Evaluation and Comparison on the Techniques of Vertex Chain Codes

Linghua Li
College of Computer Science and Engineering, Dalian Nationalities University, Dalian, China
Email: linghl@139.com

Yining Liu
College of Communication Engineering, Jilin University, Changchun, China
Email: 1337687488@qq.com

Yongkui Liu
College of Computer Science and Engineering, Dalian Nationalities University, Dalian, China
Email: ykliu@dlnu.edu.cn

Borut Žalik
Computer Science, University of Maribor, Maribor, Slovenia
Email: borut.zalik@um.si

***Abstract*—This paper firstly describes the techniques of six representative vertex chain codes, they are: original vertex chain code, extended vertex chain code, variable-length vertex chain code, variable-length compressed vertex chain code, dynamic vertex chain code, and equal-length compressed vertex chain code. The description includes the main idea and encoding method of each vertex chain code. Then the chain length, namely the code numbers, the memory occupancy, namely the general binary bits, the code average length of each code, namely bits per code of each vertex chain code were compared respectively by large numbers of experiments. In the end, the evaluation and comparison were given from the view of chain code efficiency. The goal of the paper is to provide convenience and reference for the chain code researchers and users.**

***Index Terms*—chain code, vertex chain code, comparison, evaluation**

I. Introduction

Chain code has been a research topic for more than five decades. Since the pioneer work of Freeman in 1961, different approaches of chain coding have been proposed to improve the various aspects involved in chain code [1-5]. Because of the comprehensive applicability of chain code in many parts of pattern recognition and image processing [6-12], the techniques of chain code increase rapidly. Chain code is an efficient representation of binary images composed of contours [13-15]. The idea of a chain code is based on identifying and storing the directions from each pixel to its neighbor pixel on each contour. The technique of chain code includes two aspects: the chain codes based on pixel and the chain codes based on edge. For the former, some representative pixel-based chain codes include 8-direction Freeman chain code proposed by Freeman, 4-direction Freeman chain code commonly used by people, angle differences Freeman chain code (ADFCC) proposed by Y. K. Liu et al. [1], enhanced relative 8-direction Freeman chain code (ERDFCC) proposed by S. Zahir et al. [2], orthogonal three-direction Freeman chain code (3OT) proposed by H. Sánchez-Cruz et al. [14], arithmetic coding applied to 3OT chain code (Arith_3OT) proposed by H. Sánchez-Cruz et al. [3], and modified directional Freeman chain code in eight directions by a set of nine symbols (MDF9) proposed by H. Sánchez-Cruz et al. [4], etc. For the latter, the six vertex chain codes introduced in the paper are all edge-based chain codes. Viewing from the compression of the information, there are two kinds of compression: with loss of information and with lossless of information. The six vertex chain codes evaluated and compared in the paper are all lossless-compression chain codes.

II. Overview of Techniques of Vertex Chain Codes

We mainly describe the techniques of six representative vertex chain codes including the original vertex chain code (VCC), the extended vertex chain code (E_VCC), the variable-length vertex chain code (V_VCC), the variable-length compressed vertex chain code (VC_VCC), the dynamic vertex chain code (D_VCC), and the equal-length compressed vertex chain code (EC_VCC).

A. Original Vertex Chain Code (VCC)

In 1999, E. Bribiesca first introduced the original vertex chain code (VCC) [16]. The VCC is based on the numbers of cell vertices which are in touch with the bounding contour of the shape. This code determines the number of pixels of the binary shape that are in touch with the observed vertex of the shape's boundary contour.

doi:10.4304/jsw.7.12.2840-2848

The latter represents a connected sequence of edges and vertices on the border between the shape and its exterior.

In the VCC, the boundaries or contours of any discrete shape that are composed of regular cells can be represented by chains. Therefore, these chains represent closed boundaries. The minimum perimeter of closed boundary corresponds to the shape composed only of one cell. An element of a chain indicates the number of cell vertices, which are in touch with the bounding contour of the shape in that element position.

Figure 1 is an illustration of VCC which presents a shape. It is previous that there are only three numbers of 0, 1, and 2 which are needed to present a boundary composed with pixels of quadrate grids.

In order to digitally represent these numbers of cell vertices, two bits are needed for each number. The element 1, 2, 3 of VCC can be represented by their 2-bit binary equivalents, as shown in Table I.

TABLE I.

