Robust Least Square Baseline Finding using a Branch and Bound Algorithm

Thomas M. Breuel *

Xerox PARC, 3333 Coyote Hill Rd., Palo Alto, CA 94304, USA

ABSTRACT

Many document analysis and OCR systems depend on precise identification of page rotation, as well as the reliable identification of text lines. This paper presents a new algorithm to address both problems. It uses a branch-and-bound approach to globally optimal line finding and simultaneously models the baseline and the descender line under a Gaussian error/robust least square model. Results of applying the algorithm to documents in the University of Washington Database 2 are presented.

Keywords: document analysis, layout analysis, skew detection, page rotation, text line finding, affine transformations

1. INTRODUCTION

Precise identification of baselines is an important first step in some OCR systems,[7] and it is also a useful early step in document layout analysis. Numerous methods have been proposed for text line and baseline finding. Some methods attempt to find just text lines, e.g. Hough transforms, projection profiles, and Radon transforms. Others find text lines as part of more general document layout analysis tasks, e.g., XY cuts, whitespace segmentation, Voronoi diagrams, and distance-based grouping. Generally, such methods have in common that they begin by performing a "rough" analysis of the layout, often based on the proximity of bounding boxes of connected components or connected components themselves. If more precise baseline models are matched, they are usually matched in a second step, after a collection of connected components has already been identified as likely coming from the same text line. For a survey of techniques, see the references[1,4,11]; the paper by Okun *et al.*[8] contains a good reference list.

This paper describes an alternative approach to text line and baseline finding that integrates a coarse-to-fine strategy with precise least-square (or, equivalently, maximum likelihood under a Gaussian error model) matching of a geometric model of the text line. Unlike many other coarse-to-fine strategies, the branch-and-bound search method described in this paper guarantees that the solutions returned by the algorithm are optimal with respect to the least square error model. The use of a geometric model that incorporates both the baseline and descenders makes the algorithm more robust

2. THE TEXT LINE MODEL

Text lines in the Latin alphabets are composed of a mix of characters with different heights and locations relative to one another. In most Latin fonts, each alphanumeric character (or the largest connected component making up an alphanumeric character, as in the case of the letter "i") either rests on the baseline or on a line delimiting all descenders. We model the baseline and the descender line and assume that they are at a constant distance to one another throughout the text line (this is not satisfied if there are font size changes within the line–something that is conventionally avoided). For the purposes of the algorithm, this model is parameterized by the line parameters of the baseline, (θ, r), and the distance from the baseline to the descender line, d (see Figure 1).

There are two additional imaginary lines associated with a text line: the line indicating the x-height (the height of a lower case letter) and the ascender line, the line indicating the height of capital letters and some lower case letters like "f". In most Latin fonts, the top of the bounding box of the largest connected component making up a character also coincides with one of these two lines.

* Send correspondence to tbreuel@parc.xerox.com

Figure 1. Illustration of the text line model used in the line finding algorithm. Each text line is modeled as a baseline and a parallel descender line. The baseline is parameterized by its angle ρ and its distance from the origin r. The descender line is parameterized by its distance d from the baseline. Characters (connected components) are represented by the location of the center of the lower side of their bounding box.

3. THE METHOD

Like most other text line finding and segmentation methods, the method begins by computing the connected components of the input document. For each connected component, we compute the bounding box. We compute statistics of the connected components to obtain rough estimates of the predominant character dimensions; while this step is not necessary, it helps with noisy documents or documents containing line drawings.

We use the center of the bottom side of the bounding box of each connected component as the reference point that needs to rest on either the baseline or the descender line.[6] This choice tends to be is more robust to page rotation than using the bottom left corner of the bounding box because because many characters do not have any ink in the bottom corners on the bounding box, but instead, either have a vertical stroke located in the center or a rounded bottom. If such a character is rotated clockwise, for example, the location of the bottom left corner of the bounding box will be below the correct baseline, while the center of the bottom side of the bounding box will still be in approximately the correct location.

