Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

A Novel Data Compression Scheme for Chinese Character Patterns

Jim Z. C. Lai and Wha-Chen Liaw

Dept. of Information Engineering and Computer Science
Feng-Chia University, Taichung, Taiwan 407
R. O. C.
Email address: lai@fcu.edu.tw
Fax: (886-4) 451-6101

Abstract

This paper presents an efficient lossless compression scheme for Chinese character patterns. The scheme analyzes the characteristics of line-segment structures of Chinese character patterns. A novel matching algorithm is developed for the line-segment prediction which is used in the encoding and decoding processes. The bit rates achieved with the proposed lossless scheme are 0.2653, 0.2448 and 0.2583, for three Chinese fonts, respectively. Due to the black and white points in Chinese character patterns are highly correlated, subsampling and interpolation schemes are considered to further increase the compression ratio. With these schemes, the low bit rates are achieved. Compared to the available algorithm, our approach gives the better performance. Two types of interpolation techniques are presented for the enlargement of Chinese character patterns. They are the 2-D interpolation and spline interpolation. Compared with the lossless compression scheme, the 2-D subsampling scheme can further reduce the bit rates as high as 43.19%, 41.83%, and 41.61% for three widely used Chinese fonts, respectively.

1. Introduction

Chinese characters are two-dimensional square symbols. There are 5401 commonly used characters plus 7650 frequently used characters installed in a Chinese system. As the printing quality is improved, huge storage is required for storing these characters. For example, for each font, about 6.5MB data storage is required for 13051 Chinese character patterns in a resolution of 64x64. Therefore, the compression of Chinese character patterns has been a very important task. Traditional work in the area of compression of Chinese characters mainly falls into two categories: the lossless compression technique, and the expanding/shrinking technique. A survey can be found in [1]-[3]. In reference [1], it is stated that a bit rate of 0.3 to 0.6 can be achieved by the 2-D prediction with Huffman coding. Ju[2] introduced a model including black/white pixel prediction to compress Chinese character patterns. Langdon and Rissanen [4] applied the finite-state machine model and arithmetic coding on black/white image compression. The model was based on a 10-pixel template and 1024 classes were generated. Tsay et al. [5] proposed a redundancy-gathering algorithm and applied arithmetic coding to encode Chinese character patterns. Their simulation results were good. The computational complexity is very high since a tree of huge nodes should be used. For a 12 pixel template, a tree of 2^{12} nodes will be used to set up the conditioning classes. It is noted that the above two methods use the statistical information to compress Chinese character patterns.

The 1-D Run-Length Coding Scheme (1-D RLCS) and 2-D Run-Length Coding Scheme (2-D RLCS) defined in CCITT Recommendation are capable of compressing bi-level images and are used as facsimile machine protocols [6,7]. For Chinese character patterns, these approaches, based on our experiences, may not be suitable for Chinese character compression. In this paper, we will use the characteristics of line-segment structures rather than the statistical information of Chinese character patterns to modify these methods such that they can be used more efficiently.

We also use subsampling techniques to further reduce the bit rates. We implement two interpolation schemes to restore a shrinked character patterns to its original size. The differences of black pixels between two

Chinese character patterns for different Chinese character systems may be up to 10% even though they have the same font and size. Thus the interpolation approaches used here are intended to reserve character shapes rather than their exact pixel locations in the process of character-pattern expansion.

This paper mainly consists of two parts. The first part describes an efficient lossless compression scheme called 2-D Run-Length Coding With Line-Segment Prediction (2-D RLCWLS). The second one develops subsampling schemes and the corresponding techniques of pattern expansion.

2. 2-D Run-length Coding with Line-segment Prediction

For the 2-D run-length coding scheme, it uses the vertical correlation of two consecutive lines (rows). The position of each changing point (pixel) on the coding row is coded with respect to the position of a reference point situated on either the coding line or the reference line. A changing point is defined as a point whose color (i.e., black or white) is different from that the previous point along the same line. It also contains the synchronization codeword end-of-line (EOL) and periodically inserts one dimensional modified Huffman code. The definitions and relative positions of changing points are illustrated through Fig. 1.

