Document Segmentation with Application for the Book Publishing Industry

Dan Chevion, Ehud D. Karnin, Gerry Thompson, Asaf Tzadok, Chai Wah Wu
IBM Research Division
Joe Czyszczewski, Mike Jensen, Hong Li
IBM Printing Systems Division

Keywords: Scanning, Printing, Publishing, Compression, Quality

Abstract: We describe a method of segmenting a scanned page into Text, Image, Line-Art and background. Each segment undergoes specific image processing and compression routines, based on its type, and the document is then reassembled as in the original page. This procedure improves the print quality of the document, being as close as possible to the paper original, and eliminates artifacts that would otherwise result in printing a scanned document. Moreover, the disparate compression algorithms yield a reduced size file, improving performance in printers, servers, and networks.

1. Introduction

There are many scenarios where documents are available in paper form only, and there is a business need to digitize them. Our original motivation to look at this problem stemmed from the need to reprint out of print titles [1]. These books are scanned by high volume scanners that generally support a maximum resolution of 300dpi, and use 8 bit gray-scale mode to optimize image quality.

An attempt to directly print the scanned images would result in a poor printout. The digital printer would generate a bi-level image, say of 600 dpi, by halftoning the gray scale input. The text will be far from sharp solid black characters, and the image region will show Moiré patterns. The reason for the latter phenomenon is that the originals were screened for printing, and the reprint uses a different screen grid.

Thus it is desirable to separate the different regions, so that they would undergo different image processing methods, targeted at alleviating their specific potential artifacts.

Before the segmentation process, we compute the skew angle and de-skew the whole page. Then detect the scanned page edges and make horizontal and vertical alignment and cropping adjustments.

The segmentation process [2] partitions the scan into four parts:

- **Image** part is detected by a pyramid based algorithm, and then the image is de-screened [3], to avoid Moiré artifacts when reprinted. It is kept in grayscale and compressed using standard JPEG. This is described in section 2.
- **Text** is detected by bottom up building of an hierarchical connectivity graph. We do it after the image part is taken out, just to ease on the processing. The text regions are up-scaled, say from 300dpi to 600dpi, and only then binarized. The text region is compressed by a standard binary image compression method, for example G4. This is described in section 3.
- **Line-art** regions are exposed after the image and text parts are detected. We binarize the line-art region for printing by error diffusion halftone. This is mentioned in section 3, in the context of text processing.
- **Background** or white space is not a region is the same sense as the former three, it is the remainder area. One may apply de-speckle to the scan, which result in cleaning of the background area.

Upon segmenting, processing and compression, the components are reassembled into a single file that supports the complex structure. We use PDF (PostScript is another option). The reconstructed file is much smaller the original compressed scan file. This has obvious advantages in storage, as well as saving on networking and servers demands in printing systems.

In section 4 we provide some experimental results.

Section 5 summarizes our work, and adds some concluding remarks on misclassifications.

2. Image Segmentation

3.1 Image detection

Images are best detected at low resolution, because they are relatively large objects and the decision must be region-based. On the other hand, procedures that make use of low resolution projections of images would insert annoying blocking effects. These two facts suggest looking at the problem from its both ends. Hence we work with multi-resolution structures to avoid blockiness while still making region based decisions. Many researchers have explored the multi-resolution structure, or pyramid as it is sometimes called, in order to extract objects from images [4 – 8].

The tasks of segmenting an image and the estimation of its properties are highly interdependent. Thus we use a linked pyramid structure as a framework for an iterative process, as described by Burt and Rosenfeld in their pioneering article [9].

Using the pyramid we compute a "measure of interest", which is a two dimensional array at low resolution, from which we extract the image objects. (high values in this map correspond to image regions). The ground level of our pyramid is at the original resolution of the scanned page. The value, or measure, which is assigned to each pixel, is the number of different colors (or different grey levels) at its vicinity. The vicinity is our case is a 4 x 4 square around the pixel.

Each layer of the pyramid is calculated from a layer below, as illustrated in Figure 1, which is a one dimensional description. A pixel in layer k is obtained by averaging a sub-array in layer k-1 (which is the layer below). Initially we take a full 4 x 4 array (see the four links in the 1 dimensional description in figure 1), but after climbing up few layers, we re-examine the father-sons relations.

