Document page segmentation based on pattern spread analysis

Phillip E. Mitchell
Hong Yan, MEMBER SPIE
University of Sydney
School of Electrical and Information Engineering
Sydney, NSW 2006, Australia
E-mail: mitchell@ee.usyd.edu.au

Abstract. This paper introduces an algorithm designed to segment black-and-white documents for the purpose of compression. A single document is segmented into two documents suitable for pattern-based and run-length-based compression. With some modification the same algorithm may also be used for optical character recognition. The segmentation is performed in two main steps: pattern extraction and classification. Patterns are extracted using a fast scan method that does not need to scan every pixel, and classification uses pattern characteristics such as spread and pattern context to segment the patterns. Documents may be segmented with an accuracy of at least 98%, depending on the content. Furthermore, text of any size and orientation may be successfully classified without the need for skew estimation or correction. This paper presents the segmentation algorithm and discusses the complete compression system. © 2000 Society of Photo-Optical Instrumentation Engineers. [S0091-3286(00)02003-1]

Subject terms: document page segmentation; pattern spread analysis; image compression; OCR; pattern classification.

Paper 990141 received Mar. 26, 1999; revised manuscript received Aug. 27, 1999; accepted for publication Aug. 30, 1999.

1 Introduction

With the ever-increasing use of computers and the Internet, electronic storage and transmission of information is a growing need. Much of this information is initially in the form of a physical document, and needs to be converted to digital information. Segmentation and compression algorithms are required to reduce the storage and transmission costs.

Bilevel documents may contain a wide variety of patterns, including text, lines, images, and noise. Furthermore, these patterns may vary in style and size. This makes such document images difficult to compress, because most compression algorithms are designed to be optimal for only one document style.

Pattern-matching-based compression algorithms are very powerful, but they perform best when the images contain text and it is regularly positioned.[1,2] The aim of this segmentation algorithm is to produce one image that is suitable for pattern-based compression. The remaining image will contain mostly large images and may be compressed with an algorithm based on run-length coding. Many segmentation algorithms have been published,[3–12] but not all are suitable as a first step for image compression. Our algorithm is ideal for text segmentation, as it easily handles irregular-shaped regions that may be very close to each other, or nested within each other. It is also orientation-independent, meaning that it is not necessary to employ skew detection and correction.[13]

Towards the end of the paper, a comparison is made with two other algorithms.[3,4] As this algorithm generally segments an image into text and nontext, it may also be suitable for optical character recognition (OCR).

1.1 *Overview of Algorithm*

The algorithm presented in this paper is outlined in Fig. 1. The first two steps explain how patterns are extracted from the document, which is the subject of Sec. 2, and the remaining steps explain the classification process, which is the subject of Sec. 3. The classes *type P* and *type I* loosely refer to text and nontext respectively: they are more formally defined in Sec. 3.

2 Pattern Extraction

The approach taken in this algorithm is to extract all the patterns from an image, and store them in a linked list. This enables the patterns to be quickly and easily traversed for classification. Other data structures and methods such as dynamic arrays,[4] attributed relational graphs (which are often used for recognition),[14–19] and spatial map grids[20] were considered, but a simple linked list was sufficient for the task. The main advantage in using arrays is easy random access, but for our purposes this was not required.

The first step in pattern extraction is to locate rectangular regions called *rects*. A rect may be thought of as a rectangular region of loosely connected black pixels. More specifically, a rect has at least one black pixel in every 9-pixel square. A rect is defined in this way so black regions may be found without scanning every pixel. The second step is to merge adjacent rects to form complete patterns. Section 2.1 explains the process used to locate the rects.

2.1 *Locating Rects*

The idea behind using rects to build up patterns is that rects are completely described by their corners: it is not neces-

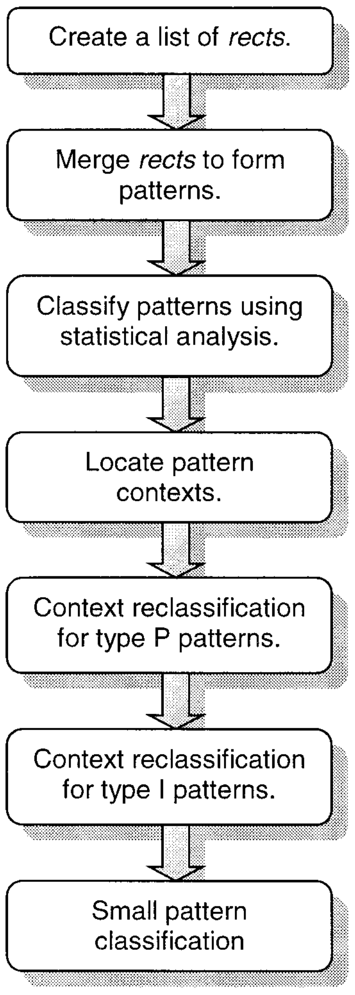

Create a list of *rects*.
Merge *rects* to form patterns.
Classify patterns using statistical analysis.
Locate pattern contexts.
Context reclassification for type P patterns.
Context reclassification for type I patterns.
Small pattern classification

Fig. 1 The segmentation and classification algorithm.

sary to describe boundaries. The search for rects begins at the top left corner of the image, and continues through to the bottom right corner, creating a list of rects as it goes. At the end of the procedure, every black pixel in the image is included in at least one rect. The process of extracting rects is described using pseudocode, in which the following definitions are used:

- (x,y) is the current pixel position with the origin in the top left corner of the image.
- w and h are the image width and height respectively.
- Im is the set of image points, i.e., $(x,y) \in Im$ if $0 \leqslant x < w$ and $0 \leqslant y < h$.

Then we have

```
(x,y)=(0,0);

while (x,y) ∈ Im
    if (xstep=search_right(x,y,xstep))
        Create new rect
        rect.left=x;
        rect.top=y;
        x=x+xstep;
        while (xstep=search_right(x,y,xstep))
            x=x+xstep;
        rect.right=x-1;
        ystep=lowest_row(rect);
        ry=y+ystep;
        while (ystep=search_below(rect,ry,ystep))
            ry=ry+ystep;
        rect.bottom=ry-1;
        Store new rect and mark pixels as taken
    else
        x+=3;
    if (x,y) ∉ Im
        (x,y)=(0,y+3);
```

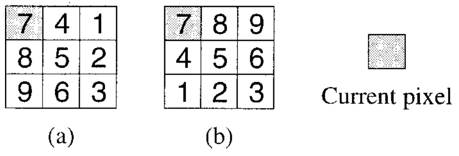


Fig. 2 Scan orders used for locating rects: (a) for *search_right*, (b) for *search_below*.

The functions *search_right*, *lowest_row*, and *search_below* are described below.