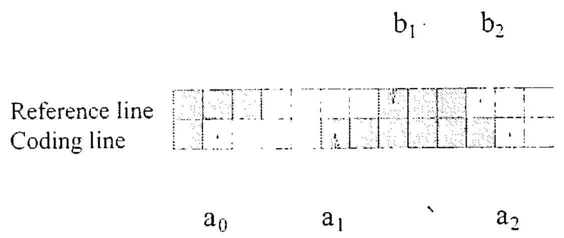


Fig. 1. The relative positions of changing points on the coding line and reference line.

In Fig. 1, a_0 is the current point on the coding line; a_1 is the next changing point to the right of a_0 on the coding line; a_2 is the next changing point to the right of a_1 on the coding line; b_1 is the first changing point on the reference line (upper line) to the right of a_0 and whose color is opposite to a_0; and b_2 is the next changing point to the right of b_1 on the reference line. To code the position of each changing point on the coding line, three kinds of coding modes are used in the traditional 2-D run-length coding scheme which is given as follows [7].

(1) If the position of b_2 lies to the left of a_1, then a pass mode is encoded and a_0 is reset to the point just under b_2 for the next coding.

(2) When the position of b_2 lies to the right of a_1 and the relative distance a_1b_1 in number of pixels is less than or equal to 3, a vertical mode and a codeword representing the relative distance a_1b_1 are encoded and the position of a_0 is reset to a_1.

(3) If neither a pass mode nor a vertical mode is classified, then a horizontal mode is encoded. In this case, both the run lengths a_0a_1 and a_1a_2 are encoded by the codewords $H+M(a_0a_1)+M(a_1a_2)$, where H is the flag codeword and $M(a_0a_1)$ and $M(a_1a_2)$ are codewords corresponding to the runs a_0a_1 and a_1a_2, respectively. After a horizontal mode coding, the position of a_0 is reset to a_2.

2.1 Line-segment prediction

The Chinese character pattern is a kind of binary image, so the 2-D run-length coding scheme in the CCITT Recommendation may be applied to it. However if this scheme is used to encode Chinese character patterns, a low compression may be resulted. A Chinese character is a two-dimensional pattern which contains various strokes. Each stroke consists of a cluster of black points, and the outside area is filled with white points. A stoke can be divided into many segments. In this paper, we analyze the characteristics of the line-segment structures rather than the statistics of Chinese characters to compress character patterns. Our proposed scheme, which is called 2-D Run-Length Coding With Line-Segment Prediction (2-D RLCWLS), is based on the prediction of the next changing point a_1. To increase the prediction accuracy and consider acceptable computational complexity, two rows are chosen as reference rows as shown in Fig. 2. These two references rows are denoted as RR(1) and RR(2), respectively.

The line segment information consists of the terminal points and the relationship between neighboring line segments for each line segment of a character pattern. If two line segments b_1b_2 and c_1c_2 as shown in Fig. 3 belong to a stroke of a Chinese character, then b_1b_2 and c_1c_2 should overlap each other, where b_1, b_2 and c_1, c_2 are terminal points of segments b_1b_2 and c_1c_2, respectively. In this case, points c_1 and c_2 are called the corresponding points of reference points $b_1$and b_2, respectively. If $\mathbf{b}_1\mathbf{b}_2$ and $\mathbf{c}_1\mathbf{c}_2$ do not overlap each other as shown in Fig. 4, then they do not belong to the same stroke and points $\mathbf{c}_1$ and $\mathbf{c}_2$ are not the corresponding points of $\mathbf{b}_1$and $\mathbf{b}_2$, respectively.

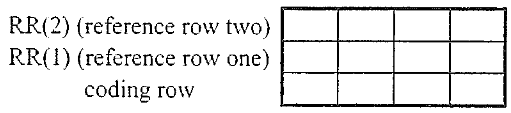


Fig. 2. Two references rows

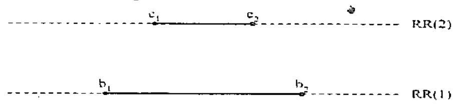


Fig. 3: An example of overlapping segments

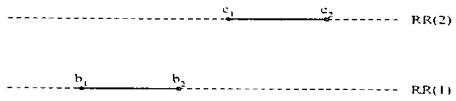


Fig. 4: An example of not overlapping segments

2.2 Position prediction

We use relationships between two segments on RR(1) and RR(2), respectively, to determine the corresponding point of a reference point. For a reference point $\mathbf{b}_1$, which is a terminal point of a segment on RR(1), the corresponding point on RR(2) cannot be simply determined by choosing the nearest terminal point on RR(2). Let's give two cases to explain why this does not work.

