2009 10th International Conference on Document Analysis and Recognition

Compression and String Matching Method for Printed Document Images

Hajime Imura and Yuzuru Tanaka
Department of Information Science and Technology Hokkaido University,
N-13, W-8, Sapporo, 060-8628, Japan
{hajime, tanaka}@meme.hokudai.ac.jp

Abstract

This paper describes a compression technique for printed document images and string matching method on the compressed images. To send digitized document images over the Web, compression of the document images is required. Moreover, in order to deal with historical letterpress printing collections, it is important to provide a full-text search method for them. The proposed compression scheme is based on character Pattern Matching & Substitution approach using a string matching technique of document images. The proposed string matching method is independent from the difference of languages and fonts because it uses the pseudo-coding that is based on statistical character shape features. We also use the pseudo-codes in a string matching of compressed documents. The system is as fast as the full-text search of machine-readable texts. Our method was evaluated in the compressed size, calculating recall-precision curves for n-gram-based query strings. The experiments have shown that about 100 pages of document in gray-scale at 300 dpi can be compressed down to around one megabyte.

1. Introduction

In recent years, a publication over the Web such as digital library is spreading widely. Some companies and libraries are promoting the electronic publishing of scanned document images over the Web. A distribution of a document image that consists of hundreds pages in a traditional image format such as JPEG or GIF terribly increases a network load. Thus, various image compression methods specialized in document images were proposed in order to reduce the network load.

In the document image compression, recent researches have shown that a Pattern Matching based approach works effectively. Document images generally have many occurrences of the same characters. The Pattern Matching-based approach compresses documents by collecting the same symbol characters using Image Pattern Matching, and encoding the collected characters effectively.

Howard *et al.* [5][4] have proposed image compression methods for bi-level and half-tone images that is employed by a "JBIG2" standard. These methods are lossy or lossless compression for images based on the Image Pattern Matching. For the lossy compression, they proposed Pattern Matching and Substitution (PM&S) methods. The PM&S method encodes the collected characters of the same symbol by substituting a representative character for the images of the same character.

LeCun and Haffner *et al.* [6][1][2] have reported an effectiveness of their compression method called "DjVu" for gray-scale images. The method separates a document image into three images: a foreground, a background, and a mask image. They achieved high compression rate by compressing each image using respective methods. The foreground image contains pixels of the text and drawings. It is compressed by wavelet-based compression method named IW44 (ISO/IEC 11544). The mask image is bi-level image for separating between the foreground and background. It is compressed by PM&S based method like JBIG2.

These methods have very high compression rate compared to the traditional one. Here, an aggressiveness and a sensitivity are important to achieve higher compression rate. The 'aggressiveness' means decreasing a false negative and the 'sensitivity' means decreasing a false positive. The requirement will be expensive on computational cost, especially in a language that has complex characters such like Chinese characters. To solve this problem, we propose two-staged Pattern Matching using pseudo-codes matching of Transmedia.

Another important requirement for publishing documents is a search method of the documents. Some publication sites also provide a search function to the documents by using Optical Character Recognition (OCR). However, their applications are limited to the documents that were published in modern ages. As the next stage, publication and distribution of old printed documents will be carried out by converting the document of the paper medium into elec-

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.182

tronic data. However, it is difficult to apply OCR to the old documents such as letterpress-printed documents or documents in early stages of phototypesetting spreading. This is because the old documents have various fonts that are different from modern fonts. The system of OCR depends on a language and a font. Therefore, OCR is expensive to prepare for unknown languages or fonts on learning process. It is important to provide a full-text search method of images that is font independent, and requires no *a priori* knowledge about particular language.

Tanaka and Torii proposed the string matching technique named "Transmedia" in 1988 based on pseudo-code for printed English documents [9]. Yusa and Tanaka [11] expanded the work in 1988 for printed Japanese documents. In Chinese printed documents, He *et al.* [3], and Lu and Tan [8] proposed a string matching technique using the codes based on character stroke density features. Our current study is based on the prior works of Tanaka *et al.*

In this paper, we propose a lossy compression method suitable for a document image keeping readability. The method does not depend on a particular language and a font. Moreover, the compressed document is searchable by applying string matching to pseudo-code encoded texts that are also independent from a language and a font.