For this re-examination we start again at the lower level, and look, for each son, for the closest father. Links to the other fathers are pruned. (I.e., each son has only one father, and each father may have between 0 to 16 sons). Again, starting from the ground level, we assign to each father a value which is the average value of its sons. This average is now projected down from a father to all its sons.

We now iterate on these three steps (determining the fathers, computing the averages, and projecting their values top down), until the process converges. In practice about 3 iterations would suffice.

A threshold is applied to the low resolution map which results from the iterative process. Values that exceed the threshold are likely to be part of an image region, as they correspond to areas where color or grey levels are varying. On the other hand, line art, and certainly text, tend to have a single color across a large area.

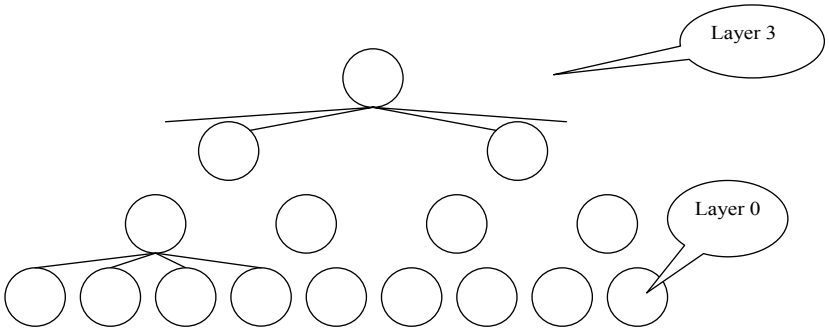

Figure 1: A linked pyramid structure

3.2 Tiling a pyramid.

The pyramid is a symmetric structure, in *x–y* axes, while the original documents are not. Therefore, we tile the pyramid as depicted in figure 2. Many small and symmetric pyramids cover the image, each one is N×N, where N = 256 in our implementation. The tiles are somewhat overlapping to avoid edge effects.

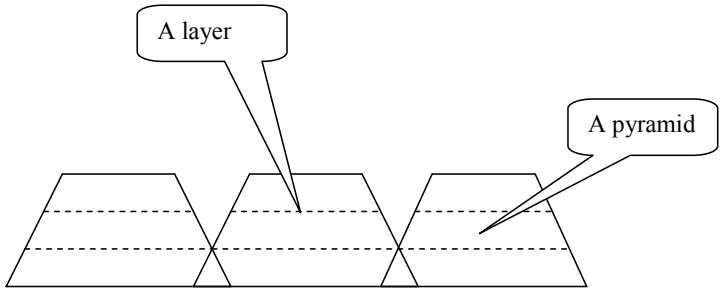

Figure 2: Tiling of an image with overlapping pyramid structures

3.3 Separation between detected continuous tone objects and line-art objects

After applying the threshold we get a mask: black regions indicate locations for which the original image is made of local alterations ("busy areas"), while white regions indicate "calm" areas. The latter may be background area, or areas painted with just a single gray level such as synthetically produced images and graphics.

This mask at its current form does not define the final image segmentation due to two problems. One is that the binary objects that hopefully correspond to images are not complete – they contain holes. The second problem is that some masks actually correspond to synthetic images that should be considered as line-art.

We solve these problems we have chosen a connected components approach to analyze those relatively large compound objects. First we combine the black regions into connected components, and assign some features to each component (e.g., area, moments, bounding box). One of these features is a histogram of the areas of its white holes.

Now, for each such a body (i.e., the black connected component), we look at its white spaces, and linked them to a connected component structure.

We now study the topology of the black bodies in order to classify them. We take advantage of the fact that connected components that were created from line-art usually look like skeletons - bones or wires wrapped around large hollow cavities. On the other hand, connected components created from continuous tone images are fat bodies with holes in them.

Once a body was decided to be an image region, we fill its holes (color it in black), as we have already found them. This is a tremendous computational advantage over using morphological operations to do this task.

In many cases continuous tone images are really rectangles, and should have rectangular masks. In these cases when the solid mask is very close to the ideal bounding rectangle, we replace the mask with its ideal representation.

The image regions are de-screened, to avoid Moiré artifacts when reprinted. They are retained in grayscale and compressed using standard JPEG.

3. Text Segmentation

3.1 Text detection

Like the image part, the text is extracted in a positive manner. Hence, in principle, we could have started from text segmentation rather than the image. Here we describe a bottom-up approach for detecting the text regions, by building a hierarchical structure.