case(a) : Suppose that segments $\mathbf{b}_1\mathbf{b}_2$ on RR(1) and $\mathbf{c}_1\mathbf{c}_2$ on RR(2), as shown in Fig. 4, are used to determine the correspondence of $\mathbf{b}_1$. If the nearest point is selected to be a corresponding point, the correspondence of $\mathbf{b}_1$ will be $\mathbf{c}_1$. However this selection is wrong since segments $\mathbf{b}_1\mathbf{b}_2$ and $\mathbf{c}_1\mathbf{c}_2$ do not belong to the same stroke and they are incompatible.

case(b) : Suppose that $\mathbf{b}_1\mathbf{b}_2$ is on RR(1) and $\mathbf{c}_1\mathbf{c}_2$ and $\mathbf{d}_1\mathbf{d}_2$ are on RR(2) as shown in Fig. 5. In this case, $\mathbf{c}_1$ should correspond to $\mathbf{b}_1$. If the nearest point is selected as the correspondence, $\mathbf{d}_2$ will correspond to $\mathbf{b}_1$. However this correspondence is wrong since $\mathbf{d}_2$ is the right terminal point of $\mathbf{d}_1\mathbf{d}_2$ and $\mathbf{b}_1$ is the left terminal point of $\mathbf{b}_1\mathbf{b}_2$.

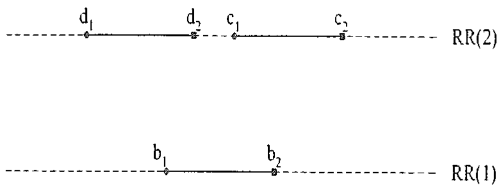


Fig. 5: An example of case(b)

The determination of corresponding point can be regarded as a matching process. Here a matching algorithm is presented. The method can be divided into two stages. In the first stage, we check whether two segments are compatible. Let $C(r,\mathbf{b}_1\mathbf{b}_2)$ describe the compatibility of the line segment r, on the RR(2), related to the line segment $\mathbf{b}_1\mathbf{b}_2$ on the RR(1). The definition of $C(r,\mathbf{b}_1\mathbf{b}_2)$ is given by

$$C(r,\mathbf{b}_1\mathbf{b}_2) = \begin{cases} 1, & \text{if } r \text{ and } \mathbf{b}_1\mathbf{b}_2 \text{ overlap each other;} \\ 0, & \text{otherwise.} \end{cases}$$

In the second stage, we use a flag to represent whether a terminal point is the left terminal point. This flag is used to avoid invalid matching. The cost $D(\mathbf{b}_1, w_i)$ between the reference point $\mathbf{b}_1$ and its candidate point w_i is determined by the following equation

$$D(\mathbf{b}_1, w_i) = | \mathbf{b}_1 - w_i | \qquad (1)$$

where $| \mathbf{b}_1 - w_i |$ is the distance between $\mathbf{b}_1$ and w_i.The matching procedures are now presented as follows.

Step 1: Determine the candidate set.

Let

$$\text{flag}(t) = \begin{cases} 1, & \text{if } t \text{ is a left terminal point;} \\ 0, & \text{if } t \text{ is a right terminal point.} \end{cases}$$

We first determine all the terminal points of the segments, which is compatible to $\mathbf{b}_1\mathbf{b}_2$, on RR(2). Find candidate points w_i from these compatible terminal points which have the same flag as flag($\mathbf{b}_1$), where $\mathbf{b}_1$ is a reference point. Let $W = \{w_1, w_2, ..., w_m\}$ be a candidate set, where m is the number of candidate points.

step 2: Find a correct corresponding point from a candidate set.

