**布局分析的两种几何算法**

Thomas M. Breuel

摘要。提出了解决布局分析中两个关键问题的几何算法:用最大空矩形寻找文档背景空白的覆盖，以及在存在几何障碍的情况下寻找几何文本行模型的约束最大似然匹配。这些算法比以前的方法更容易实现，它们返回全局最优解，并且不需要试探法。该论文还引入了一个评估函数，可靠地识别对应于列边界的最大空矩形。将这种评估函数与两种几何算法相结合，就形成了一个易于实现的布局分析系统。UW3数据库的文件证明了系统的可靠性。

**1 Introduction**

已经提出了多种用于文档图像几何布局分析的算法。 其中包括基于形态学或“涂抹”的方法、投影轮廓（递归 X-Y 切割）、基于纹理的分析、背景结构分析等（有关评论和参考，请参见 [6]）。 虽然布局分析是一个比一般图像分割更简单的问题，但它仍然在几何算法和图像统计中提出了具有挑战性的问题。

本文提出了解决几何布局分析中两个关键问题的算法。 第一个是一种高效且易于实现的算法，用于根据矩形覆盖 分析文档的空白或背景结构。 背景结构分析作为文档布局分析的一种方法已被许多作者描述 [13,2,12,8,1,9]。 贝尔德等人的工作。 [2] 根据矩形覆盖物分析背景结构，这是一种计算方便且紧凑的背景表示。 然而，过去用于计算此类矩形覆盖的算法相当难以实现，需要许多几何数据结构并处理扫描期间出现的特殊情况（Baird，个人通信）。 尽管矩形覆盖物具有吸引人的特性，但这可能限制了此类方法的广泛采用。 本文提出的算法不需要实现几何数据结构，也不需要考虑特殊情况； 它可以用不到 100 行的 Java 代码来表达。 与以前的方法相比，它还按质量递减的顺序返回解决方案。

这里介绍的第二个算法是文本行查找算法，它可以在存在“障碍物”的情况下工作。 也就是说，给定页面上已知没有文本行（例如，列分隔符）的一组区域和字符边界框的集合，该算法将在高斯误差模型下找到全局最优的最大似然匹配到文本行，受限于没有文本行跨越 障碍。 相比之下，许多以前的消息到线查找（例如，投影方法、霍夫变换方法等）要么不能可靠地用于同一页面上的多列文档或多个文本方向，要么需要将完整的物理布局分割为 在应用之前使用统一的文本行方向分离文本区域。

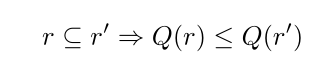
本文中介绍的每种算法在现有布局分析系统中都有有用的应用。 综上所述，这两种算法允许我们采用一种新的方法来进行文档布局分割。

传统的文档布局分析方法通常会首先尝试使用接近度、纹理或空白等特征将文档完整地全局分割成与列、标题和段落等实体相对应的不同几何区域。 然后针对文本行查找和 OCR 等任务分别考虑每个单独的区域。 这种方法的问题在于，将文档完整可靠地分割成单独的区域通常很难实现。 关于要组合哪些区域的一些决定很可能涉及对 OCR 系统输出的语义约束。 然而，为了能够首先将文档传递到 OCR 系统，我们必须已经识别出文本行，从而导致处理步骤之间的循环依赖。

相比之下，如果我们可以在存在障碍的情况下进行文本行查找，则不需要对文档进行完整的分割才能进行 OCR。 相反，所需要的只是识别垂直空间或分隔不同列中的文本的行。 事实证明，这个问题要简单得多。 使用本文中描述的空白分析算法，使用一种新颖的评估函数，可以非常可靠地完成它。

**2空白覆盖**

我们定义最大白色矩形问题如下。 假设我们在平面上有一组矩形 C = {,...,}，它们都包含在某个给定的边界矩形  中。 在布局分析中，通常会对应页面上连通组件的边界框，整体边界矩形  将代表整个页面。 此外，假设我们得到了矩形 Q 的评估函数： → R 满足，对于任何两个矩形 r 和 r'：

（1）

在 [8] 中描述的情况下，Q 函数只是矩形的面积，很容易看出满足等式 1 中表示的条件。 最大白色矩形问题是：在所有可能的矩形 r ⊆  中，找到一个矩形  ⊆  使 Q (T) 最大化，其中 r 不与 C 中的任何矩形重叠。或者，用数学符号表示：

where （2）

**2.2算法**

如上所述，最大空矩形问题有几种算法，包括来自计算几何（例如，[11]）和文档分析（例如，[2]）的算法。 然而，这种算法的实现往往相当复杂，并且没有得到广泛使用。

本文中针对最大空矩形问题提出的算法可用于点或矩形障碍物。 关键思想类似于快速排序或分支定界方法。 如图 1 所示。

图 1(a) 显示了算法的开始：我们给出了一个外边界和一组矩形（障碍物）。 如果边界内没有任何障碍物，那么我们就完成了：边界本身就是给定障碍物的最大矩形。

如果边界内包含一个或多个障碍物，我们选择其中一个矩形作为“枢轴”（图 1（b））。 一个不错的选择是在边界内居中的矩形。 鉴于我们知道最大矩形不能包含任何障碍物，特别是它不能包含枢轴。

因此，最大白色矩形问题的解决方案有四种可能性：在枢轴的左侧和右侧（图 1（c））或在枢轴的上方和下方（图 1（d））。

我们计算重叠这四个子矩形的障碍物，并评估每个子矩形内可能存在的最大空矩形的质量上限； 由于单调性（等式 1），应用于子矩形边界的质量函数 Q 用作上限。 将子矩形及其相关障碍物和质量插入优先队列，重复上述步骤，直到第一个无障碍矩形出现在优先队列的顶部； 该矩形是质量函数 Q 下最大空矩形问题的全局最优解。该算法在图 2 中以伪代码给出。

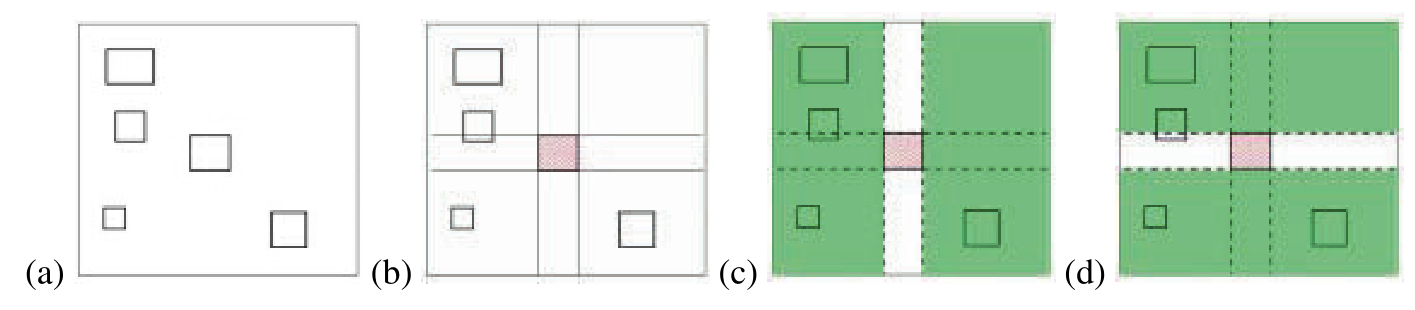


图 1. 说明分支定界空白覆盖算法的递归步骤的图。 看文字说明

找空白区域（边界，矩形）

插入边界质量，边界，矩形

如果队列不是空的：

则（q,r,障碍） = queue.dequeue\_max() 取出最大的障碍

如果没有障碍：

返回r(也就是最大的矩形边界)

枢轴 = pick（障碍）

r0 = (pivot.x1,r.y0,r.x1,r.y1)

...

subrectangles = [r0,r1,r2,r3]

对于每个子矩阵：

sub\_q = quality(sub\_r) 计算子矩阵质量（面积）

sub\_obstacles = [list of u in obstacles if not overlaps(u,sub\_r)]

