Retrieving Information from Document Images: Problems and Solutions

Fu Chang
Document Analysis and Recognition Laboratory
Institute of Information Science 20
Academia Sinica
128 Academia Road, Section 2
Nankang, Taipei 115
Taiwan R.O.C.
Email: fchang@iis.sinica.edu.tw

ABSTRACT

An information retrieval system that captures both visual and textual contents from paper documents can derive maximal benefits out of DAR techniques while demanding little human assistance for achieving its goals. This article discusses the technical problems, solution methods and integration of them into a well-performing system. Focus of the discussion is on very hard applications, for example, to Chinese and Japanese documents.

In addition to large group of potential readers, the latter types of documents create many technical issues that deserve experts' attention. The complicated Chinese or Kanji characters, for example, impose serious problem for image binarization. The coexistence of vertical and horizontal textlines on the same page renders document segmentation difficult. The large number of characters also challenges the way textual contents are recognized and retrieved.

Problems discussed in this article will be centered on these issues. Solution methods will also be highlighted, with the emphasis placed upon some new ideas, including window-based binarization using scale measures, document layout analysis as solving multiple constraint problem, and full-text searching technique capable of evading machine recognition errors.

1. Introduction

Faithful representations of paper documents need to take care of both visual and symbolic information. The former includes icons, graphics, flow charts, tables, and physical layout of documents, while the latter covers all the textual contents, includ-

ing words, textlines, paragraphs and articles. To fetch both types of information at fairly low expense is possible for today's technology. For example, scanner can be used to obtain visual contents (or images) of documents, while OCR software can be used to transcribe their textual contents. But to obtain all the contents with the same level of faithfulness as they stand in the original documents is not an easy task.

The requirement can be somewhat lessened if only the retrieval values of the contents are honored. Thus, to prepare the *retrievable* contents out of paper documents, one can do the following.

(i) Use scanner, perhaps also image-processing software, to obtain visual representation of the documents.

(ii) Use OCR software to obtain text transcription out of the same pages.

To retrieve the information thus obtained and also stored in the database, one can further carry out the following operations.

(iii) Use search engine to find interested information out of the text contents.

(iv) Retrieve the corresponding images that contain the information.

The combination of the above operations takes advantage of the fact that images are good for viewing purpose and textual contents are ideal for information search. Admittedly, machine-transcribed texts can hardly be error-proof. But the machine errors do not appear on the images and therefore do not have any impact to the viewing process. Their impact to the search process can also be minimized if certain error-tolerance mechanism is inserted in the process. The net result would be the exemption of human assistance in preparing retrievable information (except for feeding documents to scanners) and nearly no miss in fetching information from the visual contents.

Balance of loads is thus the major idea in the above consideration. What is deficient in one working component (machine transcription of textual contents) can be

compensated by other components (production of visual contents and fetch information out of them). For sure, requirements become more stringent for the components to which the burdens are shifted. First, the visual contents must be sufficiently good for human eyes. Next, retrieving information requires correct reading of not only the words, but also the word orders, out of the documents. Finally, text-searching capability also needs to be strengthened so as to bypass possible errors occurred in machine transcription.

While all these issues may already fall within the realm of DAR interests, we want to address in this article some specific problems that are associated with Chinese documents (or their similar types, for example, Japanese documents). With the last five years of efforts, our laboratory has put together a system that is capable of transforming paper documents into viewable and searchable contents. There are four major components embedded in this system. (1) Image Processing: the core is an algorithm that transforms gray-scaled images into binary images. (2) Document Analysis**:** works covered by this component include skew calibration, segregation of pictures and frames from textual contents, extraction of textual blocks, textlines, and individual characters, etc. (3) OCR Engine: it conducts recognition over printed documents consisting of Chinese, English and numeral characters. (4) Search Engine: the searching capability of this component is enhanced by an OCR-error-tolerant function.

Undoubtedly, some of the methods used in other applications (for example, English documents) can readily lend solutions to the problems associated with Chinese documents. Examples as such include skew calibration, segregation of non-textual information, and some feature extraction methods. But the complicated nature of Chinese documents create many new problems, or perhaps, harder problems. It has already been observed long time ago that Chinese character is probably the most difficult type among all others for machine recognition. The number of characters and

their complicated structures are major causes for the difficulty. The complication of characters, moreover, requires the capability of telling apart the subtle distinctions between foreground and background pixels. Chinese documents are also known to allow two types of word orders—horizontal and vertical—to appear on the same pages. This enlarges the degree of freedom for the possible document layout structures. The fact that many Chinese characters are themselves compositions of elementary entities also increases the complexity of algorithms designed for analyzing the structures.

In this article, we will thus focus the discussions on the problems that are specific to Chinese document applications and, above all, the problems that are vital from the information retrieval point of view. The rest of the paper proceeds in the following order. A discussion of binarization of document images is put forth in the next section. Section 3 covers the general issues in analyzing layout structures of Chinese documents. Section 4 discusses the textual search mechanism with machine-error tolerance. Section 5 constitutes a brief summary for the whole article.

2. Binarization of Document Images

Scanners rely upon their internal mechanism to transform gray-scale images (in which each pixel assumes one of 256 possible values) into binary images (in which each pixel assumes either black or white value). However, when one applies them to produce binary images for aged Chinese documents, many small but complicated characters will become blurred.

Years ago, before we started to work out our own solution, we thought naively that a good choice of threshold would solve the problem. Here, by threshold we mean a value, say T, based on which each pixel can be classified as either black or white,

according to whether its gray value falls below or above T[1]. To test this hypothesis, we conducted a very simple experiment. We picked an aged newspaper article and produced a gray-scale image as the source image. Out of it, we then created a series of binary images by using each possible level, ranging from 0 to 255, as the cutting value (namely, T) for blacks and whites. To our surprise, we observed the following outcomes.