2. Transmedia Engine

A proposed compression method for a document image is based on the string matching technique for document images named Transmedia Engine. The matching method treats each character as a pattern image, without recognizing each character as a character. We realized the fast full-text search to document images by using a pseudo-code that is generated from character shape features as an internal description of a document. We describe about the string matching technique.

As a preparation of images, we make pre-processing of a background removal and noise reduction to scanned images. By threshold process, we separate a pixel that constitutes characters from the image, and remove a background. Then, we convert the character pixels into an 8-bit gray-scale image.

Next, we segment a document image into rectangular areas of character. Because the segmentation is relatively easy in printed documents, we use a simple projection method. We segment the document into lines by projecting the character pixels on the orthogonal plane to the lines and segment each line into characters by projection on the plane along each line.

Next, we extract character shape features from each character images. After that, we apply a dimensionality reduction method to the feature vectors and generate pseudo-codes based on distribution of each reduced dimensional features. Finally, we assign the pseudo-code to each character. Then a text body of the document is described by pseudo-code sequence. Therefore, we can search the document images by comparing pseudo-codes with the same technique for a normal text documents.

Because the process extracts the pseudo-codes only from statistics of the target document images, we can save a cost of the operation such like a leaning process that should be prepared in advance of the search. To short, this method retrieves strings that are relatively similar to a query string in a whole target document. In the following, we describe details about the feature extraction and the pseudo-coding method.

2.1. Feature Extraction

We extract an image feature from each character image. We employed a GDF [10] as an image feature in this paper. The GDF is an image feature which calculates a gradient vector of a pixel value at each position of a target image, and generates features based on a distribution of the gradient vector. It is a shape feature of lines which constitutes a character.

2.2. Pseudo-coding

Our pseudo-code is generated by two processes: a dimensionality reduction and a quantization. The extracted feature in previous section is one high-dimensional real vector per each character image. We convert the high-dimensional features into pseudo-code described by the low-dimensional positive-integer value vector. This process enables us to reduce the amount of data descriptor for search and the computational complexity in the matching.

In the dimensionality reduction process, we reduce the dimensions of the features by Principal Component Analysis (PCA). First, we select some pages at random from the whole document and create reduced new basis vectors from the features of character in the pages by lower-order principal components of PCA. Next, we reduce the dimensions of the features in all the pages using the basis vectors. The number of a reduced dimension was determined by a criterion of 90% of an accumulated contribution rate.

In a following quantization process, we count up a distribution histogram for values in each dimension of the reduced dimensional feature vectors. Next, in each of the histogram, we divide the histogram into portions according to a contribution rate of each principal component. We set a large value as a number of the portions of a histogram that has a large contribution rate. Conversely, we set a small value as that with a small contribution rate. When the number of portions is n, the values of splitting point $(a_1, \cdots, a_{n-1})$ are determined by the formula 1. Here, $f(x)$

is a distribution of a value (x_d) of the d-th dimension in the reduced dimensional vector.

$$\int_{-\infty}^{a_1} f(x_d)dx_d = \int_{a_1}^{a_2} f(x_d)dx_d = \ldots = \int_{a_{n-1}}^{+\infty} f(x_d)dx_d \quad (1)$$

In short, we divide the each histogram into n portions with a constraint that each portion contains the same number of elements. According to the divided area where the dimensional value x_d is included, we assign the code of a positive-integer value. As a result, each character image is described by a low dimensional positive-integer vector. We call this pseudo-coding method the Scalar Quantization Coding (SQC) method. We determined the number of partitions $(32, 16, 8, 4, 4, \cdots, 4)$ from experiments in descending order of the contribution rate.

The pseudo-code sequence as an internal representation of a document enables us to process fast string matching. In searching, we give a query string by specifying string area in the target document images. By comparing a pseudo-code sequence of whole document to that of query, we obtain distances between a portion of the document and the query. As a search result, the system returns a ranked list of string areas ordered by the distance.

