Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Image and Vision Computing 22 (2004) 307–317

www.elsevier.com/locate/imavis

Newspaper layout analysis incorporating connected component separation

Phillip E. Mitchell[a,b,*], Hong Yan[a,b]

[a]Department of Computer Engineering and Information Technology, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong, China
[b]School of Electrical and Information Engineering, University of Sydney, Sydney, NSW 2006, Australia

Received 9 April 2003; received in revised form 6 November 2003; accepted 13 November 2003

Abstract

This paper presents an algorithm that performs automated segmentation and classification of newspaper images. A notable feature of the algorithm is a technique for segmenting components that are connected to other components. In particular, horizontal lines and vertical lines, which can be vital in determining the page layout, can be segmented from other lines and other components. The algorithm uses a bottom-up approach to initially segment the image, classify patterns and extract text lines. The classified patterns are then merged into complete regions. The algorithm is tested on a set of complex English and Greek newspaper images dating back to 1900.

Keywords: Document analysis; Image segmentation; Newspaper segmentation

1. Introduction

The process of converting physical newspaper pages into digital format is vital for their preservation and archival. Newspapers provide a great challenge for document image analysis because of the huge variety of layouts and qualities. Lines may not be physically connected, perpendicular to the page edge, or even straight. The space between columns may be smaller than some inter-word spaces, and possibly worst of all, separate component may actually be connected. This paper presents a complete algorithm for newspaper segmentation and classification, which includes methods for segmenting connected line components.

There are many publications that address document analysis. The segmentation methods may be broadly divided into bottom-up [1–7] and top-down [8,9] methods. Bottom-up approaches begin with low-level components, usually connected components, and the problem is then to correctly merge and classify them as document regions. One can use projection techniques to locate lines, and then group these regions into paragraphs [4]. This method, however, is highly sensitive to skew, and will not operate on complex documents such as newspapers. Other methods use the run length smoothing algorithm (RLSA) before extracting connected components [5,6]. Various rules and distance measures are then employed to merge the components. RLSAs smooth the document horizontally and vertically in an attempt to quickly locate text components lines or blocks. Larger thresholds may be used to identify text blocks, so this method is also used in top-down approaches.

Top-down approaches begin with the entire document, and the problem is then to correctly divide it into document regions. A clear problem with the RLSA method described above is being able to segment separate regions that lie very close to each other. Split and Merge methods [8,9] attempt to segment documents by splitting the document along selected lines and then merging the segmented regions. The algorithm used by Hadjar et al. [8] is reviewed below.

Alternatively, the algorithms may be defined as background approaches [10,11]. A description of the background can provide good segmentation for complex document layouts, though narrow (or zero) gaps between separate components will prevent proper segmentation. Other techniques include algorithms such as Delaney Triangulation [12] to perform text extraction. Nagy [13] and Tang et al. [14] provide comprehensive summaries of document analysis and processing.

We will now present a brief description of the algorithms submitted to the First International Newspaper Segmentation Contest. Liu et al. [5] employ a component based

* Corresponding author. Address: School of Electrical and Information Engineering, University of Sydney, Sydney, NSW 2006, Australia. Tel.: +61-2-9351-5339; fax: +61-2-9351-3847.
E-mail address: mitchell@ee.usyd.edu.au (P.E. Mitchell).

0262-8856/$ - see front matter © 2003 Published by Elsevier B.V.
doi:10.1016/j.imavis.2003.11.001

bottom-up algorithm. Lines are detected first by locating line seeds in connected components, which are grown by searching in the appropriate horizontal or vertical direction. RLSA is then applied to the image, connected components are extracted again and then attribute setting is performed. The main attributes described in the paper are font size, determined directly from the component size, and stroke width adopting the algorithm from Tseng and Lee [15]. Components are then merged using the distance measure: $D(i,j) = \alpha_{ij}\beta_{ij}d_{ij}$, where α and β are coefficients that increase when the two components i and j differ in font size and stroke width, respectively, and d_{ij} is the geometric distance between i and j. This algorithm uses detected lines well to separate regions, but the results show that this method has a very high rate of misses for text and titles.

Hadjar et al. [8] employ a split and merge algorithm, which splits the entire image along detected lines, and then attempts to merge these small zones into larger zones. The authors admit the algorithm relies heavily on being able to accurately extract all the lines from the image to fully split the image. It is also suggested that both foreground and background lines are required, but only foreground line detection is performed. The results show that the algorithm fails to segment regions correctly where lines are not present (or detected). Furthermore, the rules for merging zones are simplistic in that zones are merged vertically and then horizontally rather than selecting the best choice merge.

We employ a bottom-up, foreground approach (Section 2) to segment the components of the document into patterns, and these are classified into one of eight types (Section 3). The patterns are then modified (Section 4), which includes locating and segmenting lines from other lines and other components. Grouping the classified patterns into entire regions (Section 5) completes the algorithm. The algorithm is tested on newspaper images from the First International Newspaper Segmentation Contest [16], and the results are presented and discussed (Section 6).

2. Pattern segmentation

This algorithm uses a bottom-up approach to extract patterns from the document image. The document image is assumed to be skew-corrected. There are many published skew detection techniques [17–20]. The first step in pattern extraction is to locate rectangular regions called rects, which are described in detail in our earlier work [1]. 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. In terms of implementation, rects are stored in a linked list. This type of structure is ideal because the rects need to be quickly traversed and random access is not required.

The process of locating rects involves scanning 9-pixel squares of the image in raster order. The top left corner of a rect is defined by the first 9-pixel square found containing a black pixel. The right edge of the rect is located by searching right until a white 9-pixel square is found. Similarly, the bottom edge is located by scanning down until a white 9-pixel square is found.

A rect is defined in this way for two reasons. Firstly, it provides an efficient means of segmentation because black regions can be found without scanning every pixel. Secondly, rects are easily merged to form patterns, which are useful units as discussed below.

