A review of branch and band algorithms for geometric and statistical layout analysis

Thomas M. Breuel

A Review of Branch-and-Bound Algorithms for Geometric and Statistical Layout Analysis

Thomas M. Breuel

University of Kaiserslautern and DFKI

tmb@informatik.uni-kl.de

Résumé : *Many different approaches to the geometric and statistical analysis of document layouts have been proposed in the literature. The development of practical branch-and-bound algorithms for solving geometric matching problems under noise and uncertainty has enabled the formulation of new classes of geometric layout analysis methods based on globally optimal maximum likelihood interpretations for well-defined models of the spatial statistics of document images. I review this approach to geometric layout analysis using text line finding and column finding in the presence of noise and uncertainty as examples and compare the approach with selected other statistical and geometric layout analysis methods.*

Mots-clés : document layout analysis, geometric matching, text line finding, branch-and-bound algorithms, global optimization

1 Introduction

In addition to their purely textual content, rendered documents contain a wealth of information in the geometric arrangement of the text and figures on the page–the *page layout*. Examples of properties encoded in the page layout are information about which text corresponds to the title, author, page number, and abstract of a document, the order in which the body text is to be read (the *reading order*), and major logical divisions in the body text. Recovering this information is the problem of *document layout analysis*.

Document layout information has a variety of uses. It is a key step in the conversion of scanned documents into machine readable form ; that is, what we typically think of "optical character recognition" (OCR) actually comprises both layout analysis and recognition of individual characters. In fact, even the recognition of characters in OCR depends on correct document layout analysis, since the interpretation of certain characters is affected by their position relative to the text line and since statistical language models depend on the correct reading order of the text. Furthermore, the user of an OCR system usually expects to obtain not just a vector graphics file with thousands of characters placed at specific location in the image, but instead an editable and structured text file that contains text in its correct reading order and correctly identifies actual line and paragraph breaks.

But while OCR is perhaps the most important use of document layout analysis, it is not the only one. Document databases need to extract information that permits indexing and retrieval of documents. This information can often only be derived from the layout of the text (as opposed to the textual content or even font properties). For example, titles and authors of scientific papers tend to be printed at the top of the first page, centered, with the title immediately preceding the author and separated from the rest of the text by whitespace, properties that are recoverable by document layout analysis.

Another application of document layout analysis is image-based reformatting and reflow of documents, a technique that allows the display of scanned documents on small-screen devices without OCR errors and while preserving the appearance of the original document [BRE 02b].

2 Layout Primitives

The actual layout of a document is the result of the application of complex, interacting rules about where to place text on the page. Some of those rules are consequences of properties of the human visual system and attempts to make text more readable (e.g., keeping line lengths below a certain number of characters per line), others are the results of physical constraints (e.g., page size), constraints imposed by traditional type setting equipment (e.g., the use of straight and parallel text lines), convention (e.g., where page numbers and titles go), as well as stylistic and artistic considerations.

While layouts can become enormously complex, almost all layouts tend to be composed of a number of recurring primitives. The most important of these are text lines, text columns, sections, and paragraphs. Furthermore, these primitives have a number of common geometric relationships between them, defined by their relative size, spacing, alignment, and justification. We call the extraction of these primitives *physical document layout analysis*. We refer to the extraction of higher-level properties of a document (like titles, authors, page numbers, etc.) as *logical document layout analysis*. Logical document layout analysis generally makes use of physical document layout analysis to achieve its goals. This work deals primarily with physical document layout analysis, that is, the reliable extraction of primitives like text lines and text columns, and the geometric relationship between those primitives.

3 Previous Methods

A large number of physical document layout analysis techniques have been proposed in the literature ([CAT 98] provides a good overview). In order to perform their function, layout analysis techniques make assumptions (explicitly or

implicitly) about the geometric properties of the layouts that they are applied to. Commonly made assumptions, by different systems, are that text lines are parallel to each other, that text on the page is composed of characters of approximately the same font size, that paragraph boundaries are straight and perpendicular to text lines, that paragraph boundaries are rectangular, that larger logical divisions correspond to greater physical spacing of text, that paragraphs of text contain only limited amounts of whitespace, and many more.

Such assumptions are approximate expressions of the physical, stylistic, and perceptual constraints on layout analysis. In practice, none of the assumptions made by current layout analysis systems result in human-like performance on real documents ; that is, for each of those assumptions, there will be some classes of documents that violate those assumptions, yet are easily interpretable by a human reader. A complete taxonomy of the assumptions made by different document layout analysis methods goes beyond the scope of this paper, but we will try to examine important differences as they arise. While the models reviewed in this paper does not permit the interpretation of all documents that a human reader could easily interpret, it does relax some commonly made assumptions, which makes it applicable to a wider class of document layouts.

First, many previous physical layout analysis methods explicitly or implicitly assume that text lines on a page are (mostly) parallel to each other. For example, projection methods and Fourier transform methods for skew estimation assume that most of the text on the page is at a single orientation and then attempt to correct page rotation ("skew") and transform all text lines into lines that are oriented at multiples of 90^o relative to the page boundaries ; text lines are then identified in a second step in this corrected image (see [HAR 94, BAI 87, SRI 89, BAI 87, OKU 99] for further details). This assumption of a global page rotation of an otherwise rectilinear layout means that such methods are not applicable to some important classes of documents, such as those captured with digital cameras and hence subject to perspective distortions. Some previous methods for finding text lines do not require rectilinear layouts, but they have had to sacrifice precise geometric models of text lines (in terms of baselines, descenders, and ascenders) for simpler methods using proximity of connected components [O'G 93, KIS 98] text line finder described here does not make prior assumptions about text line parallelism, but still models the geometry of each text line precisely.