ENCODING OF THE ELEMENT OF ORIGINAL VCC

VCC	1	2	3
Binary code	01	10	11

As an illustration, consider the shape of Figure 1, when starting at the left-top point and walking along the boundary in a counter-clockwise direction, the VCC of the contour is

1 2 3 1 2 2 1 2 2 2 1 3 1 2 2 1 3 1 2 2,

as shown in Figure 1, or in binary form,

01 10 11 01 10 10 01 10 10 10 01 11 01 10 10 01 11 01 10 10.

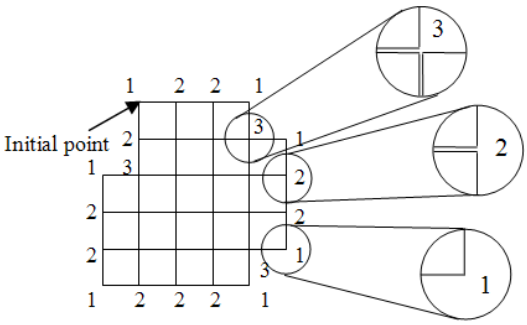

Figure 1. Obtaining the original VCC from the shape.

As we can see from the above, the numbers of chain code of VCC for the shape of Figure 1 are 20, and the numbers of binary bit of it are 40.

The VCC is invariant under translation and rotation. Using this concept of chain code it is possible to relate the chain length to the contact perimeter, which corresponds to the sum of the boundaries of neighboring cells of the shape; also, to relate the chain nodes to the contact vertices, which correspond to the vertices of neighboring cells. So, in this way, these relations among the chain and the characteristics of interior of the shape allow you to obtain interesting properties [16].

Since the VCC was proposed, it has obtained a lot of applications. For example, in [17], VCC was used for image recognition. In [18], VCC was used in neural network corner detection. In [19], VCC was used in skew detection for form document. In [20], VCC was used for calculating the compact and posture ratio of an image region.

B. Extended Vertex Chain Code (E_VCC)

Considering the compression efficiency of chain code, in 2007, Y.K. Liu et al. proposed extended vertex chain code (E_VCC) which is developed based on the original VCC [21]. Considering that the original VCC uses 2 bits to represent only three code elements 1, 2, and 3, E_VCC is introduced to add an element 0 without increasing the average bits per code. In E_VCC, the element "0" is added according to the experiments which show that the combination of element 1 and 3 is the most often occurring combination. Then the combination of element 1 and 3 is substituted by code 0.

In order to digitally represent these codes, two bits are needed for each code, and the binary bits of each code are not increased according to the original VCC. Table II shows the code relationship between E_VCC and original VCC and the binary form of E_VCC which consists of four code symbols.

TABLE II.

RELATIONSHIP BETWEEN E_VCC AND ORIGINAL VCC

E_VCC	0	1	2	3
VCC	1 and 3	1	2	3
Binary of E_VCC	00	01	10	11

As an illustration, consider the previous shape of Figure 1, when starting at the left-top point and walking along the boundary in a counter-clockwise direction, the E_VCC of the contour is

1 2 3 1 2 2 1 2 2 2 0 1 2 2 0 1 2 2,

as shown in Figure 2, or in binary form,

01 10 11 01 10 10 01 10 10 10 00 01 10 10 00 01 10 10.

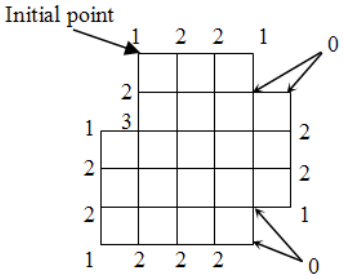

Figure 2. Obtaining the E_VCC from the shape.

As we can see from the above, the numbers of chain code of E_VCC for the shape of Figure 1 are 18 which are less than the original VCC, and the numbers of binary bit of it are 36 for the same condition.

From the example it is clear that the length of the E_VCC is less than the original VCC because of the two code symbols in original VCC are substituted by one code symbol in E_VCC but with the same symbol length with the E_VCC.

C. Variable-Length Vertex Chain Code (V_VCC)

Reference [21] also proposed a new vertex chain code named variable-length vertex chain code (V_VCC) which

is also developed based on the original VCC. The V_VCC also uses the codes of VCC which has three elements 1, 2, and 3. The difference between them is that the definition of the V_VCC considers the probability of three codes occurring. The experiments showed that the probability of code 2 is greater than the other two. Therefore, in V_VCC, the one bit binary digit 0 is used to represent for code 2, the two bits binary digit 10 and 11 is used to represent respectively for code 1 and code 3 which is with variable-length coding by applying the concept of Huffman coding concept.

Table III shows the code relationship between V_VCC and original VCC and their binary form of each code symbol are also listed.

TABLE III.