For low values of T, simple characters appear broken while complicated characters look all right. As the value of T elevates to a higher value, simple characters become intact but complicated characters turn into blurred (an example is now shown in Figure 1). Furthermore, we could not find any value of T at which both simple and complicated characters appeared okay at the same time.

(a) **(b)** **(c)**

Figure 1. The binary images of two Chinese characters, one simple and one complicated. The three sets of images are obtained by setting (a) T = 65, (b) T = 80, and (c) T = 95. As the threshold value increases, simple character becomes intact while complicated character becomes blurred.

As we scrutinized the source image, we found the reason as follows for this unhappy finding. Like many other characters, Chinese characters are composed of strokes. Within a small but complicated character, the background regions that are surrounded by strokes become extremely small. Because of the point-spread function that is used by scanners to calculate gray values, the pixels within these “tiny valleys” appear darker (i.e., their gray values are lower) than the average background pixels. Thus, when a low value of T is chosen as the cutting value for blacks and whites, the

[1] By convention, low gray values represent dark grades.

tiny valleys are classified as white, but some weak pixels on the simple characters are also classified as white, resulting in broken characters. When T elevates to certain higher level, the small characters become intact, but the tiny valleys are classified as black, resulting in blurred characters.

Our finding thus proves that a global threshold does not solve the binarization problem well. It also suggests that making local comparison of gray values may be necessary to take care of the subtle distinctions between background and foreground pixels. Let us then consider the following method.

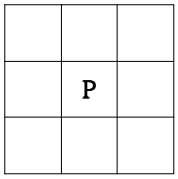


Figure 2. $W_p(3)$ gets all its members from a window of size 3×3 centered at pixel p.

For each pixel p, let $W_p(n)$ the collection of all neighboring pixels of p that falls within a window of size $n \times n$ centered at p (figure 2). Let

$$\text{Mini}(p) = \min\{g(q): q \, \varepsilon \, W_p(n)\},$$
$$\text{Maxi}(p) = \max\{g(q): q \, \varepsilon \, W_p(n)\}.$$

We then consider the difference of g(p) from Mini(p) and from Maxi(p). If the former difference is larger, then g(p) is closer to the highest gray value than the lowest value in this window. Since a pixel with high gray value is most likely a background pixel, we thus classify p as a background (i.e., white) pixel. Otherwise, we classify p as a foreground pixel.

The merit of the above method is that it does not rely on any threshold value for the classification. This method, however, does depend on the proper choice of window size, as we will see. For this reason, we call it *window-based binarization method* or *local binarization method*. In Figure 3, a result of applying this method is displayed,

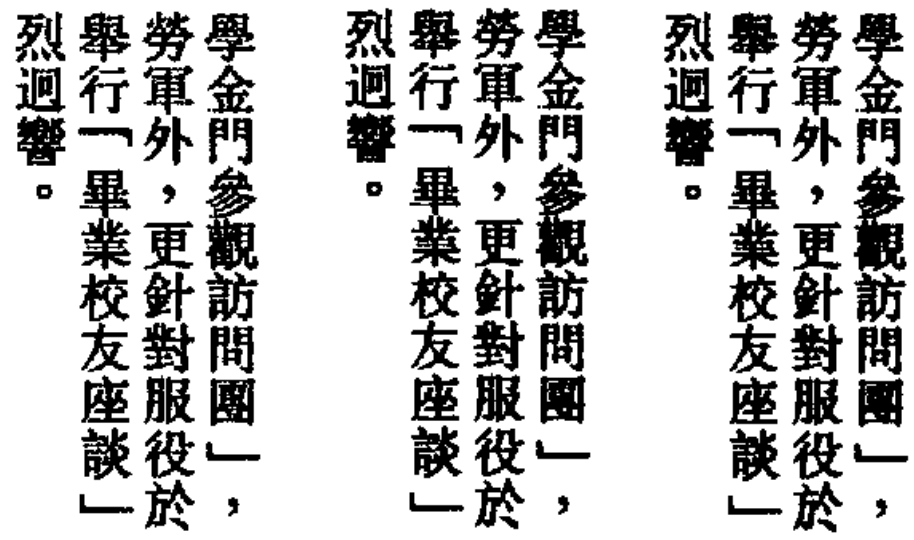
where the window size is set as 3×3. It is seen that all the characters surface out rather nicely. The clear drawback is the appearance of noises all over the background area.

Figure 3. The result of applying the window-based binarization method, where the window size is set as 3×3.

A moment's reflection would immediately suggest a possible way to improve the solution: perhaps we should restrict the application of window-based binarization only to where it is really useful. But where is the method useful anyway? Presumably, it is where the gray value fails to reflect the true status of a given pixel. Normally, when a gray value is very high or very low, it is a sure evidence for the white or black. The uncertain area is somewhere in between the extreme values.

This fact suggests that we can make a combined use of global threshold method and local binarization method. When the gray value g(p) of pixel p is far away from the global threshold T, we classify p as black or white according to whether g(p) lies on the far left or far right of T. If, however, g(p) is in the near neighborhood of T, the window-based binarization method will be used for determining the binary value of p.

The results of applying this hybrid method as described above are shown in Figure 4, where global threshold is calculated by Otsu's method [1]. The window sizes are set in three different values so that we can make comparisons. Obviously, all the results look better than the previous one. Background noises now disappear. But a



(a) **(b)** **(c)**

Figure 4. Outputs of the combined use of global threshold method and window-based binarization method with window size set as (a) 5×5, (b) 15×15, and (c) 21×21.

problem remains to be solved. What window size should we set for the local binarization method? In Figure 4, complicated characters become blurred as window size increases. So, as far as this figure is concerned, it seems that we should choose the size as 5×5 or smaller.

But if we look at Figure 5, we would conclude differently. It seems that a larger size is more appropriate in this case. For as the window size decreases, foreground noises (or white noises) also show up.