Patterns are defined as collections of adjacent rects [1]. In other words, each rect in a pattern must lie within 1 pixel of another rect in that pattern. Patterns are also stored in a linked list, which enables them to be quickly and easily traversed for classification. The process used to construct patterns involves traversing the (initially empty) list of patterns for each rect in the rect list. Each rect is subsequently added to a single pattern or used to define a new pattern. If a rect is found to be adjacent to more then one pattern, the patterns are merged into a single pattern. Note that each pattern contains its own rect list, and each rect is included in one and only one pattern. Also note that since the patterns are defined directly from the rects, there is no need to refer to the actual image.

For text regions in a document, patterns are often single characters, though they may be multiple characters or words, depending on the text style, size and image resolution. For non-text regions, patterns may be all kinds of shapes and sizes. There are two main advantages in using patterns rather than connected components for segmentation. Firstly, a standard document contains far fewer patterns than connected components, which makes subsequent processing much more efficient. Secondly, pattern boundaries are quickly and accurately represented by the contained rect boundaries, which is very useful for quickly determining the location and proximity of patterns. Furthermore, the definition of a rect ensures that patterns do not contain components that are more than 3 pixels apart. At a resolution of 300 dpi, that corresponds to about 1/4 mm.

3. Pattern classification

At this point we have extracted a list of patterns from the document image, but we do not know what each pattern represents: it might be a text character, a photograph, or something else. Ultimately, our aim is to define regions in a document, each of which is classified as one of the first seven classes listed in Table 1. However, before we can correctly form these regions, we must classify each pattern individually. A set of rules is used to determine the class of each pattern based on pattern size, shape, black pixel numbers and run length characteristics. These rules classify each pattern into one of the eight classes in Table 1. Class 8

Table 1
This table lists the seven final possible classes, along with two intermediary classes, for patterns and regions in the document

Class	Description
1	Text
2	Title
3	Inverse text
4	Photograph
5	Graphic/drawing
6	Vertical line
7	Horizontal line
8	Small

is an intermediary class: it is re-classified into one or more of the first seven classes at a later stage.

The newspaper pattern classification (NPC) rules are discussed in the text below, and displayed in the form of a flowchart in Fig. 1. Definitions for the variables used are given in Table 2 and threshold values are given in Table 3.

NPC A—Large Patterns. This is an efficient test for quickly identifying large photographs or graphics.

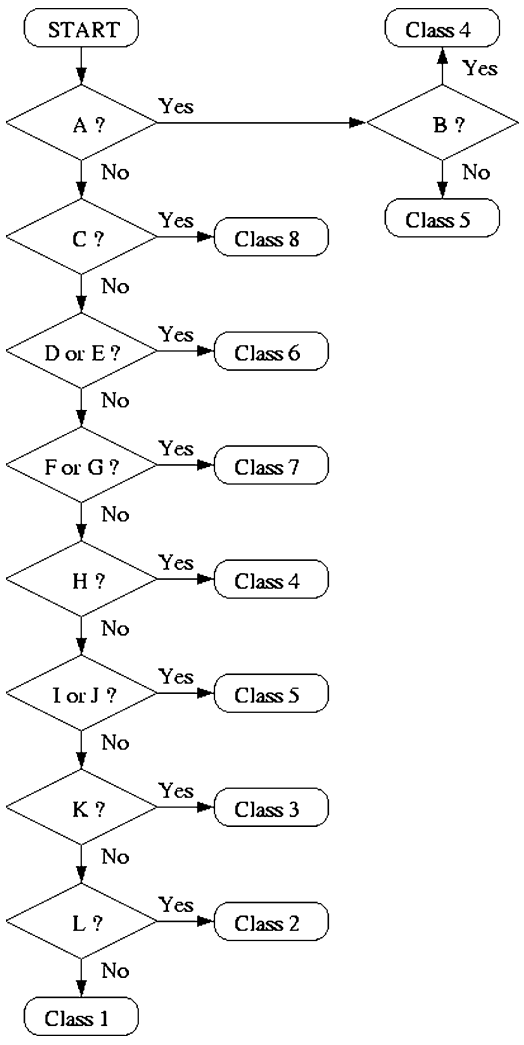

Fig. 1. Flowchart of the patterns classification decision algorithm.

Table 2
Definitions of variables used in the NPC rules

Term	Definition
w	Width of pattern
h	Height of pattern
a	Area of pattern: $w \times h$
b	Number of black pixels
d	Black pixel density: $b/(a-b)$
n	Number of black runs
m	Maximum black run length
s	Spread (defined in Rule I)
v	SD of black run lengths
h_E	Average text height (estimate)
h_T	Average text height

The assumption is that all sufficiently large patterns must be a photograph or graphic, which is a very reasonable assumption. This rule locates large patterns by testing the pattern area, width and height. Rule A is true if the following conditions hold true:

$$a > A_1 \text{ and } w > D \text{ and } h > D$$

Document dependent values are used for these thresholds to obtain good results over a range of newspaper styles and image resolutions. The most important factor for these thresholds is to ensure they are large enough to exclude the largest size text characters on the page. A good measure to use as a base for the thresholds is the average text height. The patterns are not yet classified, so we make an estimate of the average text height by calculating the average height of patterns with a height of less than 25 pixels. This upper limit of 25 pixels is safely larger than the actual text size in standard newspaper scans, and serves the purpose of ignoring excessively large patterns, which would adversely affect the calculated average. This calculation assumes that regular text patterns are the most common type of pattern, which is a very safe assumption for newspapers. This estimate of the average text height is given the label h_E.

Table 3
This table gives the values of all the thresholds used in the NPC rules

DD	DI	Ratios
$A_1 = 400h_E^2$	$A_2 = 32$	$\alpha = 0.75$
$A_3 = 80h_E^2$	$B = 16$	$\beta = 0.04$
$A_4 = 64h_E^2$	$S_1 = 350$	$\kappa = 1.4$
$D = 5h_E$	$S_2 = 500$	$\lambda_1 = 0.16$
$H = 0.7h_E$	$V = 5.5$	$\lambda_2 = 0.1$
$L = 3.0h_E$		$\lambda_3 = 0.13$
$W_1 = 1.5h_E$		$\rho_0 = 0.3$
$W_2 = 10h_E$		$\rho_1 = 0.72$
		$\rho_2 = 0.5$
		$\rho_3 = 1.65$

They are divided into document-dependent (DD) and document-independent (DI).

