图片验证码识别算法的设计与实现

张扬

摘要：

图片验证码作为一种防止机器人使用重复手动暴力尝试获取网络资源的一种手段，其生成方式简单且种类多样，主要用于区别人和机器。但各种验证码生产程序生成的图片验证码保护强度不一。对验证码机器识别的研究可以为日后研究更强的验证码生成规则提供基础。本文主要针对一种样式的图片验证码，给出一系列图像处理的方法，包括二值化，修复，去噪。并针对性地设计一种简单高效且有较高正确率的字符切割方法最后使用支持向量机进行机器训练和预测。

1. 引言

图片验证码技术是现代网络安全技术的一部分，用于区分访问对象是人还是机器人，以防止使程序进行大流量的有目的的访问。一般的验证码可以阻止一些简单的机械试的访问，在如今机器识别已经成熟的时代，对于简单排列的数字和字母的图片识别已经不是什么难题了。国内外已经有很多学者对验证码的识别进行研究。尝试机器识别验证码的目的不是破解而是发明更有效的，难以机器识别的验证码来保持发明验证码的初衷。验证码的识别包括图片预处理阶段，包括图片灰度化，图片二值化，图片去噪，图片增强等；字符分割阶段，也是识别验证码最难最关键的阶段，字符分割的好与坏会直接影响到下一步分类学习识别阶段；机器学习与预测阶段，这一阶段常用的分类及预测的有KNN和SVM算法。现阶段的研究表明，字符分割是验证码处理的重难点，目前还没有通用的完美的算法去解决复杂粘连字符的分割问题，有些甚至人眼也需要一定时间去区分，这是阻碍机器识别验证码的主要障碍，也成为如今验证码依然可以成为人机区分手段的基础保障。国外学者指出，当一组样本的验证码能够被机器有效识别（完全识别正确）超过10%则该验证码（系统）已经不再安全。

本课题旨在以验证码识别为媒介学习图片处理图片分割以及机器学习的一些算法，尝试设计一套简单的图片验证码识别算法。

第一节对图片样本的特性作出一些分析，第二节主要对图像预处理各步骤算法作介绍和一些性能分析，第三节主要介绍和分析字符分割算法，第四节对实际预测效果和性能作出一些测试和评价。

1. 原始样本的特性分析

针对一种特定规则生成的图片验证码设计图像处理机器识别算法是改进其生成算法以防止后续扔被机器暴力破解的前提，因此针对特征图片进行图像处理时可以避免大量嵌套一系列通用图像处理的算法而针对性地设计特征处理算法，因此需要在算法设计前期针对原始的图片样本做一些图像的特征分析。本案例选取500张样式样本作为机器学习的学习样本。

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| 图1. 一些原始验证码图片样本 | |

通过肉眼观察上述验证码图片的样本，我们可以发现一个样本图片主要由背景，干扰，字符三部分组成，而且这类生成规则下的验证码字符元素的粘连不大甚至几乎没有，这对于图片后期的字符切割是极为有利的，同样也就预示着这类验证码极其容易受到机器的暴力破解。

* 1. 图像结构及描述的定义

|  |
| --- |
|  |
| 图2. 图像定义 |

本文中处理的图像坐标原点设在图像左上角，图像宽度方向为X轴，高度方向为Y轴对于其中的给定一点像素为方便表述及后续处理，定义像素位置向量

|  |  |
| --- | --- |
|  | （1） |

将图像看作复平面，定义像素位置复数

|  |  |
| --- | --- |
|  | （2） |

。

对于输出为实数的（离散）函数 ，其二元函数表达形式 二维向量表达形式，复变表达形式 在本文中表达相同意义，并取其表述简单的形式对算法进行描述。

图像区域

|  |  |
| --- | --- |
|  | （3） |

其中width表示图像宽度，height表示图像高度。

* 1. 样本集中主要图像结构元素的分布

我们将样本集中所有的样本图像叠加后求平均灰度值，便可以观察到整个样本集中干扰字符等的分布状况。

对样本集做如下处理

|  |  |
| --- | --- |
|  | （4） |

处理后我们可以得到如下结果：

|  |
| --- |
|  |
| 图3. 叠加后的灰度平均值分布情况 |
| 图中，蓝色矩形框表示主干扰存在区域，红色矩形框表示字符的主要存在区域。  我们可以发现，有较多分布的较大干扰反而对后期字符处理影响不大，由于与字符间隔较大，它们可以很容易得被去除。因此我们主要需要处理的噪声为穿过字符存在区域较细的长干扰线。 |