3. Document Image Compression

The proposed compression method is a lossy image-compression method for document images. The method compresses document images by collecting the image pattern of characters and by substituting a representative character for the character occurrences.

The document images in a traditional format such as JPEG or GIF generally contain redundant representations from an aspect of describing a document. It is because the traditional format stores all of the character images that appear in the document.

By the proposed method, we match a character pattern image using the string matching of Transmedia described in the previous section. We build a registration dictionary of the character pattern image, and describe an occurrence of the same character symbol by a common representative character image. As a result, we can reduce the character image patterns that should be stored.

We describe the procedure below. Here, each character of Transmedia contains three elements: a character image, a pseudo code, and character area in a page. We build a character group registration dictionary, reading characters in the document sequentially. The group dictionary is a list of character groups that pools character patterns appearing in the document.

In building the dictionary, first, we check the existence of current reading character in the dictionary. The checking procedure consists of two steps. The first step prunes candidate character groups by pseudo-code matching in the dictionary with a threshold distance. By a second step, we check the groups matched in the first step using a pattern matching on character image bitmaps. Then, if the pattern is already registered, we encode current character instance by a matched group ID, otherwise, we add a new character group into the dictionary and encode the character by the ID of the new group. Note that we maintain the dictionary over multiple pages.

In the JBIG2 and DjVu, they employed simple methods of direct comparing bitmaps as the Image Pattern Matching. The Pattern Matching should be aggressive (decreasing a false negative) but sensitive (decreasing a false positive) about difference between two character bitmaps. In particular, the false negative is not allowed because it breaks the readability of documents. The false negative reduces the compression rate. The requirement will be expensive on computational cost, especially in complex characters such like Chinese characters.

The pseudo-code matching in the first step accelerates the matching process by pruning the candidates that should be checked in the Image Pattern Matching. The pattern matching in the second step has the purpose of avoiding the false positive. Therefore, we must set up strict constraints on the algorithm of the image pattern matching. We employed the Pattern Matching method proposed by Liang *et al.* [7].

As a result of the process, the each group contains 1) a group ID, 2) a representative image of characters, 3) representative pseudo-code of characters in the group, and 4) a list of characters in the group. The processes make it possible to describe a document as a sequence of group IDs. For example, if given text is "abracadabra," a example of built dictionary will be like Table 1. Note that a group describing one character symbol may separate in multiple groups. We permit it in order to avoid substitution errors; for example, both Group IDs '1' and '5' in Table 1 describe the same character 'a'. We obtain the four types of data as a result of the compression process: 1) Group ID sequence of the text body, 2) Tiled character image table of representative characters 3) Character dictionary such like Table 1, 4) Coordinates of each character in the document.

Table 1. Character Dictionary

Group ID	Image	Pseudo-code	Char List
1	a	[3,0,3,1]	1, 4, 8
2	b	[5,3,1,3]	2, 9
3	r	[0,2,0,3]	3,10
4	c	[5,2,2,3]	5
5	a	[4,0,4,1]	6,11
6	d	[2,3,4,3]	7

4. String Matching on Compressed Document

We realize a string matching on compressed documents by using representative pseudo-codes of each character groups. The text body of document is described by a sequence of Group IDs. Then, we give a query from the sequence of text body, such as $[GID_1, GID_2, \cdots, GID_m]$. However, a group corresponding to the same character symbol will be divided into multiple groups. Therefore we must search the dictionary for candidate groups using representative pseudo-code of the query group. The matching in the dictionary is a pseudo-code matching with a threshold. The threshold is more relaxed compared with the one of the compression process. Then, we obtain multiple candidate groups from the each character of the query. We search the text body sequence for matched strings using a regular expression (shown in 2) generated from a combination of the candidates. We obtain a set of character groups $(GID_{1(1)}|\ldots|GID_{1(n_1)})$ as a candidates for the group in the query (GID_1). Although the expression also contains incorrect patterns, the incorrect combinations do not appear easily on a probabilistic model of a language. We can match the regular expression by a simple Deterministic Finite Automaton (DFA).

$$(GID_{1(1)}|\ldots|GID_{1(n_1)})\ldots(\ldots) \\ \ldots(GID_{m(1)}|\ldots|GID_{m(n_m)}) \quad (2)$$

Here m is a length of a query string, n_i is a number of candidates in i-th character, and $GID_{m(j)}$ denotes j-th candidate Group ID of m-th character.