RELATIONSHIP BETWEEN V_VCC AND ORIGINAL VCC

V_VCC	1	2	3
VCC	1	2	3
Binary of VCC	01	10	11
Binary of V_VCC	10	0	11

As an illustration, consider the previous shape of Figure 1, when starting at the left-top point and walking along the boundary in a counter-clockwise direction, the V_VCC of the contour is the same as the original VCC, it is

1 2 3 1 2 2 1 2 2 2 1 3 1 2 2 1 3 1 2 2,

as shown in Figure 1, but the binary form of the V_VCC (shown in figure 3) is different from the original VCC, it is

10 0 11 10 0 0 10 0 0 0 10 11 10 0 0 10 11 10 0 0.

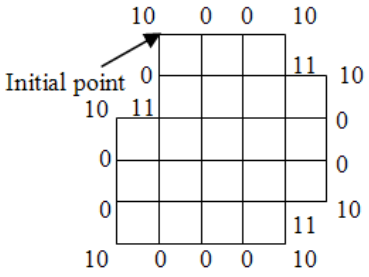

Figure 3. Obtaining the V_VCC from the shape.

As we can see from the above, the numbers of chain code of V_VCC for the shape of Figure 1 are 20 which are as same as the original VCC, and the numbers of binary bit of it are 30 because that the numbers of binary bit for code 2 are decreased.

From the example it is clear that the function of the V_VCC is the same as the original VCC, but the length of the V_VCC is less than the original VCC.

D. Variable-Length Compressed Vertex Chain Code (VC_VCC)

Reference [21] also proposed a third new vertex chain code named variable-length compressed vertex chain code (VC_VCC) which is developed based on the original VCC and Huffman coding concept by considering the different probabilities of the codes. The VC_VCC is constructing by taking account the E_VCC and V_VCC and it consists of five codes: Code 1, Code 2, Code 3, Code 4 and Code 5. The two new codes added for representing respectively for two combinations, one is the combination with first code 1 and second code 3, the other is the combination with first code 3 and second code 1. The probabilities of the codes were obtained experimentally [21]. From the experiments, it gets that the probability of the two combinations are equal. According to the statistical probabilities, the binary Huffman codes 0, 10, 110, 1110, and 1111 are assigned to the code values of Code 1, Code 2, Code 3, Code 4 and Code 5 respectively. Table IV shows the relationship between VC_VCC and original VCC and the probability and binary form of each code.

TABLE IV.

RELATIONSHIP BETWEEN VC_VCC AND ORIGINAL VCC

VC_VCC	Code 1	Code 2	Code 3	Code 4	Code 5
VCC	2	1 and 3	3 and 1	1	3
Probability	0.657	0.138	0.138	0.034	0.033
Binary of VC_VCC	0	10	110	1110	1111

As an illustration, consider the previous shape of Figure 1, when starting at the left-top point and walking along the boundary in a counter-clockwise direction, the VC_VCC of the contour is

c4 c1 c3 c1 c1 c4 c1 c1 c1 c2 c4 c1 c1 c2 c4 c1 c1,

as shown in Figure 4, where the c1, c2, c3, c4 and c5 are the respective abbreviations for Code 1, Code 2, Code 3, Code 4 and Code 5, or in binary form

1110 0 110 0 0 1110 0 0 0 10 1110 0 0 10 1110 0 0.

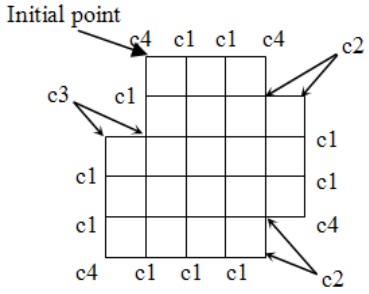

Figure 4. Obtaining the VC_VCC from the shape.

As we can see from the above, the numbers of chain code of VC_VCC for the shape of Figure 1 are 17 which are the least one in the vertex chain codes among the above, and the numbers of binary bit of it are 33 which are not the least among the above for the reason that the advantage of Huffman coding relies in the conditions where the length of the shape contours which will be described is more longer. Namely, the more the length of the shape, the less the numbers of binary bit required by VC_VCC compared to the chain codes without Huffman coding.

Within all of the vertex chain codes described above, the result of the comparison shows that the VC_VCC is the most efficient [21]. Up to now, VC_VCC remains one of the most efficient to compress binary objects. In [22], VC_VCC was further compressed by applying the concept of segment on it.