Another assumption commonly made by other layout analysis systems is that of a physically represented layout hierarchy. That is, layout components like sections, paragraphs, and text lines form a hierarchy based on inclusion : sections are composed of paragraphs, and paragraphs are composed of text lines. It seems plausible to take a top-down or bottom-up approach to recovering these layout components. That is, such methods first identify sections, then paragraphs, and finally text lines. Examples of such approaches are approaches like X-Y cuts [NAG 84] that successively split the page into smaller and smaller units top-down, and approaches that use proximity of connected components [O'G 93, KIS 98] or mathematical morphology [CHE 95] group elements on the page into successively higher level grouping in a bottom up manner. However, in practice, the logical layout hierarchy does not appear to be very well represented in simple geometric properties of the physical layout ; higher-level layout properties do not, in general correspond to larger amounts of whitespace or other large-scale features. Instead, logical layout components like paragraphs and sections are often represented only through subtle means like indentation or changes in font style (e.g., bold section headings), while physical layout structures like text columns or page breaks that do not form part of the logical layout hierarchy often have much simpler geometric representations on the page. The layout analysis techniques described here are not organized around a logical layout hierarchy ; instead, they concentrate on identifying the visually most salient properties of the page layout–its columnar structure and its text lines. Logical layout properties like paragraphs and sections can then be recovered from that information.

4 Statistical and Geometric Models

4.1 Text Lines

Probably both for typesetting and for perceptual reasons, text in many writing systems is set along straight lines. For Latin scripts, which we focus on in this work, each character rests either on a baseline or on a line of descenders (Figure 1). The top of each character reaches either up to the x-height, or to the line of ascenders. Note that the location of each character "on" the baseline or line of descenders is only approximate–typographic conventions resulting from perceptual phenomena dictate that in well-designed fonts, the actual location of individual characters differs slightly from their precise location relative to the baseline.

As is commonly done, we approximate the shape of each character by its (axis-aligned) bounding box and use the bottom center of the bounding box as a reference point [BAI 87]. If there is no rotation present at all, this reference point will lie on the baseline or line of descenders if the character does ; for small page rotations, the position of the reference point will differ from the actual location of the character relative to the baseline, but for the range of page rotations encountered in practice, the difference is slight. The amount of difference depends, however, on the character shape, and it will be larger for characters whose bounding box is defined by straight, off-center lines (e.g., "H") than for characters like "T" or "O".

Deviations from the baseline, errors resulting from using the bounding box approximation, image noise, and scanner quantization all result in some noise in the location of reference points relative to the baseline or line of descenders. We approximate this noise by a Gaussian distribution, but bound the maximum allowable deviation ; that is, beyond a certain deviation from a given baseline, it is more likely that the character in question belongs to a different baseline or background noise rather than belonging to the same baseline with a large deviation.

The text line model shown in Figure 1 is parameterized by two line parameters for the baseline (we use angle θ and distance from origin, r), plus three parameters for the size of the

FIG. 1 – The text line model used by the algorithm.

descenders, the x-height, and the ascenders. However, every character rests either on the baseline or line of descenders. Therefore, in order to identify all characters contributing to a text line, it is sufficient to model the baseline and the line of descenders, a three-parameter problem. [1]

Ignoring the line of descenders for a moment, the likelihood with which a character is found at some point p given a set of line parameters (θ, r) is then (up to normalization factors) :

$$P(p|\theta, r) \propto \max(\beta, G_\sigma(d(p, \ell_{\theta,r}))) \tag{1}$$

Here, $\ell_{\theta,r}$ is the baseline, $d(p, \ell_{\theta,r})$ is the distance of p from the baseline, $G_\sigma(x)$ is a Gaussian distribution with zero mean and standard deviation σ, and β is a background probability. This model is analogous to the model for geometric matching described by [III 97], where the reader can find a more extensive justification.

Taking into account the line of descenders, we arrive at a mixture distribution

$$\begin{aligned} P(p|\theta, r, d_d) \quad &\propto \quad \max(\beta, \\ &\qquad (1-\lambda)\, G_\sigma(d(p, \ell_{\theta,r})) + \\ &\qquad \lambda\, G_\sigma(d(p, \ell_{\theta,r-d_d}))) \qquad (2) \\ &= \quad \max(\beta, \\ &\qquad (1-\lambda)\, G_\sigma(d(p, \ell_{\theta,r})), \\ &\qquad \lambda\, G_\sigma(d(p, \ell_{\theta,r-d_d}))) \qquad (3) \end{aligned}$$

Here, d_d is the distance of the line of descenders from the baseline, and λ is the frequency with which descenders occur. We have assumed that the ranges for which the distributions around the baseline and around the line of descenders are greater than the background distribution are well separated, which has allowed us to replace the sum in Equation 2 with the max in Equation 3.

