1. Background Introduction

Document layout recognition/classification is one of the hot topics in document analysis area. There has been a lot of works concentrating on \textit{structured} layout classification, i.e., layout that features in regular paragraphs such as academic papers, for which, block based methods have been proposed and have achieved good results. The hard thing is to deal with non-structured document images, such as forms, handwritten letters, receipts, checks, etc. In these cases, even forming a clear descriptive definition for layout is difficult, and most often, classification of documents becomes heavily dependent on special patterns, such as logos or special formats of titles, signatures, etc.

This application is carried out to see (through TimeSearcher2) whether a special document image description code can be used to characterize layout features.

2. Preprocessing on Document Image

A document is first scanned as a TIFF format image and is passed through a text line detector. Based on the information of output text lines (each text line’s position, length, orientation, and font height), a binary/monotone abstract image is generated. These lines represent features which will be used to visually characterize the structure.

![](image)

---

Figure 1.

In figure1, the filled white rectangles correspond to textlines output from textline detector, which often contains false alarms and missing text lines. Fig1 (b) is the result of low level preprocessing, including noise removal, rotation and binarization.

Since we are dealing with layouts based on text lines, the fringe shape of a document need to be excluded, we detect the textline bounding box and resize the image by a factor of 5 to lessen computation load. To make each scan line easier to detect in the code, we pad zeros on the left.
side, and we also pad zeros in the bottom side to make the final images same size, see (c) in Fig.1, though this step is not essentially necessary for feature detection by TimeSearcher2.

3. Encoding the Document Image

For simplicity, we use “0” represent a black pixel, and “1” for a white pixel. A run length encoding is used to represent the lines as follows: for each scanned line “0000000… (consecutive m₁ “0”) 1111… (consecutive n₁ “1”) 0000… (consecutive m₂ “0”) 11111… (consecutive n₂ “1”)…”, a series “-m₁+n₁-m₂+n₂…” will be formed with positive numbers for the length of white segments and negative numbers for black segments.

As shown in Figure2, encoding for the red bounding box in (a) is shown in (b). Suppose the filled rectangles of the textlines take up three scan lines, and the black part between the up and down textlines takes up two scan lines. One scan line is highlighted in the left image. So each “M” shape in the right figure corresponds to one scan line in the left image.

Encoded in this way, a document image now becomes a number series. Different images will not have same length of codes any more, because though original normalized images are of same size, they may different in the number of segments in scan lines. It is not difficult to align scan line between different codes, but this is not supported by current TimeSearcher2, so we limit our application in this respect.

Note that this kind of code is totally based on the text line information; one can easily read the textline out by viewing the code. One may associate this kind of code as a way for compressing image (and it is certainly not a good code compared with jpeg or others), but how does it used for layout analysis, especially when it give you no more information than the textline information already available from text line detector? Why bother to code such derived information? The reasons are:

a) By forming a curve to visualize a document image, we are trying to analyze document through the shape of the curve, or more accurately, the open polygon.

b) It is easier to form a coarse to fine comparison on two curves than on two sets of text lines.

c) On curves, most of the time noise is comparably easier to identify.
d) It might be more natural to tell a regular part (full of repeated patterns) from a curve intuitively than to extract it by applying heuristic rules on a set of data.

e) Based on the code, Hidden Markov Models can be used to do further analysis on layout recognition.

f) A different view angle will contribute to a more complete cognition about things and if lucky, sparks a new idea.

However, even if the above advantages were valid, we need examine the feasibility of this special encoding approach. It is better to analyze the code as much as possible before setting out to implement rashly.

4. Specific Goals for Feasibility Testing

Treating each code as a time series, we can use TimeSearcher2 to analyze the codes. The answers for the following questions are expected, as they tell whether the idea of document image code is feasible for layout recognition/classification:
1) Will similar layouts present similar codes?
2) Can discriminative layout features be easily identified by this kind of code?
3) Do the discriminative features correspond to normal layout description?
4) How robust are the discriminative features to image noise and text line detector faults?

As from TimeSearcher2 point of view, the above questions are equivalent to:
1) Do similar curves (in coarse granularity) come from same layout category? (compare to ground truth)
2) Are there discriminative patterns to distinguish different curves? (in fine granularity)
3) Can these patterns be translated into a layout description language? (e.g, a pattern corresponds to a slopped line.) No evident correspondence means that the patterns are less controllable, no matter how effective they are for layout classification.
4) Can noise be easily identified and does it impact the discriminative patterns?

5. Dataset

There are 12 classes/layouts in total with tens to hundreds samples for each as follows:

<table>
<thead>
<tr>
<th>Layout Category</th>
<th>Shorthand Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-column printout</td>
<td>1c</td>
</tr>
<tr>
<td>two-column printout</td>
<td>2c</td>
</tr>
<tr>
<td>three-column printout</td>
<td>3c</td>
</tr>
<tr>
<td>printout with long title and two columns</td>
<td>1r2c</td>
</tr>
<tr>
<td>printout with two one-column block and one two-column blocks</td>
<td>1r1r2c</td>
</tr>
<tr>
<td>printout with two-asymmetric-column</td>
<td>2c_asym</td>
</tr>
<tr>
<td>printout with one two-asymmetric-column block and one two-column block</td>
<td>2c2c_asym</td>
</tr>
<tr>
<td>forms with printout and English handwritten</td>
<td>class1</td>
</tr>
<tr>
<td>informal letters with signatures in Arabian</td>
<td>class2</td>
</tr>
<tr>
<td>Arabian handwritten resume</td>
<td>class3</td>
</tr>
</tbody>
</table>
6. Experiments

6.1 Layout Class Overview

Figure 3 shows representative document image codes for the above binary images, with title above each curve showing the layout category. The overview shows that this kind of code presents evident difference for different layouts.
Figure 3. representative code of each layout

Figure 4 shows the first two layouts with four samples for each. It can be seen that documents with same layout labels present similar codes in overview. For space reason, all layouts with their samples cannot be shown here.
Figure 4. (a) 4 sample codes from the first layout, (b) 4 sample codes from the second layout

6.2 Pattern Search in Layout Representatives

The following figures are in the order of the layout name given in section 5.