- *search_right( ):* Searches the current 9-pixel square [see Fig. 2(a)] for a black pixel. The search terminates as soon as a black pixel is found. The scan order is chosen so that the rightmost pixels are scanned first. This allows for fast scanning through black regions, as less pixels need to be scanned. If all the pixels are white, the function returns *false*, meaning that the right edge of the rect is found. Otherwise, the function returns *xstep*, which is related to the rightmost column that contains a black pixel. Specifically, if x is the current column and *rightx* is the column containing the rightmost black pixel, then $xstep = rightx - x + 1$. Here *xstep* has a value between 1 and 3, and is used to update x for the next search. Note: *xstep* is also used by the function to ensure that columns are not scanned twice. For example, if a call to *search_right* returns an *xstep* of 1, it means that the two rightmost columns in the 9-pixel square are white. The current column is then incremented by *xstep*, and *search_right* is called again. It is now only necessary to scan the rightmost column in the current 9-pixel square, because we know the leftmost two columns are white.
- *lowest_row( ):* This function returns a value corresponding to the lowest row, in the current rect, that contains a black pixel. The current rect only has a left, a right, and a top edge so far, so this function searches the top three rows, between the left and the right edge, for the lowest black pixel.
- *search_below( ):* Searches every 9-pixel square [see Fig. 2(b)] in the current row of the rect for a black pixel. If any 9-pixel square is all white, it returns *false*, meaning the bottom of the rect is found. Otherwise, the function returns *ystep*, which is related to the low-

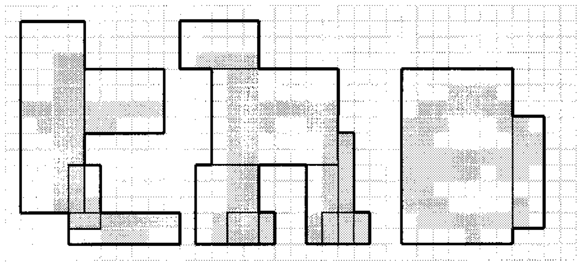


Fig. 3 An example of rects located in the pattern "the."

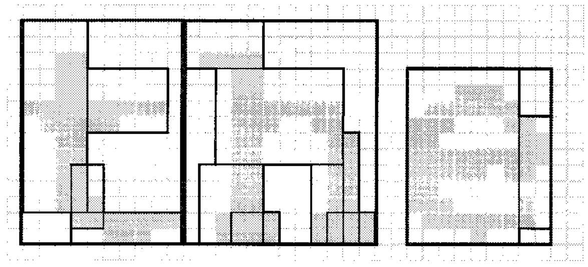


Fig. 4 Three patterns formed by merging adjacent rects.

est row, in the search range, that contains a black pixel. Specifically, if the current row is y and the lowest row with a black pixel is *lowy*, then $ystep = lowy - y + 1$. Here *ystep* has a value between 1 and 3, and is used to update y for the next search. Note: *ystep* is also used by the function to ensure that rows are not scanned twice. For example, if a call to *search_below* returns a *ystep* of 2, it means the row two rows below the current row has no black pixels. The current row is then incremented by *ystep*, and *search_below* is called again. Each time a 9-pixel square is checked now, it is only necessary to check the lower two rows, since it is known that the top (current) row is all white.

A feature of the above algorithm is that it locates black regions without needing to scan every pixel in the region. This allows groups of black pixels to be located quickly, while ensuring that unconnected components are merged only if they are very close to each other. An example of some rects located in an image is given in Fig. 3.

The algorithm, as explained above, implies that the entire document image must be in memory whilst locating rects. This is desirable, as it simplifies the process, but not necessary. If the document image is very large, or memory availability is low, then only a subset of the document image may be allowed in memory at a given time. Initially the top left corner region of the document will be put into memory; then, whenever the search needs to proceed beyond the boundary of the area currently in memory, an appropriate new area may be read from the file. Because the search for rects proceeds in an ordered fashion, the number of memory swaps required will be minimal.

2.2 *Grouping the Rects to Form Patterns*

In this step, patterns are created from the rects. The image is not referred to in this step, because all the information required is in the rects. A rect is merged with another rect if they are adjacent, that is, if the edges are within one pixel of each other. Using this definition, a list of patterns is built up from the list of rects. Each pattern is a group of adjacent rects, meaning that each rect in a group is adjacent to at least one other rect in the group. Figure 4 shows three patterns ("t," "h," and "e") that have been formed by merging the adjacent rects in Fig. 3. Each pattern stores all the rects that define it in a rect list, as well as the four edges that bound the pattern as shown in Fig. 5.

For text regions in a document, patterns are generally single characters or words, depending on the text style and size. For nontext regions, patterns may be all kinds of shapes and sizes. The rest of this paper deals with the problem of deciding which compression scheme is most suitable for each pattern.

3 Classification

The ultimate aim of this program is to segment a document image into two separate images: one suitable for pattern-based compression (P), the other suitable for image-type compression (I). To achieve this, patterns from the original document must be classified as type P or type I. Type P patterns should be small- to medium-sized text, while type I patterns should be large or irregularly shaped patterns. Large text containing a lot of black pixels should be classed as type I, as they will be compressed most efficiently with a run-length compression algorithm. Classification is performed in two steps: pattern classification and context classification. Each step is explained in detail below.

3.1 *Pattern Classification*

In this first step, patterns are classified individually based on size, black-pixel numbers, and run-length statistics. All patterns are initially assumed to be type P, so the classification rules are designed to locate type I patterns. A pattern is classified as type I immediately if any test is true, and type P if all tests are false. The four tests shown in Table 1 (classification rules A) are used to detect big, narrow, and small patterns. They are performed first, because aside from the small patterns, they are not subjected to context classification. The small patterns are classified first, because there are often many of them; later they are dealt with separately in the context classification step. The order of the remaining tests, shown in Table 2 (classification rules B), does not affect the results and is chosen mainly so that

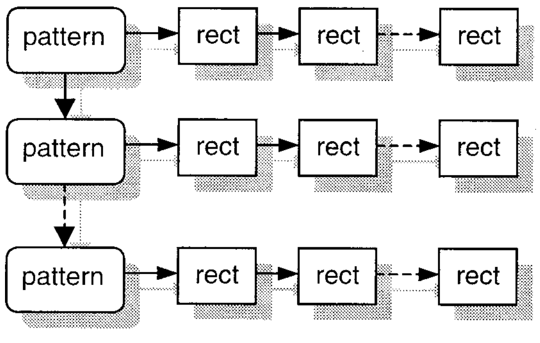


Fig. 5 The patterns are stored in a list. Each pattern contains a list with at least one rect that defines the pattern.

Table 1 Classification rules A.

Name	Classification rules	Explanation
Small patterns	$nblk < NBLK$ or $parea < PAREA\_S$	All small patterns are extracted, which can greatly reduce the number of patterns.
Big patterns	$parea > PAREA\_B$	This test locates large graphics, large text, and figure boxes.
Narrow patterns	$\max(w,h) > MAXWH$ and $\frac{\max(w,h)}{\min n(w,h)} > THIN$	This test locates long vertical or horizontal lines such as column divides.
Big rect area	$rarea > RAREA$	This test locates graphics that have a larger than normal black content.

the faster tests are performed first. The values of the constants used in the following tests are given at the end of this subsection.