With the above statistical model, we can now formulate text line finding as that of finding a maximum likelihood solution for our statistical model :

$$\hat{\ell}_{\theta,r,d_d} = \arg\max_{\theta,r,d_d} \prod_i P(p_i|\theta, r, d_d) \tag{4}$$

For the purpose of finding maximum likelihood solutions, it is easier to take the logarithm of this equation (the logarithm is a monotonic function) :

$$\begin{aligned} \hat{\ell}_{\theta,r,d_d} \quad &= \quad \arg\max_{\theta,r,d_d} \sum_i \log P(p_i|\theta, r, d_d) \qquad (5) \\ &= \quad \arg\max_{\theta,r,d_d} \sum_i \max(\log\beta, \\ &\qquad \log(1-\lambda) + \log G_\sigma(d(p_i, \ell_{\theta,r})), \\ &\qquad \log\lambda + \log G_\sigma(d(p_i, \ell_{\theta,r}))) \qquad (6) \\ &= \quad \arg\max_{\theta,r,d_d} \sum_i \max(0, \\ &\qquad \log(1-\lambda) + \log G_\sigma(d(p_i, \ell_{\theta,r})) \\ &\qquad\quad - \log\beta, \\ &\qquad \log\lambda + \log G_\sigma(d(p_i, \ell_{\theta,r})) \\ &\qquad\quad - \log\beta) \qquad (7) \end{aligned}$$

By choosing parameters appropriately and writing $x = d(p, \ell_{\theta,r})$ this reduces to a particularly simple form :

$$\hat{\ell}_{\theta,r,d_d} = \arg\max_{\theta,r,d_d} \sum_i \max(0, c_1 - c_2 x^2, c_3 - c_4 x^2) \tag{8}$$

Here, the parameters c_1, ..., c_4 depend on β, σ, and λ.

This is the optimization that we need to perform in order to find the maximum likelihood solution to the text line finding problem under our statistical model. It is closely related to robust least square fitting in robust statistics [HUB 81]. Optimizing these kinds of functions is difficult with traditional methods, such as gradient descent, because they have many local minima and large "flat" regions. However, below, we will see how we can find globally optimal solutions to such optimization problems.

As stated above, the approach assumes that there is only a single text line on the page plus background noise. However, for each reference point that contributes to the total likelihood for a particular choice of line parameters, we not only get a contribution to the total likelihood, but also an indication of whether it was considered part of the text line or whether it contributed as a background point (β). This means that once we have found a maximum likelihood solution, we can divide the reference points that contributed to it into those that contributed as background points and those that did not. The reference points that did not contribute as background points are then counted as being "part of" the line, while the background points can be re-examined for the next-best maximum likelihood solution[2]

[1] Modeling the line of ascenders and the x-height can be used to reject characters from a match that accidentally fall on the baseline or the line of descenders. That can improve robustness for very noisy documents or documents with unusual layouts.

[2] A formal justification of this approach goes beyond the scope of this

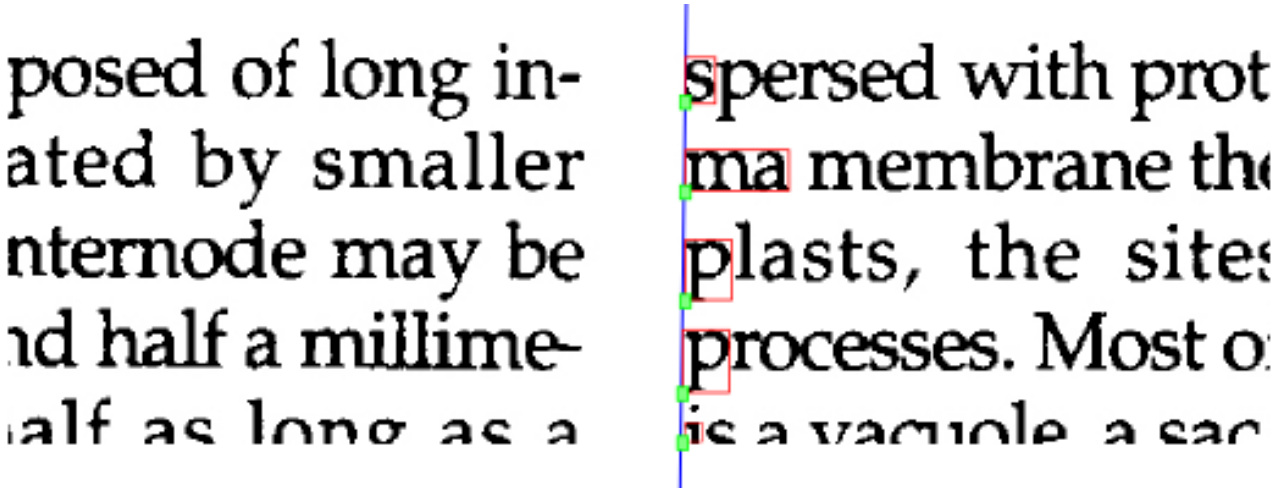


FIG. 2 – A simple column model. The red rectangles are bounding boxes for connected components, the green dots are the reference points used for each bounding box, and the blue line is a robust linear fit to the reference points.