如果障碍物与当前子矩阵重叠

queue.enqueue(sub\_q,sub\_r,sub\_obstacles)

插入子矩阵质量，子矩阵，与当前子矩阵重叠的障碍

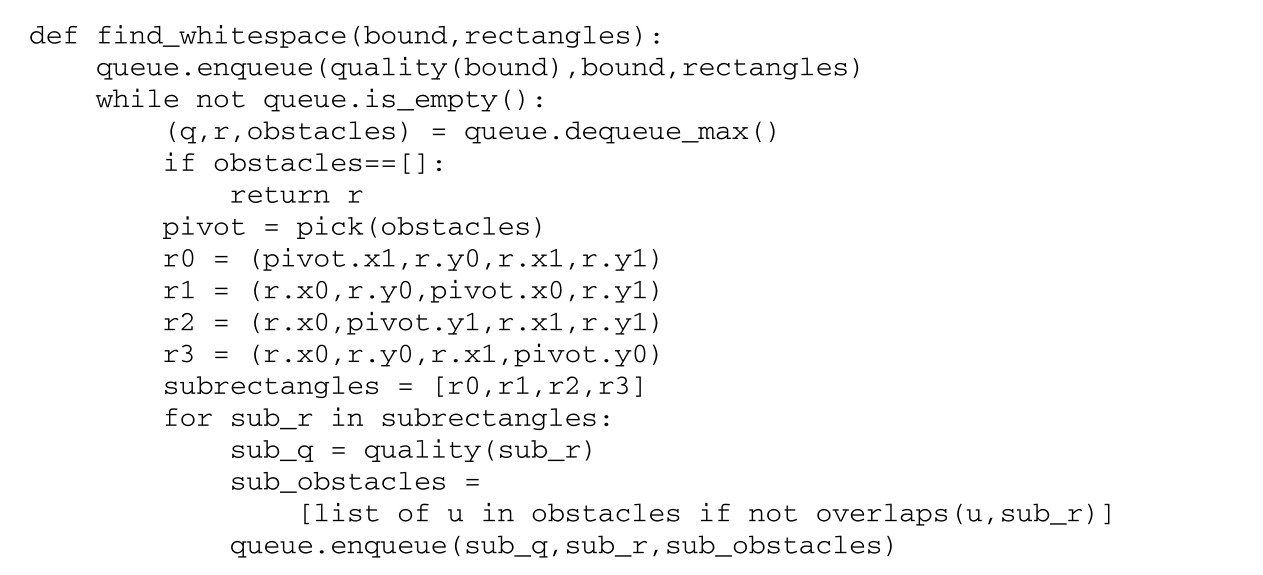


图2.用于查找全局最佳空白矩形的伪代码。完整的Java实现是约200行代码（语句）。

为了获得最优解，我们可以从优先级队列中不断扩展节点，直到按质量递减的顺序得到解。然而，这些解决方案中的许多将在很大程度上重叠。下面是寻找n个最佳解决方案的贪婪算法。找到最大空矩形后，我们可以将其添加到障碍列表中并继续扩展。当我们退出搜索状态时，我们检查障碍物列表是否已更改，如果已更改，则重新计算节点的质量并将其重新排队。这将导致贪婪地覆盖具有最大矩形的空白，并且比重新启动算法要快得多。

此外，我们可以允许它们之间的部分或绝对重叠，而不是坚持完全不相交的矩形的覆盖。仔细地寻找这种部分重叠的最大空矩形可能会在分区过程中将重叠约束合并到上界的计算中。然而，该算法在实际问题上运行速度足够快，我们需要的解的数量通常足够小，仅生成最大的空矩形就足够了，以便使用未修改的算法降低质量，测试任何新解与所有先前确定的解的重叠，并拒绝任何与以前找到的解决方案重叠过多的新解决方案。

图7显示了该算法在从UW3数据库中查找具有最大空矩形的文档贪婪覆盖的应用。使用一个400 MHz笔记本电脑上的算法实现C++的常用参数设置的计算时间在秒以内。事实上，该算法可以作为[8]使用的空白覆盖算法的替代品，对于任何有兴趣实现这种页面分割系统的人来说，它都是有用的。然而，在下文中，本文描述了使用不同评估标准的算法的替代用途。

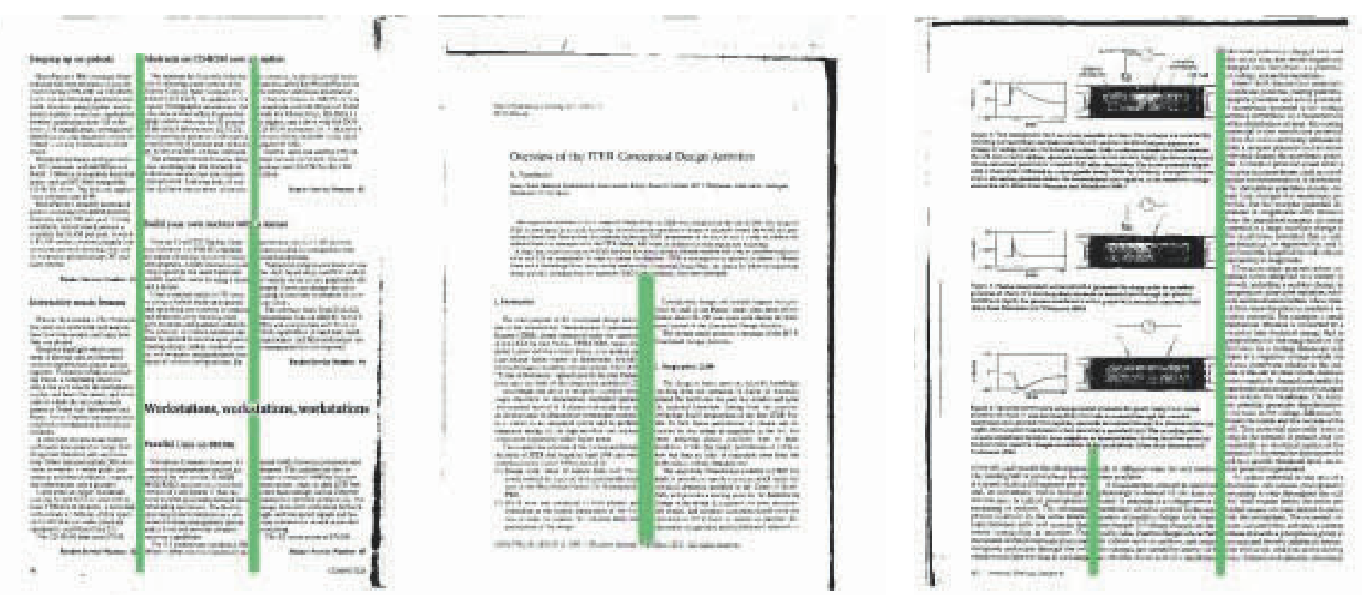


图7.用于检测具有复杂布局的文档（UW3数据库中的文档A00C、D050和E002）中的列边界的空白评估结果示例。请注意，即使是复杂的布局也由一小部分列分隔符描述。

**3约束线发现**

**3.1问题定义**

我们现在将转向第二个几何算法，一个用于在存在障碍物的情况下查找文本行的算法。。“障碍”将变成矩形（空白覆盖），由上一节中描述的算法找到的空白覆盖和下一节中描述的评估标准组成。通过采用类似的算法方法：分支和定界，约束寻线算法也与前一节中描述的算法相关联。

约束行查找在文档分析中解决的问题如下。许多文档包含多列文本。有些文档或文档图像甚至可能包含多个方向的文本，这可能是因为复杂的文档布局，也可能是因为（更常见的情况是）在同一图像中以稍微不同的旋转方式扫描书籍的两个面对面页面。因此，彼此靠近的文本行可能仍然具有不同的行参数。一些情况如图3所示。