The following definitions are used in classification rules:

w	Pattern width
h	Pattern height
parea	Pattern area$=(w-1)(h-1)$
rarea	Sum of the areas of the rects in the pattern
nblk	Number of black pixels in the pattern
nwhite	Number of white pixels in the pattern
maxbrl	Maximum length of black runs in the pattern
avbrl	Average black-run length in the pattern
spread	A measure of how spread out a pattern is: $(\#\text{black runs}) \times \min(w,h)^2/(\#\text{black pixels})$. This definition is explained in the following paragraphs.
skew	True if the pattern is skewed as defined by Fig. 6
sdbrl	Standard deviation of black runs.

A few comments need to be made about the classification rules in the tables. With the exception of the small patterns, classification rules A detect type I patterns that will not be considered in context classification. The reason is that these patterns are definitely best compressed as images. They are large or narrow patterns that are clearly not text.

On the other hand, classification rules B are less stringent; hence they are reconsidered in the context classification step. One of the more complex concepts is what we call *spread*. The definition of *spread* given above is designed so that its value increases when the patterns appear more spread out. Experimentation has shown that this is a very useful test for detecting unusual-shaped patterns that might be part of a line drawing or image, but that are similar in size to text. The other test of interest is the standard deviation of black-run lengths. In a way similar to the spread test, the variation of black-run lengths is usually higher for nontext patterns.

All the rules explained above are designed to locate type I (nontext) patterns regardless of orientation. The pattern extraction process allows patterns to be extracted individually, provided they are not too close, which means that the entire process is quite orientation-independent.

The values of the constants are as follows:

$$NBLK=16,$$

$$PAREA\_S=32,$$

Table 2 Classification rules B.

Name	Classification rules	Explanation
Large area ratio	$parea/rarea > PRRATIO$ and $parea > PAREA\_R$	Patterns with a large ratio (above 8.8) are most often graphs or text boxes.
Large black runs	$maxbrl > MAXBRL$ or $avbrl > AVBRL$	These patterns are usually graphics or large bold text.
Small black-to-white ratio	$\frac{nblk}{nwhite} > BWRATIO$	Similar to the large-area-ratio test, this test detects small graphs, boxes, and diagrams.
Large spread	$spread > SPRD1$ and $skew = False$	Detects spread-out diagrams and images. The skew test prevents skewed text being misclassified.
Large spread and SD of black runs	$spread > SPRD2$ and $sdbrl > SDBRL$	This test locates irregular-shaped diagrams and images

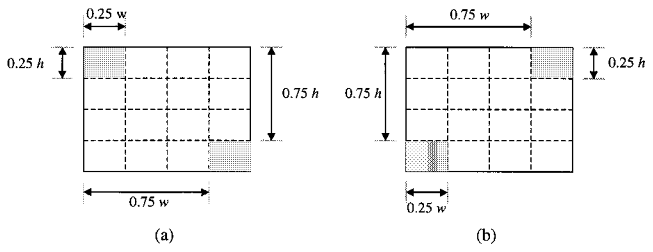


Fig. 6 A pattern is termed *skew* if no rects lie in the shaded regions of (a) or (b).

$PAREA\_B = 15{,}000,$

$MAXWH = 40,$

$THIN = 6.8,$

$RAREA = 6000,$

$PRRATIO = 8.8,$

$PAREA\_R = 100,$

$MAXBRL = 25,$

$AVBRL = 15.0,$

$BWRATIO = 0.15,$

$SPRD1 = 900,$

$SPRD2 = 300,$

$SDBRL = 5.5,$

3.2 *Context Classification*

Patterns usually occur in groups, so it is very effective to use neighboring patterns in the classification process. As an example, a text character in the middle of a text region that has been incorrectly classified as type I may be reclassified to type P, given the right conditions. These conditions relate to the number of nearby patterns and their class. Before classification can begin, each pattern's context, or pattern neighborhood, must be located.

A pattern is part of another pattern's context if it is within *GAP* pixels of it, where *GAP* is a constant with a value of 30. Formally, a pattern is part of another patterns context if

$$r_t > l_p - GAP \quad \text{and} \quad l_t < r_p + GAP \quad \text{and} \quad b_t > t_p - GAP \quad \text{and} \quad t_t < b_p + GAP,$$

where t, b, r, and l are the top, bottom, left, and right edges of the patterns respectively, and the subscripts t and p stand for pattern being tested and the pattern whose context is being found. A value of 30 for *GAP* was settled on after testing several images with different values. The classification is not very sensitive to changes in the value of *GAP:* it just needs to be large enough to encompass other patterns. Figure 7 illustrates pattern context and the meaning of *GAP*. Note that patterns are always considered to be a part of their own context, so their context is never empty.

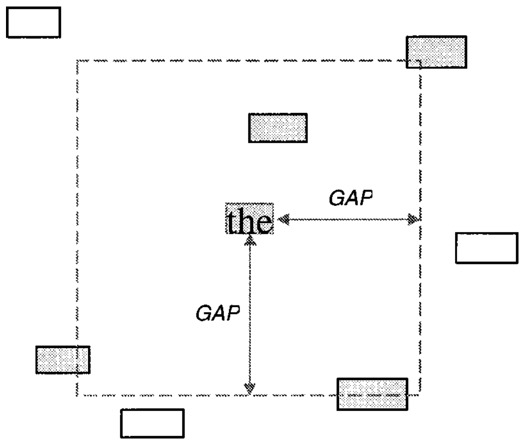


Fig. 7 Pattern context: the patterns within *GAP* of the center pattern (shaded) form its context.

Originally, all the patterns in the image were stored in a single list (see Fig. 5). However, after the pattern classification step, the patterns have been split up into three separate lists: *pats*, *bigpats*, and *smallpats*, as illustrated in Fig. 8. The *bigpats* list stores all the type I ''big patterns,'' *smallpats* stores all the ''small patterns,'' and *pats* stores the rest regardless of their current classification. This organization of pattern types simplifies the context classification step.

Pattern contexts are located for the patterns in *pats* before classification begins, by searching through the pattern list. This process can be performed very quickly because the list of patterns in *pats* is ordered first by row and then by column. Thus only a limited range of patterns needs to be searched for each pattern context. Once located, each pattern stores a list of pointers called *adj_pats* to all of its context patterns, as illustrated in Fig. 9.

There are three sets of rules for context classification. The first is designed for type P patterns, the second for type I patterns, and the third for small patterns. The tests for type P and type I patterns are performed twice, allowing reclassified patterns to be used in context. Classification is very fast, as the pattern contexts have been made directly available.

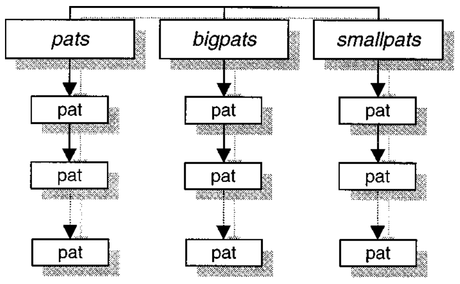


Fig. 8 Three pattern lists—*pats, bigpats*, and *smallpats*—store all the patterns after the pattern classification step.