4.2 Column Boundaries

At first sight, the problem of finding text columns is quite similar to the problem of finding text lines. That is, text is typeset in columns so that the leftmost character in each column aligns on a straight line with the other leftmost characters on each text line that is part of the column. We can therefore pick the bottom left corner of each character as an alignment point and perform a line matching operation similar to that used for text line finding. The likelihood model and optimization problems are also analogous to that for text line finding, except that modeling of baselines is not required. This approach is illustrated in Figure 2. Such an approach has many of the advantages of the text line finding algorithm mentioned above : it can find column boundaries even if they are not parallel to each other (e.g., in perspectively distorted document images), and it does not rely on any global properties of the page. An example of the application of a combined text line and column finding method can be found in Figure 3, where it is applied to the problem of removing perspective distortions from images captured with a hand-held camera.

However, while useful in some instances, unlike the text line model, such a column model is not satisfactory for complex documents because of the statistics of real documents. One simple way of understanding that difference is the following observation. If we pick out an arbitrary connected component corresponding to a character, it is with high probability part of some text line. In contrast, only a small fraction of all characters are actually part of a column boundary (namely, only those characters at the beginning of a text line). Therefore, there is almost no possibility for false positives during text line finding, since there are very few connected components not part of text lines, and those components are unlikely to be aligned linearly, while there is a significant probability of false positives during text line finding, namely when characters that are otherwise not part of any column boundary accidentally align linearly.

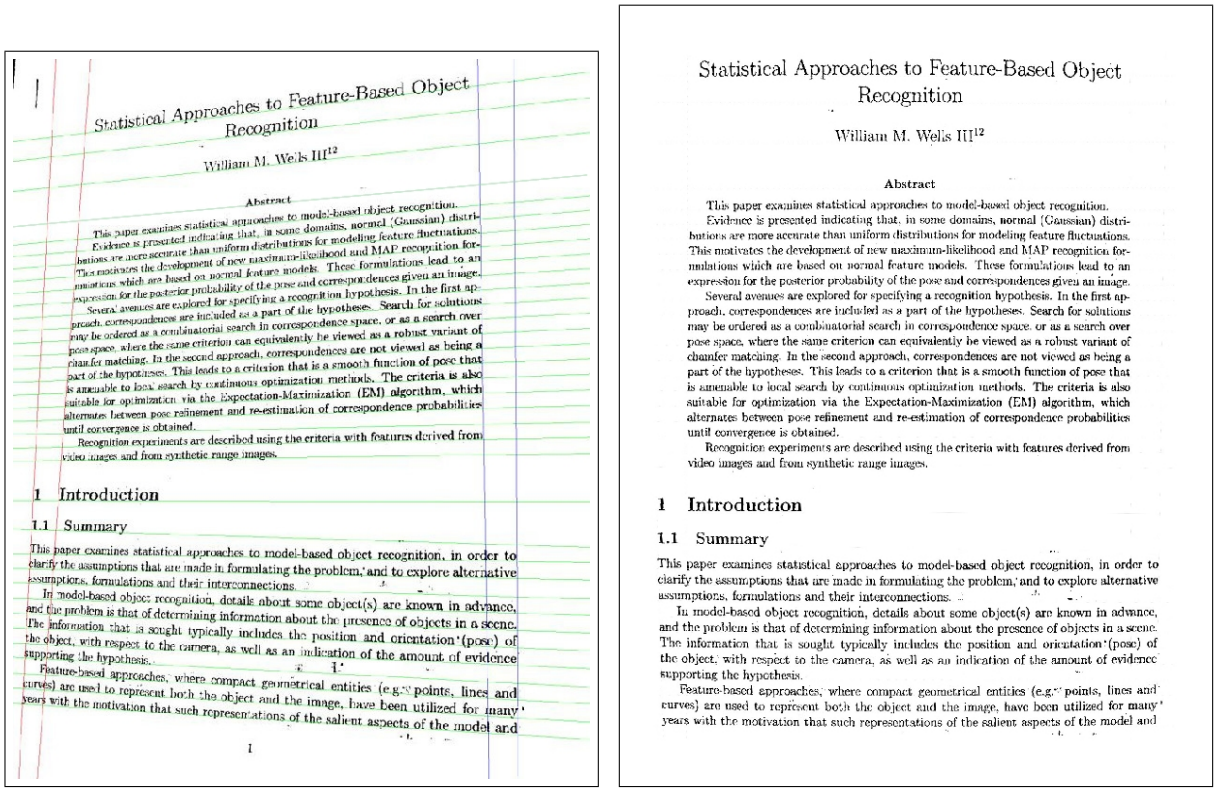
In order to recognize text columns reliably, we need to take advantage of additional information that helps us distinguish actual text columns from accidental alignments of characters. One property that suggests itself is the use of whitespace. In fact, many algorithms for geometric layout analysis rely only on whitespace for detecting column boundaries. Examples are the use of X-Y segmentations [NAG 84], and whitespace covers [BAI 94, BAI 90]. However, in our experience, whitespace covers by themselves, without the use of additional information, are not a reliable method for column detection either. But combining the computation of whitespace covers with the computation of columnar alignments turns out to result in a method that empirically recognizes column boundaries with lower error rates than either whitespace-only or alignment-only methods alone.