The red pattern shows the dominant text lines are of similar font-height and lengths, similar horizontal places and similar intervals. The pink circle in the binary image is highlighted for later comparison with Layout6.
Figure 6 above shows the two-column printout representative document. The arrow points out an outlier which is due to the inaccuracy of text line detector (two text lines are connected due to image noise). The most frequent pattern (the red ones) interprets this two-column layout very well: in each scan line, these two white lines are of different lengths, so there are repeated one higher and one lower teeth; and similar span length of the red patterns imply similar font height; and similar y-coordinates imply similar horizontal position of text lines. The bottom right figure shows an unmatched pattern (the blue one): for this textline, the right column is too short compared to the pattern being searched.

Figure 7 shows the three-column printout representative document. Except for the title part, there are many high zones due to the false long lines in the body part of the printout (caused by inaccuracy of the textline detector). The juxtaposition of three columns is captured by the three turnings in the red patterns in the bottom left figure. The dotted olive box in the bottom right figure shows some wrongly matched pattern. They look very un-similar with the designated target pattern.
Figure 7. three-column printout representative document

Figure 8. long title + two-column printout: different from Layout2 in the title part, which is pointed out by the pink ellipse.
Figure 9. Printout with one one-column block and two two-column blocks: former part of the code features in one-column feature and rear part features in two-column feature.

Figure 10. Asymmetrical two-column printout. The red low-high pattern mirrors the high-low pattern in Figure 11; and since this asymmetry is caused by juxtaposition of a shorter line and a longer line, the pattern is different from the one indicated by the pink circle in Figure 5, in which the short lines do not juxtapose with other long text lines.

Figure 11 below analyzes the code for asymmetrical two-column printout document, which is dominated by two patterns indicated by pink and green arrows respectively in the binary image. The green arrow points to the regular pattern in two-column printout. The pattern pointed by pink arrow expresses the asymmetry feature very well.
Figure 11  Asymmetrical two two-column-block layout

Figure 12. Form, mixed with text line, table line and handwritten text lines, features in scattered short lines. The code is full of irregular serrated segments.
Figure 13. informal handwritten letter, featured by sloped signature text lines. There are four evident sloped text lines and are captured by the four roof-shape patterns. The first roof is not labeled to keep a clear view.

The following layout features in right-alignment (indicated by the pink line in the bottom figure: the negative value before the negative one indicating a new scan line are same) and frequent wide intervals between vertically neighboring two text lines (indicated by the horizontal lines after the red vertical lines in the middle figure).

Figure 14. Arabian handwritten resume.
Figure 15 shows an example of Formal letters in Arabian with a special logo. The logo part, indicated by the red circle in the right image, is weakly detected by text line detector. And by examining other same class images, the logo parts are quite different. Our former classification method cannot classify this kind of document well either. In fact, classification by logo is not equal to classification by layout.

6.3 Discriminative Pattern Testing

It is important to test whether the discriminative patterns for these representative documents are also definitive for other documents from correspondingly same layout class. Experiments on searching common patterns for different documents with same layout, like in Figure 2, need to be carried out. In fact, two kinds of experiments are worth doing: querying the discriminative patterns on documents with the same layout and querying them on documents from different layouts. However, TimeSearcher2 cannot show results on individual items, see Figure 11, and unfortunately TimeSearcher1 cannot handle long time series. (In this application, all series are 2000 in length). Experiment one code item one time is too time-consuming (averagely, there are 200 documents per layout in the dataset).

Figure 15  Formal letters in Arabian with special logo

Figure 16. TimeSearcher2 does not show the matched patterns separately on individual series item, which makes the figure unreadable.

7. Conclusions and Expectations on TimeSearcher2
From the experiment results, it can be seen that this kind of special code has its potential for document layout Recognition/classification, though the experiment is not fully carried out due to the software limitations. Based on the patterns in the code, we can extract the pattern changing style of a document image as we scan the image one line by one, which provides a chance to model a document image by Hidden Markov Models. It should be noted that the purpose of this application is to see whether this idea works, not to compare it with other methods. Flexibility and time complexity of pattern searching in series influence the efficiency a lot.

TimeSearcher2 is in its very beginning phase, it will become more powerful in the near future thus more helpful for us. As for this application, some functions are expected:
- Provide simple statistics on the searching results, e.g. count the patterns found.
- Support multiple pattern searching simultaneously, like that in TimeSearch1.
- Coarse-to-fine visualization and pattern search on each level, i.e., zooming facility. Indeed this can accelerate pattern searching.
- Show searching results on individual items when pattern searching is carried out among many items.
- Support user-defined simple patterns.