5. Experimental Result

We evaluated our proposed method in the compressed size and calculating recall-precision curves for n-gram-based query strings. We used an English document set and a Japanese document set as the test set. These images are prepared by scanning from the paper documents printed on A4 paper at 300 dot par inch (dpi) quality. The English set is an English text that contains 115973 characters and 101 pages generated from 'alice29.txt' in the Canterbury Corpus (http://corpus.canterbury.ac.nz/). The Japanese set is a Japanese text that contains 160241 characters and 120 pages generated from 'Kokoro' of the novel of Japan by Soseki Natsume.

5.1. Compressed Data Size Evaluation

We evaluated the proposed method by comparing the compressed data size on each page. Two graphs of Figure 1 and 2 illustrate the size of compressed document of our method (labeled Total) and the size of Character Table

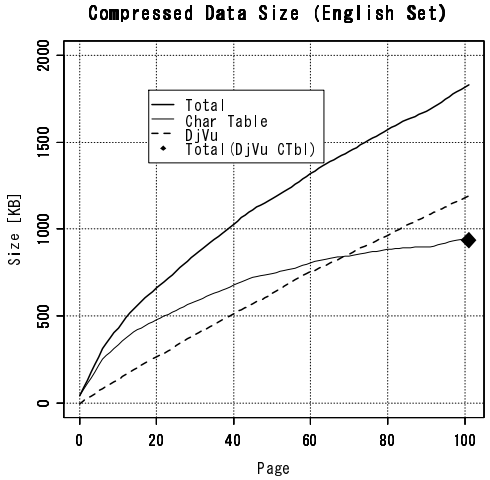

Figure 1. Compressed size of English Document.

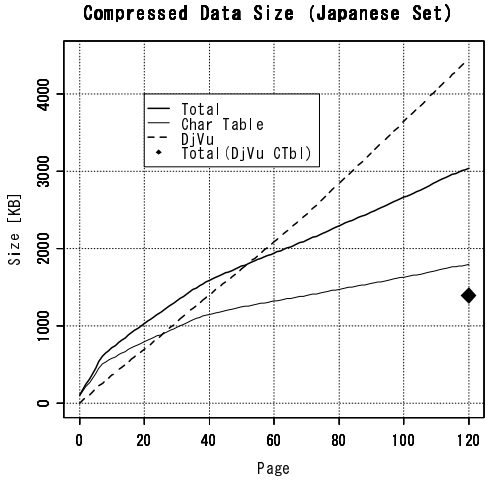

Figure 2. Compressed size of Japanese Document.

Image (labeled Char Table). In both figures, we also plotted the result of the DjVu method (labeled DjVu). Figure 1 illustrates the compression result in the English set and Figure 2 illustrates the Japanese one. From the graph of Japanese set, we can observe that the proposed method is superior to DjVu in the Japanese set. On the contrary, we can observe that the compression method of DjVu is superior in the English set.

However, by the proposed method, we can find that the image size of Character Table Image occupies a large size in the Total size. Since the Character Table Images were compressed using a general GIF format, if we compress it by the DjVu method, it is possible to improve compressed size. We plotted the results that were compressed using the DjVu format for Character Table Image (the points labeled DjVu

Table 2. Compressed Size

	proposal[KB]	DjVu[KB]	DjVuCTbl[KB]
Ja	3036	4449	1393
En	1827	1118	936

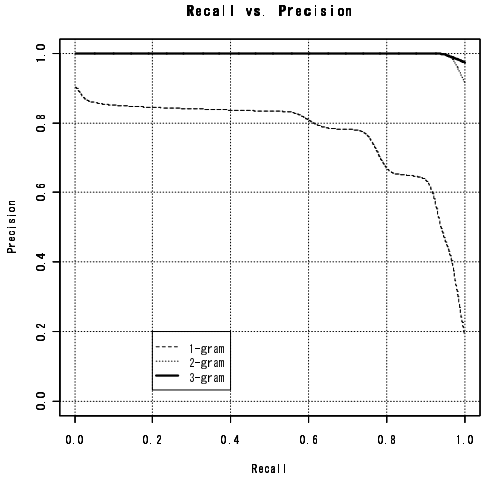

Figure 3. Recall-Precision Curves of Japanese Document.