We start by traversing the page at its pixel representation, of say 300 dpi, and generate a connected component structure. Adjacent pixels that exceed a certain level of darkness are grouped to a component. Several features are assigned to each component to characterize its geometric behavior, among which is a set of vectors pointing to neighboring components. These vectors have a crucial role in the geometric layout analysis.

The first use of these pointers from one component to its neighbors is in skew estimation. Skew is the document orientation angle with respect to the horizontal or vertical direction. Its estimation is an important step, as many document processing tasks lean on that. Many ways have been suggested in the literature to solve the skew estimation problem, like Hough transform and Fourier spectrum analysis. Here we adopt a method which is quite simple, accurate and robust, and most importantly it naturally fits to the general framework of our analysis.

For each vector that points from one connected component to its neighbor we compute the angle, relative to the horizontal direction of the scanned page. We then generate a histogram of these angles, as depicted in figure 3 below. Since text is usually arranged in lines, we expect this histogram to produce a high peak at the angle that corresponds to the lines orientation. Thus the skew is found from this histogram, and the scan can be de-skewed.

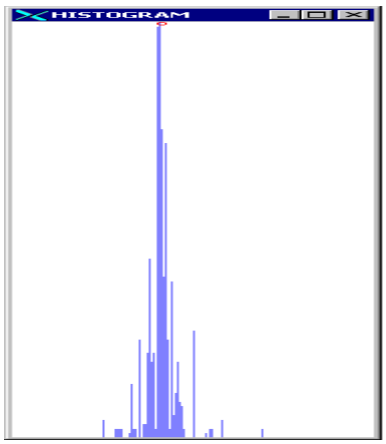

Figure 3: Histogram of angles of directions for all connected components

Once the skew is estimated, the set of vectors between elementary connected components and their neighbors is trimmed, leaving only connections in the estimated skew direction. We end up with structures that correspond to lines of text. Further, based on the length of the connecting vector, one can roughly determine the breakdown of the line to words, where a word is characterized by shorter links among its members.

On the other end, going perpendicular to the estimated skew direction reveals where lines are close or far apart, and detect the structure of paragraphs.

Thus we construct a hierarchical connectivity graph that reveals the layout of the presumed text area. The hypothesis (of a text region) can be strengthened by using other features, like components, which correspond to characters in the same line will have a similar height (the length in pixels in the dimension perpendicular to the skew).

Many important layout features can be detected as a result of this construction. Line alignment in the vertical direction, for instance, exposes blocks of text and discloses the direction of reading. Paragraph placement in the page, title detection, and other paragraph attributes such as footnotes, etc. are detected using this hierarchical connectivity graph.

Talking into consideration the features of the hierarchal connectivity graph, text region detection is very reliable. We also note that the same algorithms will work for vertically written text, which may be the default in languages like

Chinese or an exception at an English document. And of course we detect text in both left to right or right to left written languages.

3.2 Text processing and compression

As we mentioned above, the resolution of commercial scanners is typically limited to 300 dpi. We detect the text areas and upscale them to 600 dpi, while still in grey level. Only after this interpolation step we apply the binarization process that classifies every pixel to either black or white. The solid black characters at the high resolution of 600 dpi produce a much nicer print than what would have been obtained by printing the scanned page. In that latter case of the 256 grey levels the resulting printout suffers from annoying halftone artifacts, which we were able to eliminate.

Note that it is not necessary to interpolate at every pixel, but rather at border pixels. Since the characters are typically several pixels wide, a substantial saving in computational time is achieved.

As an aside, we describe another processing step that helps getting a sharper text, while we start from a color scan. We observed that color components, namely RGB, of the scanner are not perfectly aligned. It shows up as fringes of one color with respect to the other (while the text should have been black). Since the grey scale image is obtained as a linear combination of the three color components, it would be advantageous to align them prior to computing their weighted sum, an operation we termed as de-fringe. To find the displacement we formulated a minimization problem to solve, as follows:

Let r be the vector representing the correct red layer, and let g be the vector of the green layer of a grey level sub image, i.e., a piece of text. We will represent r as a linear combination of shifts of the observed red layer (thus allowing sub-pixel shifts). The shifts are represented by the matrix R, where each column in R is a vector represented by one pixel shift in the red layer. The coefficients of the desired linear combination are a vector c that multiplies R, therefore

$$r=Rc$$

The difference between the correct layers of these two colors should be minimal, hence we find the minimum of the expression

$$(r-g)^T(r-g)$$

This is a well known Least Mean Square (LMS) problem which is solved (after putting $r=Rc$) as

$$c=(R^TR)^{-1}R^Tg$$

Using the coefficient vector *c* that we just found we correct the red color component with respect to the green, and improved the alignment to produce a sharper grey level image. We repeat the same procedure for shifting the blue color component relative to the green, to complete the color registration process.