The corresponding point $\mathbf{CP}(\mathbf{b}_1)$ of $\mathbf{b}_1$ is selected from $\{w_i\}$ such that it is the closest point to $\mathbf{b}_1$. That is $\mathbf{CP}(\mathbf{b}_1)$ is determined by the following equation.

$$\mathbf{CP}(\mathbf{b}_1) = \min_{w_i} \mathrm{D}(\mathbf{b}_1, w_i) \qquad (2)$$

Once a corresponding point has been selected, we can determine the predicted position of $\mathbf{a}_1$. The predicted position of a_1 is denoted by $\mathbf{a}_1^*$ which is on the coding line (row). In the case that $C(r_i,\mathbf{b}_1\mathbf{b}_2) = 0$ for all line segments r_i on RR(2), then there is no compatible segment on RR(2). In this case we let $\mathbf{a}_1^*$ be the point just under $\mathbf{b}_1$. If $C(r,\mathbf{b}_1\mathbf{b}_2) = 1$, we first determine the intersection point, $i(\mathbf{b}_1)$, of the coding line and the straight line which passes through $\mathbf{CP}(\mathbf{b}_1)$ and $\mathbf{b}_1$. In the present case we let $\mathbf{a}_1^* = i(\mathbf{b}_1)$. The difference between $\mathbf{a}_1$ and $\mathbf{a}_1^*$ is used to determine the coding mode in the encoding procedure.

3. Encoding and Decoding Processes

Due to the resolution of a Chinese character pattern is known, a codeword EOL for each row is not necessary. When we reach the last line segment of a character pattern, we add an ending codeword to end the data stream for each character. Because codewords in the original CCITT two-dimensional code table are not suitable for the statistics of Chinese character patterns, we modify these codewords accordingly. The procedures of encoding a Chinese character pattern are now given as follows.

(1) Put a_0 before the first pixel on a new coding row.
(2) Detect a_1, b_1, and b_2.
(3) If $\mathbf{b}_2$ is on the left of $\mathbf{a}_1$, then add a codeword of pass mode and let $\mathbf{a}_0$ be the point just under $\mathbf{b}_2$ for next coding. Go to step (2).
(4) If $\mathbf{b}_2$ is not on the left of $\mathbf{a}_1$, determine $\mathbf{a}_1^*$.
(5) If $|\ \mathbf{a}_1 - \mathbf{a}_1^*\ | > 3$, then detect $\mathbf{a}_2$ and add codewords for horizontal mode coding. Let $\mathbf{a}_0 = \mathbf{a}_2$, and go to step (7).
(6) If $|\ \mathbf{a}_1 - \mathbf{a}_1^*\ | \le 3$, add codewords for vertical mode coding and let $a_0 = \mathbf{a}_1$ for next coding.
(7) If the current row is not ended, go to step (1). Otherwise go to step (8).
(8) If no line segment exists for all the following coding rows, add an ending codeword and finish the encoding process of the current character pattern. Otherwise go to step (1).

The procedures of decoding a Chinese character pattern are just the reverse processes of encoding and are presented below.

(1) Label relative changing points.
(2) Decode the coded data stream to get a codeword.
(3) If the current codeword is of pass mode. Let $\mathbf{a}_0$ be the point just under $\mathbf{b}_2$ and go to step (1).
(4) If the current codeword is of vertical mode, determine $\mathbf{a}_1$*, and obtain the difference between $\mathbf{a}_1$ and $\mathbf{a}_1$* through decoding the corresponding codeword. Set $\mathbf{a}_1 = \mathbf{a}_1$*+ the difference between $\mathbf{a}_1$ and $\mathbf{a}_1$*. Let $\mathbf{a}_0 = \mathbf{a}_1$ and go to step (1).
(5) If the current codeword is a horizontal mode codeword, perform 1-D run-length decoding. Set $\mathbf{a}_0 = \mathbf{a}_2$ and go to step (1).
(6) If the current codeword is an ending codeword, stop decoding.

3.1 Analysis of coding efficiency

The distributions for three kinds of coding modes are listed in Table 1. The probability of the vertical mode is the highest. The coding efficiencies are 0.97724, 098033, and 0.93370, for three Chinese fonts, respectively, as shown in Table 2. The coding efficiency is defined as the ratio of the entropy divided by the average length of the codeword. The experiment results of coding efficiency are listed in Table 2.