A large variety of newspapers was examined to determine the relative size of the largest title text. The largest page titles and major headings were found to be in the range of 15–20 times the height of the regular text, corresponding to approximately 4 cm with a regular text height of 2 mm. The area threshold, designed to be larger than the largest title area, is set at $A_1 = 400h_E^2$. The dimension threshold, set smaller to allow long and narrow graphics to be identified, has the value $D = 5h_E$.

NPC B—Photograph or Graphic. This rule is only applied to large patterns detected with Rule A (see Fig. 1). Photographs may be distinguished from graphics by measuring the amount of data contained in the pattern. The test measures the ratio of black to white pixels (density) in the pattern. Rule B is true, meaning that the pattern is classified as a photograph, if the following condition holds true:

$$d > \rho_0$$

NPC C—Small Patterns. This rule is used to identify patterns that are small in both area and number of black pixels. These often numerous patterns may be punctuation marks, graphics, or noise and are too small to correctly classify individually. Rule C is true if:

$$b < B \text{ and } a < A_2$$

NPC D—Vertical Lines. This rule identifies vertical lines by measuring the width to height ratio. There is also a minimum height restriction designed to prevent text characters such as 'I' and 'l' from being identified as lines. The ratio threshold, λ_1, is document independent, but the width and length thresholds are document-dependent. Rule D is true if:

$$h > L \text{ and } w < \min(\lambda_1 h, W_1)$$

NPC E—Vertical Lines. This rule identifies long and thick lines. Rule D set a limit on the maximum allowable line width, but newspaper images may contain very long and thick lines. The ratio constant is set lower than in Rule D to ensure the patterns are valid lines. Rule E is true if:

$$w \geq W_1 \text{ and } w < \lambda_2 h$$

NPC F—Horizontal Lines. This rule locates horizontal lines and is analogous to Rule D but with extra conditions to avoid incorrectly labeling long word patterns as horizontal lines. Only one of the following extra conditions is required to be true: the pattern height is smaller than text height, the height to width ratio is lower than expected for text, the maximum black run length is at least twice the pattern height, or, the black to white pixel ratio is higher than expected for text. Rule F is true if:

$$w > L \text{ and } h < \min(\lambda_1 w, W_1)$$

and if one of the following extra conditions is true:

$$h < H \text{ or } h < \lambda_3 w \text{ or } m > 2h \text{ or } d > \rho_1$$

NPC G—Horizontal Lines. This rule locates thick horizontal lines and is exactly analogous to Rule E. Rule G is true if:

$$h \geq W_1 \wedge h < \lambda_2 w$$

NPC H—Photographs. This rule identifies photographs based on the pattern area and number of black runs. The number of runs is a good measure because photographs generally contain a lot of information and detail, which implies a large number of black runs. The lower limit for the number of black runs is a function of the pattern area. Rule H is true if:

$$a > A_3 \wedge n > \beta a$$

NPC I—Graphics. This rule locates graphics based on area, *spread* and variation in black run lengths. Spread is a measure of the spatial distribution of black pixels in a pattern. Formally, spread is defined as:

$$s = \frac{n}{b} \cdot \min(w, h)^2$$

The number of black runs, n, is proportional to spread because a higher number implies greater spatial distribution. Similarly, spread is inversely proportional to the number of black pixels, b, because a greater density implies a lower spread. The value of spread also increases with area, and in particular with squarer shaped patterns. The following function is used to identify 'squareness':

$$sq = \min(w, h)/\max(w, h).$$

Combining this function with the pattern area we get the area function:

$$area \times sq = w \cdot h \cdot \frac{\min(w, h)}{\max(w, h)} = \min(w, h)^2$$

Fig. 2 shows three separate patterns and their corresponding values of spread. Setting a lower limit on the value of spread distinguishes graphics from text.

The variation in black run lengths, represented by v, is in fact calculated as the standard deviation in black run lengths. This is a useful test for smaller patterns because text patterns have a limited variation in black run lengths, while graphics may have larger variations. Rule I is true if:

$$a < A_4 \wedge s > S_1 \wedge v > V$$

NPC J—Graphics. This rule locates graphics in larger patterns. The measure of spread is again used, with a larger threshold more suitable for the larger patterns. The pixel density is also used to ensure the black to white

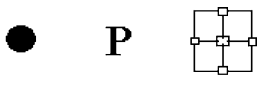

Fig. 2. Three patterns with their corresponding values of spread.

pixel ratio is smaller than expected for larger text or inverse text regions. Rule J is true if:

$a \geq A_4$ and $s > S_2$ and $d < \rho_2$

NPC K—Inverse Text. This rule identifies inverse text patterns in a logical way: the pattern must be reasonably wide, contain a high ratio of black pixels, and at least one black run must stretch most the way across the pattern. Rule K is true if:

$w > W_2$ and $d > \rho_3$ and $m > \alpha w$

NPC L—Text or Title. All the remaining patterns are assumed to be text and the average text pattern height is recalculated and given the label h_T. Rule L is then applied to distinguish the titles from the general text. Title patterns are identified as patterns with a height greater than the average height times a constant. Formally, Rule L is true if:

$h > \kappa h_T$

Clearly some tall text patterns may be mistaken for titles and vice versa for short title patterns, but it provides a useful starting point and adjustments are made later.

4. Extraction and construction of lines

Segmentation becomes very difficult when two or more different components are connected in an image. The term component is used in this section to refer to logically distinct parts of an image document. Some common examples are line components connected to text characters, other lines or noise. This section presents a solution to this problem, which works if any part of a line has been identified. Fig. 3 shows a region of a newspaper badly affected by noise that connects several separate components. Our method can separate many of these connected components. The idea is to locate lines connected to other components by extending known lines through other patterns. An outline of the algorithm is presented below, followed by an explanation of each step.

