Skew Detection, Skew Normalization and Segmentation of Document Images using Segmented Block Code

by

Masayuki OKAMOTO*, Hashim M. TWAAKYONDO**
and Hajime NISHIZAWA**

(Received October 31, 1987)

Simple efficient algorithms are presented which detect and normalize skews of scanned document images and separate these images into constitutional elements such as text areas, field separators, and table or picture areas. Conventional algorithms are based on pixel by pixel processing and deal with skew detection and normalization separately from segmentation, so that they require a lot of processing time on a general purpose computer. The new algorithms use only segmented block (SB) coded data for whole processing. An SB coding scheme featured by easiness of manipulation of image data is proposed for data compression. As compared with pixel data of document images, the number of SB coded data is very few and the form of this code is easy to process on conventional computers. The use of SB coded data has enabled us to devise the simple efficient algorithms.

1. Introduction

The construction of document image recognition systems[1] requires the skew detection, skew correction, and segmentation of inputed documents as preprocessings. There are many studies on segmentation methods[3]~[6] which use projection profile, stroke density, neighbourhood line density (NLD), size of enclosed rectangle, Fourier transformation characteristics, etc.. Generally, we make segmentation more precisely by using more than one characteristic, but in this case more processing time is required. Complete recognition of document structure requires the semantics of documents, so there are some limitations on complete segmentation when we use only structural or statistical characteristics.

We do not deal with the above problem, but we propose an efficient method for detection and correction of skewed documents and for segmentation of documents which have some kinds of common structures such as paper, manual,

* Associate Professor.
** Student of master's Course.

business letters, etc.. In order to obtain an efficient method, it is desirable to reduce the pixel by pixel processing which requires a lot of time on conventional computers.

Segmented block (SB) coding scheme[2,7] is proposed for the following reasons :

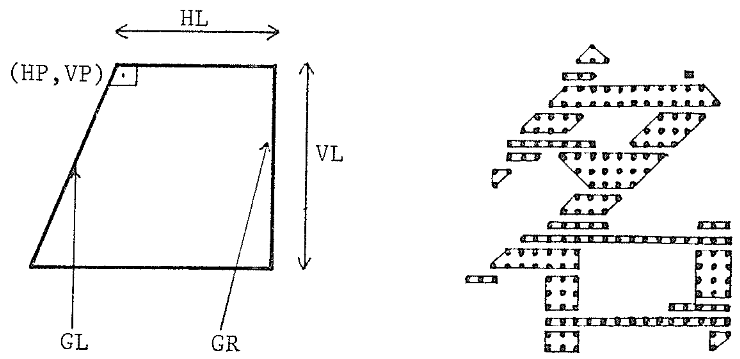
(1) Efficient data compression of document images,

(2) Easy image processing on coded data.

In SB coding scheme, a contiguous black pixel area is represented by its position and figure, and therefore some kinds of image processings are carried out efficiently by using SB coded data. We describe the SB coding scheme briefly in section 2, the detection and correction of skews in section 3, and the segmentation in section 4.

2. SB Coding Scheme

In the SB coding scheme, document images are first divided into rectangular areas which are called segments, and then, blocks which have certain slopes in left or right edges are coded by using positions relative to segments and some kind of information about figures. Figure 1 shows an example of a block and its representation.



(a) Representation of a block (b) Example of blocks

HP, VP: Relative position of a block in a segment.
HL, VL: Horizontal and Vertical length of a block.
GL, GR: Right and Left slopes of a block.

Fig. 1 Representation of SB coded data.

3. Detection and Correction of Skew

Our method detects and normalizes skewed documents by using SB coded data. Generally, the skew of document images can be detected by finding slopes of character strings. Slopes of character strings are obtained from the baseline of strings. These baselines are estimated from the squence of enclosed rectangles for characters. The correction of skewed documents is performed by rotating blocks with their starting points (HP, VP) referenced.