4. Subsampling and Interpolation

In the previous section, we have presented an efficient lossless compression scheme. We will investigate the subsampling technique to further increase the compression ratio in this section. The differences of black pixels between two Chinese character patterns for different Chinese character systems may be up to 10% even though they have the same font and size. Thus, in the upsampling process, we will reserve the shape rather the exact pixel locations of a character.

Table 1. Distribution in different modes.

	Formal-style	Running-style	Sung-style
Pass	185956	187459	169252
Vertical	4691194	4652336	5599068
Horizontal	376393	301334	339390

Table 2. The average length and coding efficiency for three fonts.

	Formal-style	Running-style	Sung-style
Entropy	1.98208	1.98917	1.63098
Huffman code	2.02822	2.02908	1.74679
Efficiency	0.97724	0.98033	0.93370

4.1 Two dimensional subsampling and interpolation

The patterns of black and white points for Chinese character patterns are not random, but highly correlated. The missed points may be recovered by efficient interpolation techniques. For the 2-D subsampling, an original pattern of size 64x64 is reduced to a pattern of size 32x32. Fig. 6 indicates the 2-D subsampling process. After this process, our proposed scheme described in section 3 is used to encode the reduced patterns.

Let $\{f(u,v);\ u,\ v = 0, 1, \ldots, 63\}$ denote an original pattern of size 64x64 and $\{F(U,V);\ U,\ V = 0, 1, \ldots, 31\}$ be the reduced pattern of size 32x32 obtained by subsampling $\{f(u,v)\}$. Their relations are

$$F(U,V) = f(2U,2V) \qquad (3)$$

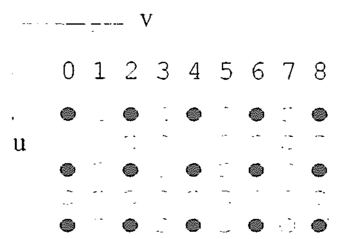


Fig. 6: The 2-D reduction process.

Many researchers have studied the enlargement of gray level images. However, the antialiasing techniques used for binary images should be different from those for gray level cases. A new approach for enlarging Chinese character patterns is developed here to reserve their shapes. Besides horizontal and vertical directions, the slant direction is also taken into consideration in the expanding process. This approach is especially suitable for Chinese character patterns.

The steps of character enlargement are summarized as follows.

(1) Decode the reduced pattern $F(U,V)$ of size 32x32.
(2) Expand the reduced pattern of size 32x32 into the enlarged pattern g(u,v) of size 64x64. Let
$g(u,v) = F(U,V)$
where $u = U \times 2$ and $v = V \times 2$.
(3) Interpolate the unknown $g(u,v)$ in the horizontal and vertical directions, respectively.

4.2 Horizontal and vertical interpolations

For the horizontal interpolation, $g(u,v+1)$ is determined by its two neighboring states (colors) $g(u,v)$ and $g(u,v+2)$. If $g(u,v)$ and $g(u,v+2)$ are the same, then $g(u,v+1) = g(u,v)$. On the contrary, if $g(u,v)$ is different from $g(u,v+2)$, then $(u,v+1)$ becomes an unknown point. For the unknown points, they are filled with white colors.

For vertical interpolation, we will consider the variations of corresponding segments. If ls_1 and us_1 overlap each other and are compatible as shown in Fig. 8(a), then we say ls_1 is a corresponding segment of us_1 or vice versa. We consider four cases of vertical interpolations as follows.

(a) For a given segment in the upper row (lower row), there is no corresponding segment in the lower row (upper row) as shown in Fig. 7.
(b) There is one segment in the lower row (upper row) corresponding to a given segment in the upper row (lower row) as shown in Fig. 8(a).
(c) There are two segments in the lower row (upper row) corresponding to a segment in the upper row (lower row) as shown in Fig. 8(b) or 8(c).
(d) There are more than two segments in the lower row (upper row) corresponding to a segment in the upper row (lower row) as shown in Fig. 8(d) or 8(e).

