A NEW CHAIN CODING BASED METHOD FOR BINARY IMAGE COMPRESSION AND RECONSTRUCTION

Saif Zahir and Kal Dhou
Image Processing, Graphics, and Multimedia Lab
3333 University way, CS, University of Northern British Columbia
Prince George, BC, **Canada**, V2N 4Z9
zahirs@unbc.ca

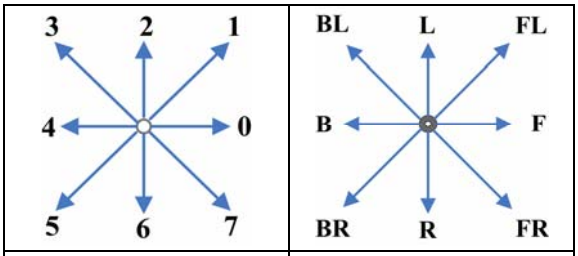
ABSTRACT

In this paper, we present a hybrid chain coding based scheme for contours and binary image compression and reconstruction. The proposed scheme comprises of a lossless and a lossy parts. This scheme is designed in such a way to generate extensive number of replicate links in the contours that can be assembled according to our (n10, 5) rule that can be highly compressed. Furthermore, for the lossy part, we introduce a new line processing technique to smooth the contours while maintaining high image quality. The experimental results show that the proposed method surpasses all published chain coding methods including FCC, DCC, DCC-8, VCC, CRCC, and L&Z. In addition, this scheme produced significantly higher compression ratio than WinZip, G3, G4, JBIG1, and JBIG2 standards.

Index Terms— Chain coding, image reconstruction, contour representation, binary image compressing

1. INTRODUCTION

In the eight connectivity encoding scheme introduced by Freeman [1], a link represents the direction between two points (pixels). These directions are numbered 0-to-7 as shown in Figure 1 (a). In order to digitally represent these directions, three bits are needed for each direction. Bons and Kegel [2] introduced the differential chain coding, DCC, method. This method exploits the correlation between two successive links via calculating the difference between the two links and assigning a variable length code to the result. Hwang, et al [3] improved the DCC method by narrowing the range of the results to one-half by employing the modulus of 8 to the result (i.e., D_i-D_{i-1} mod 8). The result is always positive, and within the range 0 to 7 instead of the range -7 to 7. Bribiesca [4] proposed a new method he called: vertical chain code, VCC that is suitable for shape representation. This method is based on the numbers of cell (link) vertices, which are in touch with the bounding contour of the shape. This method uses three elements 1, 2 and 3 to represent the bounding contour of a shape composed of pixels in the rectangular grid. He used, 1 for the outside corners, 2 to represent no change in the direction, and 3 for the inside corners. Yeh et_al [5] presented the Ideal-segmented Chain Coding (IsCC) method. They used the ideal 4-connected chains that can move in the right down / left up or left down / right up. Liu and Zalik [6] presented a new chain code method based on the eight-directions of Freeman code where each element in the chain is coded with a relative angle difference between the current and the previous direction and then they applied Huffman coding to compress the edges bit streams.



(a) Freeman code (b) Zahir & Dhou code

Figure 1 (a) Freeman chain code and (b) Z&D Code

2. THE PROPOSED HYBRID SCHEME

In this paper, we propose two lossless and lossy methods for compressing and reconstructing contours and binary images. A schematic diagram to show the steps of both methods is depicted in Figure 2. The following is a brief description of both methods:

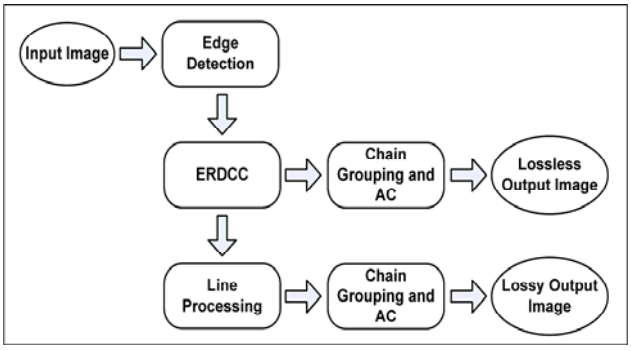


Figure 2 A schematic diagram of the proposed method

2.1 The Lossless Method

This method uses our enhanced relative directional chain of directions, ERDCC chain code provided in Figure 1 (b). It comprises of three main steps. The first step includes edge detection and the generation of

ERDCC; the second step is the grouping of the **Front** direction links in accordance to our (n10, 5) rule; and the third step is the encoding of the resultant chain sequence of directions using arithmetic coding.