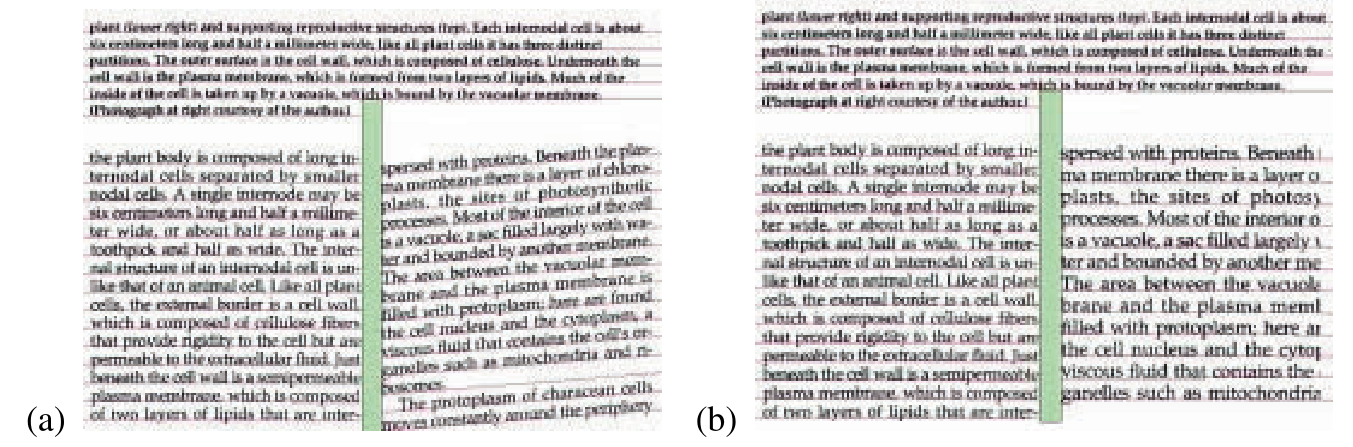
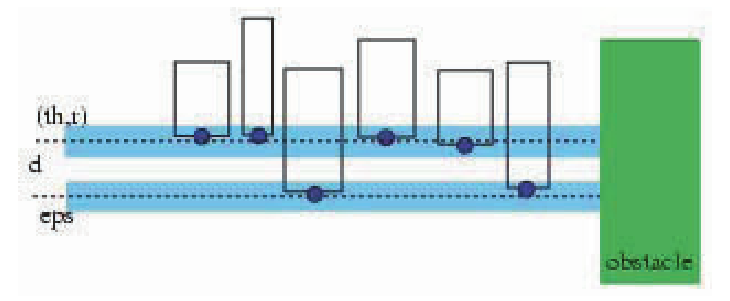


图3。约束寻线算法在页面模拟变体中的应用。 水槽（障碍物）是使用本文描述的算法自动发现的，并以绿色显示。文本行是使用约束行查找器找到的，显示为淡红色。（a） 两个相邻的列具有不同的方向（这通常发生在扫描书籍书脊的两侧）。（b） 两个相邻列的字体大小不同，因此基线不对齐。

传统的方法试图处理这种情况，首先找到一个完整和正确的页面分割，然后在每个文本块内执行行查找；也就是说，它们采用自上而下的分层方法。不幸的是，在不了解行结构的情况下，很难找到完整和正确的页面分割。页面布局分析的全局集成解决方案，如Liang等人[10]提出的解决方案，避免了这一问题，但实施起来似乎很复杂，到目前为止还没有得到广泛应用。

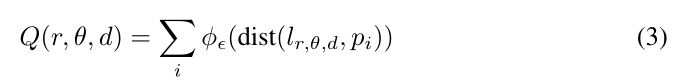
约束线查找提供了一种更简单的选择。一个受约束的测线器只需要一个文本行不会穿过的障碍物列表。这些障碍物通常是檐槽和一些图形元素，如图形或细垂直线。根据下面给出的结果，与完整（即使是临时）布局分析相比，查找檐槽似乎是一个简单得多的问题，即使是复杂的布局也往往有一个简单的檐槽结构（参见图7中的示例）。使用上一节中描述的空白覆盖方法可以很容易地识别这些檐槽。此外，本文描述的约束行查找方法还可以与方向无关的布局分析技术结合使用，允许我们在不完全分割的文本中查找任意方向的文本行。

本文中算法的约束文本行查找方法之前已用于几何对象识别[3]，并应用于文本行查找[5]。让我们用边界框底部中心的点（对齐点）表示页面上的每个字符。在没有错误的情况下，对于大多数罗马字体，每个这样的点要么位于基线上，要么位于与基线平行的另一条线上，即下行线。这如图4所示。



为了找到文本行模型与页面边界框的“最佳”匹配，我们使用了稳健的最小二乘模型。 也就是说，每个字符对文本行的整体匹配分数的贡献将受到对齐点到基线或下行线的距离的平方的惩罚，直到达到阈值。 如文献 [7] 中所示，该匹配分数对应于存在高斯位置误差和存在均匀噪声特征背景时的最大似然匹配。

让我们假设这些线是由它们与原点的距离r和它们法线的方向 θ 来参数化的。 附加参数 d 给出下行线到基线的距离。 这三个参数 (r, θ, d) 确定文本行模型。 如果页面上所有连通分量的对齐点由 {p1,...,pn} ⊆  给出，我们可以将匹配质量（与对数似然单调相关）函数表示为：



这里，dist(·,·)是欧几里得距离，φ是阈值函数



在所有参数上最大化 Q(r, θ, d) 为无约束寻线问题提供了全局最优解。 对于约束线寻找问题，我们考虑线段而不是线，并且需要找到不与给定障碍物相交的最大线段。

**3.2算法**

**几何匹配的分支定界法技术一：寻找无约束文本行查找问题全局最优解的算法**

文献[5]在前人关于几何匹配的分支定界法[4]的基础上，提出了一种寻找无约束文本行查找问题全局最优解的算法。这里我们将简要回顾一下无约束方法。

基本思想：考虑文本行参数的三维空间的矩形子集（参数框，行参数区间的笛卡尔积），并计算在这些子集上可实现的质量函数得到值的上界。（可以理解为：参数都是区间的，两两配对，计算出使得结果最优的一组参数）

假设集合A={a, b}，集合B={0, 1, 2}，则两个集合的笛卡尔积为{(a, 0), (a, 1), (a, 2), (b, 0), (b, 1), (b, 2)}。

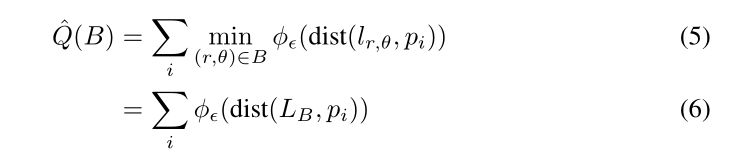
上界较大的子集被细分为较小的子集并重新求值。最终，在这个过程中得到的矩形子集足够小，能够以任何期望的数值精度约束优化问题的最优解，这是分支定界算法的一个实例。

为了适用于几何优化问题，需要克服两个困难：

首先，我们需要能够找到某个区域上质量函数 Q 的上界 ，

其次，我们需要能够有效地计算出上界 。[4] 描述了一框线参数 [,] × [,] 的上界  函数的计算。

让我们在这里简要回顾一下这种方法。 目前，为了简化讨论，只考虑基线，而不考虑下降线。 考虑用包含在参数框 B =[,] × [,] 中的参数的线条扫出的区域 。 我们使用作为我们的上界。 利用  的单调性，很容易看出这个界限是



其中阈值函数：x越大，结果越小

区域是蝴蝶结形状（ bow-tie shaped）的区域。 它在四个边上由线参数框的极值给出的线界定。 第五边以一条小圆弧为边界。 为了计算上界  ，我们需要计算一个点 p 与该区域的距离，或者至少是一个下界。 这个计算可以通过使用第五条线来限制圆弧来简化。 然后可以使用五个点积以及最小和最大操作的组合来计算距离 dist(,pi) 的下限，如 [4] 中更详细的描述。 对于下降线的计算，我们将 dist(,p) 替换为 min(dist(,p),dist,p))，其中 是由 与基线平行的下行线（有关更多详细信息，请参阅 [5]）。