* 1. 图像背景灰度值分布的前期分析

通过对验证码图片的样例进行观察可以得知图片背景灰度在图片高度方向几乎没有变化，在沿着图片宽度方向逐渐变亮。我们选取几行不经过字符主要存在区域的行作为背景采样行，用500张样本进行灰度分布的采样。

做如下操作

|  |  |
| --- | --- |
|  | （5） |

|  |
| --- |
|  |
| 图4. 主要背景区域采样行灰度值分布情况 |

对上述采样图像及原始样本图像进行分析可以得知，在宽度20和45位置的两个波谷是由黑色的主干扰线引起的平均灰度值下降，第49行的平均灰度出现另外四个明显波谷是由于有些字符如p,g，向下延伸较大，在第49行仍有黑色的字符像素分布引起的。四个波谷几乎恰好对应四个字符的分布位置。

略去黑色干扰线和个别字符引起的灰度值分布波动，背景的灰度值从图像左侧到右侧明显呈上升趋势，从采用图像上粗略估计背景灰度的最小值应该是不低于185 。

1. 图像预处理算法设计及性能分析
   1. 灰度图像二值化

本案例中样本灰度图像的二值化主要需要讨论两个方面，使用常规的二值化阈值确定方法还是根据样本特性设计一种更简单的二值化阈值确定算法；另一方面对样本进行二值化时，每个样本确定一个二值化阈值还是所有样本使用一个全局的二值化阈值，两者效果是否有较大的差距。