1. Locate box patterns and extract lines (see Fig. 4).
2. Construct complete lines. For each line pattern (referred to as λ).
   (a) Create a list of segments.
   (b) Create a list of partials.
   (c) Sort all pieces of λ.
   (d) Construct complete lines.

In Step 1, a pattern is identified as a box if it is large enough and all of its rects lie within a distance, δ, of the pattern edge. The size restriction is required to avoid defining text symbols as boxes. Specifically it specifies a minimum size for the longest pattern edge: $\max(w, h) > 3h_T$. The maximum allowable distance from the pattern edge is

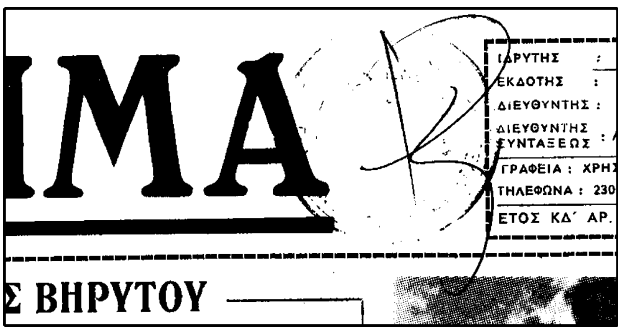

Fig. 3. An example of markings on a newspaper, which connects text, lines and a photograph. [Image: 'To Vima', 31/12/1968].

calculated as $\delta = \min(W_1, \min(w, h)/4)$. The terms used in these equations are defined in Tables 2 and 3.

If a pattern is identified as a box then up to four new line patterns may be extracted—one for each edge. The new line patterns are constructed by associating each rect in the box pattern with a pattern corresponding to the closest edge. However, a rect that lies in a corner, within δ of two edges, is associated with the farther edge pattern if the closer edge pattern does not contain other rects. The process of associating each with an edge pattern is illustrated in Fig. 4. Once the new line patterns are constructed and created, the original box pattern is removed and deleted. In Fig. 4 two line patterns are extracted from the original pattern: the left edge pattern (*l_edge*) is set to class 6 and the top edge pattern (*t_edge*) is set to class 7.

In Step 2, each detected line, referred to as λ, is processed with the aim of: extracting lines that are connected to other patterns, and merging line patterns to form complete lines. In Step 2a, a list of segments is created and in Step 2b a list of partials is created. A segment is a complete pattern, while a partial is only part of a pattern. Segments and partials are both located by examining patterns that overlap with the alignment region of λ. This region extends across the image (vertically or horizontally) in the direction of λ and extends one line thickness either side of λ to allow for variations in thickness and position. Formally, if λ is a class 6 line pattern

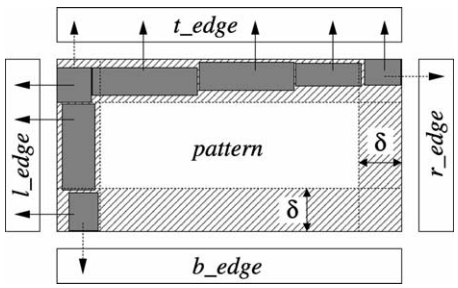

Fig. 4. A pattern containing seven rects (shaded boxes). It is defined as a box shape because all the rects lie within a distance (δ) of the pattern edge. Each rect is then associated with a pattern corresponding to one of the four edges.

(vertical) with left edge l and right edge r, then column x lies within the alignment region if $2l - r < x < 2r - l$. Similarly, if λ is a class 7 line pattern (horizontal), with top edge t and bottom edge b, row y lies in the alignment region if $2t - b < y < 2b - t$. It is assumed the origin is in the top left corner of the image.

Segments are patterns of any class (including lines) that lie wholly within the alignment region of a line λ. Partials are patterns of any class, *except* lines, that lie partly within the alignment region and contain at least one rect that lies wholly within the alignment region. The segments and partials lists are constructed by examining all eligible patterns for each line λ. Note that more than one partial may be extracted from a single pattern. One partial is separate from another if the gap between them, parallel to λ, is greater than twice the thickness of λ. See Fig. 5 for an illustration of the alignment region, segments and partials.

Clearly, not all of the partials or segments that are located will belong to genuine line components. The following two steps are designed to identify only genuine lines. Let us define the pieces of λ as being all the segments, all the partials and the line λ itself. In Step 2c, all the pieces are sorted from the lower end of the line to the higher. The lower end refers to the left end for horizontal lines and the top end for vertical lines. In Step 2d, the pieces are merged into new patterns, starting from the lower end. Pieces are merged if the space between them, measured parallel to λ, is less than twice the thickness of line λ. Each new pattern is then classified using the appropriate NPC rules: Rules D and E if λ is a class 6 line (vertical), or Rules F and G if λ is a class 7 line (horizontal). If a new pattern is classified correctly (same class as λ) then it is added to the pattern list, otherwise it is rejected.

Note, the segments that are used to construct valid new line patterns are removed from the pattern list and erased. Similarly, the original line λ is removed if it is used successfully to construct a new line. However, the partials that are used must be segmented from their patterns. This is the key step that actually segments patterns that have been detected to consist of more than one image component. After the partials have been segmented from the pattern, the remaining rects from the pattern are merged again using the method presented in Section 2, and classified again using the NPC rules of Section 3. This ensures that the one or more patterns constructed from the remaining rects are properly segmented and classified.

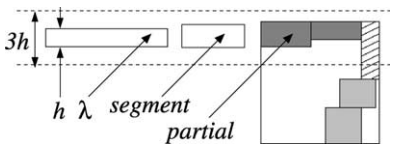

Fig. 5. Locating segments and partials within the alignment region (indicated with two dashed lines) of the line pattern λ.

5. Region formation

At this stage we have a list of patterns, which have all been individually classified. However, the goal is to define entire regions, such as paragraphs, from the document image, so the final step is to group patterns of the same type into blocks. A block is a structure containing a list of one or more patterns that are deemed to be part of the same region of the image. The issue here is how to decide if separate patterns, of the same type, are part of the same region or different regions. Clearly, the distance between patterns is a major factor: the smaller the separation, the more likely the patterns are part of the same region.