**几何匹配的分支定界法技术二：匹配列表**

使用分支定界方法实现几何匹配问题变得简单高效的第二种技术是使用匹配列表。 也就是说，对于每个行参数框 B，我们维护一个列表，其中包含对质量函数 Q 做出非零贡献的所有对齐点。我们称这个列表为“匹配列表”。 当框 B 被细分时，只需要考虑匹配列表上的对齐点。

到目前为止，本节已经回顾了先前关于全局最优寻线的工作。 现在让我们转向如何将几何障碍引入到文本行查找的框架中的问题。 在查找有障碍物的文本行时，我们不允许文本行模型与障碍物相交的匹配。 这在图 5 中进行了说明。该图显示了两条候选线（虚线）。 一条线避开障碍物并匹配两侧的点。 另一条线更好地匹配障碍物一侧的点，但不能“拾取”障碍物另一侧的对齐点。 事实上，在约束文本行查找问题中，解决方案是文本行段，而不是无限行。

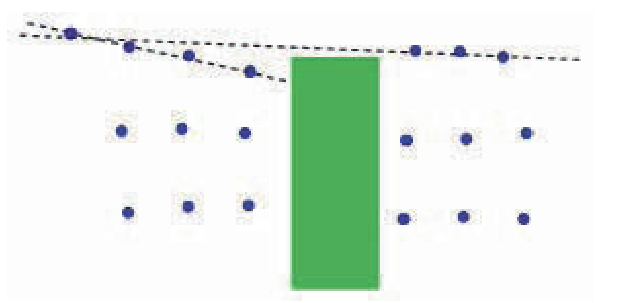


图 5. 带有障碍物的约束线寻找问题的图示。 矩形是障碍物，圆点代表要由一条线匹配的点。 显示了两条候选线：一条虚线匹配四个点但被障碍物挡住，另一条虚线匹配五个点并勉强避开障碍物。

也许令人惊讶的是，在分支定界文本行查找算法中加入障碍很简单，并且不会显着增加算法在实践中通常遇到的问题的复杂性。方法如下。

在分支定界评估期间，我们依次考虑较小的行参数框 B。当这些框很大时，它们的参数所隐含的一些线可能会与障碍物相交，而另一些则可能不会。

但是，随着参数的框越来越小，在某些时候，这些参数值对应的线要么都与障碍物相交，要么都无法与障碍物相交。

* 在所有线都无法与障碍物相交的情况下，我们只需在该参数框的后续细分中从进一步考虑中移除障碍物。
* 在所有线与障碍物相交的情况下，我们将可能匹配的对齐点集分成两个子集，位于障碍物左侧的子集和位于障碍物右侧的子集。

然后，我们使用相同的线参数框 B 和两个单独的匹配列表继续搜索，一个是障碍物左侧对齐点的匹配列表，另一个是障碍物右侧对齐点的匹配列表。 该算法在图 6 中的伪代码中给出。

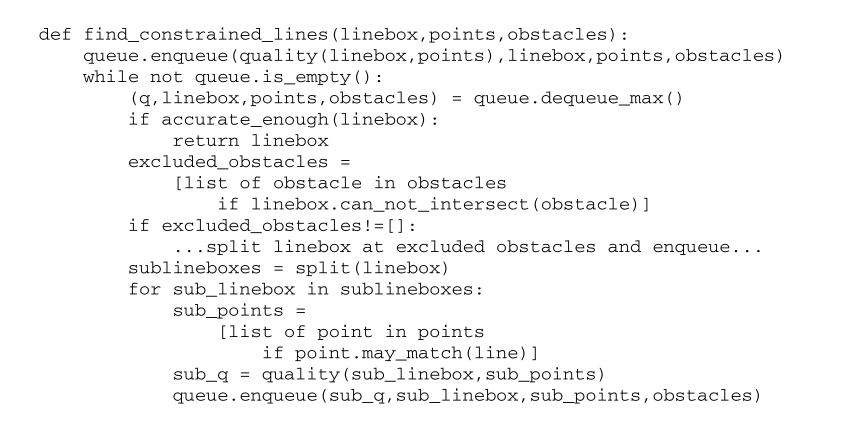


图 6. 寻找线模型对一组点的全局最优约束匹配的伪代码

这种与障碍物进行线匹配的方法不仅将匹配列表用作优化，而且还用于构建搜索并从进一步考虑中删除点。 算法找到的线段由匹配列表上的一组对齐点、障碍物和线隐式定义。 与我们直接在线段空间中进行搜索相比，这是一种更有效的方法。 为了找到具有基线的无障碍线段，这将是在五维参数空间上的搜索，而基于限制匹配列表的方法只需要在原始的三维参数空间中进行搜索。 因此，使用这种方法，有障碍物的文本行查找的运行时间与无障碍物的文本行查找的运行时间大致相同。

**4布局分析**

到目前为止，本文已经提出了两种在文档图像分析系统的实现中可能有用的几何算法。 计算空白覆盖的算法可用作 [8] 中使用的方法的易于实现的替代品。 在这项工作中，具有特定纵横比的矩形是首选，总体而言，较大的空白矩形优于较小的矩形。 它们的评估功能基于对实际文档中空白矩形分布的统计测量，旨在支持那些有意义的水平或垂直分隔符的矩形

为了测试基于面积、纵横比和页面位置的评估函数的性能，将上述空白覆盖算法应用于从UW3数据库中的文档图像获得的字符边界框。对于每个文档图像，提取200个最大的空白矩形集合，其成对重叠小于80%。正如预期的那样，这产生了一个几乎总是完全覆盖背景的空白矩形集合，以及侵入文本段落的额外空白矩形。为了进行布局分析，需要一个评估函数，该函数允许我们只选择其并集构成空白的矩形，以隔离文档布局的组件。

为了获得这样一个评价函数，训练一个决策树来估计给定的空白矩形是页面背景的一部分的概率。未尝试对性能进行正式评估，但目视检查表明，UW3数据库中的很大一部分文档无法使用此方法完全分割。如[8]中所述，高空白矩形通常分类正确，但对于宽空白矩形（将段落或部分彼此分隔开的矩形），则会出现大量正面和负面错误。Ittner和Baird的系统通过计算宽空白矩形来处理这些问题，但忽略伪宽矩形，直到后期处理阶段（在评估其方法时，它们不会被视为不正确）。此外，目视检查表明，仅基于空白矩形形状的规则或评估函数在所有情况下都不会可靠工作——UW3数据库包含的文档种类繁多，存在固有的歧义。

这意味着，虽然仅基于空白矩形形状的求值函数对于某种程度的文档集合可能有用且可靠，但对于非常异构的集合，我们可能需要另一种方法。综上所述，这些结果建议采取一种方法，将高空白单独分类，并在评估中考虑除空白矩形的形状和位置之外的其他特征。此外，一些观察结果表明，宽空白虽然有时在视觉上很突出，但对于沿垂直轴的文档布局分析来说既不必要也不充分。例如，在许多美式文档中，段落分隔符是通过缩进（而不是额外的空格）来表示的，从文档标题到正文文本的转换的转换是通过更改对齐（居中、左对齐、右对齐）来表示的，有些章节标题不是通过额外的间距来表示的，而是通过字体大小和样式的变化来表示的。这将导致文档布局分析的以下四步过程：

1. 找到高大的空白矩形并将它们评估为间隔、列分隔符等的候选者。
2. 查找符合文档柱状结构的文本行。
3. 根据相邻文本行的关系（缩进、大小、间距等）和内容（字体大小和样式等）识别垂直布局结构（标题、标题、段落）
4. 使用几何和语言信息确定阅读顺序