Step 1: in this step, the first contour point (pixel) is located in a raster scan fashion and its coordinates are stored. Then the direction to the next point (i.e., link to the next pixel) is identified according to the eight-connectivity convention of Freeman rule. We call this first direction, **Front**. In our method, we have named the direction of the links to reflect their relative direction with respect to the current link, as shown in Figure 1 (b). The links are labeled as follows: Front (**F**), Front Right (**FR**), Front Left (**FL**), Right (**R**), Left (**L**), Back Right (**BR**), Back Left (**BL**) and Back (**B**). If a link has the same direction as the one before it based on Freeman rule then we name it **Front**. For example, a chain of links that has Freeman code 2, 2, 2 then it will be encoded as F, F, F. To elaborate further, in Freeman code, the *East* direction is always a zero (**0**), in our scheme; the *East* direction may take any of the eight directions (F, FR, FL, R, L, BR, BL, or B) depending on the link before it. To further explain, when we say a direction, we refer to Freeman rule. For example R R R is possible in our method, because the first direction may be East; the second is South; and the third is West. This is different from Freeman 7 7 7, which is South-East.

In order to determine the frequency of occurrence of each of our directions, we have experimented on a large number of contours and binary images. Table 1 shows the results of such probabilities for the ten image of Figure 3. These results illustrate that the directions BR, BL and B have near zero percent probability of occurrence in contours and edges and hence removed from the table (to save the space in the Table); followed by the FL and FR directions with less than 25% for each on average, and the R and L directions with less than 2%, each and finally the F direction with more than 55% on average. Based on the aforementioned results we have designed our scheme to maximally exploit such characteristics.

Table 1
The probability distribution of our directions
for the first ten images

Image	F	FL	FR	L	R
1	.451	.233	.284	.006	.025
2	.583	.170	.183	.031	.030
3	.547	.184	.212	.023	.032
4	.513	.236	.246	.002	.003
5	.441	.257	.275	.010	.015
6	.444	.225	.263	.030	.036
7	.561	.191	.200	.019	.026
8	.595	.169	.186	.023	.026
9	.448	.239	.252	.030	.029
10	.505	.213	.242	.019	.020
Average	**.509**	**.212**	**.234**	**.019**	**.024**

Step 2: in this step, we implement our (n10, 5) rule. This rule has been developed to ensure maximum exploitation of the sequence of the repeated **Front** directions (links) that are inherent in the nature of contours/edges winding, looping, and curving as illustrated in Table 1. The (10n, 5) rule works as follows: the method counts the number of consecutive **Front** directions in the chain and replaces each ten, 10, Front directions by an X and each five, 5, Front directions by a Y. It is evident that a multiple of tens is possible and allowed, hence the (n10), but multiples of five Fronts are not possible as two fives make an X; and therefore, we have the rule (n10, 5) that allow for multiple X directions and one Y. To further explain how the rule works, consider the following: (i) when the scheme encounters the first Front direction, it starts counting the number of consecutive Fronts, if applicable. If the number of Fronts is seven, the scheme will encode these links as YFF, a Y for five consecutive Fronts and two Fs for the remaining two Front directions; (ii) if the number of Fronts is 28, as another example, then it will be encode it as XXYFFF, (two Xs for twenty Fronts, one Y for five Fronts, and three Fs for the remaining three Fronts). In other words, it uses six characters to represent the 28 links instead of 28 Fs, which makes a great deal of reduction in the chain links to be encoded. As for the reason why we introduced only two new variables (X and Y) in our enhanced directional codes, it is of two folds: (a) the possibility of negative impact on compression outcome if we use arithmetic coding (i.e., more variables means more probabilities and hence more bits); (b) other directions are not likely to repeat consistently in a substantial amount that is feasible to ensemble them in groups as mentioned earlier. These results led to the choice of our compression method (i.e., arithmetic coding), which is our third step in the scheme.

Step 3: in this step, we implement arithmetic coding on the sequence of directions as we know their probabilities of occurrence. We have experimented with several lossless compression methods such as run length coding, LZW, Huffman and found that arithmetic coding produces much better results and is superior to Huffman coding used by other researchers such as in L&Z [6].

Figure 3 Sample of ten test images

2.2 The Lossy Method

The aim of this part is to maximize the number of repeat **Front** directions further so as to increase the compression ratio. This is done with one objective on mind: maintaining the contour/image quality. This method comprises of four steps as follows:

Step 1: this step is the same as in the lossless method.

Step 2: in this step, we introduced our "line processing" technique. This technique is realized in following way:

(i) For each contour chain code, we replace BR and BL directions by R and L respectively. This is done because the BR and BL directions have a minimal effect on the contour representation, if any, and because their removal will reduce the number of variables for arithmetic coding by 2. Doing so will increase the efficiency of the arithmetic encoder at a minimal or no cost. In addition, such replacements are indiscernible.