The approach taken in this algorithm is to define a minimum horizontal and vertical gap. These spacings are defined in terms of the average text pattern height (*avh*). The values are chosen to allow most text paragraphs to be merged into one block, without merging separate regions. The horizontal and vertical spacings are, respectively, defined as:

$$hgap = 1.1 \times avh; \qquad vgap = 0.8 \times avh$$

The algorithm used to construct the blocks is specified below in pseudocode, and is followed by an explanation.

```
p = patterns
for all p
    grouped = F
    b = getblocks(p.id)
    for all b
        found = F
        if near(p,b)
            q = b.patterns
            for all q
if near(p,q)
found = T
if found = T
if grouped = F
addpat(p,b)
else
merge(g,b)
            grouped = T
            g = b
        if grouped = F
    b = newblock(p)
    addpat(p,b)
```

A search is made for each pattern (p) in the pattern list (*patterns*). For each p, the function getblocks( ) returns the list of blocks with the same type (*id*) as p. Initially these lists are empty, but as each pattern is processed the block lists grow. A search is then made for each block (b) in

the list. The function near( ) is used to test if p is close to a pattern in b. The first call, near(p, b), determines if p is close to the block b. If so, each pattern (q) in the block b is tested with the call near(p, q). The flag *found* is set to true (T) if any pattern in b is near the pattern p. Now each pattern p may be close to more than one block. If the flag *grouped* is set to false (F) then b is the first block found near to p. In this case, p is added to b, then *grouped* is set to true (T) and b is saved to g. If later another block b is found near to p then the blocks b and g are merged as they belong to the same region. This is performed with the call merge(g, b). If the flag *grouped* is set to false (F) after all the blocks have been searched, it means no block was found to be close to p. In this case, a new block is created with the function newblock(p), and p is added to the new block.

To determine if two patterns are near each other the function near(x, y) compares the edges of the two patterns x and y. The edges are represented with a single character: t, b, l, and r, corresponding to top, bottom, left and right, respectively. The function returns true if both of the following conditions are true:

1. $x.l < y.r + hgap$ and $x.r > y.l - hgap$
2. $x.t < y.b + vgap$ and $x.b > y.t - vgap$

This method of region formation produces good results, but fails under certain circumstances. One of the problems is that text and title patterns may be incorrectly classified. It may happen that most of a title is classified as title while a few of the smaller characters are classified as text. Similarly, a text paragraph may contain a few odd patterns classified as titles. The way to fix this problem is to merge nearby text and title patterns. The procedure is analogous to the one described above to form blocks. However, in this case we group text and title blocks together into what we call metablocks. Block A and block B are merged into a metablock if they contain patterns p and q, respectively, that satisfy the following conditions:

1. $p.t < q.b$ and $p.b > q.t$
2. $p.l < q.r + hgap$ and $p.r > q.l - hgap$

Once all the text and title blocks have been merged into metablocks, they are reclassified and converted back into blocks. Reclassification means the most common pattern class is used to set the class of the metablock. For example, a few title blocks in a text paragraph block should be merged and classified as text.

Region formation is almost complete: the final step is to merge title blocks. Title blocks are more difficult to merge because they can vary greatly in size, and consequently, the inter-word and inter-line spaces can be quite large. Our method for merging text blocks calculates the maximum gap allowed between two patterns as a function of the pattern heights from both blocks. The function makes the maximum gap large for similar sized titles, but small for dissimilar sized titles.

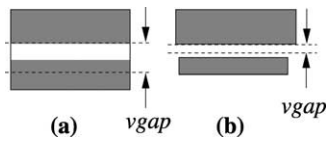

Fig. 6. The process of merging title blocks. It shows how *vgap* varies with the average pattern height of both blocks. In (a) the heights are similar so *vgap* is large enough for the blocks to merge. In (b) *vgap* is much smaller since the heights are not so similar so the blocks are not merged.

The procedure is again analogous to the one described at the beginning of this section to form blocks. However, in this case only title blocks are merged and *hgap* and *vgap* are redefined. Let us define h_A and h_B as the average pattern height in block A and block B, respectively, then the maximum gap allowed between title blocks is defined by:

$$hgap = IWF \cdot \min(h_A, h_B) \cdot \frac{\min(h_A, h_B)}{\max(h_A, h_B)}$$

$$vgap = ILF \cdot \min(h_A, h_B) \cdot \frac{\min(h_A, h_B)}{\max(h_A, h_B)}$$

where *IWF* and *ILF* are the inter-word and inter-line factors, respectively. In our implementation we use the values: $IWF = 1.05$ and $ILF = 0.75$. The first two elements of the equations define the allowable gap as a ratio of the minimum average height, while the third element takes into account the difference in height. Fig. 6 illustrates the usefulness of this procedure in distinguishing title regions. Note, when calculating the average pattern height, 50% of the patterns are ignored: the shortest 25%, and the tallest 25%. This prevents distorted averages arising from short punctuation symbols or tall patterns due to noise or connected characters.

6. Results

Our program was tested on newspaper images from the set used for 'The First International Newspaper Segmentation Contest' [10]. These images are all front pages from various English and Greek newspapers dating back to 1900. One of the test images is shown in Fig. 7. Some markings from a stamp and pen lines have degraded this image, which can be seen in more detail in Fig. 3. The result of our algorithm applied to this image is shown in Fig. 8, and the key to the shadings is given in Fig. 9. Overall the segmentation is very successful for this image, though the markings in the upper right corner have caused some problems. Let us look at that region in more detail.

The patterns located in the test image are shown in Fig. 10 for the region in the upper right corner. This figure shows how the stamp and pen markings cause the text box, some characters, a horizontal line and a photo to be connected. However, significant improvement is seen after the line


ΕΧΕΙ ΠΑΓΩΣΕΙ ΟΛΗ Η ΙΤΑΛΙΑ!