识别间距的关键思想，我们在这里指的是作为布局分析有意义部分的高大空白矩形，除了矩形的形状和位置之外，还要考虑它们与相邻文本的接近程度。这种约束是由文档结构以及在简单的最大白色矩形算法中的观察结果提出的，许多已识别的矩形将仅由其角附近的几个文本组件作为边界。基于对文档布局和可读性的考虑，我们可以暂时推导出一些我们希望应用于檐槽（沟渠）的规则（在未来的系统中，我们打算将这些约束基于预先分割的文档数据库的统计属性）：

– 沟渠的纵横比必须至少为 1:3

– 沟渠的宽度必须至少是词间空间宽度分布模式的 1.5 倍

– 此外，我们可能会包括由檐槽定义的最小文本列宽的先验知识

—沟渠必须在其左侧或右侧与至少四个字符大小的连接组件相邻（排水沟必须分隔一些东西，否则我们对它们不感兴趣）

为了测试该方法的可行性，将这些规则编码到一个空白评估函数中，并将空白覆盖算法应用于查找页面上的沟渠。为了评估性能，将该方法应用于UW3数据库“A”和“C”类中的221个文档页面。其中有73页多栏。该方法的输入由对应于文档图像的单词边界框组成。在检测到代表水槽的空白矩形后，使用约束线查找算法提取线。然后显示结果，覆盖地面真相，并进行目视检查。检查显示数据集上没有分割错误。也就是说，该方法返回的空白矩形不会拆分属于同一区域的任何行（如果空白矩形与该行的基线相交，则该行被视为“拆分”），并且作为单独区域一部分的所有行都由某个空格矩形分隔。图7显示了使用该方法实现的样本分割。

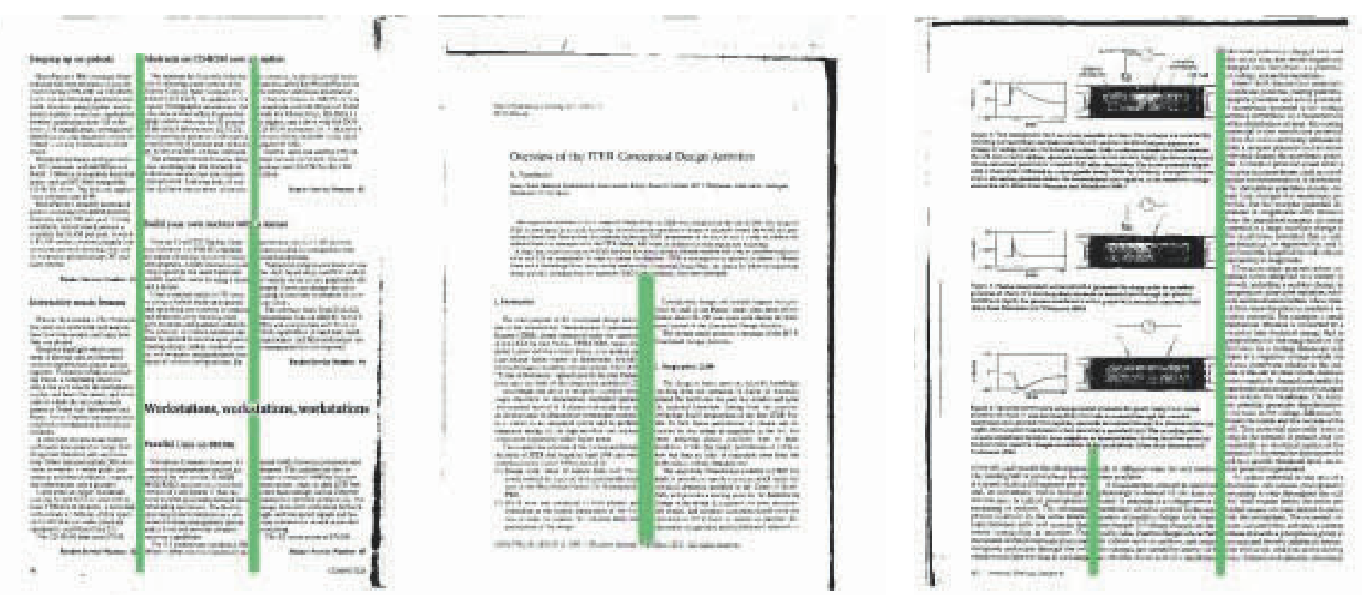


图7.用于检测具有复杂布局的文档（UW3数据库中的文档A00C、D050和E002）中的列边界的空白评估结果示例。请注意，即使是复杂的布局也由一小部分列分隔符描述。

**5 Discussion and Conclusions**

本文提出了两种几何算法。第一种算法在广泛的质量函数（包括面积）下，在矩形障碍物存在的情况下，找到最大空矩形问题的全局最优解。第二种算法在存在障碍物的情况下，找到文本行查找问题的全局最优最大似然解。这两种算法都易于实现和实用，在各种文档分析问题以及计算几何的其他领域都有应用。

这些算法构成了文档布局分析方法的基础，该方法集中于布局的两个最显著和最重要的方面：檐槽（文本的空白分隔列）和不穿过檐槽的最大文本行段。然后，可以在后续步骤中找到沿垂直维度的段落和其他布局结构。将此方法应用于UW3数据库表明分割错误率非常低（223页样本上没有错误）。结果还表明，以列分隔符、文本行和阅读顺序对页面进行描述是页面物理布局的一种非常紧凑和稳定的表示方式，与传统的层次表示方式相比，在布局分析的初始阶段可能是一个更好的目标。

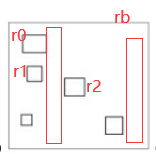
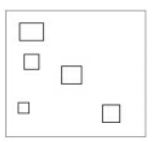
**总结：**

1. 根据矩形覆盖分析文档的空白或背景结构（按质量递减的顺序返回解决方案）
2. 文本行查找算法，可以在存在“障碍物（算法一找到的空白覆盖）”下工作（该算法将在高斯误差模型下找到全局最优的最大似然匹配到文本行，受限于没有文本行跨越障碍）。

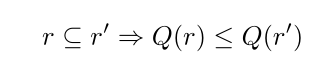
传统的文档布局分析方法通常会首先尝试使用接近度、纹理或空白等特征将文档完整地全局分割成与列、标题和段落等实体相对应的不同几何区域。这种方法的问题在于，将文档完整可靠地分割成单独的区域通常很难实现。

如果我们可以在存在障碍的情况下进行文本行查找，则不需要对文档进行完整的分割才能进行 OCR。本文中描述的空白分析算法，使用一种新颖的评估函数，可以非常可靠地完成它

**算法一：空白覆盖**



我们定义最大白色矩形问题如下。 假设我们在平面上有一组矩形 C = {,...,}，它们都包含在某个给定的边界矩形  中。 在布局分析中，通常会对应页面上连通组件的边界框，整体边界矩形  将代表整个页面。 此外，假设我们得到了矩形 Q 的评估函数： → R 满足，对于任何两个矩形 r 和 r'：你

（1）

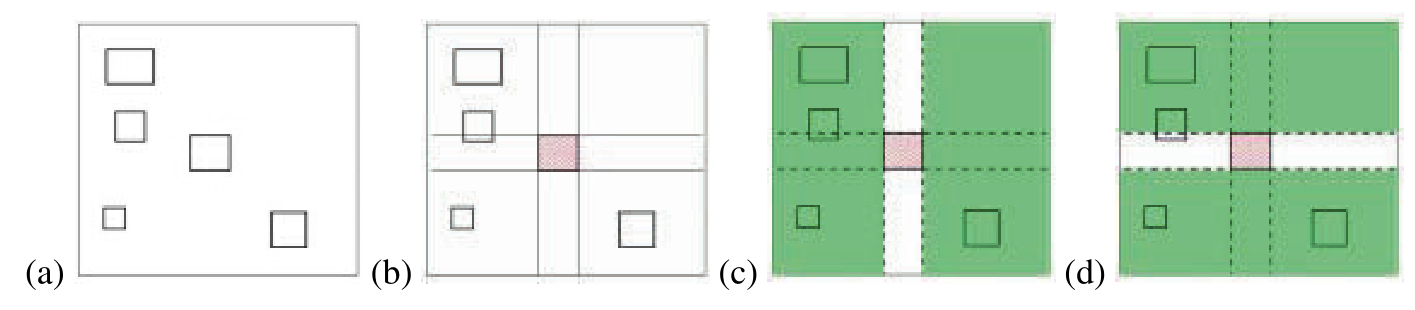
最大白色矩形问题是：在所有可能的矩形 r ⊆  中，找到一个矩形  ⊆  使 Q (T) 最大化，其中 r 不与 C 中的任何矩形重叠。（红色的矩形块）