So what is going on? Why are different window sizes required for different cases? But first of all, what is the difference between the two cases anyway? At the prelimi-

(a) **(b)** **(c)**

Figure 5. Outputs of the hybrid method. The window size for the local binarization method is set as (a) 5×5, (b) 9×9, and (c) 13×13.

nary scrutiny, we observe that the characters in Figure 4 are smaller than the character in Figure 5, and therefore have thinner strokes than the latter. Next, we note that the appropriate window size ought to be compatible with the average stroke width of the characters to which the local binarization method is applied.

But why is this magnitude so crucial? When we use the local binarization method, we compare a given pixel p with all the pixels falling within a window centered at p. If the size of this window has the right magnitude, there is a good mixture of foreground and background pixels within this window. The window-based binarization method would be able to gather all the relevant information and make the correct decision.

Let us see what would happen if we employ an inappropriate window size. Suppose that the size is taken smaller than it should be, and p is a foreground pixel. Then all the pixels falling within the window are themselves foreground pixels. We are then comparing unfairly a foreground pixel with all foreground pixels! Wrong conclusion can be drawn for p, if p happens to be not as dark as its neighboring fellows. This accounts for the white noises in Figure 5a and 5b.

On the other hand, suppose that the window size is set larger than it should be, and p is a pixel lying within a tiny valley. Then, the window centered at p may contain some background pixels lying outside of the character. Since p lies within the valley, it usually appears darker than the remote background pixels. We can thus misclassify p as a black pixel. This accounts for the blurred characters in Figure 4b and 4c.

The above consideration suggests that a kind of magnitude, to be called “scale” in this article, must be calculated before window-based binarization can proceed.

There can be many ways for obtaining this magnitude. One straightforward way is to calculate the average run-length for each character. However, there is a serious problem with this approach. The run-length measure is not defined until a binary im-

age is created. But once a binary image is initialized, it becomes a self-prophesying process: if the character appears blurred in the image, its run-length estimate would become large. Applying window-based binarization based on this estimate would still result in a blurred character.

The second approach is to infer the stroke widths of characters based on the sizes of textlines. Here the size of a textline L is defined as the height or the width of L, depending whether L is a horizontal or a vertical line. This is a workable approach, since the extraction of textline can be made independent of the binarization process. The inference of stroke widths can also be reliable, since stroke widths usually grow proportional to the sizes of textlines. But in this approach, the success of window-based binarization must hinge upon a flawless pre-step for textline extraction. It is rather burdensome.

The third approach is to obtain multi-scale measures on the source image. The idea is the following. We can use a set of “yardsticks” for the measurements. Each of them is suitable for a certain scale. We then select the best reading out of all the possible measurements.

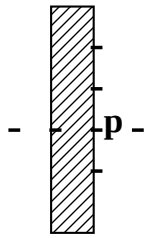
A yardstick is actually a 2n-dimensional vector, where n is an even number. When n = 2m, the general form of the vector is the following.

$$(\overbrace{1, \ldots, 1}^{m}, \overbrace{-1, \ldots, -1}^{n}, \overbrace{1 \ldots, 1}^{m})$$

The 2n-dimensional vector will be called yardstick of scale n, or in abbreviation, Y_n. Thus, in this terminology, Y_2 = (1, -1, -1, 1), Y_4 = (1, 1, -1, -1, -1, -1, 1, 1), etc. There is no yardstick of scale 3, since we do not allow odd-numbered scale. The reason will be give later.

To make the measurement, a yardstick must be applied at a specific point. Thus, if a pixel p is given, we first gather its n neighboring pixels, all lying on the horizontal

line passing through p. We then form a vector H_n out of the n gray values. The *horizontal* reading of Y_n at p is then defined as $Y_n \bullet H_n$, where • is the sign for vector multiplication. To obtain the *vertical* reading of Y_n at p, we can gather n vertical neighbors of p and form a vector V_n out of their gray values. The vertical reading is then defined as $Y_n \bullet V_n$. Finally, the total reading of Y_n at p is defined as max(vertical reading, 0) + max(horizontal reading, 0).



Coefficients of Y_2	1	-1	-1	1
Coefficients of H_2	205	35	38	212
Coefficients of V_2	30	34	35	29

Figure 6. Application of Y_2 in both directions to pixel p. Pixels are indicated by dashes.

As an example, let us consider the effect of applying Y_2 at pixel p, as shown in Figure 6. The horizontal reading is (1, -1, -1, 1) • (205, 35, 38, 212) = 314. In this case, since p locates on a vertical stroke of 2-pixel width, the reading is obtained by summing up two background values and subtracting two foreground values. The result is a large positive number. The vertical reading, on the other hand, is obtained by summing up two foreground values and subtracting two foreground values also. Since all the foreground values are close to each other, the resulting value (-10) is a number close to zero.

If we apply a yardstick to all the pixels of the same stroke, sum up the readings and then take the average, we get some information about this stroke. As a general rule, a stroke reacts strongly to yardsticks of compatible scales. Moreover, if a stroke gets highest average reading from a yardstick of scale n, its average width must be

close to n. Strokes, however, are not easily identified from images. Instead, characters or parts of characters can be found at relative ease. Based on these ideas, we come up with a process for determining the window sizes as follows.

In order to apply window-based binarization method, we have to first come up with a global threshold T. We can thus employ this T to create a binary image and find all the connected components out of the image. Using some techniques not described here, we can separate textual components (i.e., characters or parts of characters) from non-textual ones (table lines, icons, graphics, etc.)

At reasonable level of resolution (200 or 300 dpi, for example), each textual component is most likely a whole character or part of a character. Thus, for such a component, say C, we can apply all the yardsticks as described above and find the one that yields the highest average reading. The scale of this yardstick is defined as the scale of C. The window size for C is then set as s+1, where s is the scale of C.