3.1 Detection of Skew

We obtain baseline of character strings from an approximated line which is estimated from the squence of bottom left points of enclosed rectangles corresponding to character strings. Enclosed rectangles for each character are obtained in the first step of segmentation procedure described in section 4. We classify these rectangles into several groups each of which corresponds to each character string, this classification is carried out as follows:

(1) Sort of rectangles. We sort rectangles in the order of x-coordinates of bottom left points. By this processing we can easily select rectangles which are located in the horizontal direction.

(2) Classification of rectangles corresponding to strings. We classify rectangles which satisfy the conditions

$$rx_j - rx_i \leq \theta_1,$$
$$|ry_j - ry_i| \leq \theta_2,$$
$$\theta_1 = 50,\ \theta_2 = 5,$$

where $(rx_i,\ ry_i)$ and $(rx_j,\ ry_j)$ are left bottom points of adjacent blocks.

These classified rectangles correspond to each character string. In the above conditions, θ_1 is determined to prevent merging of character strings which are located in the opposite sides when text lines are separated at left-and right-hand sides in a document. θ_2 is determined to prevent merging of each pair of successive text lines.

We obtain an approximate line from the sequence of bottom left points of rectangles by using the method of least squares, and then estimate the skew of document from the average value of the slopes of approximated lines.The theoretical resolution of skew detection is $\tan^{-1}\left(\frac{1}{1728}\right) = 0.033°$ when horizontal size of document is 1728 pixels, but we estimate the skew with a precision of 0.1 degree.

3.2 Correction of Skew

The pixel by pixel skew correction procedure requires a lot of processing time. Our procedure corrects the skew by rotation of blocks in order to reduce the processing time. In our procedure we first transform the coordinates $(HP,\ VP)$ of the starting point of blocks in a segment into coordinates $(HP_c,\ VP_c)$ in the whole document, and then rotate these blocks by the formulae

$$x = HP_c \cdot cos\,(\theta) - VP_c \cdot \sin\,(\theta),$$
$$y = HP_c \cdot \sin\,(\theta) + VP_c \cdot \cos\,(\theta),$$

where θ is the degree of skew.

We change no information except the starting point about blocks. This causes separation of blocks which have been located adjacent to each other in the original document before rotation. Therefore it is necessary to improve the quality of the rotated image by eliminating the overlap or separation of blocks. For this purpose we move some rotated blocks so as to maintain the adjacency relation of blocks that has existed before rotation.

3.3 Experimental results

We input some documents by G3 facsimile (7.7 lines/mm) and process these document on a work station (MC 68010, 10MHz). Table 1 shows experimental results of the skew detection.

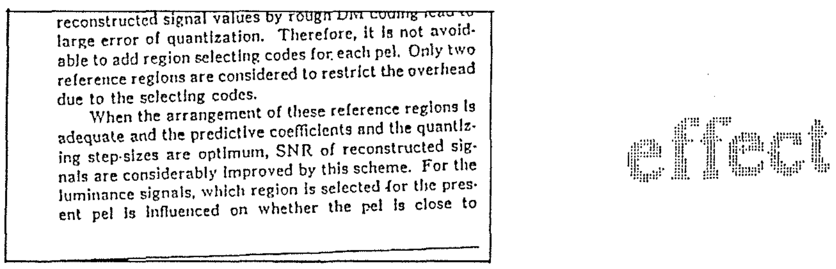
We measure skew values from the marked points on the baseline along the character strings. As shown in Table 1, the number of extracted character strings is greater than that of character string in inputed documents, which is due to the separation of extracted character strings. The average number of enclosed rectangles is greater than 35 for one character string. These values are sufficient to estimate skew value by the least squares method. As shown in Table 1, the difference between the detected and the measured skew value is less than 0.1 degree, which proves our procedure of detecting the skew value to be of good performance.

Our procedure deals only with enclosed rectangles for characters in the whole processing. Generally, the number of characters in ordinary documents of A4 size is less than the number of all pixels in the order 10^3. Therefore our method may be considered to be capable of performing skew detection more efficiently than the pixel by pixel method.