本文中针对最大空矩形问题提出的算法可用于点或矩形障碍物。 关键思想类似于快速排序或分支定界方法。

图 1(a) 显示了算法的开始：我们给出了一个外边界和一组矩形（障碍物）。 如果边界内没有任何障碍物，那么我们就完成了：边界本身就是给定障碍物的最大矩形。

如果边界内包含一个或多个障碍物，我们选择其中一个矩形作为“枢轴”（图 1（b））。 一个不错的选择是在边界内居中的矩形。 因此，最大白色矩形问题的解决方案有四种可能性：在枢轴的左侧和右侧（图 1（c））或在枢轴的上方和下方（图 1（d））。

我们计算与这四个子矩形中的每一个重叠的障碍物，并评估每个子矩形内可能的最大空矩形的质量上限； 由于单调性（等式 1），应用于子矩形边界的质量函数 Q 用作上限。 将子矩形及其相关障碍物和质量插入优先队列，重复上述步骤，直到第一个无障碍矩形出现在优先队列的顶部； 该矩形是质量函数 Q 下最大空矩形问题的全局最优解。该算法在图 2 中以伪代码给出。



为了获得最优解，我们可以从优先级队列中不断扩展节点，直到按质量递减的顺序得到解（可以理解为一开始左边范围，从整个最大的绿色矩阵开始，有包含障碍，则减小面积，如果还包含障碍，再减少面积，以此类推，直到不包含障碍物位置，此矩形为最大空白矩阵）。然而，这些解决方案中的许多将在很大程度上重叠。

下面是寻找n个最佳解决方案的贪婪算法。找到最大空矩形后，我们可以将其添加到障碍列表中并继续扩展。当我们退出搜索状态时，我们检查障碍物列表是否已更改，如果已更改，则重新计算节点的质量并将其重新排队。这将导致贪婪地覆盖具有最大矩形的空白。

查找具有最大空矩形的文档贪婪覆盖的应用：

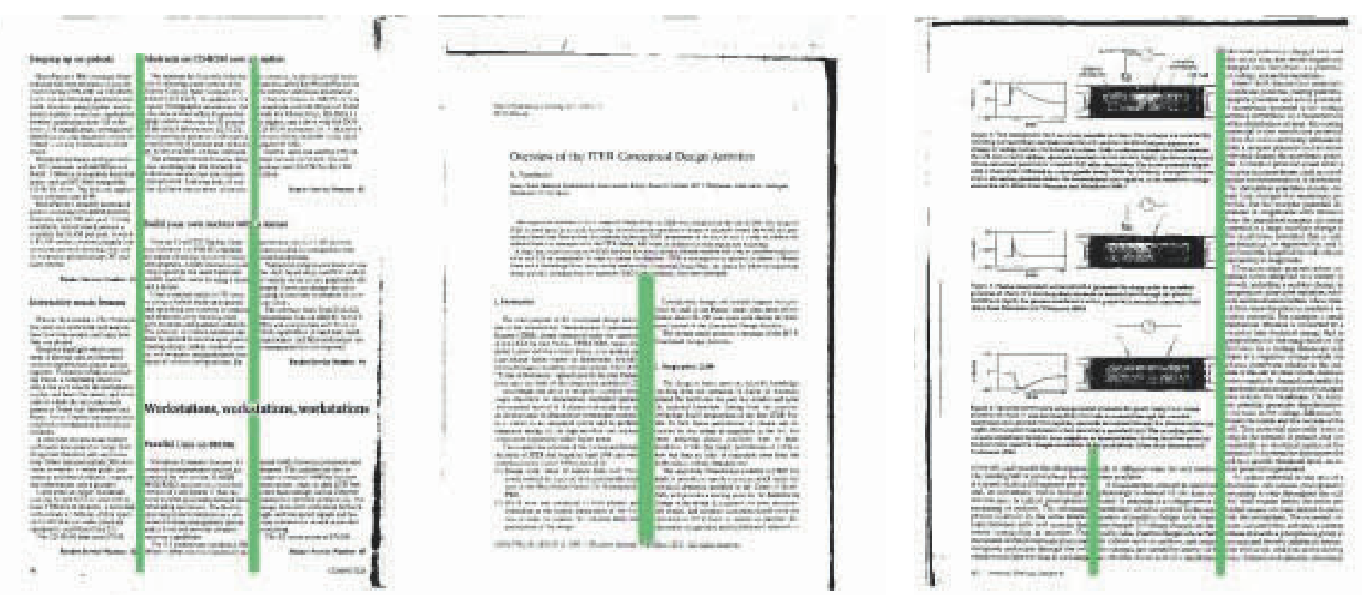
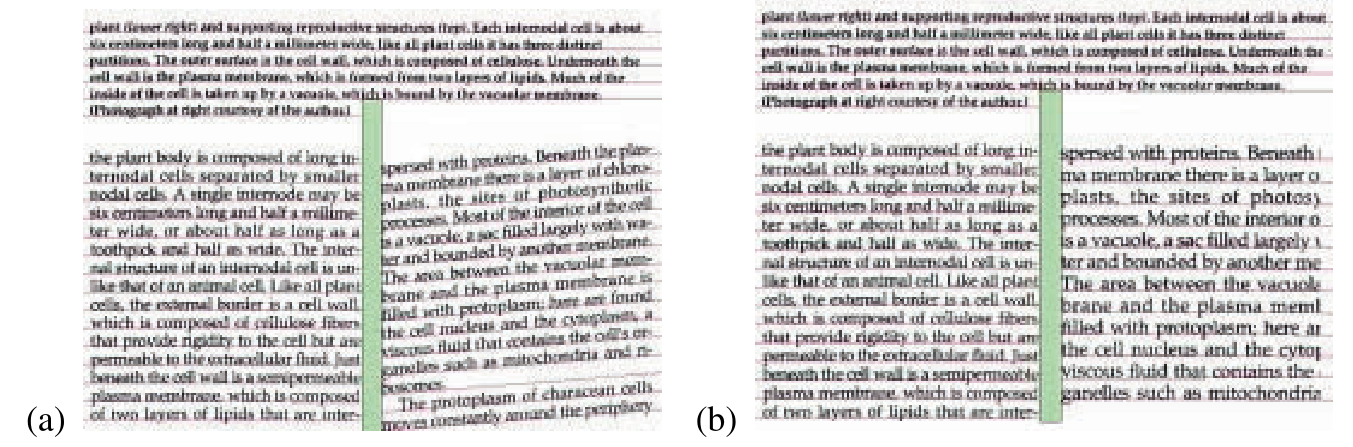


图7.用于检测具有复杂布局的文档（UW3数据库中的文档A00C、D050和E002）中的列边界的空白评估结果示例。请注意，即使是复杂的布局也由一小部分列分隔符描述。