Two facts are worth mentioning here. First, window-based binarization method is quite robust to slight variations of scale estimates. A deviation of one and sometimes even two pixels is tolerable. This is why we are satisfied with even-numbered scales. The nice thing about even-valued scales is that the corresponding windows become symmetric with respect to their centers. Secondly, window-based binarization would not add any utility for components whose scale is equal or higher than 8. This means that, for computing window sizes, we only need 4 yardsticks, of scale 2, 4, 6, or 8 respectively. Those components reacting most strongly to the yardstick of scale 8 presumably have stroke width larger than 8.

In Figure 7, we display two binary images. One of them is obtained from global threshold method, where the threshold value is calculated by way of Otsu's method. The other image is obtained by the hybrid method, that is, the combination of both global and local binarization methods. Improvements over small and complicated

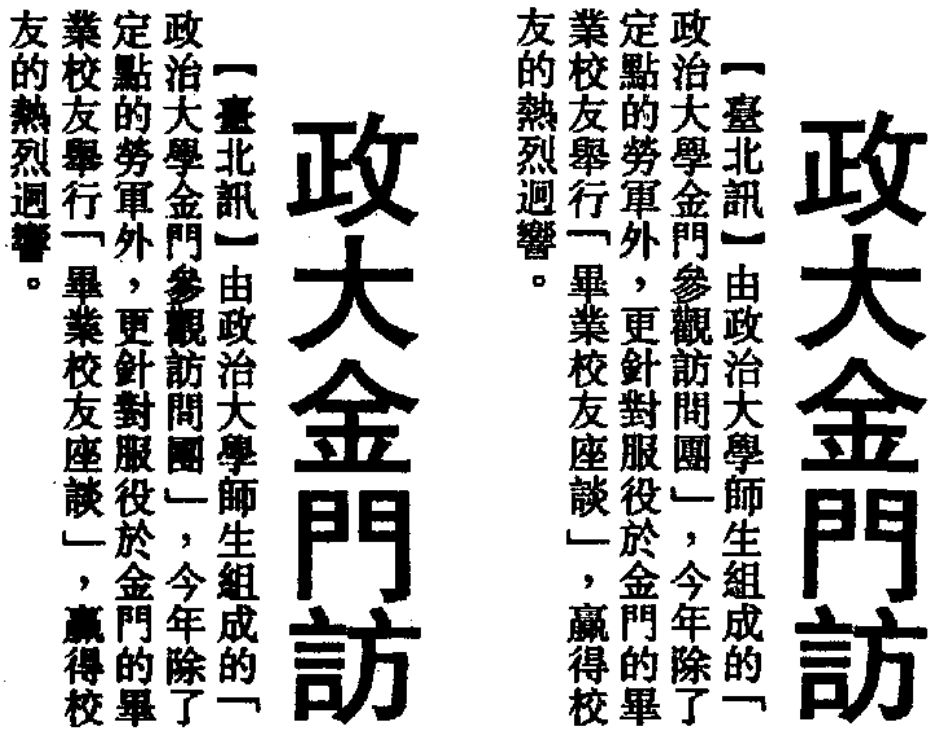


Figure 7. Left panel: binary image obtained by global threshold method. Right panel: binary image obtained by the hybrid method.

characters by the hybrid method should be clear as one compares the two output images. Note that the common input to both methods is a gray-scale image made out of a clipped newspaper article, dated July 13, 1993.

Readers may also feel happy to know that the hybrid method not only improves the visual quality of binary images but also helps to improve the performance of OCR engines. For testing this assertion, we gathered three sets of samples: recent articles, articles printed in 1996, and articles printed in 1969. Each set consists of 20 testing samples, each sample containing a few hundreds of Chinese characters.

Two OCR engines were employed for the testing. Both are fetched from commercial software so that there is no issue of possible bias. For each of the testing articles, we produced a gray-scale image and two binary images out of it, one by way of global threshold method and the other by the hybrid method. When applied the OCR

engines to one and then the other binary image, the increments in recognition rate are recorded in table I. It is clear that the effects of hybrid method are more manifest on aged articles than recent ones.

Table I. Testing Results

	Increased OCR Performance
Recent Articles	0%~10%
1996 Articles	5%~23%
1969 Articles	20%~37%

3. Document Layout Analysis

Binarization captures the visual contents of documents but do not interpret them. In the attempt of understanding printed documents, an advantage one can take is the apparent physical layout structure, from which it is possible to infer the distinct locations of characters and also their reading orders. Well-segmented characters make the job of machine recognition easier. Understanding the reading orders is a necessary ingredient for understanding the meaning of textual contents, for it is the compositions in certain orders that make characters to convey meanings. While all of these may sound so easy to human readers, they can cause a lot of problems for people who want to make machines do the same things. In this respect, Chinese documents are certainly no short of problems.

Documents organize their textual contents in hierarchical order. Individual characters are organized into textlines, which in turn are organized into paragraphs, etc. In the past, many methods look for an entering point to the hierarchical system. For example, the recursive X-Y cuts method [11] seeks to enter from the top of the hierarchy. It searches for large horizontal or vertical gap and decomposes an image into two sub-images. It then repeats the same operation to each of them. The maximal white-rectangles method [12] also seeks to enter the hierarchy from a top position. But in-

stead of decomposing the image from the perspective of projection profiles, it looks for all the maximal white rectangles that imbed in the layout structures. It then segregates the image from the large white rectangles. The run-length smearing method [13], on the other hand, seeks to enter from the lowest position of the hierarchy. It goes along each scan line and blackens the small intervals between black pixels. The smearing operation has thus the effect of gluing foreground pixels into connected pieces, that can be textlines, paragraphs, or mixtures of things, depending the parameter value being used. Textline construction method [14] may be said to enter the hierarchy from a middle position. Taking the advantage that textlines, in printed documents at least, are approximately aligned (from top and bottom for horizontal textlines, or from right and left for vertical textlines), this method endeavors to find them by means of a pair of rails (as in the railways).