There are several ways of formalizing the combination of whitespace covers with measuring the alignment of connected components as a column boundary. First, we can start with the maximum whitespace rectangle formulation as described by Baird *et al.* [BAI 94, BAI 90] and add a requirement that a minimal number of connected components be present around the outside of that rectangle. Alternatively, we can start with a column finder based on linear alignment of reference points, as in Figure 2 and add a requirement that significant amounts of whitespace be present next to the column boundary. For historical reasons–maximal rectangle whitespace covers had already proven to be fairly reliable at detecting column boundaries and since we developed simple and efficient algorithms for computing them–we adopted the first approach.

This approach and insight then leads to a computational geometry problem for finding column separators. In the case of axis-aligned document layouts, we can state this problem formally as follows (also illustrated in Figure 5.

Definition 1 Axis-Aligned Maximum Whitespace Rectangle with Halo Problem. *The input to the algorithm is a rectangular outer bound B, a collection of axis-aligned rectangular obstacles R_i, a distance ϵ, and a threshold count h. The output of the algorithm is a rectangle M with maximal area satisfying (1) $M \subseteq B$, (2) for all R_i, area$(M \cap R_i) = 0$, and (3) the number of rectangular obstacles R_i within a distance*

paper.



Statistical Approaches to Feature-Based Object Recognition

William M. Wells III[12]

Abstract

This paper examines statistical approaches to model-based object recognition. Evidence is presented indicating that, in some domains, normal (Gaussian) distributions are more accurate than uniform distributions for modeling feature fluctuations. This motivates the development of new maximum-likelihood and MAP recognition formulations which are based on normal feature models. These formulations lead to an expression for the posterior probability of the pose and correspondences given an image.

Several avenues are explored for specifying a recognition hypothesis. In the first approach, correspondences are included as a part of the hypotheses. Search for solutions may be ordered as a combinatorial search in correspondence space, or as a search over pose space, where the same criterion can equivalently be viewed as a robust variant of chamfer matching. In the second approach, correspondences are not viewed as being a part of the hypotheses. This leads to a criterion that is a smooth function of pose that is amenable to local search by continuous optimization methods. The criteria is also suitable for optimization via the Expectation-Maximization (EM) algorithm, which alternates between pose refinement and re-estimation of correspondence probabilities until convergence is obtained.

Recognition experiments are described using the criteria with features derived from video images and from synthetic range images.

1 Introduction

1.1 Summary

This paper examines statistical approaches to model-based object recognition, in order to clarify the assumptions that are made in formulating the problem, and to explore alternative assumptions, formulations and their interconnections.

In model-based object recognition, details about some object(s) are known in advance, and the problem is that of determining information about the presence of objects in a scene. The information that is sought typically includes the position and orientation (pose) of the object, with respect to the camera, as well as an indication of the amount of evidence supporting the hypothesis.

Feature-based approaches, where compact geometrical entities (e.g. points, lines and curves) are used to represent both the object and the image, have been utilized for many years with the motivation that such representations of the salient aspects of the model and

FIG. 3 – Application of the text line and column finders to the problem of removing perspective distortions of documents captured with hand-held cameras. The application is enabled by the fact that the text line and column finders do not make any assumptions about parallelism among text lines or columns.

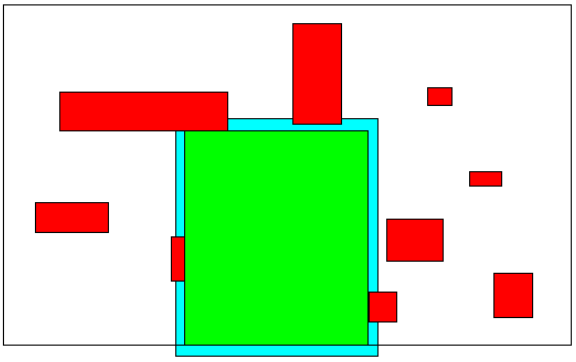


FIG. 5 – Maximal whitespace rectangles with halo. See the text for details.

of ϵ of M is greater than h, $\#\{R_i : dist(R_i, M) \leq \epsilon\} \geq h$.

This definition can be generalized easily to non-axis aligned rectangles, although (as we will see) the resulting algorithm differs significantly. Furthermore, in practice, it is useful to limit the aspect ratio of the maximal rectangles found by the algorithm.

Note that our reasoning for stating this approach to column finding has a rather different motivation from our approach to text line finding. In the case of text line finding, we started with a clearly defined geometric model of text lines (characters resting either on the baseline or line of descenders), chose an error model, and derived the maximum likelihood solution. In the case of column finding, while we were motivated by a similar model–alignment of characters along a line–we ultimately chose an algorithm based on maximal whitespace rectangles because such algorithms have been found to work well in previous work [BAI 94, BAI 90] ; the constraint of linear alignment of the connected components corresponding to the first character of each text line adjacent to the column separator comes in through the halo requirement. This requirement of having a halo of connected components under a given error bounds corresponds to a bounded uniform error noise model, as opposed to the Gaussian error model used with the text line finder.

5 Branch-and-Bound Algorithms

In the previous section, we have discussed the kinds of statistical and geometric models that correspond to text lines and column boundaries. However, we have so far left open the question of how to find optimal solutions under those models ; without practical algorithms, such models would simply not be very useful. In fact, the use of the models described in the previous section has only been enabled through the development of a new class of practical optimization algorithms. These algorithms, described below, combine ideas from branch-and-bound geometric matching algorithms in computer vision [BRE 92] and interval arithmetic optimization [HAN 80]. Furthermore, they incorporate an important optimization that we refer to as *matchlists* [BRE 92].

