使用Tesseract OCR引擎的组合脚本和页面方向估计

Combined Script and Page Orientation Estimation using

Ranjith Unnikrishnan和Ray Smith，Google Inc.，1600 Amphitheatre Pkwy，Mountain View，CA 94043ranjith @ alumni.cmu.edu，theraysmith @ gmail.com

抽象

本文提出了一种简单而有效的算法来估计图像中包含的文本的脚本和主要页面方向。使用合成渲染的文本生成每个脚本的一组形状类候选集，并将其用于训练快速形状分类器。在运行时，针对组件的每个可能方向，将分类器独立应用于图像中的已连接组件，并使用累积的置信度分数来确定页面方向和脚本的最佳估计。结果证明了该方法对以下情况的有效性： 1846年的文档包含14个脚本中的不同图像集以及四种可能的页面方向。在未来的开源Tesseract OCRengine [1]版本中，将可以使用此工作的C ++实现。

类别和主题描述符

I.7.5 [文档和文本处理]：文档捕获-光学字符识别（OCR）

一般条款

算法，语言

关键词

脚本检测，页面方向检测，Tesseract

图1：使用建议的算法检测到的15个脚本的图像样本。 （未显示拉丁语。Fraktur被视为独立脚本。有关详细信息，请参见文本。）

1.引言本文着重于估计图像中印刷文本的脚本和主要页面方向的问题。为了准确识别图像中的文本，光学字符识别（OCR）算法通常利用大量先验知识，例如字符的形状，单词列表以及它们出现的频率和样式。如果不是全部的话，很多知识都是特定于语言的并且不会在脚本之间泛化。这使得图像中包含的文本语言成为使用OCR算法时要指定的关键输入参数。脚本形成了基于外观的自然语言分组，许多语言共享同一脚本。例如，俄语，保加利亚语和乌克兰语共享“西里尔字母”脚本。大多数印度文字，例如泰卢固语，卡纳达语和泰米尔语，都只有一种语言与之关联，而拉丁文字至少由26种常见语言共享。因此，针对脚本检测问题的解决方案要么解决了语言识别问题，要么将其简化为一个较小的问题。OCR算法独立于脚本的知识，自然地假定要处理的图像中的文本是直立的。并非总是如此。例如，书中的大桌子和数字经常以横向模式打印，而书的内容则可能以纵向模式显示。否则，书籍本身可能会以错误的方向作为输入。输入和期望值之间不匹配的示例很常见，并且不可避免地导致产生垃圾输出的OCR算法。一种蛮力的解决方案是使用传统的OCR算法对每个可能的组合处理一次每个输入页面方向和语言模式。但是，这将是不切实际的缓慢，并且还需要

确定每种定向语言配置中输出的文本的有效性的单独过程。本文 duced as output in each orientation-language configuration.
提出了一种简单但有效的方法来估计脚本和主导方向，不仅具有很高的准确性，而且所花费的时间也要少得多。使用传统的OCR算法甚至需要处理一次图像。该解决方案的伸缩性足以区分14种脚本-拉丁，西里尔，希腊，希伯来语，阿拉伯语，中文（或汉族），日语，韩语，泰语，梵文，卡纳达语，泰米尔语，泰卢固语和孟加拉语-涵盖40多种语言，以及还可以区分Fraktur和非Fraktur字体。