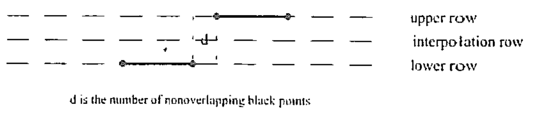
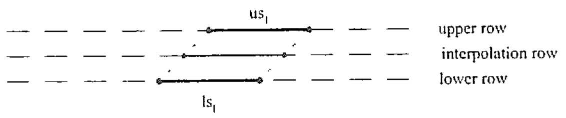


Fig. 7: No two segments overlap each other

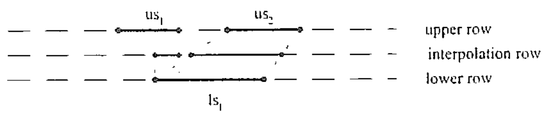
Two approaches will be used for vertical interpolation. They are the linear and zero-order interpolations. Let the left and right terminal points of a segment s be denoted as l(s) and r(s), respectively. In the case of linear interpolation, we determine the intersection of the interpolation row and the line passing through

$l(us_1)$ and $l(ls_1)$ as the left terminal point of the interpolated segment as shown in Fig. 8(a), where us_1 and ls_1 are segments on the upper row and lower row, respectively. By the same process, we can also obtain the right terminal point of the interpolated segment. After two terminal points are determined, the segment on the interpolation row is obtained accordingly. In the case of zero-order interpolation, the overlap of two segments on the upper row and lower row is determined as the interpolated segment on the interpolation row as shown in Fig. 8(b).

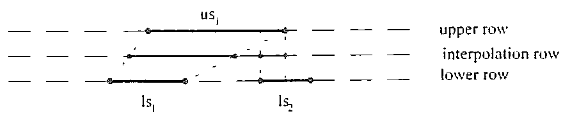
In case (a) of vertical interpolation, no interpolated segment is generated. In case (b), a segment ls_1 on the lower row has only one corresponding segment us_1 on the upper row. The interpolated black points are generated through the linear interpolation between the upper and lower segments as shown in Fig. 8(a). Case (c) is the situation that a segment has two corresponding segments as shown in Figs. 8(b) and 8(c). In this case, two pairs of corresponding segments, such as us_1 and ls_1, us_2 and ls_1 as shown in Fig. 8(b), exists. The interpolated points are generated by performing linear interpolation between a pair of segments which has the less cost and zero-order interpolation for the other pair. The cost D(r,s) between a pair of segments s and r is defined as follows.



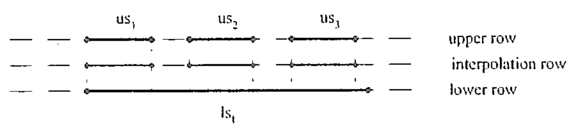
(a)



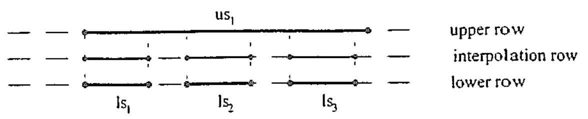
(b)



(c)



(d)



(e)

Fig. 8: The generation of interpolated points in the vertical direction. (a) one linear interpolation, (b) and (c) one linear interpolation and one zero-order interpolation, (d) and (e) three zero-order interpolations.

$$D(r,s) = |\,l(s) - l(r)\,| + |\,r(s) - r(r)\,| \tag{4}$$

Case (d) denotes the situation that there are three or more pairs of corresponding segments. In this case zero-order interpolation is performed for each pair of corresponding segment as shown in Figs. 8(d) and 8(e).

4.3 One dimensional subsampling and interpolation

In the process of 1-D subsampling, an original pattern of size 64x64 is reduced to a pattern of size 64x32. After this subsampling process, our proposed scheme described in the section 3 is used to encode the reduced patterns.

Let $\{f(u,v);\ u,\ v = 0,\ 1,\ \ldots,\ 63\}$ denote an original pattern of size 64x64 and $\{F(U,V);\ U = 0,\ 1,\ \ldots,\ 63,\ V = 0,\ 1,\ \ldots,\ 31\}$ be the reduced pattern of size 64x32 obtained by subsampling $\{f(u,v)\}$. Their relations are

$$F(U,V) = f(U,2V)$$

For the one dimensional pattern enlargement, a horizontal interpolation described in section 4.2 is used to restore a reduced character pattern to the original size.