下面我们考察一个样本及其灰度直方图的特性：

|  |
| --- |
|  |
| 图5. 样本及其灰度直方图 |
| 结合前期对于样本图片背景的研究，其中很大一部分灰阶值所拥有的像素个数少于10个，这表明这些灰度值对应的点是字符与背景的过渡点，由于字符始终是灰度为0的黑色点，而背景由左至右逐渐变亮，图片的生成规则为了使变化不唐突，使得字符与背景的过渡点随着背景的变化而变化。  我们显然不能通过二值化操作将字符过度略去，更不能将背景划为有效识别像素，因此那一系列低像素值的灰阶便是我们选取二值化阈值的范围。  传统的OSTU法取图像二值化需要多次迭代，比较耗费计算时间，主要用于处理那些构成复杂的图片，这类图片验证码背景区分鲜明，构成并不复杂，因此这里考虑不适用OSTU法，初步试探尝试使用低像素灰阶中位数法。   1. 先对原图像作出灰度直方图 2. 在定义域的子集中观测一个合理的界定值 3. 在区间中找到所有的 4. 求出的中位数 5. 记录二值化阈值   找到二值化阈值后对图像进行二值化处理  则有   |  |  | | --- | --- | |  | （6） |   我们将将作为学习样本的500张图片，分别用不同的二值化阈值计算方法计算其二值化阈值，并统计得到下图：   |  | | --- | |  | | 图6. 不同二值化阈值计算方法计算出的阈值结果 |   通过对比看出，OSTU计算出的二值化阈值较高，各样本的阈值相当接近，MLPG法计算出的二值化阈值较低，个样本比较接近，最大与最小的二值化阈值相差在20左右。但是MLPG法的计算算法复杂度远低于OSTU，不必要做重复的迭代，只需要扫描一遍像素点即可。  通过上述统计还可以发现，不同样本的二值化阈值相当接近，表明有条件进一半简化算法，对于所有的样本使用同一的二值化阈值，使得二值化前不必要针对每个样本进行计算阈值。实际上这是可行而且是较佳的选择。  下面我们给出两个样本使用不同的阈值进行二值化的效果。   |  | | --- | |  | |  | | 图7. 不同二值化阈值下的二值化效果 |   不难看出，二值化阈值越高，字符约粗实，字符边缘也越发平滑，但较高的二值化阈值也强化的干扰线部分，加大了后期的处理难度，而且对于某些样本如【mw3n】，较高的二值化阈值使得字符见出现了粘连，这对于后期字符分割是比较难以处理的，这也是较高二值化阈值的一个致命缺陷。低二值化阈值使得字符边缘凹凸不平，且字符骨干部分偏细，但是却很好地分开了各个字符，不对对后期字符分割造成困难。  由于二值化阈值选取的不同，会导致字符粗细不同，这对于后期机器学习样本字符的平均粗细不一是很不利的，使用一个全局的二值化阈值可以使得不同样本间的字符粗细均衡，对于机器学习需要的归一化条件是很有利的，这也是最后摈弃了每个样本分别计算二值化阈值的主要原因。  本案例中的二值化阈值尝试选取T=80,T=100,T=120 ，经过后期的去噪分析，字符分割分析，最终选取全局二值化阈值T=80。该阈值较好地保留了字符的骨干部分，会导致字符边缘出现一些孔洞，这使用后期的字符空洞修复算法可以完成；该阈值没有过度强调干扰线，使得干扰部分可以较为容易地去除；该阈值不易造成字符粘连，（原则上该样本集的字符间均有空隙，但可能较小，过高的二值化阈值将这些字符背景过渡点划为字符点使得出现了字符粘连）比较有利于后期的字符分割步骤。   * 1. 字符边缘孔洞修复   由于二值化算法选取的阈值会导致字符边缘出现孔洞，字符边缘不平滑的结果。这样的字符边缘在后期去噪可能会被误认为干扰线；其次，同一字符的不同样本在不同的位置出现孔洞与不平滑现象对于机器训练是不利的，会导致机器识别误判；再次，后期的字符分割默认认为字符是连续的，字符内部是不应该出现空隙的，字符边缘零散越出的白点或者黑点也可能会对字符分割造成干扰。因此字符边缘的修复这一步是必要的。  这里先介绍取一个像素周围四连通域或者八连通域的算法：  取四连通域的四个像素即取该像素位置的上下左右四个像素，在该案例中使用像素的位置复数表达较为简单。  由复数的有关知识可知在上述坐标系中 所表示的位置分别为复数 所表示的像素位置的 右下左上 四个像素。  对于取八连通区域的八个像素，即像素的上下左右，左上左下右上右下八个像素。  为方便处理定义旋转用模长为1的复数   |  |  | | --- | --- | |  | （7） |   同时为保证旋转45°时依旧是格点复数，即x,y是整数，需要在旋转45°时做模长伸缩变换。  像素八连通域的八个像素位置可以用如下方式取到   |  |  | | --- | --- | |  | （8） |   为方便后续使用，定义   |  |  | | --- | --- | |  | （9） |   在上述坐标系下 其中k=0,1,2,3,4,5,6,7时分别表示原像素的右，右下，下，左下，左，左上，上，右上 八个位置的像素位置复数。  这里我们额外定义一条规定：对于图像区域外即简单地定义此时的即白色点。  取四连通域像素进行判断时，我们认为一个白点周围有三个即以上的黑点，那么该白点应当被修正为黑点；   |  |  | | --- | --- | |  | （10） |   取八连通域像素进行判断时，我们认为一个白点周围有五个及以上的黑点，那么该白点应当被修正为黑点。   |  |  | | --- | --- | |  | （11） |   所有的修正后的信息直接输出在一个新建的图像上，对原始的输入图像不直接做修改。   |  |  | | --- | --- | |  |  | |  |  | |  |  | | 二值化后图像 | 边缘散点修复后图像 | | 图8. 字符边缘孔洞修复效果对比 | |   通过字符边缘孔洞修复后的图片明显得字符边缘变得平滑，且白点黑点零散分布的现象明显减少，有利于确定字符的主体位置，且有利于后续的去噪手段，不至于把应属于字符的黑色像素点过分地去除。   |  |  |  | | --- | --- | --- | |  |  |  | | 3pfg | 525d | cdg8 | | 图9. 修复掉的白色噪点 | | |   由效果图还可以得知，虽然这一算法步骤是去掉白点增加黑点，但主要是增加了字符边缘的黑色点，对干扰线并没有明显地加粗，因为干扰线周围的白点是不符合被黑色点大半包围这一判定条件的。  通过实际的测试，该算法的实现及相关阈值的设定，达到了不错的预期效果，可以为后续步骤的实施提供前提保障。   * 1. 图像去噪算法      1. 噪声来源   通过前文的描述可以得知，经过二值化后，背景几乎不会产生噪声，图像噪声来源主要在于厚度较大的左侧两条干扰线，其中厚度较大的部分在图像的主要字符存在区域外部，这一部分干扰后续可以在字符切割的时候直接切除实现，现处理阶段实际并不会造成过多的影响，这一阶段主要处理的是一段贯穿主字符存在区间的细干扰线，它会对后期机器训练识别字符造成一定的影响，比如字符误认；也可能对字符切割时造成困难，比如切割时机器认为这是字符粘连而错选了切割位置。  因此，图像去噪这一阶段的算法设计，主要是针对如何消除贯穿主字符区间的干扰线而特别设计的，避免了叠加使用通用图像处理方法。这里设计的集中针对性去线算法可以在每一个案例中单独使用，也可以进行设定不同的参数以组合使用达到更好的去噪效果。   * + 1. 深度（垂直）方向探索去噪算法   我们把在图像的一列中，沿着高度方向，图像的y坐标增大的方向叫做图像的深处方向。深度方向探索即检查当前像素位置深处方向的像素点。  根据我们要着重去除黑色干扰线部分的特点：连续较长，贯穿字符存在区间；大部分图像的干扰线较细（少数特例）厚度小于三个像素。同时，我们探查了字符厚度，大致在5个像素厚度以上。 |
| 对于大部分图片样本而言，生成的黑色干扰线厚度大致相同，因此我们选择几例干扰线和字符明显分离的样本在做厚度采样测试。  样本中的蓝色矩形框为我们对噪声区域像素厚度的采样，红色矩形区域为我们对字符区域像素厚度的采样。  记录的信息为我们每找到一个黑色像素点，这个黑色像素点深度方向的厚度。   |  |  | | --- | --- | | 其中 |  | |  |  | |  |  | |