Table 1 Results of Skew Detection

Documents No.	Character Strings in Documents	Extracted Character Strings	Detected Skew Values (deg.)	Measured Skew Values (deg.)	Processing Time (sec.)
1	66	66	1.70	1.80	28.80
2	66	82	0.50	0.56	33.20
3	85	105	−0.80	−0.78	69.70
4	92	99	1.40	1.43	87.90
5	92	92	3.00	2.95	85.80
6	84	84	1.90	1.82	51.00

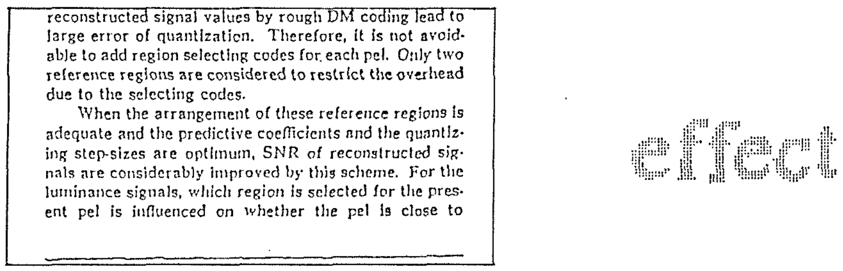
No. 6 document is Japanese paper and the others are English paper.
No. 1, 2 and No. 4, 5 are the same documents except for skewed values.



reconstructed signal values by rough DM coding lead to large error of quantization. Therefore, it is not avoidable to add region selecting codes for each pel. Only two reference regions are considered to restrict the overhead due to the selecting codes.

When the arrangement of these reference regions is adequate and the predictive coefficients and the quantizing step-sizes are optimum, SNR of reconstructed signals are considerably improved by this scheme. For the luminance signals, which region is selected for the present pel is influenced on whether the pel is close to

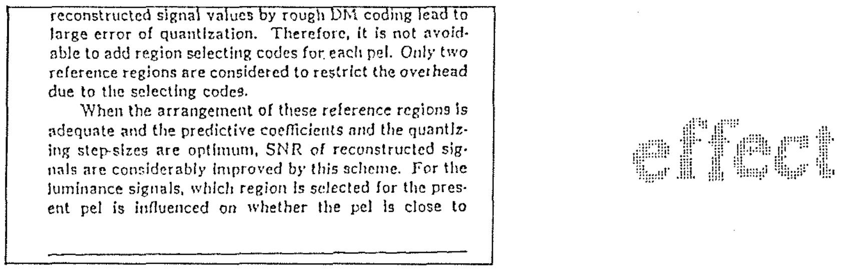
(a) Inputed image.



reconstructed signal values by rough DM coding lead to large error of quantization. Therefore, it is not avoidable to add region selecting codes for each pel. Only two reference regions are considered to restrict the overhead due to the selecting codes.

When the arrangement of these reference regions is adequate and the predictive coefficients and the quantizing step-sizes are optimum, SNR of reconstructed signals are considerably improved by this scheme. For the luminance signals, which region is selected for the present pel is influenced on whether the pel is close to

(b) Skew correction by our method.



reconstructed signal values by rough DM coding lead to large error of quantization. Therefore, it is not avoidable to add region selecting codes for each pel. Only two reference regions are considered to restrict the overhead due to the selecting codes.

When the arrangement of these reference regions is adequate and the predictive coefficients and the quantizing step-sizes are optimum, SNR of reconstructed signals are considerably improved by this scheme. For the luminance signals, which region is selected for the present pel is influenced on whether the pel is close to

(c) Skew correction by pixel by pixel method.

Fig. 2 Example of skew correction image.

Figure 2 shows a comparison of qualities of skew correction, where the inputed document has 1.7 degree skew. The qualities of skew correction of both the methods depend on skew values, so it is difficult to rank the qualities of both the methods generally, but as far as our experiment is concerned, both the methods may be regarded as having nearly the same qualities. The right-hand side image of Figure 2(a) has 3.0 degree skew. In this example skew images for some characters are corrected better by our method than by the pixel by pixel method.

