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） |

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。  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。 |
|  |