In this way, we derive a large collection of reference points that need to be organized into baselines and corresponding descender lines. As we indicated above, our text line model consists of a straight baseline together with a parallel descender line at some fixed distance. This text line model can be parameterized by three parameters: the line parameters (θ, r), where θ is the orientation of the line and r its distance from the origin, and a distance d between the descender line and the baseline. Our goal is to find a collection of parameters (θ, r, d) for each line in the text document that maximizes the number of bounding boxes matching the model and that minimizes the distance of each reference point from the baseline in a robust least square sense.

More precisely, let the locations of the reference points present in the document be given by $\{x_1, \ldots, x_N\}$. If we assume a Gaussian error model on location and a uniform background probability of finding reference points that do not belong to the baseline we are matching, it can be easily shown in analogy to well-known results[5]

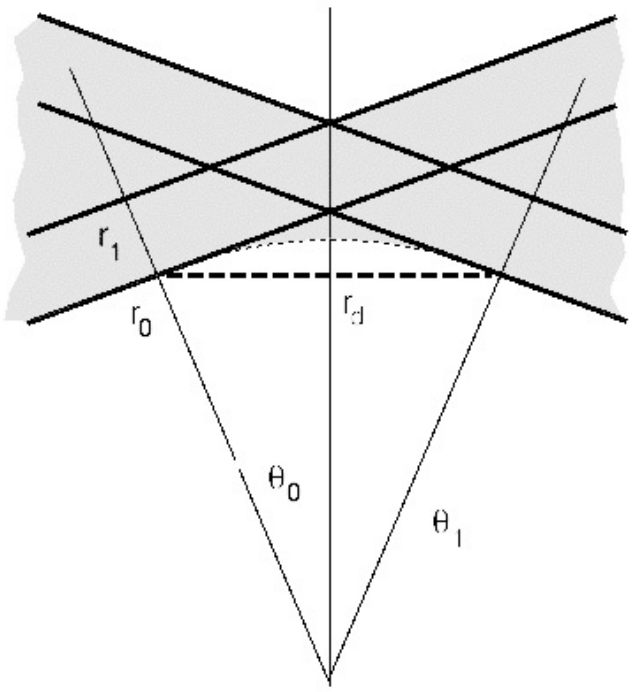

Figure 2. Diagram illustrating the region swept out (gray) by varying line parameters over the rectangular region in parameter space, $\mathcal{T} = (\theta_0, \theta_1) \times (r_0, r_1)$. Note the small area bounded by two lines and a circular arc at the bottom of the region.

that the log likelihood $L_{\theta,r,d}$ of any particular set of parameters (θ, r, d) is linearly related to:

$$L_{\theta,r,d} = \sum_{i=1\ldots N} \max(0, 1 - \epsilon^{-2}\text{distance}(x_i, l_{\theta,r,d})^2) \tag{1}$$

Here, $l_{\theta,r,d}$ is the pair of parallel lines representing the baseline and the descender line, and distance(x, l) is the minimum distance of the point x from the point set l. We see from this equation that optimizing the log likelihood is equivalent to a robust least square line fit.

Traditionally, such fits are carried out iteratively and heuristically, with a certain risk that the results found are merely locally optimal solutions in parameter space. However, there is now a general framework for geometric matching under a variety of error models that uses a branch and bound approach and guarantees that it finds globally optimal solutions[2,3] (variants of these algorithms have been described more recently by some other authors[9]). Furthermore, such a branch-and-bound approach can efficiently extract the n-best matches, not just the top match. We will describe the approach briefly here; for additional references, the reader is referred to the references.[2,3]

The basic idea is to perform matching using a subdivision of parameter space and to bound the quality of the possible match for each such subdivision. If a subdivision potentially contains an optimal match, it is explored further, otherwise it is pruned. The application to text line finding is as follows.

A particular choice of parameters (θ, r, d) describes a pair of parallel lines, the baseline and the descender line (Figure 1).

Considering just the baseline, if we allow the baseline parameters to vary over a rectangular region in parameter space, $\mathcal{T} = (\theta_0, \theta_1) \times (r_0, r_1)$, the baseline will sweep out a bow-tie shaped region in image space

(Figure 2). As θ_1, θ_0 and r_1, r_0 converge to limit values of θ and r respectively, this region converges to the baseline described by the parameters θ and r. Computing a distance of any point from the region shown in Figure 2 gives us an upper bound on the distance of that point from any of the baselines whose parameters are contained in $\mathcal{T} = (\theta_0, \theta_1) \times (r_0, r_1)$. Four sides of that region are bounded by lines; the fifth side is bounded by a circular arc; see Figure 2.