ΤΟ ΒΗΜΑ

ΤΡΙΤΗ 31 ΔΕΚΕΜΒΡΙΟΥ 1968

ΜΕΤΑ ΤΗΝ ΙΣΡΑΗΛΙΝΗΝ ΕΙΣΒΟΛΗΝ ΕΙΣ ΤΟ ΑΕΡΟΔΡΟΜΙΟΝ ΤΗΣ ΒΗΡΥΤΟΥ

ΠΑΓΚΟΣΜΙΟΣ ΑΝΤΙΔΡΑΣΙΣ ΚΑΙ ΕΝΤΟΝΟΣ ΑΝΗΣΥΧΙΑ

Προτείνεται άμεσος σύγκλησις τετραμεροῦς διασκέψεως

ΜΕ ΠΑΡΑΜΟΡΦΩΜΕΝΗΝ ΜΠΑΛΛΑΝ ΤΟΥ ΓΚΟΛΦ ΟΜΟΙΑΖΕΙ Η ΣΕΛΗΝΗ!

Εδημοσιεύθησαν αι πρῶται 12 φωτογραφίαι

Διάγγελμα τοῦ Ἀντιβασιλέως

Ὕψους 14.500 ἑκ. δραχ. αἱ δημόσιαι ἐπενδύσεις

Θά ταξιδεύωμε σέ ἄλλους κόσμους

ΚΑΙ ΠΕΡΑΝ ΤΗΣ ΣΕΛΗΝΗΣ

2.- Φόβος καὶ Δῆμος, τα δυό «παιδιά» τοῦ Ἄρη

Τό τραγούδι τοῦ Γένους

Ραδιοφωνικόν μήνυμα τοῦ κ. Πρωθυπουργοῦ

Ο ΘΡΙΑΜΒΟΣ τῆς φαντασίας

Ὁ Πρωθυπουργός τῆς Ἰαπωνίας δέρνει τήν σύζυγόν του μέ «ταχύν ρυθμόν»

Fig. 7. Example of a test image [‘To Vima’, 31/12/1968].

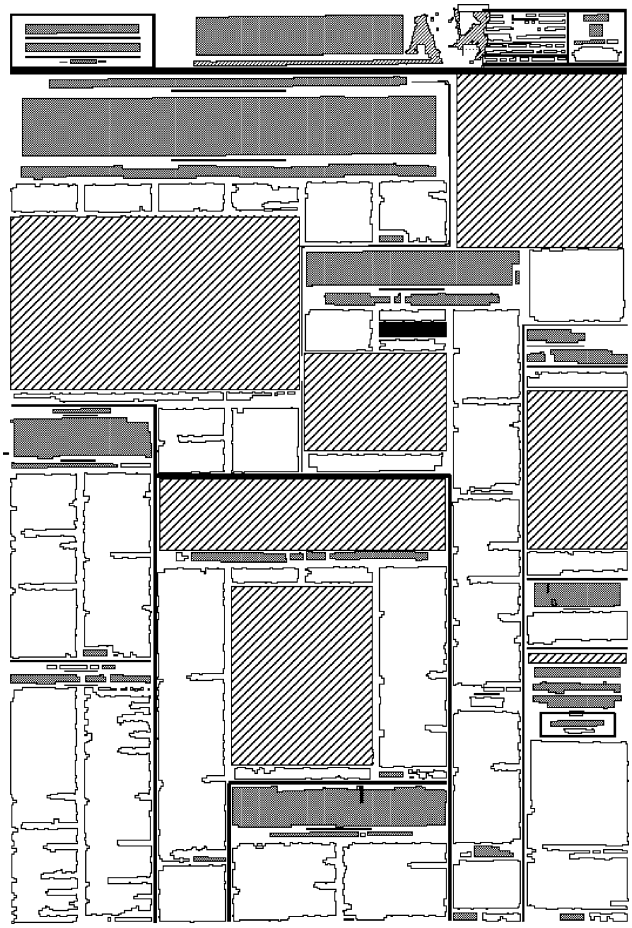

Fig. 8. The results of our algorithm applied to the test image shown in Fig. 7. The key to the classification is shown in Fig. 9.

extraction step (see Fig. 11). The photograph, horizontal line and top and bottom edges of the text box have all been segmented successfully and correctly classified. In fact, the only lines that could not be segmented correctly were either connected to noise, like the left edge of the text box in Fig. 12, or connected to other lines in an unusual shape, such as a ‘T’ shape or ‘H’ shape.

To evaluate our algorithms performance, we use the Newspaper Component Detection Metric ($NCDM_i$) [16]. This method of evaluation is based on counting the number of matches between the regions detected by the algorithm and the regions in the ground truth. In our algorithm a region corresponds to a block. A ground truth is a list of all the ‘true’ regions in a document along with their correct class.

A *MatchScore* table is created with one row for each result region and one column for each ground truth region. Each value in the table indicates the strength of the match between a result and a ground truth region. Values are in the range of zero to one, where zero indicates no match and one indicates perfect match. Let I be the set of all foreground pixels, R_x the set of all points inside the xth result region, G_y the set of all points in yth ground truth region, g_x the class of region x, r_y the class of ground truth region y, and $T(s)$ a function that counts the elements of s. Then the *MatchScore* table is defined as:

$$MatchScore(x, y) = a\frac{T(R_x \cap G_y \cap I)}{T((R_x \cup G_y) \cap I)}$$

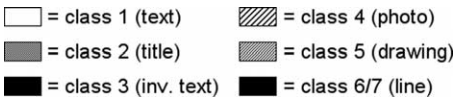

Fig. 9. Key for region shading in Fig. 8.

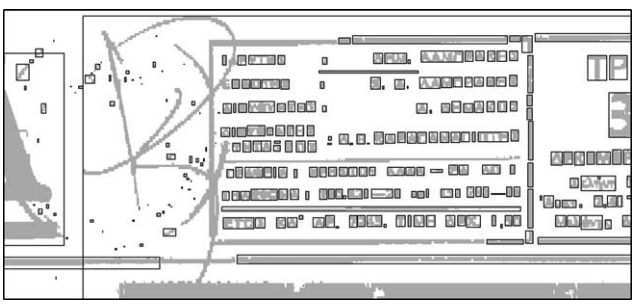

Fig. 10. Patterns located in the upper right corner of the test image (Fig. 7).