|  |
| --- |
|  |
| 10-1 样本图片 |
|  |
| 10-2 干扰线像素点深度方向厚度 |
|  |
| 10-3 字符像素点深度方向厚度 |
| 图10. 样本深度方向厚度的一个采样 |
|  |

通过一个样例的采样分析我们可以看出，属于干扰线部分的黑色像素，深度方向的厚度绝大部分在3以下，少数达到4，而属于有效字符的黑色像素，只有少量在深度方向的厚度不达3，这些黑色点在字符下端片区，即“字符笔迹”在垂直方向的末端点。

由此我们给出一个尝试消去黑色干扰线的深度方向去噪算法：

D为给定的深度方向探索阈值，

若

|  |  |
| --- | --- |
|  | （11） |

则计算

|  |  |
| --- | --- |
|  | （12） |

则去噪修正后的灰度为

|  |  |
| --- | --- |
|  | （13） |

所有的修正后的信息直接输出在一个新建的图像上，对原始的输入图像不直接做修改。

我们在这里先讨论单一方式去噪的效果，后续再进行组合方式去噪的讨论。

下面我们对单一进行深度方向去噪的效果给出一些性能测试：

|  |  |  |  |
| --- | --- | --- | --- |
| 样本 |  |  |  |
| 阈值D=1 |  |  |  |
| 阈值D=2 |  |  |  |
| 阈值D=3 |  |  |  |
| 阈值D=4 |  |  |  |
| 图11. 单一深度方向去噪算法效果 | | | |