All the methods are good in certain contexts but all have their limitations also. The recursive X-Y cuts is a handy method in the context of a single article. But on a full page of newspaper, there can be many articles and this method does not apply. The maximal white-rectangles method is applicable to more complicated environments. However, the size of white rectangles may not be a reliable clue for segregation. In Chinese documents, textlines can go in either horizontal or vertical direction. Textlines of different orientations are naturally segregated from each other, even though the white rectangles that go between them are not very large. In Figure 8, for example, white rectangle A is a natural region for segregation but its size is smaller than some other white rectangles (B and C) that are not.

The run-length smearing method has the problem in the parameter setting. At the level of 300 dpi, the inner spacing of small textlines are but 2 to 3 pixels wide, but they can grow to 80 pixels wide in headlines. But even within the same textline, sizes of gaps can vary wildly. For example, the gaps around punctuation marks can be 18

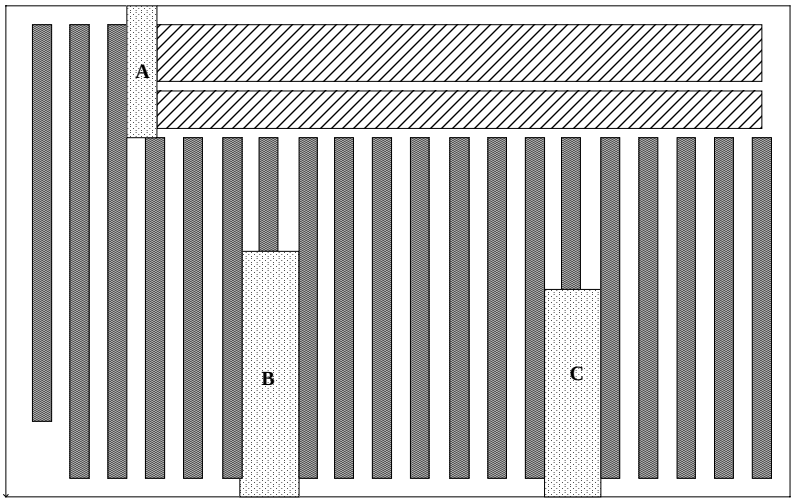


Figure 8. The layout structure of a Chinese article is displayed. Headlines (hashed boxes) are horizontally oriented. The rest of textlines (half-toned boxes) are vertically oriented. Dashed lines form the borders of the image. Dotted boxes are white rectangles.

pixels wide, while those between characters remain 2 or 3 pixels wide. Thus, counting on a constant size to bridge the gaps would loose all the flexibility that is desired.

Textline construction method takes more structural elements than other methods. But two issues arise when it is applied to Chinese documents. First, stemming from the same starting point it is possible to obtain two textlines, one going horizontally and the other vertically. Only one of them is legitimate, but none of them could be excluded a priori (Figure 9a). Secondly, a textline so constructed can overextend to the site of another textline, resulting in conflicts that have to be resolved (Figure 9b). Both issues are related with the fact that two possible orientations are allowed for textlines in Chinese documents. The higher degree of freedom lies in the heart of the problems here.

In fact, all the methods reviewed in the above can be useful in certain contexts. But it is important to first identify the contexts correctly before applying them. In our view, all methods for layout analysis employ some *split and merge* operations. Run-length smearing, for example, is a merge operation, while X-Y cut is a split operation.

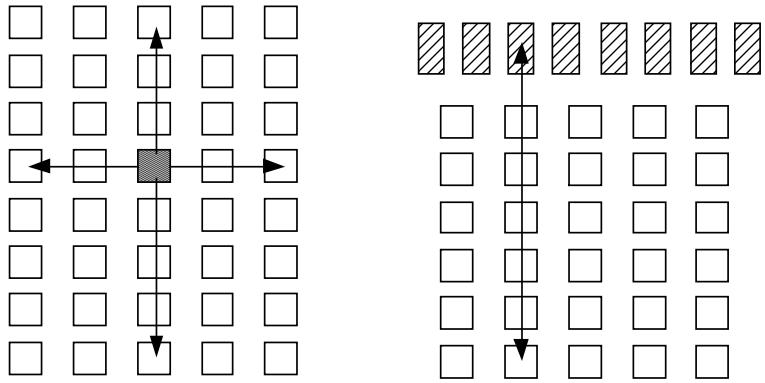


Figure 9. All boxes represent characters. (a) Starting from the same character (half-toned box), two textlines can be constructed, one going in horizontal direction, and the other vertical direction. (b) A vertical textline overextends to the site of a horizontal textline (where all the boxes are hashed).

To work out reasonable outcomes, these operations rely upon certain parameter values to function. In the previous section, the value of scale needs to be computed before the window-based binarization can proceed. But unlike the situation there, most of the parameter values associated with the document layout problem can not be computed independently. They are part of the problems that are being solved.

By nature, document layout analysis is a *multiple constraint problem*, where the objects (paragraphs, textlines, and characters, etc.) to be constructed or identified are the *unknown variables* built in the constraints. There are actually three sets of constraints that are met by most Chinese documents.

A. Composition Constraints: paragraphs are composed of contiguous textlines of similar sizes, which in turn are composed of contiguous characters of similar sizes; characters are composed of contiguous components, i.e., clusters of connected foreground pixels.

B. Alignment Constraints: characters contained in horizontal (vertical) textlines are properly aligned on their top (right) and bottom (left) edges; textlines

contained in horizontal (vertical) paragraphs are properly aligned on their top (right) and bottom (left) edges.

C. Spacing Constraints: the spacing within a textline is less than the spacing between textlines; moreover, the spacing within a paragraph is less than the spacing between paragraphs (Figure 10).