The basic idea behind these algorithms is to formulate the search for an optimal solution as a search over a multidi-

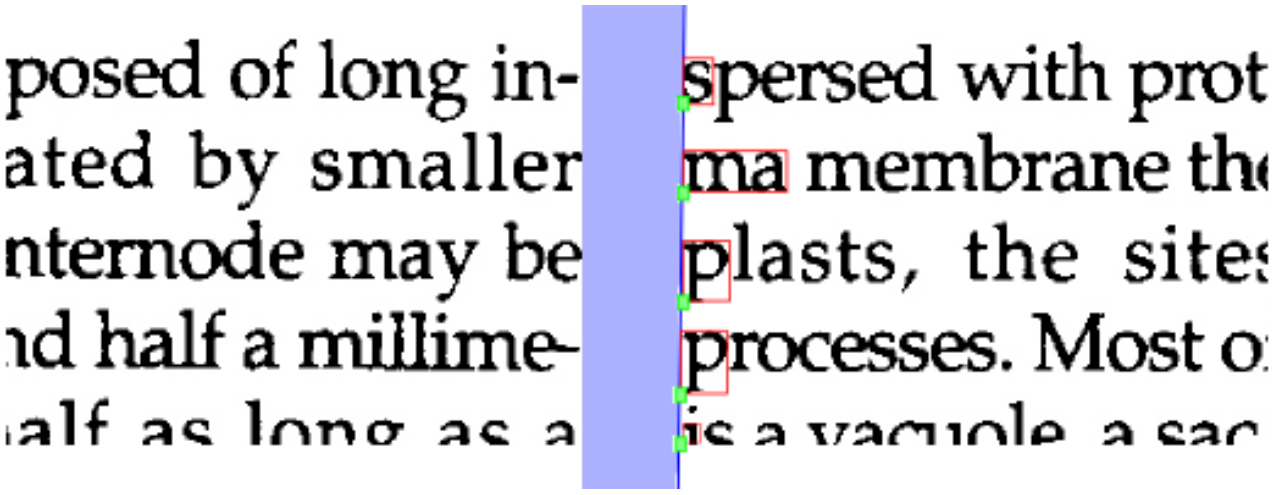


FIG. 4 – A column model that takes into account whitespace. The red rectangles are bounding boxes for connected components, the green dots are the reference points used for each bounding box, and the blue line is a robust linear fit to the reference points. The blue area to the left of the blue line is the columnar whitespace.

mensional parameter space. In the case of finding maximum likelihood text line solutions, the parameter space is three-dimensional : it consists of a range of angles $[\underline{\theta}, \overline{\theta}]$, a range of distances from the origin $[\underline{r}, \overline{r}]$, and a range of distances for the line of descenders from the baseline $[\underline{d_d}, \overline{d_d}]$. In the case of finding non-axis aligned rectangles with halo, the parameter space is four dimensional[3] and consists of a range of rectangle centers $[\underline{x}, \overline{x}]$ and $[\underline{y}, \overline{y}]$, a range of orientations $[\underline{\theta}, \overline{\theta}]$, and an aspect ratio $[\underline{r}, \overline{r}]$.

In both cases, the objective functions (Equation 8 and the area function subject to the constraints in Definition 1) are ill-suited to the usual gradient-based optimization methods because they contain large "flat" regions and many local minima. In order to find good solutions to these problems, we need global optimization techniques. The combination of branch-and-bound methods together with interval arithmetic has proven to be a powerful tool for these kinds of problems [HAN 80, BRE 03b]. The idea is that we subdivide the parameter space into hyper-rectangular subregions and compute upper and lower bounds on the value of the objective function over each subregion.

This can be accomplished using interval arithmetic. Interval arithmetic allows us to take an objective function in algebraic notation and replace the arithmetic operations occurring in that objective function with their interval equivalents. (This can be accomplished automatically in languages like C++ through overloading.) The resulting interval objective function then computes upper and lower bounds on the value of the objective function over each hyper-rectangle in parameter space. Those bounds are not guaranteed to be tight, but they are guaranteed to be *convergent* [JAU 01]–informally, as the hyper-rectangle in parameter space becomes smaller, the bounds are guaranteed to approximate the true value of the objective function closer and closer.

In order to make this approach practical for these applications, we need to apply an important optimization. Observe that the objective function for text line finding, given by Equation 8 is a sum of individual contributions. Furthermore, we have rewritten that objective function so that the contribution of background features to the total sum is zero. The interval equivalent of this objective function will be evaluated repeatedly in the inner loop of the search algorithm. But it can be shown easily that if a contribution from some feature point p_i is zero for a hyper-rectangle in parameter space, it will remain zero for any subrectangle of that hyper-rectangle. All points making zero contributions therefore need not be considered during further evaluations of the objective function for subrectangles in parameter space. We keep track of the points making non-zero contributions on a simple data structure, the matchlist. As we consider subrectangles of a hyper-rectangle in parameter space, points are removed from the matchlist and need not be evaluated any further in the computation of the objective function for such subrectangles. Using these insights, we can now write down pseudo-code for the global optimization algorithms for these geometric problems (for brevity, we use the term `region` instead of "hyper-rectangle in parameter space"); this is shown in Figure 6. Actual code for this problem is, in fact, very close to the pseudo-code shown in Figure 6.

```
def globally_optimize(region,points):
    q = objective_function(region,points)
    queue.enqueue(q,region,points)
    while not queue.is_empty():
        (q,region,points) = queue.dequeue_max()
        if accurate_enough(region): return region
        subregions = split(region)
        for subregion in subregions:
            subpoints =
                [list of point in points
                    if point contributes to objective_function]
            subq = objective_function(subregion,subpoints)
            queue.enqueue(subq,subregion,subpoints)
```

FIG. 6 – Pseudo-code for finding globally optimal solutions to the maximum likelihood problems described in the text.