1.1相关工作解决脚本检测问题的过去方法可能分为三大类。第一种可能被称为“全局”或基于纹理的方法。此类别中的算法使用图像过滤器来计算文本块上的判别特征，以确定语言或脚本所独有的模式。 Chaudhury等。等人[2]提出了使用水平文本行投影轮廓的频域表示。布希等文献[3]对多种纹理特征，包括投影轮廓，Gabor和小波特征以及用于检测脚本的灰度级共现矩阵，进行了广泛的评估。这类方法的缺点是在一个脚本中需要较大且对齐的文本均匀区域，并且所讨论的功能通常在具有嘈杂或偏斜文本的情况下对计算而言既不具有区分性也不可靠。第二类方法可以称为基于“本地”或连接组件。这些利用了各个连接组件的形状和行程特征。霍赫伯格（Hochberg）等。 [4]等人建议通过将频繁出现的字符或单词形状聚类来使用脚本特定模板。斯皮兹等[5]等人[5]构造了捕获字符凹面的形状代码，并使用它们首先将它们分类为基于拉丁语或汉族的文字，然后使用其他形状特征将其分类。 Ma等[6]使用Gabor过滤器和最近邻分类器来确定单词级别的脚本和字体类型。还建议了几种局部和全局方法的混合变量[7]。方法的第三类可以称为“基于文本的”。此类算法的工作原理是使用一个或多个“试点”语言模式使用传统的OCR引擎处理整个页面，然后使用单独的过程来分析（可能不准确）输出的统计信息，以猜测原始图像是哪种语言这类方法巧妙地利用了这样一个事实：尽管在未经训练的班级上对形状分类器进行评估时，形状分类器是可预测的错误，但它们造成的错误却往往是可重复的。因此，以汉语言模式处理阿拉伯文本的图像将给出垃圾文本，但是具有输出字符的特征频率。然后可以分析输出文本的统计信息，以估计输入图像的脚本。但是，此类技术无法很好地适应大量的脚本/语言，我们在第5节中的实验表明，它们的准确性低于本文档中提出的方法。

2.方法我们提出的方法属于“本地”方法类别，并且通过对各个连接的组件进行独立分类来进行操作。这种策略具有以下优点：不需要分词或文本行查找作为预处理步骤，并且能够处理较小的输入图像。所提出的方法的基本思想很简单。所有感兴趣的脚本中的字符（类）。在运行时，分类器根据图像中的每个连接组件（CC）独立运行，并且在将每个CC旋转到其他三个候选方向（输入方向为90°，180°和270°）之后，重复该过程。 。该算法跟踪给定方向上每个脚本中估计的字符数，并跟踪所有候选方向上累积的分类器置信度得分。页面定向估计值被选择为累积置信度得分最高的脚本，而脚本估计则被选择为该脚本中字符数最多的脚本，以获得最佳方向估计。采用这种策略的主要困难是与所有14个脚本关联的类的总数。仅Hanscript就有数千个字符经常使用。在像梵文和阿拉伯文这样的文字中，字母的形状可以根据上下文采用不同的形式，并且与拉丁文不同，字母可以结合起来形成单个或多个连接的组件。脚本的这些属性在组合上增加了可能出现的形状的总数，并且由于我们的方法要求将一个类（和脚本）与每个连接的组件相关联，因此训练一个形状分类器的有效类数可能不切实际地很大。已经为所有脚本识别了形状类，然后问题就变成了如何选择这些类的子集来训练形状分类器。为脚本选择代表形式的类的一种可能方法是通过判别选择。这涉及到在脚本中选择与其他脚本具有较高“距离”的形状类。当在一个形状上进行训练并在另一个形状上进行评估时，或者通过直接嵌入形状，可以将这两个形状之间的距离计算为分类器置信度的函数。在适当的特征空间中，我们发现区分类别选择方法的两个主要问题。首先，判别类不一定是频繁的。例如，假设我们选择小写的“ q”作为拉丁文字的类，因为它与不同文字中的任何字符都没有形状相似性。给定文本图像，出现字母“ q”的频率可能很小。因此，给定一种对连接组件的随机采样集进行分类的策略，将脚本识别为拉丁语的预期时间将会很高。当输入图像的文本很少时，这会限制算法的准确性。第二个问题是由于观察到整个算法的性能不仅取决于分类器在训练集中的类的准确性，还取决于其在行为集上的行为。分类者尚未受过训练的形状。例如，假设形状分类器未针对字符“ t”的形状进行训练。当输入的形状为“ t”时，理想的结果是分类者声明该形状可能是以下形状之一：

任何具有相同可能性的候选脚本。但是，这种理想的结果在实践中永远不会发生，并且通常的行为是使脚本具有不等的后验概率。这种偏见自然会影响算法整体对脚本的估计。因此，我们采用生成类选择的方法。这是通过按频率和其他感兴趣的统计信息对类别进行排序，然后按覆盖率修剪此列表来完成的。此方法是出于以下观察目的