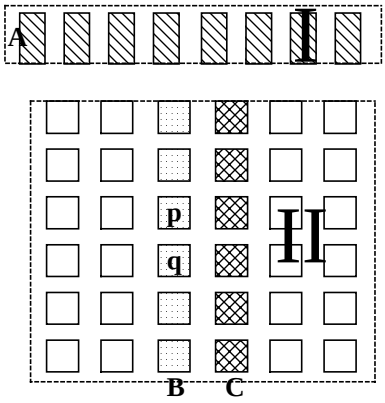


Figure 10. Two paragraphs I and II are enclosed in dashed lines. Paragraph I contains textline A, that is horizontally oriented. Paragraph II contains B, C and other vertical textlines. Each labeled textline is identified by the same fill pattern. The spacing within B (= distance between p and q) is less than that between B and C. The spacing within paragraph I (= distance between B and C) is less than that between I and II.

Each set of constraints involves more than one unknown variable and each unknown variable also occurs in more than one set of constraints. The third set of constraints, moreover, incorporates some parameters (inner spacing and outer spacing) whose values depend on other variables (textlines and paragraphs).

All the methods reviewed in the above attempt to solve the problem in a straightforward manner. They seek for a solution for some unknown variable, which in turn would help to solve for another unknown variable, etc. Unfortunately, for most hard constraint problems, straightforward solutions rarely exist. As an alternative, we propose the following method. We start with an approximate solution, that is, a solution satisfying some but not all the constraints. We then base on this temporary solution to

estimate the unknown parameter values appearing in some other constraints. We can then derive a better solution (namely, a solution that meet more constraints) by taking reference of the estimated values. The solution so reached can still be improved if more or better knowledge is generated.

At the start, the choice of primitives is important. Foreground pixels can certainly be taken as primitives but they are not as handy as components. A component can be represented by its enclosing box so that four numerical values (the x- and y-coordinates of its upper left and lower right corners) are sufficient for determining its location. The distance between two components can also be defined in terms of the distance between their enclosing boxes.

When working on the components, run-length smearing and textline construction become almost the same method. They both look for collections of contiguous components that are properly aligned on the top and bottom edges, or on the right and left edges. The textline construction method tolerates small amount of misalignment and is thus more robust. Our solution method employs the same technique to find all the textlines from document images.

(a) **(b)**

Figure 11. (a) Two logical components join into one physical component. (b) The enclosing boxes of three physically separated components come into contact with each other.

Many Chinese characters are composed of more than one component. But it is possible that two logical components of the same character actually join into one physical component on the images (Figure 11a). On the other hand, even though some of the components are physically separated, their enclosing boxes get in touch with each other and can be joined into a single box (Figure 11b). Due to all these reasons, it

becomes a fact that almost every legitimate textline on the images contains at least a single-box character.

To take advantage of this fact, we can pick out the component boxes whose width-to-height ratio is close to 1. To ensure that they are not arbitrary boxes, we can further check if there are any other boxes whose sizes are similar to theirs. We then restrict textlines to grow only from those boxes whose sizes are common to some others. If, after this stage, there are still component boxes uncovered by any textlines, we go ahead to search in their neighborhood for any other components that can be put together to form a box of larger size and better (closer to 1) width-to-height ratio. If so, we proceed the textline construction for the newly formed boxes.

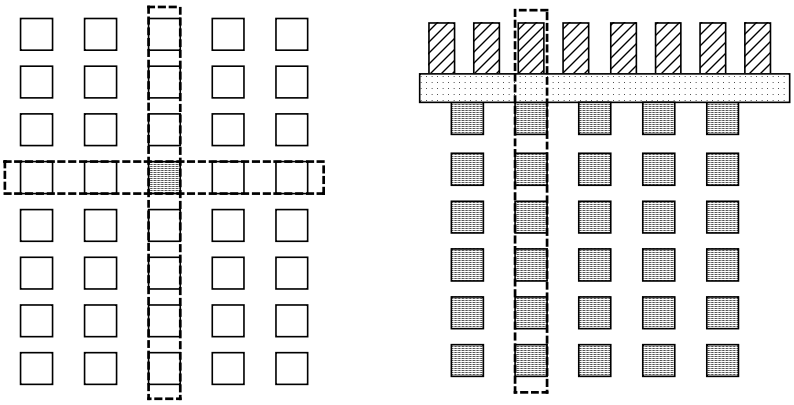
After the stage of textline construction, two more things need to be taken care. First, those textlines growing in the wrong orientation must be eliminated. Next, those textlines growing too far need to be pruned.

Eliminating textlines of incorrect orientations has the effect of adjusting the solutions in line with the spacing constraint. According to this constraint, the inner spacing[2] of a textline is smaller than its outer spacing. Thus, if two textlines grow out of the same component box, the one that has wider inner spacing than its counterpart ought to be eliminated (Figure 12a).

For the purpose of textlines pruning, we first find all the white rectangles on the document image. They serve as clues for possible segregation regions. So we want to examine those textlines that cross some white rectangles. However, rather than simply pruning them, we make a further comparison. For those textlines whose inner spacing is much smaller than the spacing created by the white rectangle (Figure 12b), we go

[2] The gaps within a textline may not be constant in size. So the inner spacing of a textline is defined to be the gap whose size is close to most of the gaps within the textline.

ahead to make the pruning.



(a) **(b)**

Figure 12. (a) Two textlines (enclosed by dashed lines) grow from the same (half-toned) component box. The horizontal line has wider inner spacing than the vertical line. (b) The vertical textline (enclosed by dashed lines) extends beyond a white rectangle (dotted box) with relatively wide spacing.

The pruning operation has the effect of forcing the solutions to comply with the spacing constraints for both textlines and paragraphs, since the combination of the two constraints implies that the inner spacing of a textline is smaller than the outer spacing of the paragraph.

After the stage of textline consolidation, we proceed to paragraph construction. This is rather a simple operation. It starts with any textline and proceeds to find, from any of the four possible directions, textlines of similar sizes. The same operation is then applied recursively to each of the textlines thus obtained.

Textlines and paragraphs are not simply end products of the process. They also serve as inputs to other adjustment procedure. Thus, after the stage of textline construction, those irregular boxes appearing within the same textline can be put together to form regular boxes. In Figure 13, for example, boxes A and B can merge into one character box, in reference to the alignment requirement for textlines. Boxes C and D in the same figure can also merge into one box, conforming to the requirement that

each textline is composed of similar-sized characters.