我们很显然地发现，使用这种深度方向去噪算法虽然可以有效消除掉薄的干扰线，但是也会削减掉字符的厚度，因此阈值的选择尤为重要，这里阈值的选择需要考虑能把大部分有影响的干扰线去除，尽量不把字符像素削减的过于薄以至于在字符内部出现空隙对后期字符分割造成不利的影响。

我们参照上述效果图可以看出，如果单一对这种验证码图片使用这种去噪算法，阈值D=2或者阈值D=3是可以考虑使用的阈值，这也与之前像素采样结果相一致。

该算法的有点为针对性强，专门针对薄干扰线设计，且算法执行简单，只需要遍历图片所有像素一次即可，没有迭代操作；该算法缺点也很明显，很明显地削薄了表示字符的像素，过度的去噪可能会产生字符内部不该有的空隙；如果要很好地保留字符像素就会去噪不当，干扰线去除不充分。该算法与其他去噪算法组合使用的效果将在后面测试。

* + 1. 八方向检索去噪算法

检索一个像素八连通域方向上的像素是否总体符合一定的要求来判断这是不是一个待去除的噪声干扰点。在这里，我们先对一个像素八方向及层数的定义做一些说明。

|  |
| --- |
|  |
| 图12. 一个像素的八方向及其层数 |

取一个像素第一次的八方向像素即取一个像素八连通域的八个像素的方法在上文已经介绍过，这里再说明一下取不同层数的八方向像素的方法。

一个像素的八方向像素与中心像素的辐射角满足如下的位置关系：

|  |  |
| --- | --- |
|  | （14） |

对应各数字分别是中心像素的右，右上，上，左上，左，左下，下，右下八个位置。那么待选取的第s层第k个像素可以如下表示：

|  |  |
| --- | --- |
|  | （15） |

其中zz为待选取像素，z为中心像素。

我们依旧选取一些样本对这些样本的干扰线区域和字符区域统计其中黑色像素点（待处理像素点）的八方向像素点信息，主要是统计八方向白色像素点的个数。

|  |
| --- |
|  |
| 图12. 八方向数据采集样例（蓝色为干扰区，红色为字符区） |

|  |  |
| --- | --- |
|  |  |
| 层数=2 干扰区域统计结果 | 层数=2 字符区域统计结果 |
|  |  |
| 层数=3 干扰区域统计结果 | 层数=3 字符区域统计结果 |
|  |  |
| 层数=4 干扰区域统计结果 | 层数=4 字符区域统计结果 |
| 图13. 对样例进行2,3,4层八方向白色像素个数统计结果 | |

我们给出如下的八方向多层检索去噪算法：

实际所取的层数S，和设定的参考阈值T为经验参数。

|  |  |
| --- | --- |
|  | （16） |

当一个黑色像素点检索到的多层八方向像素中白色像素点的个数大于设定的阈值时，这个黑色像素点就会被当做噪声点给予去除。

所有的修正后的信息直接输出在一个新建的图像上，对原始的输入图像不直接做修改。

在对设置探索层数及对应的处理阈值前，我们对样本每层及各个阈值去除掉的干扰线点比例和字符点比例进行统计。

|  |  |  |
| --- | --- | --- |
| 表1. 探索层数=2时各阈值上述样本去噪效果 | | |
| 阈值 | 干扰区黑色像素点去除率（应去率） | 字符区黑色像素点去除率（误去率） |
| 5 | 80.24% | 38.79% |
| 6 | 68.14% | 30.85% |
| 7 | 54.28% | 16.96% |
| 8 | 41.89% | 11.41% |
| 9 | 29.50% | 4.55% |
| 10 | 24.78% | 2.73% |
| 11 | 17.70% | 0.74% |