1.类的覆盖图通常很陡。对于具有相似形状的字符的拉丁文和西里尔文（Cyril-lic）等脚本，此属性尤其适用。因此，实际上，仅需要选择少量的形状类来表示此类脚本。

2.前面提到的关于区分选择的两个问题仅与分别在训练集中或不在训练集中的类的出现频率成比例地发生。与文本中频繁出现的字符相关联的形状类的选择减少了识别其脚本的预期时间。同样，在前面的示例之后，无法识别字母“ t”的形状，以及随之而来的后脚本概率估计值的分配仅与文本中字母“ t”的自然出现频率成正比。

在第3节中，我们详细介绍了识别与给定脚本相关联的shapeclass，然后一般性地选择这些类别的子集来训练Tesseractshape分类器的过程。第4节介绍了在运行时处理文本图像时的算法。然后，第5节说明了我们用于测试算法的数据集以及也经过评估以进行比较的替代方法。然后，我们在第6节中总结了一些一般性的观察和未来工作的方向。

3.训练训练阶段的目的是准备一个经过特殊训练的分类器，在我们的案例中，这是Tesseract OCR发动机使用的静态形状分类器[8]。它以我们感兴趣的脚本中的一组文本语料库为输入，并为每个脚本提供一组用于训练分类器的类和形状作为输出。这个阶段可以分为三个步骤-候选班级创建，生成选择和分类培训。

3.1候选类的创建训练的第一步的目标是详尽地识别一组图像形状原语和关联文本，它们共同代表该脚本中的任何文本体。由此产生的形状-文本对集合将形成一组候选类别。随后的步骤中，我们将从中选择一个子集来训练形状分类器。此步骤的策略是在每个脚本中使用自然文本（通过从网上抓取的文本语料库中获取），通过渲染文本并隔离连接的组合-与渲染页面中的文本关联的名词。从渲染过程中可以知道每个组件的文本，并且可以使用连接的组件提取算法轻松获得文本的形状[9]。这给

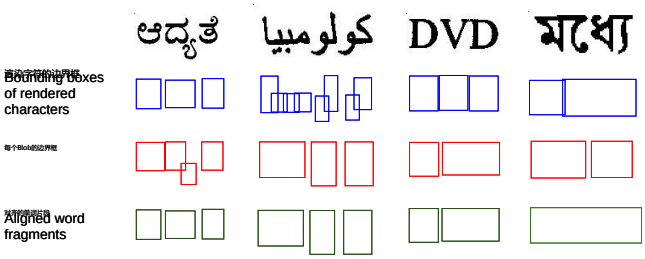

Figure 2: Examples of word fragment generation (left-to-right) Kannada, Arabic, Latin and Bengali, illustrating different cases of character overlap and resulting fragment generation. See text for details.

图2：单词片段生成的示例（从左到右）卡纳达语，阿拉伯语，拉丁语和孟加拉语的单词片段提取示例，说明了字符重叠和导致片段生成的不同情况。有关详细信息，请参见文本。

a set of shape-text pairs, and allow us to obtain relevant statistics from them for use in the subsequent selection step.

择步骤使用。更详细地，我们的首选实现通过以下过程生成候选集，该过程针对每个感兴趣的脚本执行：

1。使用标准渲染引擎，以一种或多种支持的字体将脚本中文本语料库中的单词渲染为具有可变降级的图像。我们的实现使用Unicode国际组件（ICU）布局引擎，该引擎为具有复杂渲染规则的各种非拉丁脚本（包括梵文）提供支持。降解包括不同程度的形态腐蚀/膨胀和噪声添加，并且是为了训练分类器而使其健壮。此步骤在图像中每个字符以及与之相关的字符周围产生了按阅读顺序排列的一组边界框。每个盒子。图2在每个渲染的字符周围显示了蓝色的边框，例如，用坎纳达语，英语，阿拉伯语和孟加拉语书写的单词。