**算法二：约束线查找**

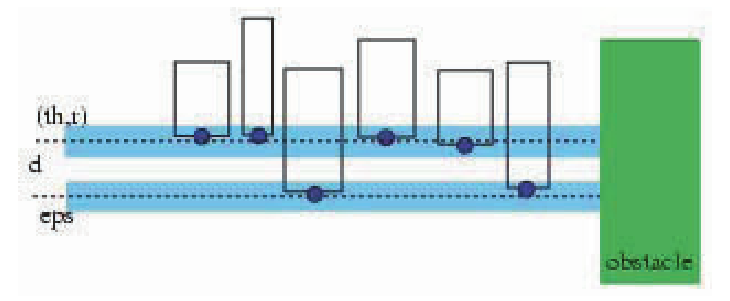
一个用于在存在障碍物的情况下查找文本行的算法，“障碍”将变成矩形（空白覆盖），由上一节中描述的算法找到的空白覆盖和下一节中描述的评估标准组成。

许多文档包含多列文本。有些文档或文档图像甚至可能包含多个方向的文本，主要是扫描原因，（在同一图像中以稍微不同的旋转方式扫描书籍的两个面对面页面）



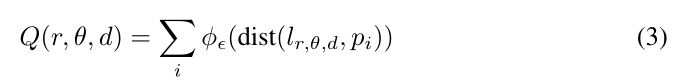
传统的方法是要得到完整核正确的页面分割（前提，很难做到），然后在每个块内执行行查找，而本文提出的约束线查找，只需要一个文本行不会穿过的障碍物列表。这些障碍物通常是檐槽和一些图形元素。使用上一节中描述的空白覆盖方法可以很容易地识别这些檐槽。

边界框底部中心的点（对齐点）表示页面上的每个字符。在没有错误的情况下，对于大多数罗马字体，每个这样的点要么位于基线上，要么位于与基线平行的另一条线上，即下行线。如下图所示。



为了找到文本行模型与页面边界框的“最佳”匹配，我们使用了稳健的最小二乘模型。 也就是说，每个字符对文本行的整体匹配分数的贡献将受到对齐点到基线或下行线的距离的平方的惩罚，直到达到阈值。 如文献 [7] 中所示，该匹配分数对应于存在高斯位置误差和存在均匀噪声特征背景时的最大似然匹配。

让我们假设这些线是由它们与原点的距离r和它们法线的方向 θ 来参数化的。 附加参数 d 给出下行线到基线的距离。 这三个参数 (r, θ, d) 确定文本行模型。 如果页面上所有连通分量的对齐点由 {p1,...,pn} ⊆  给出，我们可以将匹配质量（与对数似然单调相关）函数表示为：



这里，dist(·,·)是欧几里得距离，φ是阈值函数



在所有参数上最大化 Q(r, θ, d) 为无约束寻线问题提供了全局最优解。 对于约束线寻找问题，我们考虑线段而不是线，并且需要找到不与给定障碍物相交的最大线段。

**算法详解：**

使用分支定界方法实现几何匹配问题变得简单高效的第二种技术是使用匹配列表。 也就是说，对于每个行参数框 B，我们维护一个列表，其中包含对质量函数 Q 做出非零贡献的所有对齐点。我们称这个列表为“匹配列表”。 当框 B 被细分时，只需要考虑匹配列表上的对齐点。

转向如何将几何障碍引入到文本行查找的框架中的问题。 在查找有障碍物的文本行时，我们不允许文本行模型与障碍物相交的匹配。

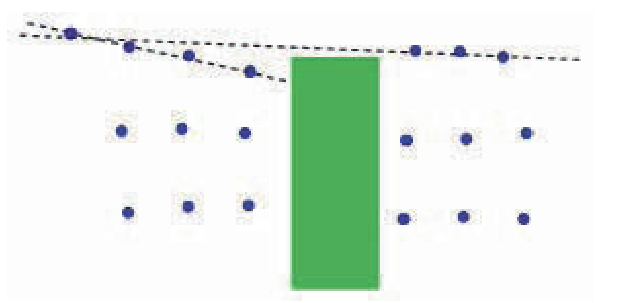


图 5. 矩形是障碍物，圆点代表要由一条线匹配的点。 显示了两条候选线：一条虚线匹配四个点但被障碍物挡住，另一条虚线匹配五个点并勉强避开障碍物。

在分支定界评估期间，我们依次考虑较小的行参数框 B。当这些框很大时，它们的参数所隐含的一些线可能会与障碍物相交，而另一些则可能不会。

但是，随着参数的框越来越小，在某些时候，这些参数值对应的线要么都与障碍物相交，要么都无法与障碍物相交。

* 在所有线都无法与障碍物相交的情况下，我们只需在该参数框的后续细分中将障碍物从进一步考虑中移除。
* 在所有线与障碍物相交的情况下，我们将可能匹配的对齐点集分成两个子集，位于障碍物左侧的子集和位于障碍物右侧的子集。

然后，我们使用相同的线参数框 B 和两个单独的匹配列表继续搜索，一个是障碍物左侧对齐点的匹配列表，另一个是障碍物右侧对齐点的匹配列表。 该算法在图 6 中的伪代码中给出。

这种与障碍物进行线匹配的方法不仅将匹配列表用作优化，而且还用于构建搜索并从进一步考虑中删除点。 算法找到的线段由匹配列表上的一组对齐点、障碍物和线隐式定义。 与我们直接在线段空间中进行搜索相比，这是一种更有效的方法。 为了找到具有基线的无障碍线段，这将是在五维参数空间上的搜索，而基于限制匹配列表的方法只需要在原始的三维参数空间中进行搜索。

**评估：**

具有特定纵横比的矩形是首选，总体而言，较大的空白矩形优于较小的矩形。

它们的评估功能基于对实际文档中空白矩形分布的统计测量，旨在支持那些有意义的水平或垂直分隔符的矩形

对于每个文档图像，提取200个最大的空白矩形集合，其成对重叠小于80%。正如预期的那样，这产生了一个几乎总是完全覆盖背景的空白矩形集合，以及侵入文本段落的额外空白矩形。为了进行布局分析，需要一个评估函数，该函数允许我们只选择其并集构成空白的矩形，以隔离文档布局的组件。

训练一个决策树来估计给定的空白矩形是页面背景的一部分的概率

如[8]中所述，高空白矩形通常分类正确，但对于宽空白矩形（将段落或部分彼此分隔开的矩形），则会出现大量正面和负面错误。Ittner和Baird的系统通过计算宽空白矩形来处理这些问题，但忽略伪宽矩形，直到后期处理阶段（在评估其方法时，它们不会被视为不正确）。此外，目视检查表明，仅基于空白矩形形状的规则或评估函数在所有情况下都不会可靠工作——UW3数据库包含的文档种类繁多，存在固有的歧义。

这意味着，虽然仅基于空白矩形形状的求值函数对于某种程度的文档集合可能有用且可靠，但对于非常异构的集合，我们可能需要另一种方法。综上所述，这些结果建议采取一种方法，将高空白单独分类，并在评估中考虑除空白矩形的形状和位置之外的其他特征。

**文档布局分析的四步过程：**

1.找到高大的空白矩形并将它们评估为间隔、列分隔符等的候选者。

2.查找符合文档柱状结构的文本行。

3.根据相邻文本行的关系（缩进、大小、间距等）和内容（字体大小和样式等）识别垂直布局结构（标题、标题、段落）

4.使用几何和语言信息确定阅读顺序

**应用于沟渠的规则：**

识别间距的关键思想，我们在这里指的是作为布局分析有意义部分的高大空白矩形，除了矩形的形状和位置之外，还要考虑它们与相邻文本的接近程度。这种约束是由文档结构以及在简单的最大白色矩形算法中的观察结果提出的，许多已识别的矩形将仅由其角附近的几个文本组件作为边界。基于对文档布局和可读性的考虑，我们可以暂时推导出一些我们希望应用于檐槽（沟渠）的规则