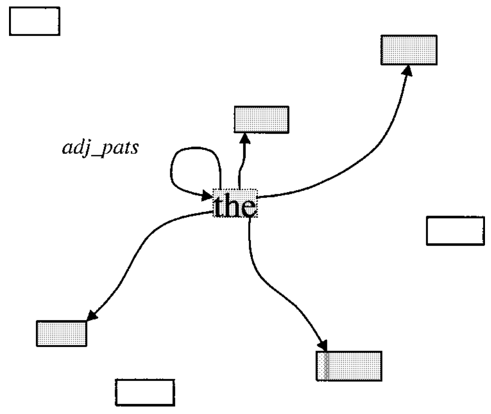


Fig. 9 A list called *adj_pats* stores all the patterns in a pattern's context.

Table 3 Context rules for classification from type P to type I.

Name	Rule
A. Standard deviation of context pattern area	$ni>3$ and $nc>4$ and $SD_{\mathrm{area}}>1.0$
B. Within a big pattern	Within a big pattern and [($ni>3$ and $nc>4$ and $ns>4$ and $nblk<45$) or ($np<5$ and $ni+ns>3$ and $nblk<80$)]
C. Within a small big pattern	Within a small big pattern and $np=1$ and $ni+ns>0$
D. More type I than type P	$np<5$ and $ni>0$ and $ni+ns>2\cdot np$

3.2.1 *Context rules for classification from type P to type I*

Several abbreviations are used in the following tests for simplicity, and are defined here:

np	Number of type P patterns patterns (from *pats*) in context
ni	Number of type I patterns (from *pats*) in context
nc	Total number of patterns (from *pats*) in context
ns	Number of small patterns (from *smallpats*) in context
$nblk$	Number of black pixels in current test pattern.

The tests are summarized in Table 3.

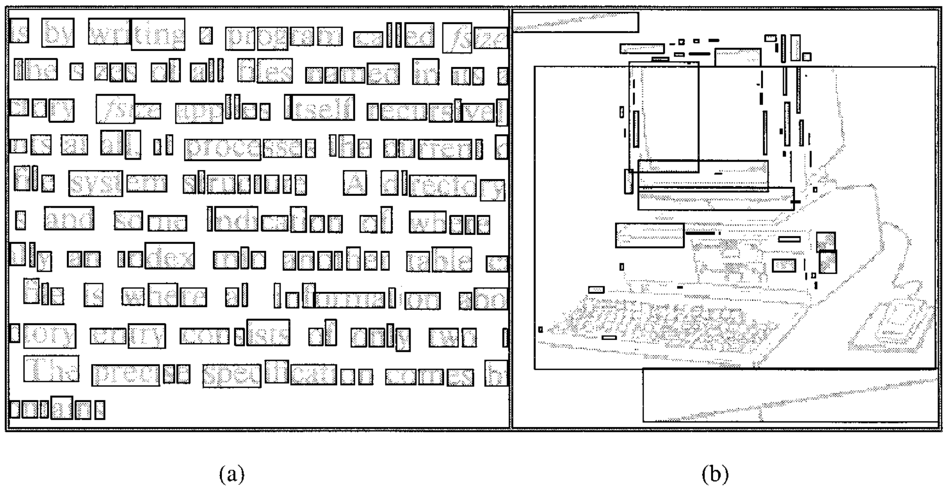
A. Standard deviation of context pattern area. SD_{area} is the normalized standard deviation of the area of the context patterns. It is derived from the variance of the *nc* context patterns, v_{nc}, which is defined recursively as

$$v_n=\frac{1}{n}[(n-1)(v_{n-1}+\overline{a}_{n-1}^2)+a_n^2]-\overline{a}_n^2,$$

where n is the pattern number, a_n is the area of pattern n, and $\overline{a_n}$ is the average area of the first n patterns. The average area is also calculated recursively as $\overline{a_n}=(1/n)[(n-1)\overline{a}_{n-1}+a_n]$. The normalized standard deviation is defined as $SD_{\mathrm{area}}=v_{nc}^{1/2}/\overline{a}_{nc}$.

This test relies on the fact that patterns in a text region do not vary greatly in area. If there are a significant number of context patterns (*nc*) with many being type I (*ni*), and the standard deviation of area (SD_{area}) is large, then the test pattern is most likely type I.

Figures 10 and 11 illustrate how the variation in pattern area changes with different pattern types. The patterns in Fig. 10(a) have a low area variation, indicating type P patterns, while the patterns in Fig. 10(b) vary greatly in area, indicating type I patterns. Figure 11 is a graph of all the pattern areas from a single document. It is clear from the plot that the type P patterns are spread over a much nar-



(a) (b)

Fig. 10 The variation in pattern area may be used to deduce the region type. The patterns in (a) vary slightly in area, indicating a type P region, while the patterns in (b) vary greatly, indicating a type I region.

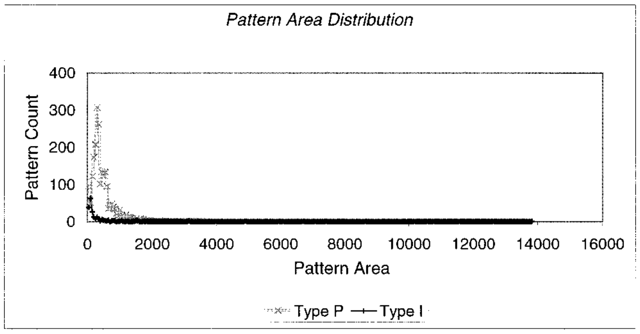


Fig. 11 This plot illustrates the spread of pattern areas for different pattern types.

rower range of areas than the type I patterns. This explains why type P pattern regions generally have a low area variance.

B. Within a big pattern. This test checks if the pattern is surrounded by a big pattern. Although text may appear within a big pattern (for example, inside a text box or page border), it is a useful test for graphics. The test ensures that the pattern is relatively small (*nblk*) and that there are at least as many type I (or small) patterns as there are type P patterns. This test is very useful for finding small graphic patterns that could be easily mistaken for text.

C. Within a small big pattern. This obscure-sounding test is useful for locating graphics. A small big pattern is a big pattern (in *bigpats*) that has an area less than 50,000. Such a pattern may be a table, figure, or graphic, but if it is also not near any type P patterns, the test pattern is unlikely to be type P.

D. More type I than type P. This test simply counts the number of type I and small patterns, and checks if there are more than double that number of type P patterns.

3.2.2 *Context rules for classification from type I to type P*

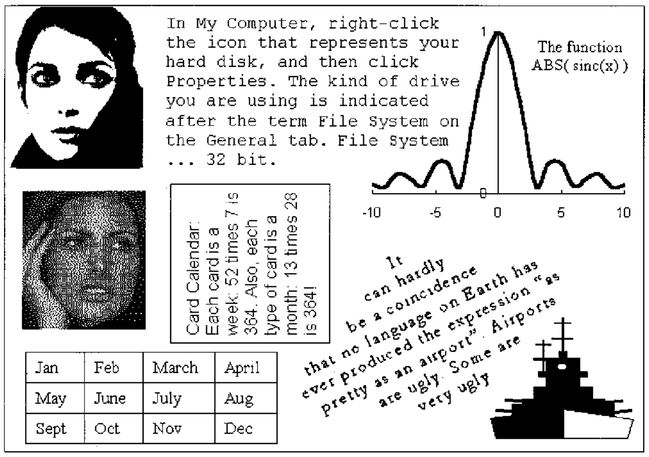
This section explains the rules (Table 4) used to test type I patterns for reclassification to type P. The abbreviations defined in Sec. 3.2.1 are again used here.