2.处理图像以找到其连接的组件，并按阅读顺序排列其边界框（图2中的红色）。在这一步中，我们使用现有的Tesser-act实现来处理页面.Tesseract预处理器还根据水平重叠条件将连接的组件组合在一起，称为嵌套的组件列表（称为斑点）。小写字母“ i”形状的点与“ i”的竖线分组在一起，因此可能不会被视为单独的字符。图2显示了示例单词在每个获得的Blob周围的红色包围框。在训练过程中使用Tesseract，而不是仅在分类过程中使用，具有使连接的组件和形状产生屈服组的好处，其方式与在运行。

3.将字符周围的两个边界框序列与斑点周围的两个边界框序列对齐，以形成一组单词片段。这种对齐方式可以通过测试每个Blob和字符的边界框之间的重叠并形成簇来贪婪地束缚起来。通过一次渲染一个单词，可以确保片段边界不跨越单词边界，从而进一步简化了对齐问题

Figure 3: Portion of a synthetic rendered training image for the Kannada script. Bounding boxes of the word segments chosen for this script are shown colored in green.

图3：卡纳达语脚本的合成渲染训练图像的一部分。为此脚本选择的单词片段周围的边框以绿色显示。

图2说明了在不同脚本中生成单词片段（显示为绿色框）的一些可能情况。卡纳达语单词的第二个字符由两个相互连接的组成部分组成，它们之间没有明显的重叠，而阿拉伯语的多个字符组合在一起构成了连接的组成部分。虽然拉丁字符通常不会合并，但上面的示例碰巧有两个字符重叠，这与所选字体和渲染级别降低有关。孟加拉语单词中的第二个字符由两个不相交的连接组成，但其左部分与前一个字符重叠，结果是由两个字符组成的单个单词片段。因此，一个片段可能由一个或多个字符和一个或多个斑点（或相连的组件）组成。

4.如果片段的长宽比超过某个阈值，或者具有其他指示符，很容易将其与其他字符或非文本形状混淆，则拒绝该片段。如果未拒绝某个片段，请跟踪它在输入文本语料库中被看到的次数，它代表的字符和连接的组件数，与之关联的文本以及该文本的脚本，用于呈现该字体的字体的属性。片段以及其他任何相关统计信息。

所得的单词片段集形成特定脚本的候选形状类集。然后对所有脚本和支持该脚本的每种字体重复此步骤，以产生覆盖所有有趣脚本的一组片段。这标志着候选类别识别阶段的结束，并为下一步的生成类别选择让路。

3.2生成选择在此步骤中，从先前形成的候选集中选择具有代表性的类别子集。在我们的实现中，这是通过首先按照出现频率的降序对每个脚本的片段进行排序来完成的。如果两个片段在频率范围内，则按照它们所代表的字符数的升序对它们重新排序，然后按它们所包含的连接组件进行排序，以此类推。此重新排序步骤会优先考虑具有简单非复杂形状的片段，因为它们倾向于形状分类器更易于编码和表示。然后，将每个目标脚本的有序集修剪到总xs%的顶部出现次数（覆盖率）。xs的值控制要学习的类的数量脚本，并且每个脚本可以独立更改，以权衡处□□理时间和分类错误。我们的实现方式是将所有脚本的xs值设置为相等的低值，例如10%，然后逐步逐步增加每个脚本的值，直到针对受训字体的所有脚本的分类误差不再减小为止。第4节详细介绍了分类时执行的步骤。此步骤的最终结果是为每个脚本选择了一组单词片段。