4.4 Spline interpolation

The principle that is common to all interpolation schemes is to determine the parameters of a continue signal representation from a set of sample points [9]. A more refined approach is the cubic B-spline interpolation scheme. Fast algorithms for the calculation of the B-spline transforms have been proposed by M. Unser et al [10]. Using these algorithms, we can obtain better quality of character reconstruction.

5. Experimental Results

Our experiment was performed on an IBM-PC with programs written in C language. In the simulation, the test patterns include three Chinese fonts; every font contains 13051 Chinese characters in a resolution of 64x64 points(pixels).

A Chinese character is a two-dimensional pattern which contains various strokes. Each stroke consists of a cluster of black points. A stoke can be divided into many segments. The black/white ratios of the 13051 Chinese character patterns for three Chinese fonts in a resolution of 64x64 are given in Table 3.

Table 3. The black/white ratios of the original patterns.

	Formal-style	Running-style	Sung-style
Black/White Ratio	0.26/0.74	0.19/0.81	0.29/0.71

Table 4 shows the results using the UNIX command "Compress". Table 5 gives the bit rates using 2-D RLCS. The simulation results of 2-D RLCWLSP are shown in Table 6. The bit rates of applying 2-D subsampling and 1-D subsampling schemes are presented in Table 7 and Table 8, respectively.

Table 4. The bit rates using the UNIX command "Compress".

	Formal-style	Running-style	Sung-style
Bit rate	0.4349	0.4268	0.4561

Table 5. The bit rates of 2-D RLCS.

	Formal-style	Running-style	Sung-style
Bit rate	0.2810	0.2723	0.2686

Table 6. The bit rates of 2-D RLCWLSP scheme.

	Formal-style	Running-style	Sung-style
Bit rate	0.2653	0.2448	0.2583

Table 7. The bit rates of using 2-D subsampling and 2-D RLCWLSP scheme.

	Formal-style	Running-style	Sung-style
Bit rate	0.1507	0.1424	0.1508

Table 8. The bit rates of using 1-D subsampling and 2-D RLCWLSP scheme.

	Formal-style	Running-style	Sung-style
Bit rate	0.2259	0.2110	0.2304

Two character sets with large number of strokes in each font are given to show shat the proposed pattern interpolation techniques can preserve the shapes of Chinese characters during the restoration process. It is noted that complicated Chinese characters are more difficult to preserve the shapes during pattern enlargemnet. Figs. 9 to 11 show the original Chinese characters with different fonts in a resolution of 64x64, respectively. The expanded character patterns using three interpolation techniques are shown in Figs. 12 to 13. Every one of these figures contains three rows, character patterns on the upper row are obtained by using 2-D interpolation, character patterns on the middle row are obtained by using spline interpolation, and character patterns on the lower row are enlarged through 1-D interpolation. All characters are in a resolution of 64x64.

Comparing the results shown in Tables 4, 5, and 6, we can conclude that "Compress" is inefficient for compressing Chinese character patterns. The improvements of 2-D RLCWLSP scheme over 2-D RLCS are 5.59%, 7.65% and 3.74% for three fonts of Chinese characters, respectively. However, the improvements of 2-D RLCWLSP with 2-D subsampling over 2-D RLCS are in the range of 43.0% to 46.4%. The bit rates of using a redundancy-gathering algorithm with 10-pixel template and full search are shown in Table 9 [5]. Comparing Tables 7 and 9, we find that 2-D RLCWLSP with 2-D subsampling is still outperform the method used in reference [5].

Comparing Figs. 9 - 13, we can find that the three interpolation techniques presented in this paper preserve the pattern shapes during character enlargement. The 2-D interpolation method also gives the character patterns almost as beautiful as the original ones.

Table 9. The bit rates of using a redundancy-gathering algorithm with 10-pixel template and full search

	Formal-style	Running-style	Sung-style
Bit rate	0.1851	0.1841	0.1701

6. Conclusions

In this paper, we have proposed the two schemes for the coding of Chinese character patterns. A lossless compression scheme called 2-D run-length coding with line-segment prediction, is developed. A novel matching algorithm is presented for the line-segment prediction which is used in the encoding and decoding processes. For improving the compression ratio, subsampling and interpolation schemes are also developed. The improvements of bit-rate using 2-D RLCWLSP with 2-D subsampling over the traditional 2-D RLCS are in the range of 43.0% to 46.4%. Compared to the redundancy-gathering algorithm used in reference [5], our approach gives the better performance. As shown by experiments, the three interpolation techniques presented in this paper preserve the pattern shapes rather than the exact pixel locations during character enlargement. As shown in Figs. 9 - 13, the 2-D interpolation method gives the expanded character patterns almost as beautiful as the original ones.