A. Standard deviation of context pattern area. As explained in Sec. 3.2.1 A, the standard deviation of the context pattern area is a useful test. Here, if the variation in area is low, the pattern is most likely type P.

B. More type P than type I. This test compares the numbers of the various types of pattern in the context. If there are more type P patterns than type I, the pattern is reclassified to type P.

Table 4 Context rules for classification from type I to type P.

Name	Rule
A. Standard deviation of context pattern area	$np>2$ and $ni=1$ and $SD_{\text{area}}<1.1$
B. More type P than type I	$np>2$ and $ni<3$ and ($nc>4$ and $np>2\cdot ni$)


In My Computer, right-click the icon that represents your hard disk, and then click Properties. The kind of drive you are using is indicated after the term File System on the General tab. File System ... 32 bit.

The function ABS(sinc(x))
1
0
-10
-5
0
5
10

Card Calendar: Each card is a week: 52 times 7 is 364. Also, each type of card is a month: 13 times 28 is 364!

It can hardly be a coincidence that no language on Earth has ever produced the expression "as pretty as an airport". Airports are ugly. Some are very ugly

Jan	Feb	March	April
May	June	July	Aug
Sept	Oct	Nov	Dec

Fig. 12 A test document containing a wide range of text and graphics styles.

3.2.3 *Small-pattern classification*

The only step that now remains is small-pattern context classification. Small patterns are very often punctuation marks, but may include other patterns such as scan noise, figures, or diagrams. These patterns are all suitable for type P compression, provided that they are close to other type P patterns. Pattern-based compression is most efficient when the many patterns are similar, and when they are close to each other. Therefore, any small pattern that lies in a type P pattern context is classed type P, the rest remaining type I. This system of classification would also work well for OCR.

4 Results

In this section, the results of our segmentation algorithm are presented and compared with two other algorithms: bounding-box projection[3] and description of the background.[4] It will be seen that our algorithm operates on a larger range of document styles than the other methods. To illustrate this point, a test document was created that contains text and graphics with a range of styles and orientations. It is shown in Fig. 12.

This test document contains three different styles of images. The face in the top left was created by thresholding a gray-scale image, the face below that was created by dithering a gray-scale image, and the battleship graphic in the bottom right is a clip-art image. There are three orientations of text, some appearing inside text boxes or tables, and lastly there is a graph with labels along the axis. A brief description of the two other algorithms is presented, followed by the results.

A. The bounding-box projection technique: This technique segments documents into text blocks, by first locating connected component bounding boxes. Text lines are found by using horizontal and vertical histograms of the bounding boxes to locate white space between lines. Words are then

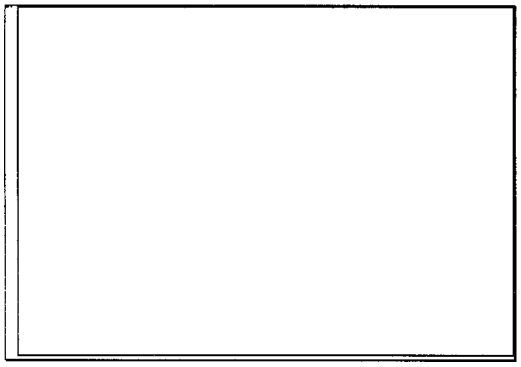


Fig. 13 Segmentation of the test document (Fig. 12) by the bounding-box projection technique.

located within the text lines and grouping the text lines makes text blocks. The technique is quite successful provided that:

1. The document is skew-corrected, which implicitly restricts the document to contain only one orientation of text.
2. The document does not contain irregular-shaped graphics or varying column layout. This restriction is not specified, but the algorithm fails in such cases.

Otherwise, the algorithm is quite fast, since it does not refer to the actual image once the connected component bounding boxes are created.

B. The description-of-the-background technique: This algorithm builds up a description of the white background of a document image to segment the information regions. The first step is to determine the average baseline distance (the vertical distance from one line of text to the next). Taking sample histograms and locating maxima and minima does this. The next step is *vertical smearing*, which involves blackening all the pixels between two black pixels if they are (vertically) within the smearing value. The smearing value is defined as 2/3 of the baseline distance. The smearing attempts to merge text in the same paragraph, while keeping different paragraphs and images separate. The algorithm then creates white tiles, which tightly cover the white regions in the image. Without going into further details, the white tiles are then used to create a description of boundaries of the black smeared content, which are effectively holes in the tiles. This algorithm can successfully segment a document with quite complicated regions. Specifically, it can segment nonrectangular and skewed regions within a document. However, the regions must be sufficiently separated; otherwise they will be merged. Clearly, if two different regions are separated (vertically) by less than the smearing value, they will be merged into one region. This makes this algorithm unsuitable for the purposes of compression if the document contains nontext regions close to text regions.

As might be expected, the bounding-box technique performed very badly on the test image. The irregular layout of the document made the segmentation task impossible with this technique. Figure 13 shows that only one block

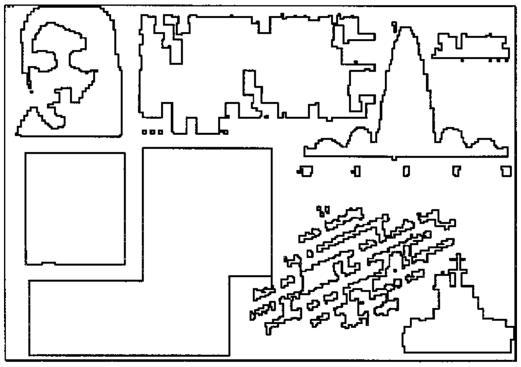


Fig. 14 Segmentation of the test document (Fig. 12) by the description-of-the-background technique.

was segmented from the image: the block containing the entire image content.

The description-of-the-background method performed much better, as seen in Fig. 14. The lines in the image illustrate the boundary of the segmented regions. The three images, the graph, and most of the text were successfully segmented. However, the text box, the table, and part of the skewed text have been merged together. Furthermore, the text inside the table and text box has not been segmented at all. Reapplication of the algorithm may be able to segment nested regions, but this is not very convenient in practice.

Unlike the other algorithms, ours can quite successfully segment the test document. Figure 15 shows the type I patterns, while Fig. 16 shows the type P patterns. Our method is successful because it assumes very little about the document style, and because it can segment regions that are very close. Components are treated as separate as long as they are 3 or more pixels apart. At a resolution of 300 dots/in., this is equivalent to about 1/4 mm.