Because it is computationally expensive to determine the distance of a point from this region, we use a somewhat larger region that is entirely bounded by linear constraints; that is, in Figure 2, we replace the circular arc at the bottom of the region swept out by the line with the triangle bounded by the two lines with $r = r_0$ and a third line (shown heavily dashed in the diagram). The parameters for this third line are given by $\theta_d = \frac{\theta_0+\theta_1}{2}$ and $r_d = r_0(1 - \cos\frac{\theta_1-\theta_0}{2})$ (a more detailed explanation can be found in the literature[3]).

The five lines θ_0, r_0, θ_1, r_0, θ_0, r_1, θ_1, r_1, and θ_d, r_d, describe a region as illustrated in Figure 2. The distance of a point from that region can be easily computed with five linear operations. Furthermore, as $\theta_1 - \theta_0$ and $r_1 - r_0$ approach 0, the distance of a point from the region computed in this way is easily seen to converge to the distance of the point from the line described by the limit of these parameters θ, r.

With these preliminaries, to compute an upper bound on the log likelihood in Equation 1 for a range of parameters $(\theta_0, \theta_1) \times (r_0, r_1)$, we now proceed as follows. First, we compute an upper bound of the distance of each point representing a connected component from baselines, i.e. lines with parameters $(\theta_0, \theta_1) \times (r_0, r_1)$. We also compute an upper bound of the distance of each point representing a connected component from baselines, i.e. lines with parameters $(\theta_0, \theta_1) \times (r_0 - d_1, r_1 - d_0)$. We convert each of these bounds on the distance into bounds on likelihoods using Equation 1; because likelihood is monotonically decreasing with distance, this yields an upper bound on the likelihoods. We then label a point as being on the baseline or on the descender line depending on which likelihood is larger. Finally, we total up all the likelihoods, resulting in the overall likelihood for the line.

Given this computation of an upper bound on the likelihood of any text line with parameters in the region $(\theta_0, \theta_1) \times (r_0, r_1) \times (d_0, d_1)$, we can now organize the search as follows:

1. Pick an initial region of parameter values $\mathcal{T}$ containing all the page rotations, baseline offsets, and descender sizes we are interested in.
2. Maintain a priority queue of regions $\mathcal{T}_i$, where we use as the priority the upper bound $\hat{L}_{\mathcal{T}_i}$ on the log likelihood as described above.
3. Remove a region $\mathcal{T}_i$ from the priority queue; if the log likelihood associated with the region is too small to be of interest, terminate.
4. If the region is small enough to satisfy our accuracy requirements, accept it as a solution; remove all connected components involved in that solution from further consideration and continue at Step 3.
5. Otherwise, split the region $\mathcal{T}_i$ along its largest dimension and reinsert the subregions back into the priority queue; continue at Step 3.

This algorithm will return all maximum likelihood matches greedily in decreasing order of likelihood.

In order to make this approach practical and avoid duplicate computations, we use a *matchlist* representation.[2] That is, with each region kept in the priority queue in the algorithm, we maintain a list (the matchlist) of all and only those points that fall within the ϵ bound in Equation 1. The justification is that points for which the distance from the baseline is greater than ϵ do not contribute to the computation of the likelihood function at all. Furthermore, it is easy to see that if $\mathcal{T}_j \subseteq \mathcal{T}_i$, the upper bound satisfies $\hat{L}_{\mathcal{T}_j} \leq \hat{L}_{\mathcal{T}_i}$. When we split a region in Step 5, we therefore never have to reconsider points in the children which have already failed to contribute to the likelihood computation in the parent.

4. IMPLEMENTATION DETAILS

There are several implementation details for improving the performance and robustness of the algorithm.

