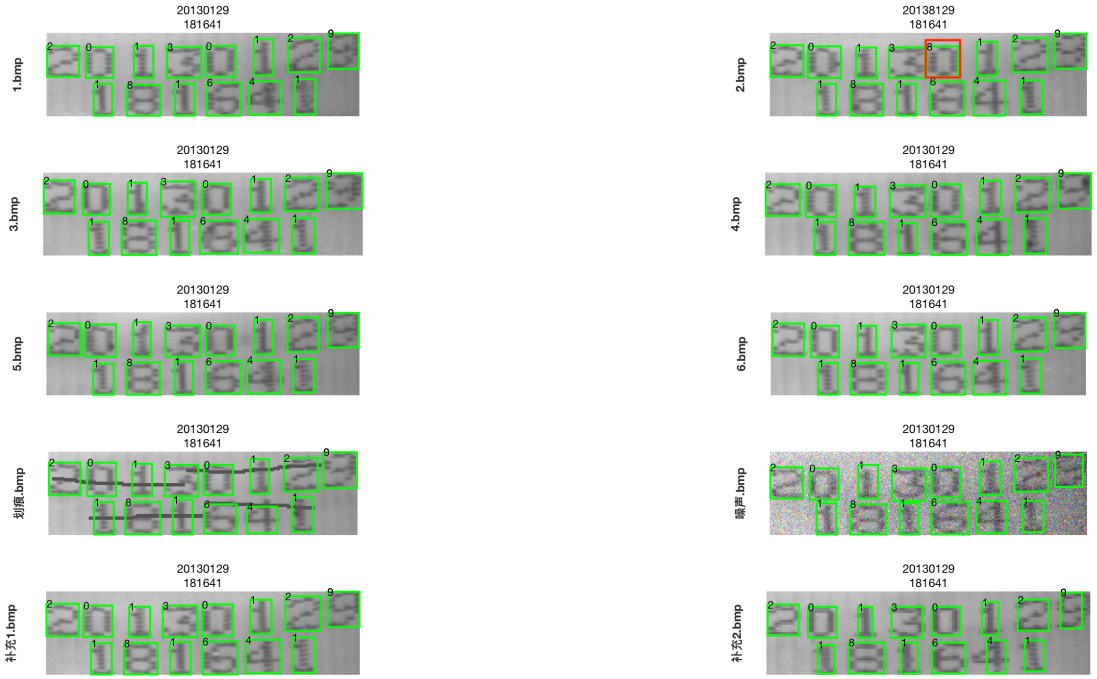



Figure 7: 多种特征识别结果

$$TP = \frac{139}{140} \approx 99.3\%$$

### 3.1 单特征识别效果

- `blockwise_counter`
  - 参数:

    ```
    norm_size = 128;
    block_size = 8;
    step = 2;
    ```

  - 最佳识别率: 137/140
  - 备注: 该特征对数字字符识别相当有效，出现误识别的字符基本由切割不精确导致（错误地切掉了一小部分数字本体）。
- `blockwise_white_counter`
  - 参数:

    ```
    norm_size = 128;
    block_size = 8;
    step = 2;
    ```

  - 最佳识别率: 138/140
  - 备注: 该特征对数字字符识别相当有效，效果甚至略优于黑色像素点计数。
- `colwise_hist`
  - 参数:

    ```
    bin_count = 15;
    ```

  - 最佳识别率: 70/140
  - 备注: 单从列方向提取的特征直接用于识别效果太差，尽管该方法不需要 `resize` 图像，但对 补充 `2.bmp` 识别效果极差。因为小尺寸图像在使用归一化宽度时，取整带来的误差影响很大。
- `rowwise_hist`
  - 参数:

    ```
    bin_count = 10;
    ```

  - 最佳识别率: 94/140
  - 备注: 单从行方向提取的特征直接用于识别效果也较差。但该特征要优于列方向特征，因为数字字符普遍是“瘦高”形态，按行提取的特征信息更多。另外，该特征有时能区分出其他特征无法区分的 `0` 和 `8`。
- `hog`
  - 参数:

```
norm_size = 20;
cell_size = [2 2];
```

  - 最佳识别率：122/140
  - 备注：由于模板本身尺寸不大，HOG 特征默认的 8x8cell 难以体现效果。而测试发现归一化尺寸和 cell 尺寸的选取对 HOG 特征的提取影响非常大。

# 4 结果分析及总结

## 4.1 结果分析

简单的特征（局部像素计数）对本作业的数字字符识别任务有良好效果，可惜最终还是没能完全正确，0 和 8 在多个特征维度下都难以明确地被区分。

## 4.2 总结

- 在多种特征综合时，可考虑赋予不同特征不同的权重，通过调节权重获得更好的识别效果；
- 利用归一化的尺度单位（百分比）可以避开对图像块的 resize 操作，但在处理小尺寸图像时一定要注意，百分比映射到实际行列坐标的取整操作误差对小尺寸图像块来说可能是无法忽略的！
- 由于模板是相切图像，在计算 HOG 特征时，边缘切点的梯度信息不完整，应考虑用 zero-pad 的方式略微扩展模板，可能会提高 HOG 特征的识别性能。