Another image is now presented (Fig. 17) to illustrate our algorithm's ability to segment documents with very irregular-shaped regions. Many algorithms have trouble segmenting such documents, for two main reasons:

1. The different regions (text and graphics) are irregular in shape.
2. The different regions are very close.

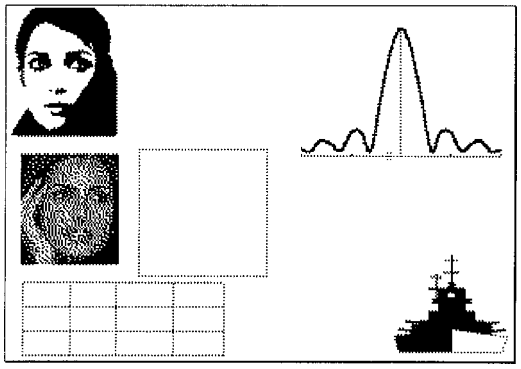
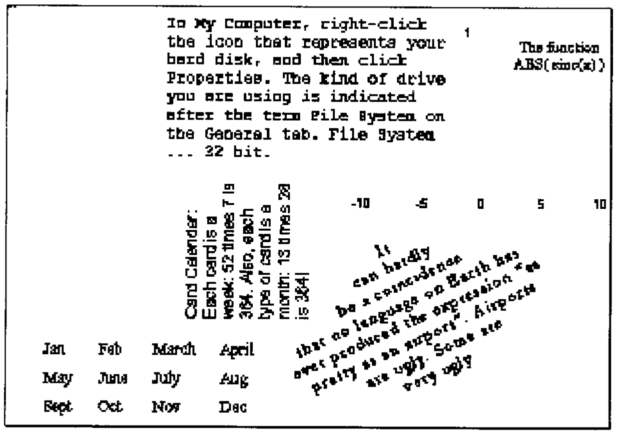


Fig. 15 The type I patterns (images) extracted from the test image (Fig. 12) using our method.



In My Computer, right-click
the icon that represents your
hard disk, and then click
Properties. The kind of drive
you are using is indicated
after the term File System on
the General tab. File System
--- 32 bit.

1

The function
ABS(sinc(x))

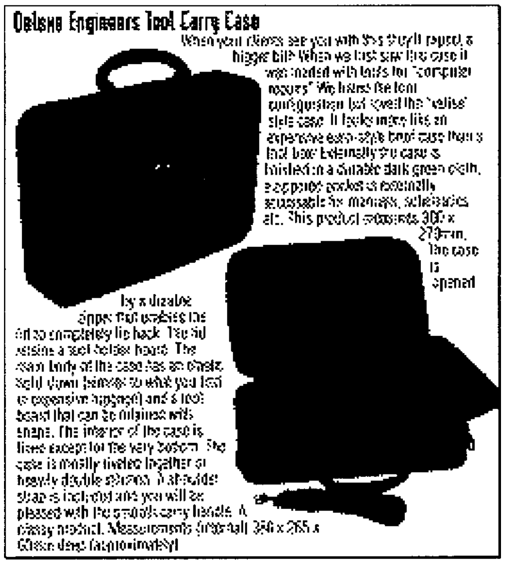
-10 -5 0 5 10

Card Calendar:
Each card is a
week: 52 times 7 is
364. Also, each
type of card is a
month: 13 times 28
is 364!

It
can hardly
be a coincidence
that no language on Earth has
ever produced the expression "as
pretty as an airport". Airports
are ugly. Some are
very ugly

Jan	Feb	March	April
May	June	July	Aug
Sept	Oct	Nov	Dec

Fig. 16 The type P patterns (text) extracted from the test image (Fig. 12) using our method.



Deluxe Engineers Tool Carry Case
When your clients see you with this they'll expect a
bigger bill! When we first saw this case it
was loaded with tools for "computer
repairs". We hated the tool
configuration but loved the "deluxe"
style case. It looks more like an
expensive euro-style brief case than a
tool box! Externally the case is
finished in a durable dark green cloth,
a zippered pocket is externally
accessible for manuals, schematics
etc. This product measures 300 x
275mm.
The case
is
opened
by a durable
zipper that enables the
lid to completely lie back. The lid
retains a tool holder board. The
main body of the case has an elastic
hold-down (similar to what you find
in expensive luggage) and a tool
board that can be retained with
snaps. The interior of the case is
lined except for the very bottom. The
case is mostly riveted together or
heavily double stitched. A shoulder
strap is included and you will be
pleased with the smooth carry handle. A
classy product. Measurements (overall) 350 x 265 x
60mm deep (approximately)

Fig. 17 A document image with text wrapped tightly around graphics.

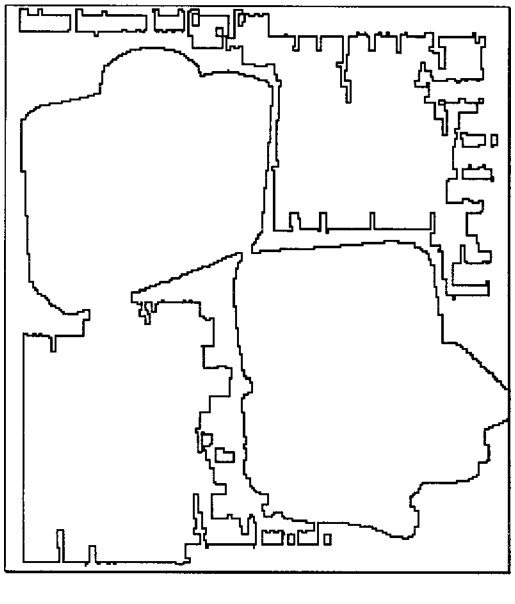


Fig. 18 Segmentation of the document shown in Fig. 17 using the description-of-the-background method.

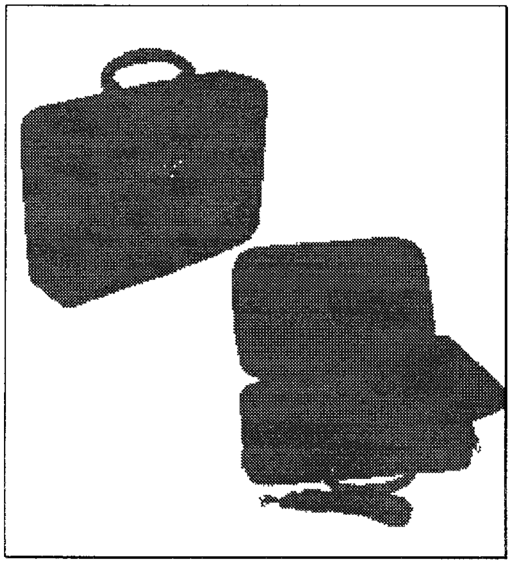
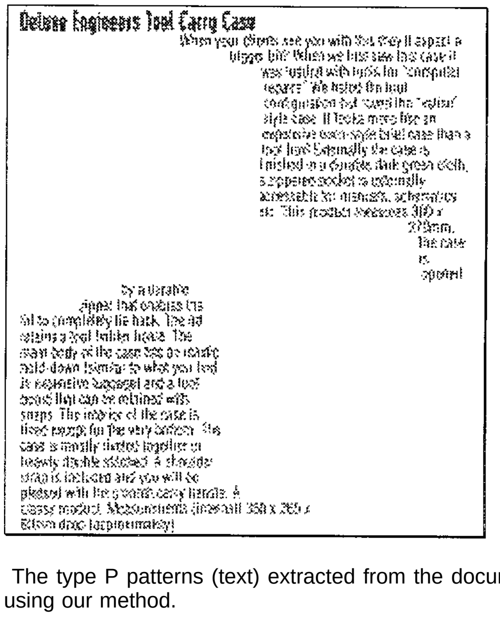


Fig. 19 The type I patterns (images) extracted from the document in Fig. 17 using our method.