More Strict Filtering To improve robustness to noise of this algorithm further, we can impose the constraint during the evaluation of the log likelihood function that all the bounding boxes that make up a single line overlap vertically and are close to each other. Additionally, to improve robustness to spurious bounding boxes further, we might impose the constraint that the upper center of the bounding box lines up at the x-height and the ascender height, although that was not found to be necessary on the database of documents used for testing.

Weighting of Boxes To avoid counting assigning too much weight to very small bounding boxes (which often represent noise), each box is assigned a weight that is one if the bounding box has the same dimensions as that of the average character (as determined by size statistics of bounding boxes), and decreases to zero as the bounding box gets larger or smaller.

Baseline Ambiguities When a line has no descenders, the log likelihood is the same whether the actual baseline is matched against the baseline of the model or against the line of descenders of the model. To avoid this ambiguity, we weight matches against the descender line of the model slightly less than matches against the baseline.

Removing Diacritics and Punctuation Marks To obtain a complete collection of bounding boxes corresponding to the text found on a text line, we also need to identify diacritics and punctuation marks whose bounding boxes do not match the baseline or descender line. To do this, we compute the bounding box (aligned with the rotated coordinate system of the text line) of the alphanumeric characters that constitute a text line and assign all connected components falling within this bounding box to the text line.

5. RESULTS

The algorithm as described above was implemented in Java and C++ (the latter is a more general line finding algorithm). Both implementations are about 300 lines of source code. The running time of the C++ implementation is about 2 seconds on a 1GHz PC for a page consisting of connected components. One of the advantages of this method is the small number of parameters it has. For most applications, all the user needs to decide on is the range of permissible text line parameters (maximum page rotation and maximum font size), and the error bound ϵ.

5.1. TEXT LINE FINDING

To test the algorithm, it was applied to the documents in the ENGLISH / MEMO / REAL subdirectory of the University of Washington Database 2. The database consists of 62 memos, photocopied and scanned. The documents in this database exhibit a wide variety of noise, document types, degradations, fonts, and manual annotations. The output of the text line algorithm described in this paper was examined for all documents. The text line fitting algorithm correctly identified all lines present in the documents when the extracted bounding boxes allowed it to do so. Furthermore, each text line that the algorithm found was identified as a single line; there were no duplicate detections. Of the approximately 2000 text lines represented by the document database, only a single line in the body of a document was not recognized; it was rejected prior to geometric matching because it consisted of a single word in which all characters were touching and had been merged into a large connected component that was rejected prior to geometric matching.

These results demonstrate that the geometric modeling part of the algorithm is highly accurate and robust. An examination of the document database also suggests that the ability to model descenders helps significantly with the identification of short text lines, since both characters on the baseline and descenders contribute to the score for a textline. In contrast, projections or proximity based methods, with their less precise geometric modeling, are more susceptible to the detection of spurious lines when thresholds are set low enough to recognize short text lines with a mix of characters on the baseline and characters with descenders.