CTbl). The two results show that we can further improve the compressed size by combining the proposed method and the DjVu method. All the results are shown in Table 2.

5.2. A Search Evaluation on the Compressed Document

In order to evaluate the accuracy of the proposed string matching on compressed document image, we depicted average recall-precision curves in each length of string.

We prepared a ground truth dataset using string matching on computer generated document images. The characters of the same symbol in the images perfectly correspond with each other. Therefore, we can obtain list of correct string occurrences for any query string. We collected the strings that appeared 50 times or more as the ground truth and we removed punctuations. These ground truth dataset contains sets of the same string occurrences based on n-gram. After that, we created a scanned document images that were fitted position to the computer generated documents. Then, we obtained the precisions of our method by searching the scanned document for each query of the ground truth dataset. Note that we used the 20 pages part of the Japanese document set.

As a result, in Figure 3, we depicted the average recall-precision curve in each length of string. From Figure 3, for the case of 2-gram or more, the retrieval precisions can be interpreted that it is enough to use in practical use because the precision ratios are 1.0 in almost all the recall ratio. Moreover, while a string length becomes long, we can find that the precision is converging.

6. Conclusion and Future Works

In this paper, we have proposed an image compression method for printed document images using a string matching technique based on character shape features. Our compressed image documents contain additional data for search. Moreover, the compression and the search method are independent from a particular language and a font. The experimental results show that our method achieves higher compression rate and accuracy on printed documents than conventional methods. Furthermore, by combining our method and DjVu method, we can further improve compression rate.

References

[1] P. Haffner, L. Bottou, P. Howard, P. Simard, Y. Bengio, and Y. LeCun. Browsing through high quality document images with djvu. In *Proc. IEEE International Forum on Research and Technology Advances in Digital Libraries, ADL'98*, pages 309–318, 1998.

[2] P. Haffner, L. Bottou, P. G. Howard, and Y. LeCun. Djvu: Analyzing and compressing scanned documents for internet distribution. In *Proc. ICDAR'99*, pages 625–628, 1999.

[3] Y. He, Z. Jiang, B. Liu, and H. Zhao. Content-based indexing and retrieval method of chinese document images. In *Proc. ICDAR'99*, pages 685–688, 1999.

[4] P. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. Rucklidge. The emerging jbig2 standard. *IEEE Trans. on Circuits and Systems for Video Technology*, vol.8(7):838–848, 1998.

[5] P. G. Howard. Lossless and lossy compression of text images by soft pattern matching. In *Proc. the Conference on Data Compression, DCC '96*, pages 210–219, 1996.

[6] Y. LeCun, L. Bottou, P. Haffner, and P. Howard. Djvu: a compression method for distributing scanned documents in color over the internet. In *Proc. 1998 IS&T/SID Color Imaging Conference*, pages 147–156, 1998.

[7] J. Liang, R. M. Haralick, and I. T. Phillips. Document image restoration using binary morphological filters. In *Proc. International Symposium on Electronic Imaging, SPIE'96*, pages 274–285, 1996.

[8] Y. Lu and C. L. Tan. Word spotting in chinese document images without layout analysis. In *Proc. the 16 th International Conference on Pattern Recognition, ICPR '02*, volume vol. 3, pages 57–60, 2002.

[9] Y. Tanaka and H. Torii. Transmedia machine and its keyword search over image texts. In *Proc. RIAO '88*, pages 248–258, March 1988.

[10] K. Terasawa and Y. Tanaka. Locality sensitive pseudo-code for document images. In *Proc. ICDAR '07, vol. 1*, pages 73–77, 2007.

[11] M. Yusa and Y. Tanaka. The extension of transmedia system for japanese text. In *Proc. the 49th National Convention of IPSJ*, volume 49, pages 217–218, 1994.