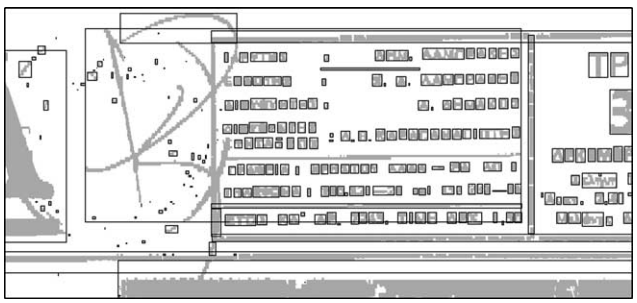

Fig. 11. Patterns from the upper right corner of the test image (Fig. 7) after lines have been extracted and segmented.

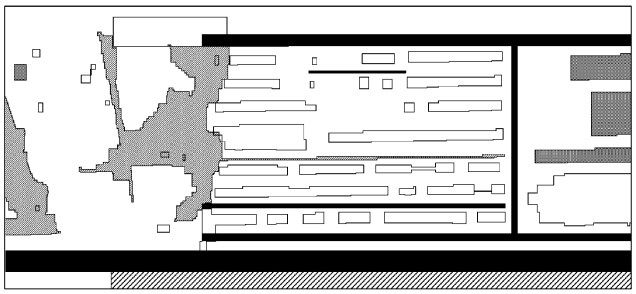

Fig. 12. Detail of the results for the upper right corner of the test image (Fig. 7).

$$\text{where } a = \begin{cases} 1, & \text{if } r_x = g_y \\ 0, & \text{otherwise} \end{cases}$$

If N_i is the number of ground truth regions with class index I and M_i is the number of results regions with class index i, we can calculate the detection rate, det_i, and recognition accuracy, rec_i, for class i as follows:

$$det_i = \frac{one2one_i}{N_i} + \frac{g\_one2many_i}{4N_i} + \frac{g\_many2one_i}{4N_i}$$

$$rec_i = \frac{one2one_i}{M_i} + \frac{d\_one2many_i}{4M_i} + \frac{d\_many2one_i}{4M_i}$$

where the undefined terms are calculated from the *MatchScore* table by following the steps of Philips and Chhabra [21]. The calculation details are omitted, but each term is now briefly defined. The term $one2one_i$ is a count of the $MatchScore(x, y)$ entries corresponding to class i with values greater than the acceptance threshold of 0.85. In other words, one result region matches well with one and only one ground truth region. The term $g\_one2many_i$ is a count of the class i ground truth regions that partially match with two or more result regions. These multiple result regions are counted to form the $d\_many2one_i$ term. A partial match requires a score greater than the rejection threshold of 0.1, but less than the acceptance threshold. The terms $d\_one2many_i$ and $g\_many2one_i$ are similarly defined as counts relating to result regions partially matching two or more ground truth regions. Note the prefixes 'g' and 'd' correspond to 'ground truth' and 'detected', respectively. Two other measures are also extracted from the *MatchScore* table: $misses_i$ is a count of the ground truth regions with no match scores above the rejection threshold and $false\_alarms_i$ is a count of the result regions with no match scores above the rejection threshold (Fig. 12).

The performance metric for each region class is extracted by combining the detection rate and recognition accuracy as follows:

$$NCDM_i = \frac{2det_i rec_i}{det_i + rec_i}$$

Finally, a global performance metric, known as Newspaper Segmentation Metric, is defined as:

$$NSM = \frac{2\sum_I det_i \sum_I rec_i}{I\left(\sum_I det_i + \sum_I rec_i\right)}$$

where I is the number of classes, which in our case is seven.

Table 4 lists the match results and performance metrics for each of the seven class types. The performance is good for most classes and Fig. 13 illustrates that our results compare very well in every class with the results from

Table 4
This table list the results of our algorithm, including the Newspaper Component Detection Metric ($NCDM_i$), for each region class

Class index, i	1	2	3	4	5	6	7
N_i	265	182	7	19	12	62	178
M_i	360	211	6	26	11	58	154
$one2one_i$	252	91	6	19	4	48	149
$g\_one2many_i$	10	53	0	0	1	0	0
$g\_many2one_i$	3	0	0	0	0	0	10
$d\_one2many_i$	8	0	0	0	0	0	5
$d\_many2one_i$	42	120	0	0	3	0	0
$misses_i$	0 (0%)	24 (13%)	1 (14%)	0 (0%)	7 (58%)	14 (23%)	19 (11%)
$false\_alarms_i$	53 (15%)	14 (7%)	0 (0%)	7 (27%)	4 (36%)	10 (17%)	0 (0%)
$DetectRate_i$ (%)	96.3	57.3	85.7	100.0	35.4	77.4	85.1
$RecogAccuracy_i$ (%)	73.5	57.3	100.0	73.1	43.2	82.8	97.6
$NCDM_i$ (%)	83.4	57.3	92.3	84.4	38.9	80.0	90.9

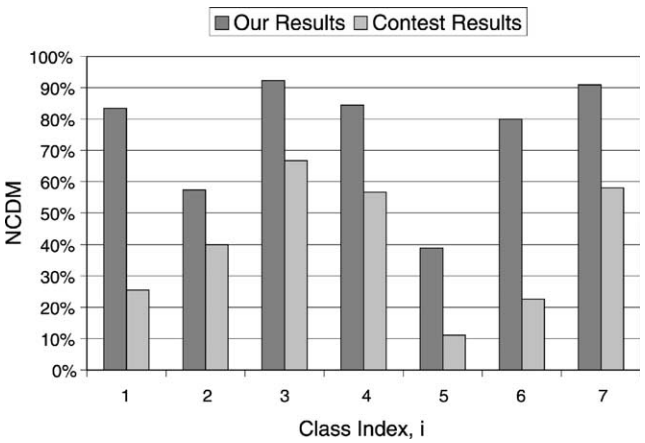

Fig. 13. This is a plot of the $NCDM_i$ for our results compared with the best results (in each class) from the newspaper contest.

the First International Newspaper Segmentation Contest [16]. It is noted that the 'Contest Results' plotted in Fig. 13 correspond to the best result in each class out of the three algorithms compared in the contest.