3.3形状分类器训练最后一步是使用选定的单词片段来训练形状分类器。我们的实现使用第4节中概述的Tesser-act的静态字符分类器，在[8]中也有详细介绍。训练分类器所需的输入包括文本图像，以及一组图像边界框坐标和相关的字符文本。每个要训练的角色。我们使用在生成选择步骤中修剪的单词片段列表来生成此数据。在候选类创建步骤（第3.1节）中，我们会跟踪文本语料库中遇到的每个单词中包含的片段。然后将每个脚本-字体对按照降序排列单词，直到它们共同包括为脚本选择的所有片段。然后将所选单词呈现为字体，并使用统计知识提取片段边界框（单词中包含的每个碎片）。脚本的所选类集中包含的片段的边界框坐标被写入文件，然后与渲染图像一起用于训练字符分类器。以这种方式使用先前从常见单词中获得的单词两个好处。首先，它允许字符分类者在经常看到的单词的上下文中获取诸如字符高度之类的辅助信息。其次，由此产生的盒像文件对形成了非常紧凑的训练数据。图3显示了与Kedage字体的Kannada脚本相对应的训练图像和片段框的示例。表1列出了脚本使用的训练类的组成。形状分类器总共训练了1808个类别，每个脚本使用一种字体。为了不被偶尔的包含大量数字表的页面所混淆，通常将常用的印度-阿拉伯数字添加到一组训练班中，并在表中的“通用”脚本名称下列出它们。

4.分类在运行时详细描述脚本和页面方向估计算法之前，我们简要概述一下Tesseract中使用的内部特征表示和特征分类器的形式。Tesseract的静态字符分类器使用两种类型的内部特征表示每个单词片段和两个阶段的分类[8]。训练模型中使用的特征是多边形轮廓的4维（（x，y）位置，方向和长度）片段，未知特征是3维的，通过将每个片段分成多个单元获得长度片段。分类的第一阶段，即类修剪器，使用类似于宽容散列[11]的技术来产生类的简短列表。人物课堂的第二阶段

Script name	#classes	Coverage (%)
Arabic	208	90%
阿拉伯语200 60%孟加拉语101		
30%普通10 100%西里尔语28		
90%德瓦纳加里223 10%希腊语7		
40%汉族578 25%汉高543		
30%希伯来语10 70%平假名28		
50%卡纳达语60 60%片假名68		
50%拉丁语30 60%泰米尔语15		
20%泰卢固语84 60%泰国14		
60%		

表1：跨脚本的训练集中的1808类的组成：覆盖率值xs反映了过滤后片段集中的选定类发生的累积频率（第4节中的步骤2a）。

选区包括将来自训练的多边形段与从未知形状的轮廓获得的多边形段进行匹配。如果两个段相对于（x，y）位置和角度是近端的，则声明为匹配。将两个形状之间的距离计算为原型形状中距离未知形状的距离段的加权平均值，反之亦然。尽管第二步可能在计算上很昂贵，但它仅在类修剪器返回的短列表中的类上执行。注意与未知形状之间的最小形状距离的类，并将与该类关联的文本的脚本视为该形状脚本的最佳估计。 Fraktur被视为一个单独的脚本，尽管它是使用与受过训练的类关联的字体属性的知识而不是关联文本的脚本来检测的。给定页面图像作为输入，运行时的过程如下：

1.对图像进行二值化并将其分割为连接的组件。具有水平信号重叠的成组连接的组件成为斑点。注意，在此阶段需要在线查找算法或连接组件的订购过程。但是，在预期的输入类型上，可能需要找到非文本区域的算法[9、10]作为此之前的预处理步骤，以便从考虑中消除那些区域中错误检测到的斑点。但是，这通常是对OCR的普遍要求，而不是对所提出算法的限制。

2.对于图像中所有斑点的随机选择的N大小子集的每个斑点：

（a）如果Blob的长宽比超过某些阈值，则其高度或宽度超出可接受的值范围以包含有效文本字符，或具有可能容易与其他字符或非文本形状混淆的其他指示符，请拒绝并继续。

（b）分类斑点，并找到最可能的脚本所属。如果此脚本的最佳估计值的信度得分低，或在脚本的下一个最佳估计的信度范围内，则拒绝该blob并继续。

（c）累积与剧本最佳估计相关的置信度分数。

（d）如果估计类的字体属性表明未知形状是用Fraktur字体呈现的，则将“ Frak-tur”脚本的计数加1。否则，查找估计类的脚本，并将脚本的计数增加1。

（e）在其他三个可能的方向（从输入方向成90°，180°和270°）旋转连接的组件后，重复2a-2d，最终结果是四组，每个检查方向一组每个感兴趣的脚本都带有针对该方向的累积信心分数。