A question that remains about these algorithms is how ef-

[3] Such rectangles are actually described by five parameters, but the requirement that the rectangles are maximal reduces the parameter space that needs to be searched to four dimensions, given a particular set of obstacles.

ficient they are. Commonly used methods for analyzing the performance of geometric and numerical algorithms determine asymptotic complexities in terms of elementary arithmetic operations. Such analyses are both difficult and not very useful for these classes of algorithms. They are difficult because the complexity of these algorithms depends on the desired accuracy of the computed results. Increasing the desired accuracy, the asymptotic complexity of the algorithm can be shown to be linear in the number of solutions and the input, but in practice, the algorithms are usually used with a fixed, finite accuracy dictated by the problem requirements. Furthermore, the problem instances themselves are limited in complexity : since they are derived from scanned documents and there are physical and perceptual constraints on the size and complexity of such documents, asymptotic complexity is of less interest than actual complexity on the range of problem complexities found in real-world applications. Having said that, we find average running times of approximately 1 second for the text line finder (finding all text lines in a document image) and less than 0.25 seconds for the white space cover on modern PC hardware on the documents in the University of Washington Database 3 (UW3), with approximately linear scaling of running times in terms of the number of connected components over the range of document complexities found in the UW3 database.

6 Discussion

This paper has reviewed well-defined models of the spatial statistics of document layouts and algorithms for finding maximum likelihood solutions under such models. These approaches to layout analysis differs from previous methods for finding geometric layout primitives in a number of important ways.

First, they are guaranteed to find globally optimal solutions under the specific geometric and statistical model chosen. That makes their results reproducible and deterministic ; if the algorithm fails to come up with a correct solution, we know that this must be a problem with the statistical model itself, no simply a failure of the algorithm to find the optimal solution to the model. Empirically, this simplifies debugging and parameter estimation for layout analysis systems based on these methods.

Second, the algorithms are expressed in terms of well-defined models of spatial statistics and errors ; the distributional parameters (error bounds, frequency of background features) can be estimated from sample images and have intuitive interpretations. At the same time, the models also incorporate strong prior knowledge about the geometry of document images (e.g., the fact that text lines are usually straight). This is in contrast to other well-founded statistical models of page layout proposed recently [LIA 99, LIA 01] ; those models also take advantage of statistical constraints, but they rely on more general statistics of the relative spatial distributions of layout elements, making both the parameter estimation and the inference problem considerably harder. It is also in contrast with projection-based methods for layout analysis [BAI 87, NAG 84], whose criteria for determining the location of text lines or paragraphs are generally not directly related to spatial statistics of the location of characters on the page.

Third, the methods presented here can operate “locally” ; that is, individual text lines and column separators can be found independently of each other. This allows us to apply the methods both to novel document types (e.g., documents with text at various different orientations) and to standard document types captured in novel ways (e.g., document capture with hand-held cameras, resulting in perspective distortion). In contrast, many commonly used methods for geometric layout analysis rely on global layout properties. For example, methods like projection-based text line finding and layout analysis generally assume rectilinear layouts and the ability to find a single, global page rotation that will “deskew” the page. There are some methods for geometric layout analysis that do not make such global assumptions, such as the Voronoi-based technique of [KIS 98] or closely related methods based on linking nearby connected components. However, while they are able, in principle, to operate on pages that have been perspectively distorted or contain text at various different orientations, those methods rely on much weaker geometric models of text lines than the methods presented in this paper. That suggests that they will be more susceptible to false positive errors in the detection of text lines ; however, this remains to be demonstrated on actual datasets.

In this short review, we have focused on motivating particular models of spatial statistics for layout analysis and outlined an approach to inference based on branch-and-bound methods that makes using such models feasible for real document analysis problems. For detailed performance data in specific applications, the reader is referred to the literature. For data on the accuracy of text line finding and deskewing using these methods, [BRE 02c] contains an evaluation on the UW2 database. For data on the reliability with which the method finds column separators, the reader is referred to [BRE 02a]. Results on non-axis aligned column separator identification can be found in [BRE 03a]. More complete document analysis systems taking advantage of the novel capabilities of these methods have also been described, including a system for image-based reflowing of document images [BRE 02b].

However, the reader should keep in mind that these methods can also be used as a drop-in replacement for commonly used existing methods in document analysis, often yielding some improvement in performance, reliability, or functionality over existing methods. For example, the text line finder described above can be used in place of projection-based text line finders for reliable skew estimation and text line identification. And the maximal empty rectangle methods outlined above can be used as a drop-in replacement for the whitespace cover approach described by [BAI 94, BAI 90].