|  |  |  |
| --- | --- | --- |
| 表2. 探索层数=3时各阈值上述样本去噪效果 | | |
| 阈值 | 干扰区黑色像素点去除率（应去率） | 字符区黑色像素点去除率（误去率） |
| 8 | 88.50% | 45.33% |
| 9 | 84.07% | 36.97% |
| 10 | 71.39% | 26.22% |
| 11 | 59.88% | 18.94% |
| 12 | 50.74% | 12.82% |
| 13 | 37.76% | 7.94% |
| 14 | 30.97% | 4.30% |
| 15 | 24.48% | 3.06% |

|  |  |  |
| --- | --- | --- |
| 表3. 探索层数=4时各阈值上述样本去噪效果 | | |
| 阈值 | 干扰区黑色像素点去除率（应去率） | 字符区黑色像素点去除率（误去率） |
| 11 | 89.09% | 59.06% |
| 12 | 86.73% | 48.06% |
| 13 | 76.40% | 38.21% |
| 14 | 74.34% | 29.36% |
| 15 | 68.44% | 22.75% |
| 16 | 57.82% | 16.63% |
| 17 | 49.56% | 11.86% |
| 18 | 40.71% | 7.69% |
| 19 | 31.86% | 5.46% |
| 20 | 26.25% | 3.39% |
| 21 | 23.89% | 2.07% |

以上效果数据为从少数样本将干扰与字符分离后的测试结果，具有一定的参考性。同时针对不同样本，实际的效果可能会有所出入。

这种针对性去噪算法显而易见地可以得出，如果要增加去噪率，必然会提高误去率，因此使用该去噪算法的关键在于控制好两者比例，使得影响字符识别和字符分割的干扰线能够被充分去除，同时保证字符框架被较好保留，因为字符被错误去除一小部分几乎是无可避免的。

下面给出单一使用这种去噪方式（组合去噪方式将在后续讨论）对于几个测试样本的实际去噪效果。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 样本 | T=4 | T=5 | T=6 | T=7 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 图14. 层数为2时八方向检索去噪法各阈值的实际效果 | | | | |

通过上述实例效果我们可以看出，进行2层八方向检索去噪时，阈值设定为5或6为理想的阈值能够达到较好效果，阈值设定大于7时，虽然字符保留比较完整但可能会使得部分干扰线去除不是很彻底，但如果某些样本使用低阈值时字符去除过多，设定阈值为7也是可以参考的。

我们选用同样的样本做3层八方向去噪实验：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T=8 | T=9 | T=10 | T=11 | T=12 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 图15. 层数为3时八方向检索去噪法各阈值的实际效果（省略原始样本） | | | | |

使用三层八方向去噪时，去噪效果随阈值设定的变化渐变，从上述样本的直观视觉看来，阈值设定9,10,11,12的实验效果均可以接受，不同的阈值虽然对字符骨干的保留均尚可，但对字符厚度还是有些许影响，如果机器训练的样本和识别用的样本去噪阈值设定不同，字符的厚度可能会影响到机器识别的效果。再者，当3层去噪与2层去噪有相当效果时，尽量选择二层去噪来适当减少计算量。

最后我们还是选择上述样本做4层八方向去噪实验：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T=14 | T=15 | T=16 | T=17 | T=18 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 图16. 层数为4时八方向检索去噪法各阈值的实际效果（省略原始样本） | | | | |

使用四层八方向去噪时，阈值设定更加细化，去噪效果越发渐变，但实际上较较少层去噪的效果提升并没有质的变化，因此实际使用时不推荐使用四层以上的检索。

综合上述去噪效果的测试，我们后续默认处理时，使用效果比较好的2层阈值为6的去噪参数以作为一个对比效果。

八方向检索去噪也是根据干扰线厚度较小的特征，针对像素厚度的一种去噪方式的变形，因此这种算法也是不可避免地会削薄有效的字符像素的厚度，但通过合适的参数设定，可以让这种“削弱”是可控且可接受的。

* + 1. 模板统计滤波器