3.选择具有较高总置信度得分的方向作为页面方向的最佳估计。

4.选择计数最高的脚本。

某些脚本（例如韩语和日语）严格来说是伪脚本，其文本由属于其他“真实”脚本的字符组合组成。例如，日文以片假名，平假名和韩文文字的组合形式存在，而韩语则以韩文汉文字的组合形式存在。我们解决了使用小数计数来识别此类伪文字的问题。如果未知形状的估计文字是汉字，则不仅将汉字的数量增加1，而且将日文和韩文的数量增加，也可以考虑到真实文字也可能是韩文或日文的事实。但比例较小。通过从后两个脚本的文本语料库中分析日文和韩文汉字符号出现的自然频率，可以估算出这些分数的最佳值。在我们的实现中，用于估计日文和韩文文字的汉字权重分别选择为0.2和0.6。

5.评估和实验结果我们对从15个脚本（包括Fraktur）中的任何一个以大致相等的比例扫描书籍获得的1846多页文档的数据集上评估了该算法。每个文档的真实性可以作为主要方向，也可以作为文档包含的一个或多个脚本的列表。该算法在文档的10个随机采样页面上运行，每页最多250个blob。定向的信心分数和脚本估计值在各个页面上取平均值。这样可以确保脚本的估计值不会因

真实的脚本 GROUND TRUTH	估计scriptara ben cyr dev frk gre han heb jpn kan kor lat tam tel tha ESTIMATED SCRIPT
ara	100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
ben	0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
cyr	0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
dev	0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
frk	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
gre	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
汉	3.14 0.00 0.00 0.00 0.00 0.00 92.15 0.00 4.71 0.00 0.00 0.00 0.00 0.00 0.00
heb	2.01 0.00 0.00 0.00 0.00 0.00 0.00 97.99 0.00 0.00 0.00 0.00 0.00 0.00 0.00
jpn	0.000.00 1.03 0.00 0.00 0.00 4.12 0.00 94.85 0.00 0.00 0.00 0.00 0.00 0.00
kan	2.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 97.50 0.00 0.00 0.00 0.00 0.00
kor	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 3.12 0.00 96.88 0.00 0.00 0.00 0.00
lat	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00
tam	0.95 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0 .00 0.00 0.00 0.00 99.05 0.00 0.00
tel	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
tha	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.01 0.00 0.00 0.00 0.00 0.00 98.99

表2：1846年文档数据集上脚本检测结果的混淆矩阵：值准确无误，以百分比为单位。行对应于真实的脚本名称，列对应于由代码给出的估计脚本名称：ara =阿拉伯文，ben =孟加拉文，cyr =西里尔文，dev = Devanagari，frk = Fraktur，gre =希腊文，han =韩文，heb =希伯来语，jpn =日语，kan = kannada，kor =韩国语，lat =拉丁语，tam =泰米尔语，tel =泰卢固语，tha =泰语。

即使文档可能主要使用不同的脚本，但包含序言或作者前言的页面都带有密集文本（例如英语（拉丁语））。脚本和页面方向检测是按类别进行分类的问题，因此，选择的错误度量标准是修改的指示符函数，如果脚本的最佳估计不在地面脚本列表中，则定义为1，否则定义为0。由于我们的方法将按置信度分数排序的脚本列表作为输出，因此可以考虑本质上离散和连续的其他度量。例如，如果两个置信度最高的脚本中“前两个错误”度量标准都可以定义为1分数在真实的脚本列表中。为简洁起见，本文仅报告经过修改的指标函数产生的错误。在1846年的文档数据集中，我们的实验记录了方向错误率为0.2%，脚本识别错误率为1.84%。我们不使用任何班级先验，并且所有候选脚本和方向都经过同等对待，并被认为具有同等的可能性。表2显示了真实脚本和估算脚本的混淆矩阵。处理页面的时间因页面内容而异，通常可与Tesseract中使用的阈值算法所花费的时间相媲美。我们观察到阈值器花费的时间为0.3-1.2秒，而分类器（以允许在页面中处理的最大Blob数量为时间限制）花费0.5-0.6秒。