– 沟渠的纵横比必须至少为 1:3

– 沟渠的宽度必须至少是词间空间宽度分布模式的 1.5 倍

– 此外，我们可能会包括由檐槽定义的最小文本列宽的先验知识

– 沟渠必须在其左侧或右侧与至少四个字符大小的连接组件相邻（排水沟必须分隔一些东西，否则我们对它们不感兴趣）

**空白评估函数（没给出）**

该方法集中于布局的两个最显著和最重要的方面：檐槽（文本的空白分隔列）和不穿过檐槽的最大文本行段，在检测到代表水槽的空白矩形后，使用约束线查找算法提取线

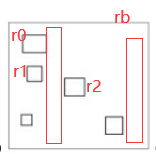
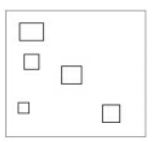
**简洁总结：**

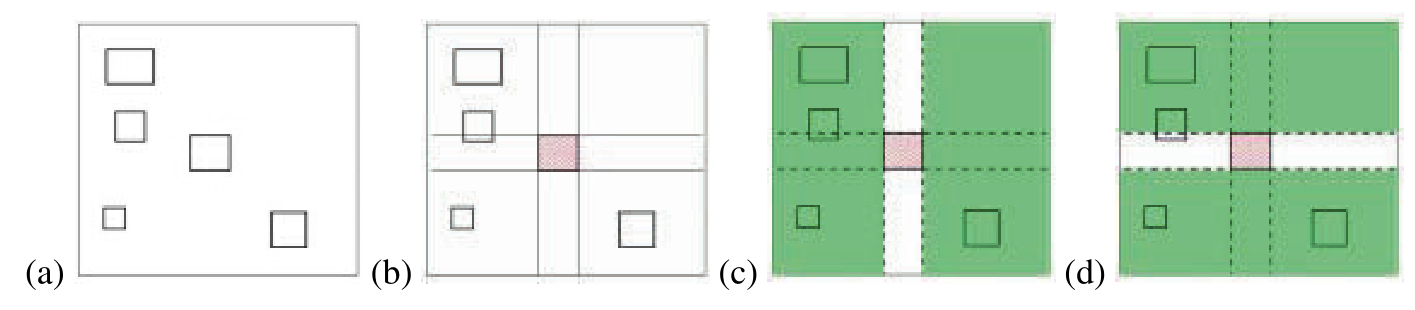
1. 根据矩形覆盖分析文档的空白或背景结构（按质量递减的顺序返回解决方案）
2. 文本行查找算法，可以在存在“障碍物（算法一找到的空白覆盖）”下工作

传统的布局分析方法需要将文档分割为不同几何区域，问题在于，将文档完整可靠地分割成单独的区域通常很难实现。

如果我们可以在存在障碍的情况下进行文本行查找，则不需要对文档进行完整的分割才能进行 OCR。本文中描述的空白分析算法，使用一种新颖的评估函数（只给出了规则），可以非常可靠地完成它。

**算法一：空白覆盖**





①矩形为障碍物，选择居中的矩形作为枢轴，则最大白色矩形解决方案有四种可能性（绿色）

②计算与这四个子矩形中的每一个重叠的障碍物，将子矩形及其相关障碍物和质量插入优先队列，重复上述步骤，直到第一个无障碍矩形出现在优先队列的顶部

③为了获得最优解，我们可以从优先级队列中不断扩展节点，直到按质量递减的顺序得到解（可以理解为一开始左边范围，从整个最大的绿色矩阵开始，有包含障碍，则减小面积，如果还包含障碍，再减少面积，以此类推，直到不包含障碍物位置，此矩形为最大空白矩阵）

④寻找n个最佳解决方案的贪婪算法，找到最大空矩形后，我们可以将其添加到障碍列表中并继续扩展

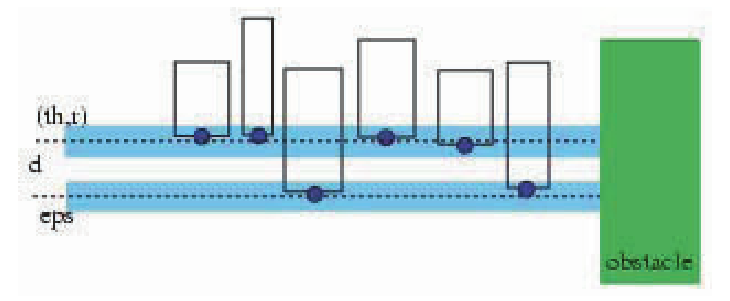
**算法二：约束线查找**

“障碍”将变成矩形（空白覆盖）

传统的方法是要得到完整核正确的页面分割（前提，很难做到），然后在每个块内执行行查找。

而本文提出的约束线查找，只需要一个文本行不会穿过的障碍物列表。这些障碍物通常是水槽和一些图形元素。使用上一节中描述的空白覆盖方法可以很容易地识别这些檐槽。

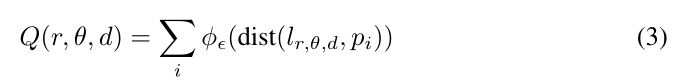
边界框底部中心的点（对齐点）表示页面上的每个字符。对于大多数罗马字体，每个这样的点要么位于基线上，要么位于与基线平行的另一条线上，即下行线。如下图所示。



为了找到文本行模型与页面边界框的“最佳”匹配，我们使用了稳健的最小二乘模型。 也就是说，每个字符对文本行的整体匹配分数的贡献将受到对齐点到基线或下行线的距离的平方的惩罚，直到达到阈值。

这些线是由它们与原点的距离r和它们法线的方向 θ 来参数化的。 附加参数 d 给出下行线到基线的距离。

匹配质量函数

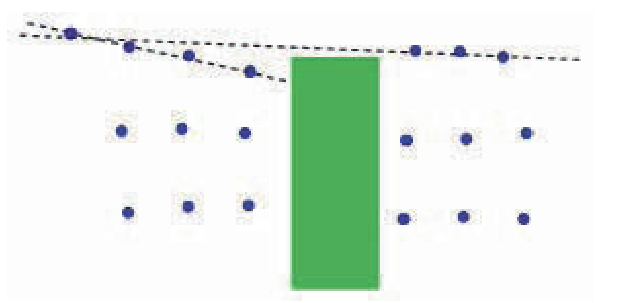




一种方法最大化匹配质量函数：是直接在线段空间进行搜索，在五维参数空间上的搜索

另外一种：匹配列表。 也就是说，对于每个行参数框 B，我们维护一个列表，其中包含对质量函数 Q 做出非零贡献的所有对齐点。我们称这个列表为“匹配列表”。 当框 B 被细分时，只需要考虑匹配列表上的对齐点。

如何将几何障碍引入到文本行查找的框架中的问题。 在查找有障碍物的文本行时，我们不允许文本行模型与障碍物相交的匹配。



分支定界评估期间，我们依次考虑较小的行参数框 B，但是随着参数的框越来越小，在某些时候，这些参数值对应的线要么都与障碍物相交，要么都无法与障碍物相交。

* 在所有线都无法与障碍物相交的情况下，我们只需在该参数框的后续细分中将障碍物从进一步考虑中移除。
* 在所有线与障碍物相交的情况下，我们将可能匹配的对齐点集分成两个子集，位于障碍物左侧的子集和位于障碍物右侧的子集。

然后，我们使用相同的线参数框 B 和两个单独的匹配列表继续搜索，一个是障碍物左侧对齐点的匹配列表，另一个是障碍物右侧对齐点的匹配列表。

**评估：**(**没有给出评估函数**)

**文档布局分析的四步过程：**

1.找到高大的空白矩形并将它们评估为间隔、列分隔符等的候选者。

2.查找符合文档柱状结构的文本行。

3.根据相邻文本行的关系（缩进、大小、间距等）和内容（字体大小和样式等）识别垂直布局结构（标题、标题、段落）

4.使用几何和语言信息确定阅读顺序

**应用于水沟的规则：**

– 沟渠的纵横比必须至少为 1:3

– 沟渠的宽度必须至少是词间空间宽度分布模式的 1.5 倍

– 此外，我们可能会包括由檐槽定义的最小文本列宽的先验知识

– 沟渠必须在其左侧或右侧与至少四个字符大小的连接组件相邻（排水沟必须分隔一些东西，否则我们对它们不感兴趣）