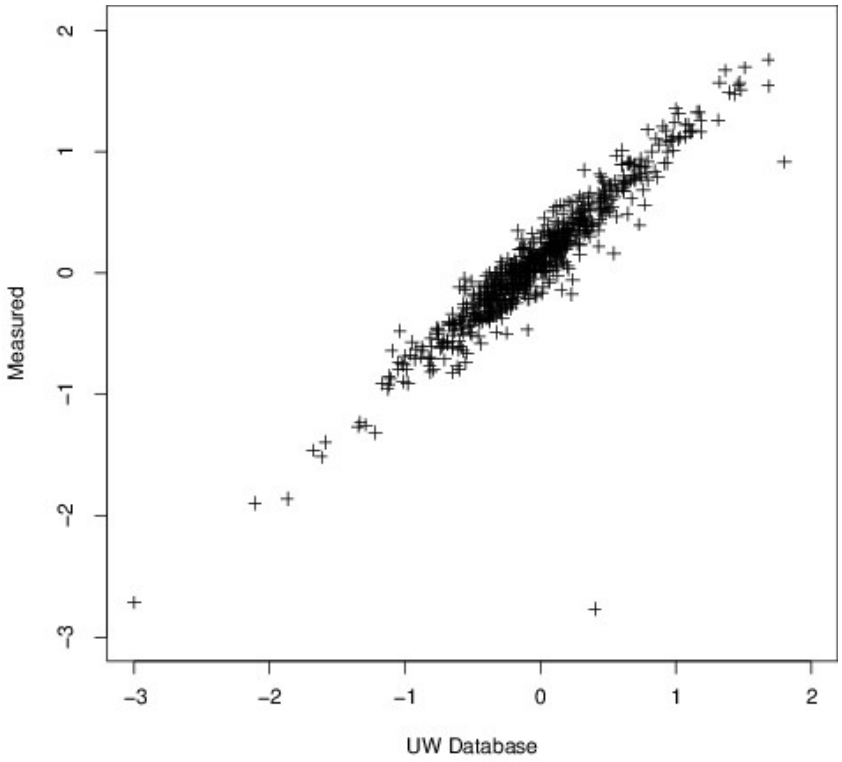

Figure 3. Scatter plot of the page rotation values given for the first 680 documents in the University of Washington Database 2, compared to the orientation of the longest line as determined by the algorithm given in this paper. The two obvious outliers are pages containing vertical text.

5.2. DETERMINING PAGE ROTATION

To answer the question of geometric accuracy, the baseline finding algorithm was applied to the first (by document label) 686 non-Japanese documents in the University of Washington Document Image Database[10] 2 (UW2).

The orientation of the top match was used as the page rotation (other choices, like the median, mean, weighted median, or weighted mean, of all the different orientations of lines gave similar results). Of these, we determined that in at least six cases, the page rotation information supplied with the UW2 database was incorrect (A03I, A03J, A05G, A06F, D05G, H00T).

For the remaining 680 cases, Figure 3 shows the relationship between page rotations determined by this algorithm and page rotation values in the data files of the UW2 database. Page rotation values supplied with the UW2 database are themselves algorithmically determined[10] and therefore do not constitute manually determined ground truth. However, they do provide an interesting basis for comparison.

We see that there is a close correlation between the page rotation determined by this algorithm and the data provided with the UW2 database ($R^2 = 0.88$). The two obvious outliers in Figure 3 correspond to vertically oriented images with very little textual content that, itself, is vertically oriented. These cases could be handled by running the baseline finding algorithm both for horizontal and vertical lines.

There is some scatter of the measured orientations compared to the page rotations recorded in the University of Washington database. Visual inspection of the baseline matches determined using the algorithm described in this paper suggests that its matches tend to be geometrically very accurate. However, close inspection also shows that the baseline orientations within the same page vary appreciably, possibly due to distortions like page skew, non-linear paper transport during imaging, etc. This means that there is no single correct page rotation