与基于文本的语言ID算法的比较：

我们还将提出的算法与基于替代文本的方法进行了比较。如第1.1节中概述的那样，竞争方法的工作原理是：使用经过飞行员语言训练的OCR引擎处理整个页面，并分析乱码输出文本的统计信息。我们采用这种方法的实现使用OCR引擎对每个图像进行了两次处理-首先以识别拉丁字符的方式进行第二次以一种模式认出汉族。在每种情况下，都对输出文本的单字母组合统计进行了分析，以确定该文本是拉丁文字还是其他一些脚本家族的可能性。由于此类方法最初并非旨在处理非垂直方向的，所以我们针对提议的算法，在一组都具有垂直方向的较小的多页文档上进行了评估。我们发现基于文本的方法的错误率约为7.52%，而提出的方法的错误率较低，为2.11%。提议的方法也快了6-10倍，这主要是因为它不需要输入任何图像全部处理。

6.结论本文提出了一种简单而有效的算法，以使用Tesser-act形状分类器来组合脚本和页面方向检测。进行了许多观察和设计选择，使建议的方法可以很好地工作。一个观察是，形状类的覆盖率分布在许多脚本中比较陡峭，这使得生成类选择方案在使用少量训练类的情况下也能很好地工作。使用“局部”方法在单个连接的组件级别上进行操作以及对斑点旋转和形状分类操作进行交错的设计选择使整个算法有效。但是，仍然有一些故障情况需要解决。从表2的混淆矩阵中还可以看到，大量的分类错误是由于日语脚本被误认为汉字，反之亦然。通常，这是脚本检测算法常见的错误来源，主要是由于这两个脚本具有许多共同的符号。其他失败情况是，文档包含退化的手写文本，或者具有不从中删除的异常图像或线条画预先考虑

处理步骤，或使用未经训练的字体脚本。这些错误源中有许多是
processing, or using fonts not trained on. Many of these sources of error are shared with
与更一般的OCR问题共享的，并且是正在进行的工作的主题。
the more general OCR problem and are the subject of ongoing work.

7. ACKNOWLEDGMENTS

7.致谢作者感谢Dar-Shyang Lee设计和实施了评估该工作的基于文本的语言识别算法，以及他的有用建议和反馈。

8.参考资料[1] Tesseract开源OCR引擎。http：//code.google.com/p/tesseract-ocr。[2] S. Chaudhury，R。Sheth：印度语言的可训练脚本识别策略，Proc.5th IEEE Intl。 Conf。文件分析与识别（ICDAR），第657-680页，1999年。[3] A. Busch，W。Boles，S。Sridharan：《脚本识别的纹理》，IEEE Trans。模式分析与机器智能（PAMI），27（11），第1720-1732页，2005年。[4] J. Hochberg，P。Kelly，T。Thomas，L。Kerns：使用基于集群的模板从DocumentImages自动识别脚本，IEEE《模式分析与机器智能》（PAMI），第176-181页，1997年。[5] A. L. Spitz：确定文档图像的脚本和语言内容，IEEE跨模式分析和机器智能（PAMI），pp。235-245，1997年。[6] H. Ma，D. Doermann：基于Gabor过滤器的多分类器，用于扫描的文档图像，Proc。第七届IEEE国际Conf。文件分析与识别（ICDAR），第968-972页，2003年。[7] L. J. Zhou，Y。Lu，C。L. Tan：孟加拉语/基于连接组件轮廓分析的英语脚本识别，第七届IAPR文件分析系统研讨会（pps），第243-254页，2006年。[8] R. Smith：Tesseract OCR引擎概述。第9 IEEE国际机场Conf。文件分析与识别（ICDAR），第629-633页，2007年。[9] Leptonica图像处理和分析库。http：//www.leptonica.com。[10] R.Smith：Proc。通过Tab-stopDetection进行混合页面布局。第十届IEEE国际Conf。文件分析与识别（ICDAR），2009年。[11] S. Baluja和M. Covell：学习宽恕的哈希函数：算法和大规模测试，国际人工智慧联合会议（IJCAI），2007年