The processing time for the skew correction by our method is one-fifth that of the pixel by pixel method. In ordinary documents, the number of blocks of SB code is one-tenth the total number of black pixels, so theoretically the processing time for the rotation of blocks is one-tenth that of the pixel by pixel rotation. However, our method provides some extra procedure of improving rotated image, so that its actual total processing time is equal to one-fifth that of the pixel by pixel method.

The method which we propose here adopts fixed threshold values in classifying character string rectangles, but we can predetermine these values independently of the kind of documents as,

$$\theta_1 = 3 \cdot R_h^f \text{ and } \theta_2 = 0.5 \cdot R_v^f$$

where R_h^f and R_v^f are defined in the next section.

4. Segmentation

In our segmentation method, basically blocks of SB code are enclosed by minimal-sized rectangles, which are in turn merged with other small-sized rectangles. In this procedure, to increase the processing efficiency we use the list structure of enclosed rectangles shown in Figure 3. In this structure, all rectangles which have the same scanning lines are linked together in the same list.

All procedures described later are constructed by the processes which are basically carried out in the order of scanning line or vice versa. Such a method makes the merging of rectangles efficient.

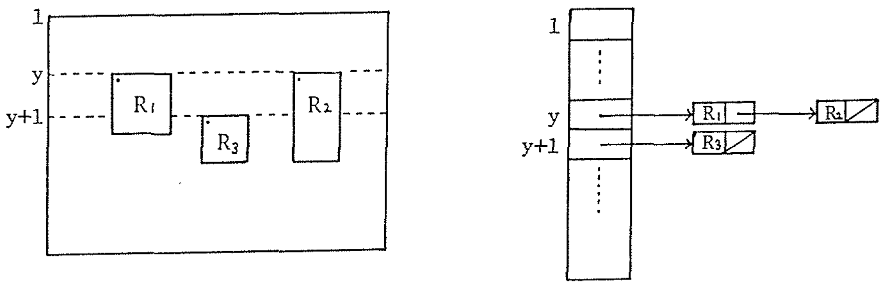


Fig. 3 List structure of rectangles.

4.1 Segmentation Algorithm

The segmentation algorithm is composed of the following procedures:

(1) Enclosing blocks by rectangles and merging of adjacent rectangles,
(2) Extraction of field separators and figure or table areas,
(3) Merging of character candidate rectangles,
(4) Merging of separated sentence rectangles,
(5) Re-extraction and merging of figure or table areas.

(1) Enclosing blocks by rectangles and merge of rectangles. We enclose all blocks of SB code by minimal size rectangles and record these rectangles in the list structure described above. Then we check the adjacency of blocks and enclose adjacent rectangles by new larger size rectangles. This procedure is carried out in the order of scanning lines and vice versa. In the second processing, we obtain the most frequent occurrence of horizontal and vertical lengths $R_h{}^f$ and $R_v{}^f$ of rectangles.

(2) Extraction of field separators and figure or table areas. We extract rectangles which satisfy the following conditions as field separators and figure or table areas. Let R_h and R_v be horizontal and vertical lengths of rectangles respectively:

(i) Condition of field separators

$$R_h/R_v > 50 \ or \ R_v/R_h > 50$$

(ii) Condition of figure or table area

$$R_h > 10 \cdot R_h{}^f \text{ and } R_v > 5 \cdot R_v{}^f$$

(3) Merging of character candidate rectangles. We classify rectangles which are not extracted in procedure (2) as character candidate rectangles and merge these rectangles by using the following conditions to obtain sentence rectangles;

(i) The case that the upper left corner of the rectangle to be merged is within 2/3 of the vertical width of already merged rectangles as shown in Fig. 4.