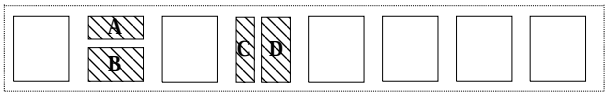
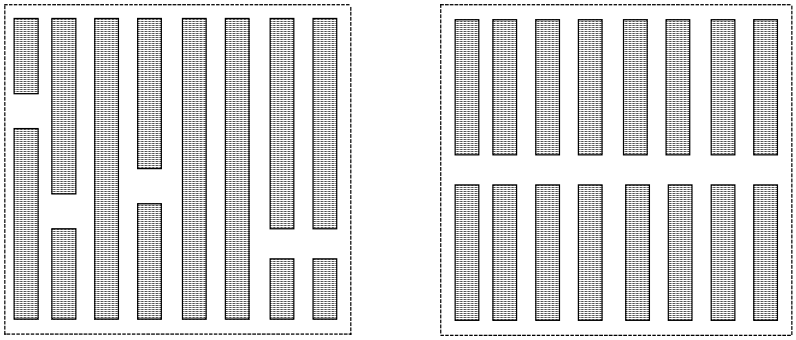


Figure 13. A horizontal textline (enclosed in dashed lines) incorporates irregular components A, B, C and D.

Similar adjustment can be made within the paragraphs. During the stage of textline construction, some textlines may not be fully connected, due to some conservative parameter values being used there. After the stage of paragraph formation, those properly aligned but disconnected pieces can join together to form a longer line (Figure 14a). However, there is a danger of joining textlines that really pertain to different columns. As a remedy, the recursive X-Y cuts can be used to undo the ill effect (Figure 14b).



(a) **(b)**

Figure 14. (a) Broken pieces falling within the same paragraph can be connected to full pieces. (b) Textlines belonging to different columns can be separated by recursive X-Y cuts.

The solution path suggested in this section thus evolves as a self-adjustment process. It starts to construct solutions complying with some easy constraints and then gradually improves them by taking reference of the better knowledge gained along the process. The goal of this process is to adjust the solutions to conform to more and more hard constraints.

The constraint conditions considered in this section can be met by many Chinese documents, but certainly not by all of them. Some special documents (business cards, for example) may require one or several of the conditions to remove or weaken, and a few others to add in. But one nice thing about this approach is that it can easily adjust the solution path according to the changes of constraint setting. The reason for the ease of adjustment is that constraint conditions can be naturally classified into categories (we categorized them into three sets). Thus, when one category of constraints remains unchanged, all the corresponding operations remain unchanged. The idea of modularity thus naturally grows into this solution method.

4. Retrieving Information from Machine-Transcribed Contents

There are thousands of distinct commonly used characters in the Chinese language. This means that for each of the characters to be recognized by a machine, it has to be matched against at least thousands of template images. Machine recognition is bound to make errors. Moreover, due to the time constraint that can be crucial to many applications, a trade-off between the time spent by the recognizer and the accuracy attained by it must be observed. For this reason, the thousands of characters can incur a significant loss to the recognition accuracy.

The usual measure for the accuracy counts the percentage of first-placed candidates that hit the targets. If more candidates are included in the count, however, the hitting rate can go near to 1. For example, the hitting rate achieved by the first five candidates generated from our recognizer can go way above 99% for most applications, even though the rate achieved by the first candidates varies from applications to applications. Since the machine-transcribed contents are used for searching purpose only, the first five candidates can be as useful as the first-placed candidates. The idea is given as follows.

In the data prepared for searching usage, we store the first-placed candidate generated by the recognizer for each of the characters extracted from original documents. Assuming users intend to look for a keyword string, say $K_1K_2...K_n$ out of the original documents. Our search engine searches for the information from the prepared data, instead. Since the data is likely to contain errors, the string $K_1K_2...K_n$ may appear as $V_1V_2...V_n$ on the prepared data, where V_i is either K_i or a character misidentified by the recognizer for K_i, for $i = 1,2,...,n.$ To ensure that users get what they want, the search engine should look for all the possible variants $V_1V_2...V_n$ out of the prepared data.

To do so, the search engine looks up a pre-stored table that keeps for each character entry those characters that can be misidentified for it. The misidentified characters for entry E are those that are expected to appear in the short-listed candidates for E. So they can be prepared in the following way. For each training sample of E, we collect the first N short-listed candidates generated by the recognizer. We then calculate the accumulated rate for the candidates and sort them accordingly to obtain the first M candidates, where M is a number close to N.

The success rate of the search engine is thus related to the hitting rate of the table that lists out all the misidentified characters. Normally, the hitting rate can reach above 99.5% when no more than 5 misidentified characters are stored for each entry.

Thus, if the keyword is a bi-gram (two-character string), the engine can to find all occurrences of the keyword from the original images, at a probability higher than 99%. If the keyword consists of more than two characters, we can relax a little bit by requiring the engine to find all the strings $U_1U_2...U_n$, where at least n-1 entries in $U_1U_2...U_n$ matches with the corresponding entries in $V_1V_2...V_n$. Then the engine can find all occurrences of the keyword from the original images, at a probability higher than 99.9%.

If textual contents are stored in the same order as they appear in the original documents, to search for a keyword and all its variants through them would require M×L×P many times of character comparison, where M is the number of misidentified characters, L is the length of keyword string, and P is size of the textual contents. The search time can be greatly reduced, if we use a pre-stored file to record for each character entry all the positions it occurs in the textual contents.

The way to use the pre-stored file is the following. When a search engine starts to find $V_1V_2...V_n$, a variant of the keyword string, we start to construct a tableau. The tableau is a matrix with n rows and P columns, where n is the length of the character string and P is the size of contextual contents. To start with, all the entries in the matrix are filled with 0's. Now, if the pre-stored file tells us that an occurrence of V_i is at the m-th position in the textual contents, then we fill 1 at position (i, m-i+1) in the matrix.