References

[1] M. Nagao, "Data compression of Chinese character pattern", Proc. IEEE, vol. 68, pp. 818-829, 1980.
[2] R. H. Ju, I C. Jou, and M. K. Tsay, "Data compression techniques for Chinese character patterns", J. Chinese Institute of Engineers, vol. 14, no. 5, pp. 447-461, 1991.
[3] T. S. Liu, G. H. Chang, R. H. Ju, E. Hsieh, B. S. Jeng, "Data compression for Chinese character patterns", TL technical journal, vol. 16, no. 3, pp. 311-328, 1986.
[4] G. G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding", IEEE Trans. Commun., vol. COM-29, pp. 858-867, 1981.
[5] M. K. Tsay, C. H. Kuo, R. H. Ju, and M. K. Chiu, "Data compression on multifont Chinese character patterns", IEEE Trans. Image Processing, vol. 3, no. 2, pp. 139-146, 1994.
[6] International Telegraph and Telephone Consultative Committee Recommendation T.4, 1980.
[7] R. C. Gonzalez, R. E. Woods, Digital Image Processing, Addison Wesley, 1992.
[8] Y. Yasuda, "Overview of Digital Facsimile Coding Techniques in Japan", Proc. IEEE, vol. 68, no. 7, pp. 830-845, 1980.
[9] Peter Burger and Duncan Gillies, "Interactive computer graphics", Addison-Wesley Publishing Company, 1989.
[10] M. Unser, A. Aldroubi, M. Eden, "Fast B-spline transforms for continuous image representation and interpolation", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, no. 3, pp. 277-285, 1991.

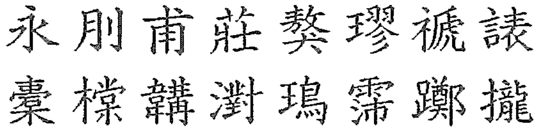


Fig. 9. The original characters with Formal-style font.

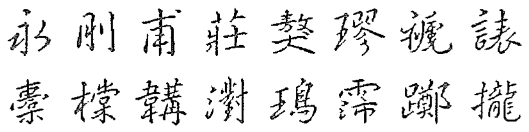


Fig. 10. The original characters with Running-style font.

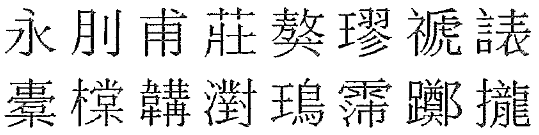


Fig. 11. The original characters with Sung-style font.

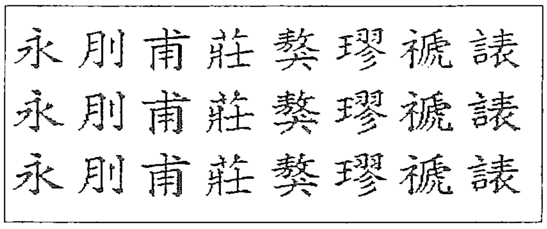


Fig. 12. The reconstructed characters with Formal-style font by different interpolation techniques : (upper row) 2-D interpolation, (middle row) spline interpolation, (lower row) 1-D interpolation.

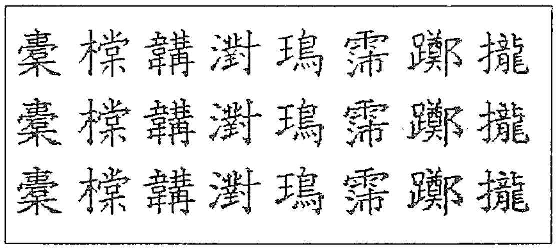


Fig. 13. The reconstructed characters with Formal-style font by different interpolation techniques : (upper row) 2-D interpolation, (middle row) spline interpolation, (lower row) 1-D interpolation.