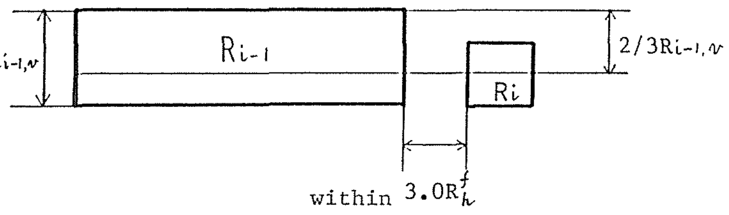


Fig. 4 Merging of object rectangles using this condition.

(ii) The other case is for 1/3 below with respect to already merged rectangles as shown in Fig. 5.

This procedure is carried out in the order of scanning lines, vice versa, and in the order of scanning lines. In the third processing we obtain the most frequent occurrence of horizontal and vertical lengths $SR_h{}^f$ and $SR_v{}^f$ of new rectagles. The reason for dividing the merge conditions into the two cases is to prevent merging of more than one character string into one rectangle

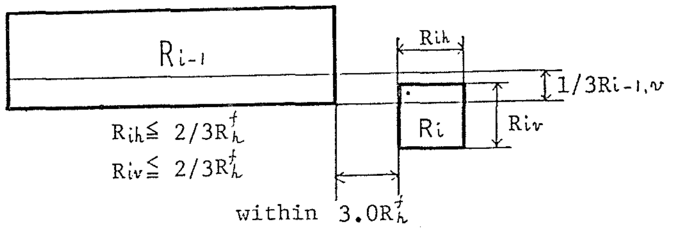


Fig. 5. Merging of object rectangles using this condition.

when line spaces between sentences are rather small. By this procedure almost all character strings are enclosed by rectangles, so that we call these rectangles sentence rectangles, which is abbreviated as SR.

(4) Merging of separated sentence rectangles. For a title in which characters are located far-apart, or a line in which there are larger spaces between character strings, sentence rectangles obtained in procedure (3) are separated. These rectangles are merged by using the condition shown in Figure 6.

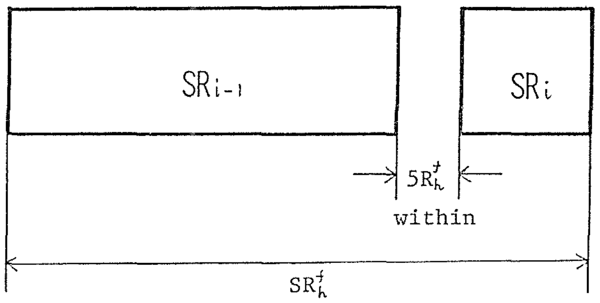


Fig. 6. Condition for merging separated sentence rectangles.

(5) Re-extraction and merging of figure or table areas. We re-extract figure or table areas which are not extracted in procedure (2) by comparing their size to most frequent occurrence of horizontal or vertical size of rectangles $SR_h{}^f$, $SR_v{}^f$. Let the sizes of the given rectangle be R_h and R_v, so we classify rectangles satisfying the following conditions as figure or table areas:

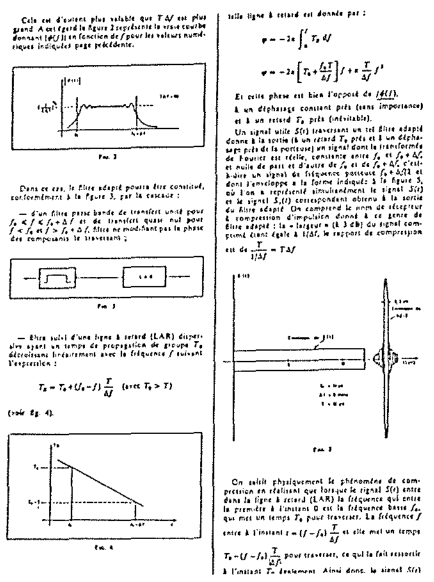
$$R_h/R_v < SR_h{}^f/SR_v{}^f$$

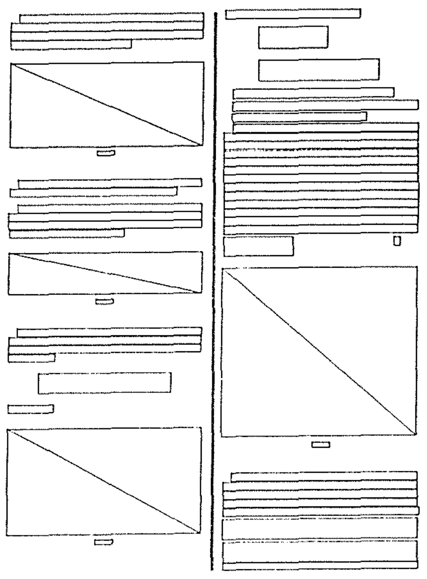
$$R_v > 3 \cdot SR_v{}^f$$

Furthermore, we merge rectangles which are contained in the imaginary area which is the extention of figure or table area with sizes of $R_h{}^f$ and $R_v{}^f$. We merge sentence rectangles which are located overlaping with the other sentence rectangles. This procedure is needed in such a case that some character strings of figure or table are extracted separately or that the numerator or denominator is extracted separately in formula sentence.

4.2 Experimental results

Documents are inputed into the work station (MC 68010, 10MHz) by facsimile G3 (7.7 lines/mm). Figure 7 shows an example of segmentation and Table 2 shows results of a segmentation for some documents.





(a) Inputed document (b) Result of segmentation

Fig. 7 Example of segmentation

Table 2 Results of Segmentation

Documents No.	Extracted Sentence Rect. (extracted/real)	Fig. or table Areas (extracted/real)	Field Separator (extracted/real)	Processing Time (sec)
1	65/ 63	2/2	0/0	156.0
2	83/ 83	1/1	0/0	210.0
3	128/126	0/0	1/1	318.0
4	87/ 88	1/1	0/0	303.1
5	69/ 69	2/2	1/1	180.3
6	77/ 81	1/1	0/1	144.0

4.3 Discussion of results

Our segmentation procedure extracts the caption of figures or the title of tables as character string areas. In number 6 document, we extract fewer sentence rectangles than real ones. This is due to extracting figure area containing captions. Also in number 6 document, a field separator is not extracted. This is due to touching of character strings into field separator. In other documents, sentence areas, table or figure areas and field separators are almost all extracted correctly.

In this segmentation procedure, most processing time is used in procedure (1), so the total required time is directly proportional to the number of blocks of SB coded documents. Generally, an A4 size document contains 5,000 to 50,000 blocks, so we can perform segmentation more efficiently than by using the pixel by pixel processing. Blocks of SB codes are reflected to some kinds of constitutional characteristics of documents. So when we use these characteristics in segmentation, we can obtain more precise or detailed segmentation results. This is the subject to future study.

References

1) K. Y. Wong, R. G. Casey, and F. M. Wahl, IBM J. Res. Dev. 22, No.6., 647 (1982).
2) M. Okamoto, K. Kawata, and M. Tamai, Trans. IECE, E78, 113 (1985).
3) O. Nakamura, M. Ujiie, N. Okamoto, and T. Minami, Trans. IECE, Vol. J67-D, 1277 (1984).
4) M. Hase and Y. Hoshino, Trans. IECE, Vol. J67-D, 1044 (1984).
5) T. Akiyama and I. Masuda, Trans. IECE, Vol. J77-D, 111 (1983).
6) T. Akiyama and I. Masuda, Trans. IECE, Vol. J69-D, 1187 (1986).
7) M. Okamoto, N. Takahashi, and K.Kawata, Trans. IECE, Vol.J69-D, 1075 (1986).
8) F. M. Wahl, K. Y. Wong, and R. G. Casey, Research Report RJ 3356, IBM Reseach Laboratory, San Jose, CA (1981).