Once we completed the processing (alignment and upscale for the text region), we compress the bilevel text region. There are several standard algorithms to do it, like the G4 method used in (binary) fax compression. Another option is JBIG.

3.3 Line art segmentation

At this point we have located the image and the text region by a positive method. While the image regions are detected (by the pyramid method described in section 2), some line-art regions are also revealed. Other line art areas are exposed as the remaining region after we detecting image and text; hence we may say that it is detected by elimination.

The line art region is not necessarily a solid black region, and we do not want to binarize it. On the other hand it may be composed of thin lines, so de-screening methods, as applied to the image part, may blur it. We choose to force it to be a black and white image, so that printers will not halftone it by uncontrolled methods. We apply error diffusion, and in this way we preserve both the sharpness of the lines and the potential grey scale appearance of these regions.

Since line-art regions are kept as a bilevel image, they are compressed by similar algorithms to those we use for text compression, e.g., G4 or JBIG.

4. Experimental Results

The algorithms that were described in the previous sections were implemented in standard C++ language, and run in a Windows environment.

We have chosen few scans to demonstrate their behavior. The left page in figure 4 below shows a page composed of a large image rectangle, and some text. The right page has lot of line-art in addition to some (non-rectangular) image regions and text regions.

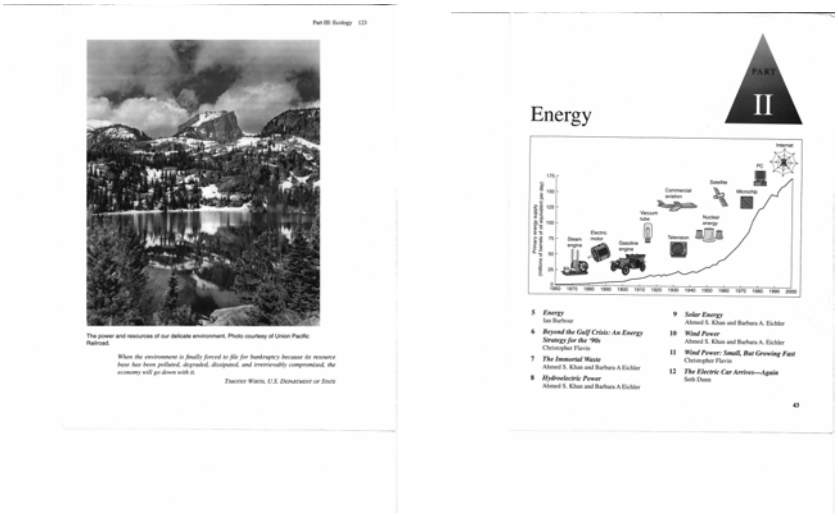

Figure 4: Original document images

The segmentation of the left document is shown in figure 5 below, the image on the left side and the text to its right. When they are separately processed and compressed, the reassembled PDF file is 0.67 MB, while the original file was 8.148 MB uncompressed, and 1.313 MB compressed with JPEG.

The right document is segmented to three different regions. We have found the text both in normal lines beneath the diagram, as well in the isolated words embedded in the diagram, near the graph. The small synthetic images (within the diagram) were correctly classified as line-art objects. Again a substantial saving in storage was obtained in the re-assembled PDF representation, 0.368 MB versus 8.148 MB in the original scan uncompressed and 0.438 MB after compression with JPEG.