Thus, suppose that $V_1V_2...V_n$ actually appears in the text and its first character appear at the 37^{th} position in the text. Then we fill 1 at (1, 37) in the matrix. Its second character must then appear at the 38^{th} position, so we fill 1 at (2, 37) in the same matrix, etc. Finally, all the entries on column 37 in this matrix are filled with 1's.

Having done all the fillings, we start to examine all the columns in the matrix. If the keyword is a bi-gram, we pick up all the columns that have two 1's as their entries. If the keyword has more than two characters, then we pick up all the columns that have either all 1's or only one 0. At the end, we output the indices of all these columns. They indicate the starting positions of all the occurrences of $V_1V_2...V_n$ in the textual contents.

The method as described in the above is essentially a fast-matching method. The only difference is it has to ensure the matching of all possible variants of the given keyword. The computational cost is then M times higher than what would be required

for an exact matching (i.e., matching of the keyword only), where M is maximal number of misidentified characters listed for each character entry.

5. Summary

In previous sections, we describe the special problems that are encountered in building a system that supports information retrieval from Chinese document images. Two types of solutions are considered: capture of visual and textual contents from paper documents and retrieving information out of the prepared data. In the introductory section, we discuss how to put together all these solutions to obtain a system with the minimal demand for human interference. For the rest of the article, we focus on some special technical problems and their solutions.

Context sensitivity lies the heart of all these problems considered here. In the binarization problem, global threshold method paves the ground and sets the condition for window-based binarization. For the latter to work properly, another process is called forth to set the window sizes right. In the document layout problem, the solution path is not so straightforward any more. The grasp of contextual information is part of the solution being sought for. The solution scheme thus evolves as a self-improving process. While context sensitivity causes the two problems hard, it actually makes the task of information retrieval easy. Advantage can be taken from the fact that a keyword used for searching purpose normally consists of at least two characters. The inaccurate and piecemeal information contained in the machine-transcribed contents can synthesize into much more reliable and useful information, when used for the purpose of information retrieval.

REFERENCES

1. N. Otsu, A threshold selection method from gray-scaled histogram, *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 8, pp. 62-66, 1978.

2. G. Johannsen and J. Bille, A threshold selection method using information measures, *Proc. Sixth Intern. Conf. Pattern Recognition*, pp. 140-143, Munich, Germany, 1982.
3. J. M. White and G. D. Rohrer, "Image thresholding for optical character recognition and other applications requiring character image extraction," *IBM J. Res. Develop.* Vol. 27, No. 4, pp. 400-411, 1983.
4. W. Tsai, Moment-preserving thresholding: a new approach, *Computer Vision, Graphics, and Image Processing*, Vol. 29, pp. 377-393, 1985.
5. J. N. Kapur, P. K. Saboo, and A. K. C. Wong, A new method for gray-level picture thresholding using the entropy of the histogram, *Computer Vision, Graphics, and Image Processing*, Vol. 29, pp. 273-285, 1985.
6. J. Kittler and J. Illingworth, On threshold selection using clustering criteria, *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 15, pp. 652-655, 1985.
7. Y. Liu and S. N. Srihari, "Document image binarization based on texture features," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 19, pp. 540-544, 1997.
8. E. Giuliano, O. Paitra, and L. Stringa, Electronic character reading system, *U.S. Patent* 4,047,15, 1977.
9. W. L. Hwang and F. Chang, Character Extraction from Documents Using Wavelet Maxima, *Image and Vision Computing*, Vol. 16, pp. 307-315, 1998.
10. F. Chang, K. H. Liang, T. M. Tan, and W. L. Hwang, Binarization of document images using Hadamard multiresolution analysis, *Proceedings Intern. Confern. Document Analysis and Recognition*, Banglore, 1999.
11. G. Nagy, S. C. Seth, and S. D. Stoddard, Document analysis with an expert system, *Proceedings Pattern Recognition in Practice II*, Amsterdam, 1985.
12. H. S. Baird, S. E. Jones, and S. J. Fortune, Image segmentation by shape-directed covers, *Proceedings 10th ICPR*, Atlantic City, pp. 820-825, 1990.
13. F. M. Wahl, K. Y. Wong, and R. G. Casey, Block segmentation and text extraction in mixed text/image documents, *Comput. Vision Graphics Image Process*, vol. 20, pp. 375-390, 1982.
14. T. Pavlidis and J. Zhou, Page segmentation and classification, *CVGIP: graphical models and image processing*, vol. 54, pp. 484-496, 1992.
15. A. Dengel, Initial learning of document structure, *Proceedings Intern. Confern. Document Analysis and Recognition*, Tsukuba, 1993.
16. D. Wang and S. N. Srihari, Classification of newspaper image blocks using texture analysis, *Comput. Vision Graphics Image Process*, vol. 47, pp. 327-352, 1989.
17. A. K. Jain, B. Yu, Document representation and its application to page decomposition, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 20, pp. 294-

308, 1998.

18. L. O'Gorman and R. Kasturi eds., *Document Image Analysis,* Los Alamitos, IEEE CS Press, 1995.
19. H. Fujisawa and Y. Nakano, A top-down approach for the analysis of documents, *Proceedings 10th Intern. Conf. Pattern Recognition,* pp. 113-122, Atlantic City, 1990.
20. J. Fisher, S. Hinds, and K. D'Amato, A rule-based system for document image segmentation, *Proceedings 10th Intern. Conf. Pattern Recognition,* pp. 567-572, Atlantic City, 1990.
21. H. Baird, Anatomy of a versatile page reader, *Proceedings IEEE,* vol. 80, pp. 1059-1065, 1992.
22. F. Chang, et al., A Document Analysis and Recognition System, *Inter. Conf. Document Analysis and Recognition,* Ulm, Germany, 1997.