(ii) Identifying and smoothing "*stitches*" in the contours. A "*stitch*" is defined as a change in the direction of a contour by one pixel length. Figure 4 shows the four possible cases and their counterparts. For example, the chain of directions F **L R** F F in Figure 5 (a) has a "stitch" **L-R** because the L is the compliment/counterpart of R in our method. The four directions and their counterparts that create "stitches" are provided in Table 2.

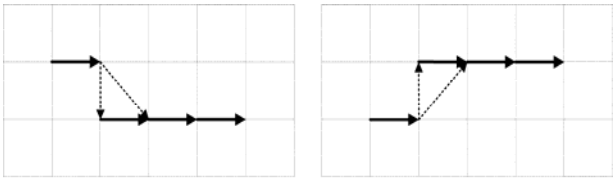
Table 2: Directions and their complements/counterparts

DIRECTION	COUNTERPARTS
R	L, FL
L	R, FR
FR	FL, L
FL	FR, R

For line processing, we consider only those stitches that have the lengths of their pre-stitch and post-stitch according to **a ≥ 2b rule**, where **b** is the number of directions before the stitch and **a** is the number of directions after the stitch.

This rule is introduced to offer a solution to some cases as follows: (1) the staircase phenomenon: contours and edges may loop; wind, or curve, hence smoothing repetitive stitches may flatten the curve and hence destroy the quality of the image or contour. To remedy this situation, we included the look-ahead feature in the scheme and restricted line smoothing to cases of a single level stitches only i.e., with no repetition; (2) the case of two complement *stitches* as depicted in Figure 5 (a): in such cases, we may have either a "trough" of one pixel deep or a "hump" of one pixel high. Both cases are smoothed in the same way as shown in Figure 5 (a, b, and c); (3) the loss of links: one significant advantage of the line smoothing is the possibility of reducing the number of directions (links) from the original sequence. These directions are limited to the L and R directions. For example, consider the chain in Figure 5 (a): the direction sequence for this contour is, F F **L R** F F F **FR FL** F. The total number of the links in this chain is nine (9) and it contains two stitches (L-R and FR-FL). These stitches are called complement stitches. When we smooth this contour as in Figure 5 (c), the chain of links becomes F F F F F F F F. The new number of links in the contour is eight (i.e., one link shorter) and they are all Fs that can be grouped using our rule. Please note that, loosing a link did not alter the length of the contour and hence connectivity of edges is maintained. In addition, such reduction in the number of directions increases the compression ratio.

Step 3: in this step, we implement our rule (n10, 5) as in the lossless scheme.



(a) (b)

Figure 4 The four cases for line processing

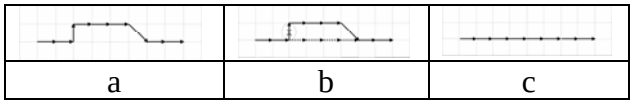


Figure 5 An example of a "hump" line processing

Step 4: Compress using arithmetic coding as in the lossless scheme.

3. EXPERIMENTAL RESULTS

In order to test the performance of the proposed scheme, we have implemented its lossless and lossy methods on a set of 46 binary images. The results of the first ten images of the database are shown in Tables 3 and 4. These results show the performance comparison between our scheme and DCC-8 [3], L&Z [6], in terms of the number of directions (chains links) needed to encode these images (compression).

In addition, we compare the compression ratio of the proposed scheme on the 46 images with WinZip, JBIG1, and JBIG2. We have tabulated the results of compression on the first ten images in Table 4, for space considerations only. The obtained results demonstrate that the proposed method surpassed all published methods. For example, the results of the ten images given in Table 3 show that the proposed lossless method is far better than all published results and it obtained a total of 49,497 links for the ten images while the nearest to it, the L&Z, obtained 59,977

links with 10, 480 links higher than ours. As for compression, the total number of bits of the compressed first 40 images using our lossless method is 255010 bits, whereas the number of bits for WinZip, JBIG1 and JBIG2 are (1,356,800), (457,128), and (450,208) bits, respectively. Such result represents a saving of 195,198 bits over the best-case compression method, (i.e., JBIG2). In conclusion, the proposed method offers more than 40% saving over JBIG2 for these images.

As for our lossless and the lossy methods performance comparison, we found that our lossy method offers, on average, nearly eight percent 8% higher compression than our lossless scheme while maintaining the quality of the contour/image near perfect. This is largely due to our proposed line smoothing scheme.