Figure 5: segmented document

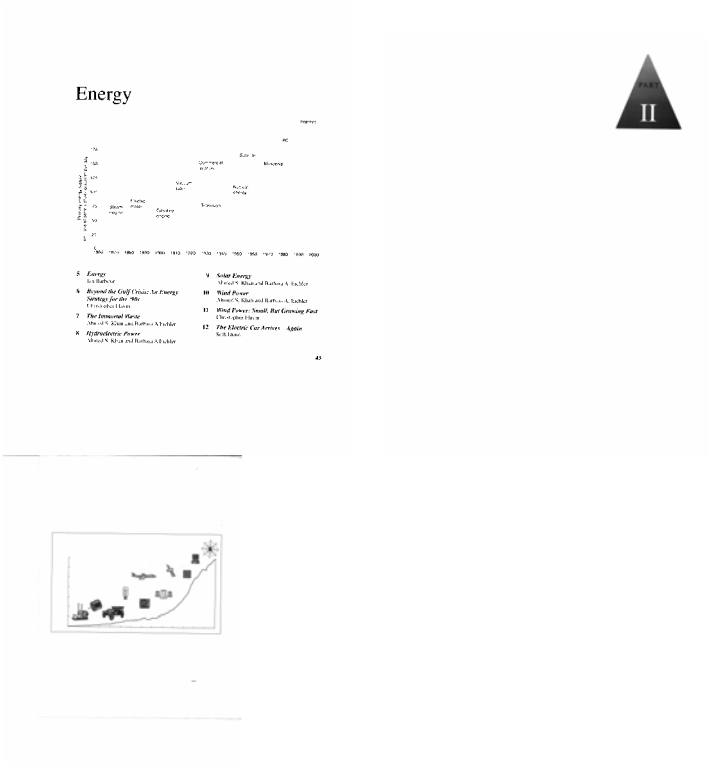

Figure 6: Segmentation into text, continuous tone image and line-art

5. Summary

We have developed an automatic method for segmenting a scanned page into Text, Image, Line-Art and background. Each segment is processed for improved quality, and compressed for reducing the re-assembled file size.

Our extensive experimentation with the prototype that we built shows quite a robust behavior. However, some misclassifications are inevitable. First we would comment that most misclassification do not damage the quality. For example, a small text area that goes into the line-art region will still appear quite nice and sharp.

To correct misclassifications and to override the decision of the automatic system, we built an editing environment. The user can work with the regions in their connected component representation. Thus the correction, which means the transfer of a region from one class to the other, can be done on a component basis (or many components selected by a bounding box). This way there is no need for precise manual definition of the region, a process that be quite time consuming.

We mentioned that our original motivation to look at this problem stemmed from the need to reprint out of print titles. However we believe that the segmentation system has wider application domains. In fact it may be useful in many office environments, where documents are scanned, stored, and proceed to further processing step like data extraction by optical character recognition.

References

[1] J. S. Czyszczewski, J. T. Smith and H. Li, "Accelerating Production Book Scanning," *DPP2003, IS&Ts International Conference On Digital Production Printing and Industrial Applications*, IS&T – The Society for Imaging Science and Technology, Barcelona SPAIN, pp. 38-39, May 18-21, 2003

[2] R. Cattoni, T. Coianiz, S. Messelodi and C.M. Modena, "Geometric Layout Analysis Techniques for Document Image Understanding: a Review," *Technical report ITC-IRST,* Italy, January 1998.

[3] J. Rydenius; "Inverse Halftoning of Scanned Colour Images," *Master's thesis*, Image Processing Laboratory Department of Electrical Engineering, Linkoeping University and Institute of Technology, January 1997.

[4] A.D Gross And A. Rosenfeld, "Multiresolution Object Detection and Delineation"; *Computer Vision, Graphics and Image Processing 39*, pp. 102-115, 1987.

[5] L. Cinque, L. Lombardi, G. Manzini, "A multiresolution approach for page segmentation," *Pattern Recognition Letters 19*, pp. 217-225, 1998.

[6] M. Bister J. Cornelis and A. Rosenfeld, "A Critical view of pyramid segmentation algorithms," *Patern Recognition Letters 11*, Elsevier Science Publishers B.V. North-Holand, pp. 605-617, 1990.

[7] D. Prewer, L. Kitchen, "Weighted Linked Pyramids and Soft Sefgmentation of Colour Images,", *ACCV2000, Taipei, Taiwn, vol. 2,* pp. 989-994, Jan 2000.

[8] M.O. Shneier, "Extracting Linear Features from Images Using Pyramids," *IEEE Transactions On Systems, Man, and Cybernetics, Vol., SMC-12, No. 4,* July/August 1982.

[9] P.J. Burt and A. Rosenfeld; "Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchial Computation," *IEEE transactions on system, man and cybernetics, Vol, SNC-11 no 12*, December 1981