The bounding-box technique again fails with this document, not only because there are images present, but also because the space between lines is small. This means that the bounding-box method cannot locate text lines, because there is no clear line of pixels separating them. The result for this method is omitted, as it provides no information.

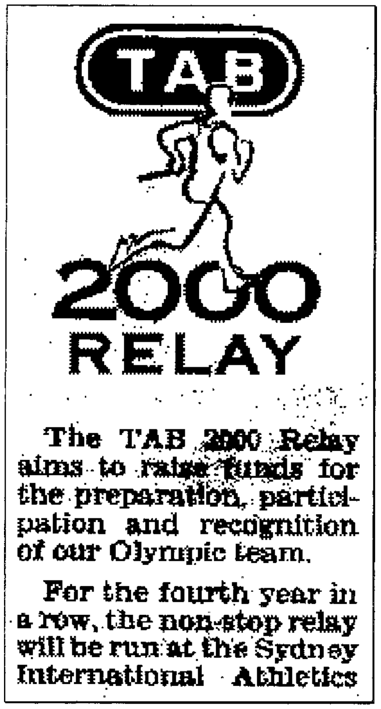
The description-of-the-background method produces better results (Fig. 18), but it fails to separate the regions completely. The text on the upper right is partially fragmented, while the text on lower left remains attached to one of the images. Altering the baseline distance manually varies the results, but the text becomes very fragmented before the baseline distance is low enough to correctly separate the image from the text.

The results of our method (shown in Figs. 19 and 20)



Deluxe Engineers Tool Carry Case
When your clients see you with this they'll expect a
bigger bill! When we first saw this case it
was loaded with tools for "computer
repairs". We hated the tool
configuration but loved the "deluxe"
style case. It looks more like an
expensive euro-style brief case than a
tool box! Externally the case is
finished in a durable dark green cloth,
a zippered pocket is externally
accessible for manuals, schematics
etc. This product measures 300 x
275mm.
The case
is
opened
by a durable
zipper that enables the
lid to completely lie back. The lid
retains a tool holder board. The
main body of the case has an elastic
hold-down (similar to what you find
in expensive luggage) and a tool
board that can be retained with
snaps. The interior of the case is
lined except for the very bottom. The
case is mostly riveted together or
heavily double stitched. A shoulder
strap is included and you will be
pleased with the smooth carry handle. A
classy product. Measurements (overall) 350 x 265 x
60mm deep (approximately)

Fig. 20 The type P patterns (text) extracted from the document in Fig. 17 using our method.



TAB

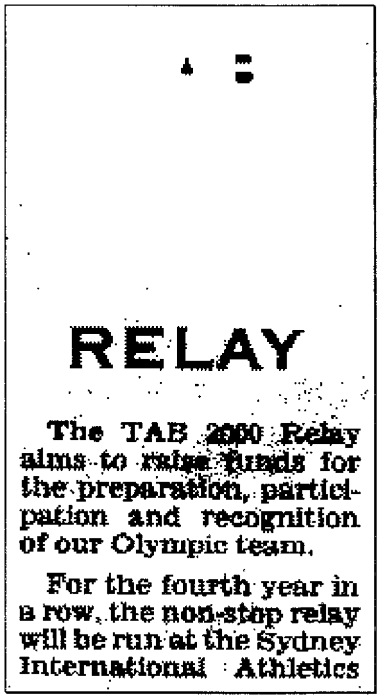
2000

RELAY

The TAB 2000 Relay aims to raise funds for the preparation, participation and recognition of our Olympic team.

For the fourth year in a row, the non-stop relay will be run at the Sydney International Athletics

Fig. 21 A noisy document image.



RELAY

The TAB 2000 Relay aims to raise funds for the preparation, participation and recognition of our Olympic team.

For the fourth year in a row, the non-stop relay will be run at the Sydney International Athletics

Fig. 23 Type P (text) part of Fig. 21.

show that our method can operate successfully on documents with text wrapping closely around irregular-shaped images.

One further example (Fig. 21) illustrates our method's ability to correctly segment and classify images in the presence of noise. Figures 22 and 23 show the results of segmentation using our method. Excessive noise, particularly when spread throughout the document, does reduce classification accuracy, but the algorithm is not overly sensitive to it. The reason is that small patterns (which are often noise) are segmented out and do not play a significant role in classification.

The algorithm presented in this paper was tested on a large range of documents with good results. Table 5 illustrates the classification accuracy for various document types. This accuracy is defined as the proportion of patterns that were correctly classified. At least 98% of patterns are correctly classified on average for any given document.

Incorrect classification is largely due to separate patterns being joined either by scan noise or low resolution. Judging whether a pattern has been correctly classified is not straightforward either: small patterns that look like characters, but form part of a graphic, can equally well be said to be type P or type I. The results are very good, considering that type P patterns are most important for efficient compression.

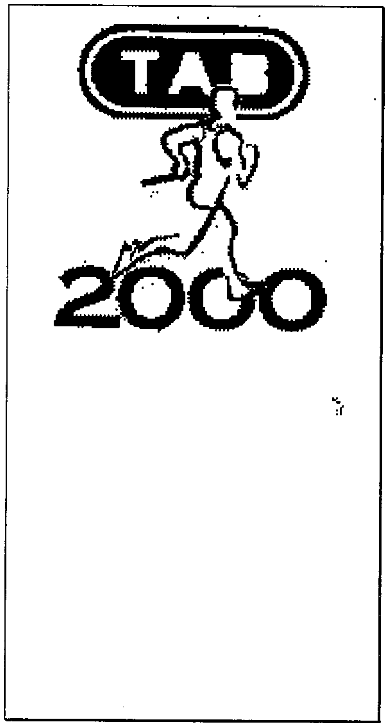


Fig. 22 Type I (image) part of Fig. 21.

Table 5 Classification accuracy for various document styles.

	Proportion correct (%)		
Document style	Type P	Type I	Total
Text; no graphics	100	—	100
Skewed text and graphics	98	94	98
Graphics	98	95	98
Total	99	95	99

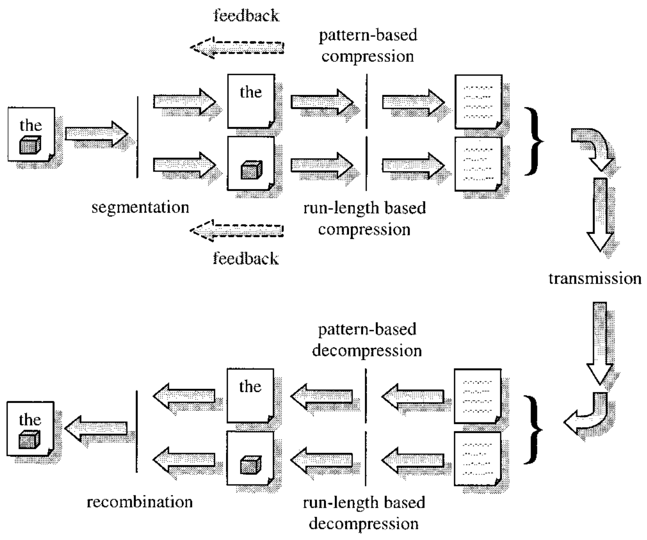

feedback
pattern-based compression
the
the
segmentation
run-length based compression
feedback
transmission
pattern-based decompression
the
the
recombination
run-length based decompression

Fig. 24 The complete segmentation and compression system. The dashed feedback arrows indicate the possibility of using compression information to assist the segmentation process to achieve greater efficiency.