The average performance in classifying type 5 graphics is due to two factors. Firstly, a high proportion of the graphics regions (50%) are misclassified as class 4 photographs. This is because many of the graphics are highly textured (see the central graphic in Fig. 7), which makes them very difficult to distinguish from photographs. The second factor is noise patterns, like those extracted from the markings shown in Fig. 3, which are mostly classified as graphics.

Title classification (class 2) is the other result that requires some discussion. Firstly it is noted that $misses_2$ and $false\_alarms_2$ are satisfactorily low. The low value of $NCDM_2$ is a result of over-segmentation, identified from the high $g\_one2many_2$ and $d\_many2one_2$ counts. The main cause of problem is the large number of small titles. The negligible difference in height between title and text characters is not enough to distinguish the two. Segmentation occurs when different parts of the same title are classified differently. However, larger titles are segmented and classified very well, especially titles and sub-titles that lie very close to each other.

The recognition accuracy for lines is very high. However, the test documents contain several groups of connected lines, not only in recognizable box shapes but also in more complicated configurations such as 'T' and 'H' shapes. These groups of lines are generally misclassified as graphics and further contribute to the relatively poor performance for graphics.

A factor that affected the results across all classes is the image quality. For example, in the original images some poor quality lines were severely segmented making them very difficult to correctly recognize. Many of the *one2many*, *misses*, and *false_alarms* are due to components being very close or actually connected.

Finally, the Newspaper Segmentation Metric (*NSM*) for our algorithm is calculated as 76.0%. This is a vast improvement on the contest results, and is close to twice the highest score of 38.7%.

7. Conclusions

This paper discusses an algorithm that performs segmentation and classification of newspaper images. The results show that many components that have been connected, by noise or design, can be correctly segmented and classified. Genuine newspaper images were used to test the algorithm. The images contain tightly positioned text and title regions as well as numerous defects that cause distinct components to be connected. The algorithm scored 76.0% using the Newspaper Segmentation Metric, which is nearly twice the highest score achieved in the First International Newspaper Segmentation Contest. The major problems for misclassifications are small titles positioned close to text, and poor image quality.

Acknowledgements

This work is supported by the Hong Kong Research Grant Council (Project CityU 1128/01E).

References

[1] P. Mitchell, H. Yan, Document page segmentation based on pattern spread analysis, Optical Engineering 39 (3) (2000) 724–734. March.
[2] P. Mitchell, H. Yan, Newspaper Document Analysis Featuring Connected Line Segmentation, Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR), Seattle, 2001, pp. 1181–1185.
[3] P. Mitchell, H. Yan, Document Page Segmentation and Layout Analysis using Soft Ordering, Proceedings of the 15th International Conference on Pattern Recognition (ICPR), vol. 1, 2000, pp. 458–461.
[4] B. Gatos, M. Papamarkos, Applying Fast Segmentation Techniques at a Binary Image Represented by a Set of Non-Overlapping Blocks, Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR), IEEE, New York, 2001, pp. 1147–1151.
[5] F. Liu, Y. Luo, M. Yoshikawa, D. Hu, A New Component based Algorithm for Newspaper Layout Analysis, Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR), Seattle, 2001, pp. 1176–1180.
[6] J. Xi, J. Hu, L. Wu, Page segmentation of Chinese newspapers, Pattern Recognition 35 (2002) 2695–2704.
[7] C. Strouthopoulos, N. Papamarkos, Text identification for document image analysis using a neural network, Image and Vision Computing 16 (12/13) (1998) 879–896. August.
[8] K. Hadjar, O. Hitz, R. Ingold, Newspaper Page Decomposition using a Split and Merge Approach, Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR), Seattle, 2001, pp. 1186–1189.

[9] J. Liu, Y.Y. Tang, C.Y. Suen, Chinese document layout analysis based on adaptive split-and-merge and qualitative spatial reasoning, Pattern Recognition 30 (8) (1997) 1265–1278.
[10] A. Antonacopoulos, Page segmentation using the description of the background, Computer Vision and Image Understanding 70 (3) (1998) 350–369. June.
[11] K. Kise, O. Yanagida, Page segmentation using thinning of white areas, Systems and Computers in Japan 29 (3) (1998) 59–68.
[12] Y. Xiao, H. Yan, Text extraction in document images based on Delaunay Triangulation, Pattern Recognition 36 (3) (2003) 799–809.
[13] G. Nagy, Twenty years of document image analysis in PAMI, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (1) (2000) 38–62. January.
[14] Y.Y. Tang, S. Lee, C.Y. Suen, Automated document processing: a survey, Pattern Recognition 29 (12) (1996) 1931–1952.
[15] Y. Tseng, H. Lee, Recognition-based handwritten Chinese character segmentation using a probabilistic Viterbi algorithm, Pattern Recognition Letters 20 (1999) 791–806.
[16] B. Gatos, S. Mantzaris, A. Antonacopoulos, First International Newspaper Segmentation Contest, Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR), Seattle, 2001, pp. 1190–1194.
[17] H. Yan, Skew correction of document images using interline cross-correlation, CVGIP: Graphical Models and Image Processing 55 (6) (1993) 538–543.
[18] H.K. Kwag, S.H. Kim, S.H. Jeong, G.S. Lee, Efficient skew estimation and correction algorithm for document images, Image and Vision Computing 20 (1) (2002) 25–35.
[19] E. Kavallieratou, N. Fakotakis, G. Kokkinakis, Skew angle estimation for printed and handwritten documents using the Wigner–Ville distribution, Image and Vision Computing 20 (11) (2002) 813–824.
[20] P.-Y. Yin, Skew detection and block classification of printed documents, Image and Vision Computing 19 (8) (2001) 567–579.
[21] I.T. Philips, A.K. Chhabra, Empirical performance evaluation of graphics recognition systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (9) (1999) 849–870.