Table 3: Number of Chain Links of the first 10 images

Image	FCC	DCC8	L&Z [6]	Our Lossy	Our Lossless
Image 1	1999	1999	1963	**1597**	**1752**
Image 2	8923	8923	8666	**5556**	**6388**
Image 3	9728	9728	9424	**6506**	**7613**
Image 4	2584	2584	2548	**1671**	**2008**
Image 5	3791	3791	3747	**2812**	**3142**
Image 6	7819	7819	7481	**6458**	**7005**
Image 7	4474	4474	4348	**2967**	**3382**
Image 8	7788	7788	7483	**4610**	**5420**
Image 9	8942	8942	8550	**7236**	**7887**
Image 10	5881	5881	5767	**4384**	**4900**
Total	61929	61929	59977	**43797**	**49497**

Table 4. Number of bits of the first 10 images

Image	Size	WinZip	JBIG1	JBIG2	Z&D Lossless
Image 1	37746	16728	6496	6376	**3346**
Image 2	85301	43256	21616	23008	**14209**
Image 3	180090	78448	26944	25800	**16063**
Image 4	39006	22864	7592	7568	**3805**
Image 5	92928	31064	9760	9384	**6161**
Image 6	125870	63568	25240	27152	**14020**
Image 7	59718	31056	13160	14144	**7079**
Image 8	72941	46680	18088	19920	**11831**
Image 9	116112	67360	31832	29640	**15818**
Image 10	90360	43216	15032	13896	**9760**
Total	**900072**	**444240**	**175760**	**176888**	**102092**

Table 5. Mean value of the 46 images

	Size	JBIG 1	JBIG 2	Ours
Mean	130386.3	11428.2	11255.2	6375.25

4. CONCLUSIONS

In this paper, we have presented a new scheme for compressing and reconstructing contours and binary images. This method comprises of a lossless method that faithfully reconstructs the contours and a lossy method that produces near perfectly results. This scheme takes advantage of the repeated relative directions links that occur in a consecutive manner, which can be grouped together according to our rule (n10, 5). For the lossy method, we have introduced a line processing technique to smooth the edges and generate more Front directions that can be highly compressed. The experimentation results of this scheme show that the proposed method is superior to all published chain encoding methods such as FCC, DCC, DCC-8, VCC, IsCC, and L&Z. Furthermore, it surpassed binary image compression standards such as WinZip, JBIG1 and JBIG 2 by a significant percentage as shown in Table 5. The proposed method has a low complexity compared to JBIG family and it is simple to implement.

5. REFERENCES

[1] H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," *IRE Trans. Elec. Computer*, EC (10) 1961, pp. 260-268

[2] J. H. Bons and A. Kegel, "On the Digital Processing and Transmission of Handwriting and Sketching," *Proceedings of EUROCON 77*, 1977, pp. 880-890

[3] Y. T. Hwang, Y. C. Wang, and S. S. Wang, "An Efficient Shape Coding Scheme and its Codec Design," *IEEE Int. Conf. Signal Processing Systems*, 2001, pp. 225-232

[4] E. Bribiesca, "A new chain code," *Pattern Recognition*, vol. 32, no. 2, 1999, pp. 235-251

[5] M.C. Yeh, Y. L. Huang, J. S. Wang, "Scalable ideal-segmented chain coding," *ICIP* (1), 2002, pp. 197-200

[6] Y. K. Liu and B. Zalik., "An efficient chain code with Huffman coding," *Pattern Recognition*, vol. 38, no. 4, April 2005, pp. 553-557

[7] L. Zhou and S. Zahir, "A New Efficient Context-Based Relative-Directional Chain Coding," proceedings IEEE-*ISSPIT*, Vancouver, BC, August 2006, pp. 787-790

[8] B. J. Chen, "Binary Shape Coding Using Segment-Based Method," Masters Thesis, 2005

[9] S. C. Hermilo and R.- D. Ramon M., "Compressing bilevel images by means of a three-bit chain code," *Optical Engineering*, vol. 44, no. 9, 2005

[10] P. Nunes, F. Pereira, F. Marques, "Multi-grid chain coding of binary shapes," *ICIP*, Santa Barbara, CA, 1997, pp. 114-9

[11] http://www.cis.scu.edu.tw/~tseng/DataCompression/ Jan 25, (2007)

[12] http://www.eee.bham.ac.uk/woolleysi/All7/intro_3.htm Jan 20, (2007)

[13] M. Salem, A. Sewisy and, U. Elyan, "A Vertex Chain Code Approach for Image Recognition," *ICGST International Journal on Graphics, Vision and Image Processing*, vol. 5, no. 3, 2005

[14] P. Nunes, F. Marques, F. Pereira, and A. Gasull, "A contour-based approach to binary shape coding using a multiple grid chain code," *Signal Process, Image Commun*, vol. 15, 2000, pp. 585-599

[15] S. Hoque, K. Sirlantzis, Michael C. Fairhurst, "A New Chain-code Quantization Approach Enabling High Performance Handwriting Recognition based on Multi-Classifier Schemes," *ICDAR*, 2003, pp. 834-838.

[16] T. L. Chia, K. B. Wang, Z. Chen, and D. C. Lou, "A parallel algorithm for generating chain code of objects in binary images," *Journal of Information Science*, vol. 149 no. 4, 2003 , pp. 219-234