5 Conclusion

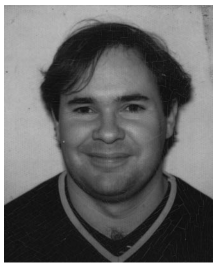
The results above show that the segmentation algorithm presented in this paper is very successful for any black-and-white document. Text of any style and orientation may be correctly classified without the need for skew estimation or correction. With some modification, this algorithm may be used for OCR, but its main purpose is for compression. Future work involves integrating the segmentation algorithm with compression algorithms to create a complete system (Fig. 24). Compression may be improved further by using information from the compression process to assist classification.

Acknowledgment

The authors would like to thank the Australian Research Council and Motorola Australia Pty Ltd, which have provided sponsorship for this project.

References

1. O. Johnsen, J. Segen, and G. L. Cash, ''Coding of two-level pictures by pattern matching and substitution,'' *Bell Syst. Tech. J.* **62**(8), 2513 (1983).
2. M. Rabbani and P. W. Jones, *Digital Image Compression Techniques*, SPIE Optical Engineering Press, Bellingham, WA (1991).
3. J. Ha and R. M. Haralick, ''Document page decomposition by the bounding-box projection technique,'' in *Proc. 3rd Int. Conf. on Document Analysis and Recognition (ICDAR)*, Montreal, pp. 1119–1122, IEEE Computer Society Press (1995).
4. A. Antonacopoulos, ''Page segmentation using the description of the background,'' *Comput. Vis. Image Underst.* **70**(3), 350–369 (1998).
5. C. L. Tan and P. O. Ng, ''Text extraction using Pyramid,'' *Pattern Recogn.* **31**(1), 63 (1998).
6. D. X. Le, G. R. Thoma, and H. Wechsler, ''Automated borders detection and adaptive segmentation for binary document images,'' in *Proc. 13th Int. Conf. on Pattern Recognition (ICPR)*, Vol III, Track C, pp. 737–741, IEEE Computer Society Press (1996).
7. K. Kise, O. Yanagida, and S. Takamatsu, ''Page segmentation based on thinning of background,'' in *Proc. 13th Int. Conf. on Pattern Recognition (ICPR)*, Vol III, Track C, pp. 788–792, IEEE Computer Society Press (1996).
8. D. Drivas and A. Amin, ''Page segmentation and classification utilising bottom-up approach,'' in *Proc. 3rd Int. Conf. on Document Analysis and Recognition (ICDAR)*, Montreal, pp. 610–614, IEEE Computer Society Press (1995).
9. L. O'Gorman, ''The document spectrum for page layout analysis,'' *IEEE Trans. Pattern. Anal. Mach. Intell.* **15**, 1162–1173 (1993).
10. I. H.Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes*, Van Nostrand Reinhold, New York (1994).
11. K. C. C Chan, X. Huang, and P. Bao, ''Fuzzy segmentation for document image analysis,'' in *Proc. 1997 IEEE Int. Conf. on Systems Man and Cybernetics*, pp. 977–982 (1997).
12. H. Bley, ''Segmentation and preprocessing of electrical schematics using picture graphs,'' *Comput. Vis. Graph. Image Process.* **28**, 271–288 (1984).
13. S. C. Hinds, J. L. Fisher, and D. P. D'Amato, ''A document skew detection method using run-length encoding and the Hough transform,'' in *Proc. 10th Int. Conf. on Pattern Recognition (ICPR)*, pp. 464–468, IEEE CS Press, Los Alamitos, CA (1990).
14. K. P. Chan and Y. S. Cheung, ''Fuzzy-attribute graph with application to Chinese recognition,'' *IEEE Trans. Syst. Man Cybern.* **22**(1), 153–160 (1992).
15. L. G. Shapiro and R. M. Haralick, ''Structural descriptions and inexact matching,'' *IEEE Trans. Pattern. Anal. Mach. Intell.* **3**, 504–519 (1981).
16. A. K. C. Wong and M. You, ''Entropy and distance of random graphs with application to structural pattern recognition,'' *IEEE Trans. Pattern. Anal. Mach. Intell.* **PAMI-7**(5), 599–609 (1985).
17. W.-H. Tsai and K.-S. Fu, ''Error-correcting isomorphisms of attributed relational graphs for pattern analysis,'' *IEEE Trans. Syst. Man Cybern.* **SMC-9**(12), 757–768 (1979).
18. A. Sanfeliu and K.-S. Fu, ''A distance measure between attributed relational graphs for pattern recognition,'' *IEEE Trans. Syst. Man Cybern.* **SMC-13**(3), 353–362 (1983).
19. J. Liu, W. K. Cham, and M. M. Y. Chang, ''Online Chinese character recognition using attributed relational graph matching,'' *IEE Proc. Vision Image Signal Process*. **143**(2), 125–131 (1996).
20. J. Lii and S. N. Srihari, ''Location of name and address on fax cover pages,'' in *Proc. 3rd Int. Conf. on Document Analysis and Recognition (ICDAR)*, Montreal, pp. 756–759, IEEE Computer Society Press (1995).


Phillip E. Mitchell received his BSc degree in mathematics and physics in 1994 and his BE degree in electrical engineering in 1996, both from the University of Sydney. From 1996 to 1997 he worked on pattern recognition using neural networks for the Department of Psychology, and he is currently a PhD student in the Department of Electrical and Information Engineering, University of Sydney. His research interests include image compression, document analysis, and pattern recognition.

Hong Yan received his BE degree from Nanking Institute of Posts and Telecommunications in 1982, MSE degree from the University of Michigan in 1984, and PhD degree from Yale University in 1989, all in electrical engineering. In 1982 and 1983 he worked on signal detection and estimation as a graduate student and research assistant at Tsinghua University. From 1986 to 1989 he was a research scientist at General Network Corporation, New Haven, Connecticut, where he worked on a CAD system for optimizing telecommunication systems. Since 1989 he has been with the University of Sydney, where he is currently a professor of electrical and information engineering. His research interests include computer animation, signal and image processing, pattern recognition, neural and fuzzy algorithms, and quantum computers. He is an author or coauthor of one book and over 200 technical papers in these areas. Dr. Yan is a fellow of the Institution of Engineers, Australia (IEAust), a senior member of the IEEE, and a member of the SPIE, the International Neural Network Society, the Pattern Recognition Society, and the International Society for Magnetic Resonance in Medicine.