Future work on these algorithms divides into fundamental algorithmic improvements on the one hand, and the exploration of additional application areas on the other. On the algorithmic side, one open problem is the application of these optimization methods to higher-dimensional parameter spaces. Currently parameter spaces beyond four or five dimensions lead to impractically long computation times. However, our lab is currently examining methods for speeding up search that may allow us to extend that limit for

many kinds of objective functions. We are also examining other approaches to coping with problems such as matching smoothly curved text lines (as they occur, for example, on photographic images of curved book pages). On the application side, one of the most important areas is the use of these methods with non-Latin scripts, languages, and page layouts. Furthermore, a detailed performance evaluation and comparison between other recently proposed layout analysis techniques [LIA 99, LIA 01, KIS 98] and these methods remains to be done.

In order to facilitate more widespread adoption of these methods, we are making available sample implementations for research purposes ; please contact the author for more information.

Références

[BAI 87] BAIRD H. S., The Skew Angle of Printed Documents, *Proc., 1987 Conf. of the Society of Photographic Scientists and Engineers*, Rochester, New York, 1987.

[BAI 90] BAIRD H. S., JONES S. E., FORTUNE S. J., Image Segmentation by Shape-Directed Covers, *Proceedings of the Tenth International Conference on Pattern Recognition, Atlantic City, New Jersey*, 1990, pp. 820–825.

[BAI 94] BAIRD H. S., Background Structure in Document Images, *H. Bunke, P. S. P. Wang, & H. S. Baird (Eds.), Document Image Analysis, World Scientific, Singapore*, 1994, pp. 17–34.

[BRE 92] BREUEL T. M., Fast Recognition using Adaptive Subdivisions of Transformation Space, *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1992, pp. 445–451.

[BRE 02a] BREUEL T. M., Two Algorithms for Geometric Layout Analysis, *Proceedings of the Workshop on Document Analysis Systems, Princeton, NJ, USA*, 2002.

[BRE 02b] BREUEL T. M., JANSSEN W. C., POPAT K., BAIRD H. S., Paper-to-PDA, *Proceedings of the International Conference on Pattern Recognition (ICPR'02), Quebec City, Quebec, Canada*, 2002.

[BRE 02c] BREUEL T., Robust Least Square Baseline Finding using a Branch and Bound Algorithm, *Proceedings of the SPIE - The International Society for Optical Engineering*, 2002, page (in press).

[BRE 03a] BREUEL T. M., An Algorithm for Finding Maximal Whitespace Rectangles at Arbitrary Orientations, International Conference on Document Analysis and Recognition, 2003.

[BRE 03b] BREUEL T. M., On the Use of Interval Arithmetic in Geometric Branch-and-Bound Algorithms, *Pattern Recognition Letters*, , 2003.

[CAT 98] CATTONI R., COIANIZ T., MESSELODI S., MODENA C. M., Geometric Layout Analysis Techniques for Document Image Understanding : a Review, rapport, 1998, IRST, Trento, Italy.

[CHE 95] CHEN S., Document Layout Analysis Using Recursive Morphological Transforms, PhD thesis, Ph.D. thesis, Univ. of Washington, 1995.

[HAN 80] HANSEN E., Global optimization using interval analysis – the multi-dimensional case, *Numerische Mathematik*, vol. 34, 1980, pp. 247–270.

[HAR 94] HARALICK R. M., Document Image Understanding : Geometric and Logical Layout, *CVPR94 : IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1994, pp. 385–390.

[HUB 81] HUBER P., *Robust Statistics*, John Wiley and Sons, New York, 1981.

[III 97] III W. W., Statistical approaches to feature-based object recognition., *International Journal of Computer Vision*, vol. 21, n° 1/2, 1997, pp. 63–98.

[JAU 01] JAULIN L., KIEFFER M., DIDRIT O., WALTER E., *Applied Interval Analysis*, Springer Verlag, Berlin, 2001.

[KIS 98] KISE K., SATO A., IWATA M., Segmentation of page images using the area Voronoi Diagram, *Computer Vision and Image Understanding*, vol. 70, n° 3, 1998, pp. 370-82.

[LIA 99] LIANG J., PHILLIPS I., HARALICK R., A Unified Methodology for Document Structure Analysis, *Workshop on Document Layout Interpretation and its Applications (DLIA)*, 1999.

[LIA 01] LIANG J., PHILIPS I. T., HARALICK R. M., An Optimization Methodology for Document Structure Extraction on Latin Character Documents, *Pattern Analysis and Machine Intelligence*, , 2001, pp. 719-734.

[NAG 84] NAGY G., SETH S., Hierarchical Representation Of Optically Scanned Documents, *7ICPR*, vol. 84, 1984, pp. 347-349.

[O'G 93] O'GORMAN L., The Document Spectrum for Page Layout Analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, n° 11, 1993, pp. 1162–1173.

[OKU 99] OKUN O., PIETIKAINEN M., SAUVOLA J., Robust document skew detection based on line extraction, *Proc. of the 11th Scandinavian Conference on Image Analysis (SCIA'99), June 7-11, Kangerlussuaq, Greenland*, 1999, pp. 457–464.

[SRI 89] SRIHARI S., GOVINDARAJU V., Analysis of textual images using the Hough transform, *Machine Vision and Applications*, vol. 2, 1989.