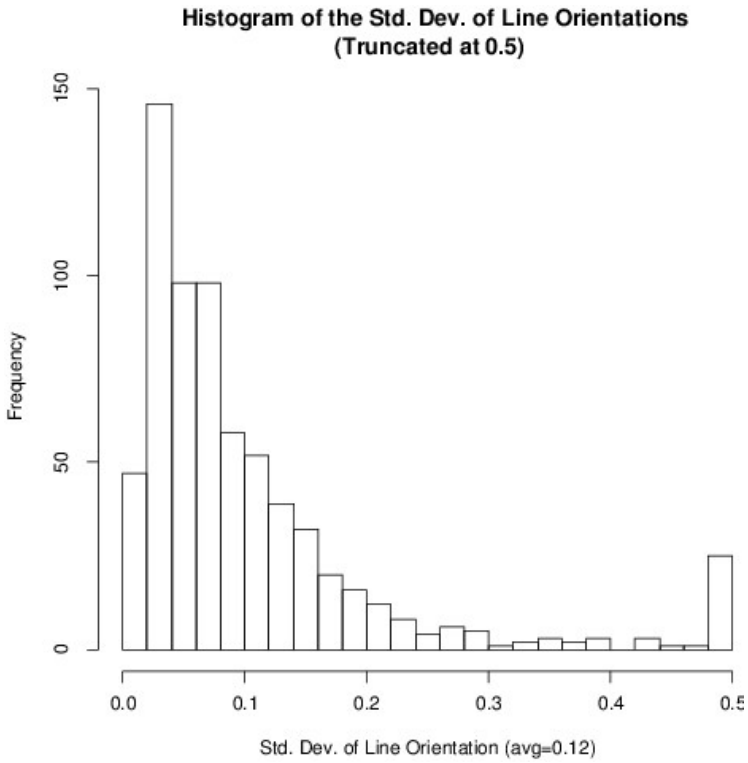

Figure 4. A histogram of the standard deviation of line orientations, truncated at 0.5 (values over 0.5 were put into the last bin). To create this histogram, the orientations of all the text lines in a page, as measured by the method presented in this paper, were determined and, within each page, their standard deviation around the mean orientation (in degrees) was calculated.

value that can be assigned to a page and the residual variance between measured and supplied page rotations may simply be due to the intrinsic variability of baseline orientations on these pages.

This hypothesis is supported by a histogram of the standard deviation of baseline orientations within each page (Figure 4). This histogram suggests that baseline orientations within a page can be quite variable. The average standard deviation is 0.12 degrees, corresponding approximately to the degree of scatter of measured vs. supplied determined page rotations exhibited in Figure 3.

6. DISCUSSION

This paper has described a simple and effective method for locating text lines in document images. The algorithm computes globally optimal maximum likelihood solutions under a Gaussian error model to the text line finding problem. By taking into account both characters on the baseline and descenders, the method can accumulate evidence for the presence of text lines in document images that would be missed by many other approaches. It is easy to implement and has only a very small number of free parameters (error bound, ranges of permissible page rotations and descender sizes, and numerical accuracy). The experimental results reported show that the algorithm is robust and yields accurate results.

There is a number of areas in which the method can be improved and extended. The algorithm shares with other layout analysis systems a certain sensitivity to document degradations where large numbers characters are merged into the same connected component. However, such degraded documents can be handled either using morphological preprocessing or by processing word outlines; incorporating support for this is planned in the future.

One particularly interesting property of this text line finding and page rotation algorithm is that it allows different text lines on the same page to have different orientations. This permits the algorithm to be applied to document images that have undergone a projective or affine transformation and is a useful step in the dewarping of document images captured by cameras rather than scanners (these results will be reported elsewhere). The line finder on which this text line localization algorithm is based tends to be robust and accurate when applied to natural scenes; this suggests that the text line localization method described here should also prove useful in the localization and analysis of text found in natural scenes.

The method can also be extended to non-linear (linear with a small quadratic or cubic contribution) baseline models without a significant increase in computational cost. Such very precise baseline modeling is helpful for some kinds of OCR systems, like the DID system.[7]

REFERENCES

1. H. S. Baird. The skew angle of printed documents. In *Proc., 1987 Conf. of the Society of Photographic Scientists and Engineers*, Rochester, New York, 1987.
2. Thomas M. Breuel. Fast Recognition using Adaptive Subdivisions of Transformation Space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 445–451, 1992.
3. Thomas M. Breuel. Finding Lines under Bounded Error. *Pattern Recognition*, 29(1):167–178, 1996.
4. Robert M. Haralick. Document image understanding: Geometric and logical layout. In *CVPR94: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 385–390, 1994.
5. William Wells III. Statistical approaches to feature-based object recognition. *International Journal of Computer Vision*, 21(1/2):63–98, 1997.
6. D. Ittner and H. Baird. Language-free layout analysis, 1993.
7. Gary E. Kopec and Philip A. Chou. Document image decoding using Markov source models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):602–617, June 1994.
8. O. Okun, M. Pietikainen, and J. Sauvola. Robust document skew detection based on line extraction. In *Proc. of the 11th Scandinavian Conference on Image Analysis (SCIA'99), June 7-11, Kangerlussuaq, Greenland*, pages 457–464, 1999.
9. C. F. Olson. Locating geometric primitives by pruning the parameter space. *Pattern Recognition*, 34:1247–1256, 2001.
10. I. Phillips, S. Chen, J. Ha, and R. Haralick. English document database design and implementation methodology. In *Proc. of the Second Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV*, pages 65–104., 1993.
11. S. Srihari and V. Govindaraju. Analysis of textual images using the hough transform. *Machine Vision and Applications*, 2, 1989.