E. Dynamic Vertex Chain code

To decrease the chain length of the vertex chain code, in 2010, G. F. Yu et al. proposed dynamic vertex chain code (D_VCC) which is developed based on the original VCC [23]. The main idea of D_VCC is to change the means of original VCC in which each code expresses the numbers of cell vertices, and assigns new code value for the elements of chain code. There are ten elements in D_VCC, from the decimal number 0 to 9. The decimal number 0 (or 9) represents code "1" of original VCC, the decimal number 9 (or 0) represents code "3" of original VCC, and the decimal number 1 to 8 represent code "2" and the numbers of sequential code "2" directly. For instance, the decimal number 1 represents the one code "2", the decimal number 2 represents the two sequential codes "22" of VCC, and so on. Table V shows the relationship between D_VCC and original VCC.

TABLE V.

RELATIONSHIP BETWEEN D_VCC AND ORIGINAL VCC

D_VCC	VCC
0	1 or 3
1	2
2	22
3	222
4	2222
5	22222
6	222222
7	2222222
8	22222222
9	3 or 1

As an illustration, consider the shape of Figure 5, when starting at the left-top point and walking along the boundary in a counter-clockwise direction, the VCC of the contour is

1 2 3 1 3 1 1 3 2 2 2 2 2 1 1 3 2 2 2 2 2 2 2 1 1 2 3 1 3 1 3 1 3 3 2 2 2 2 1 2 1 2 3 1 1 2 3 3 2 1 3 1 2 2,

as shown in Figure 5, or in binary form,

01 10 11 01 11 01 01 11 10 10 10 10 10 10 01 01 11 10 10 10 10 10 10 01 01 10 11 01 11 01 11 01 11 11 10 10 10 10 01 10 01 10 11 01 01 10 11 11 10 01 11 01 10 10.

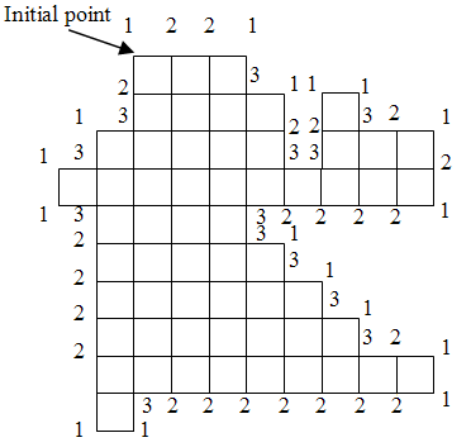

Figure 5. Obtaining the VCC from the shape.

In the same conditions, when expressing with D_VCC, the chain code of the contour is

0 1 9 0 9 0 0 9 5 0 0 9 7 0 0 2 9 0 9 0 9 0 9 9 4 0 1 0 1 9 0 0 1 9 9 1 0 9 0 2,

as shown in Figure 6, where the code "0" and "9" of D_VCC represent the code "1" and "3" of VCC respectively. When expressing in binary form, it is

0000 0001 1001 0000 1001 0000 0000 1001 0101 0000 0000 1001 0111 0000 0000 0010 1001 0000 1001 0000 1001 0000 1001 1001 0100 0000 0001 0000 0001 1001 0000 0000 0001 1001 1001 0001 0000 1001 0000 0010.

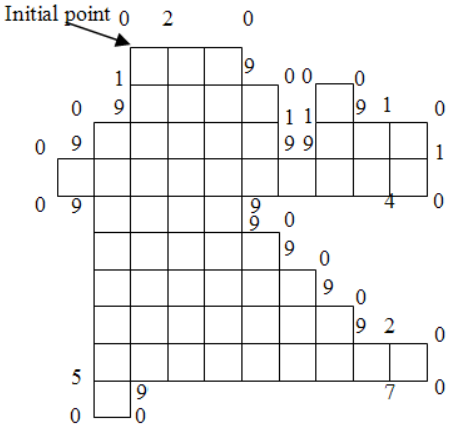

Figure 6. Obtaining the D_VCC from the shape.

As we can see from the above, the numbers of chain code of D_VCC for the shape of Figure 5 are 40 which are less than the original VCC with 54 for the reason that the codes of D_VCC apply the method of arithmetic coding, but the numbers of binary bit of it are 160 which are more than the original VCC with 108 because of the coding for the codes of D_VCC are also with equal length. Moreover, from the above description, we can see that D_VCC has ten codes but uses four binary bits for each code. It is obviously that four binary bits can encode for sixteen codes at most, it generates redundancy among the rest.

So, it is obviously that the ultimate goal of D_VCC is to reduce the general numbers of chain code by overlooking the memory capability occupied by the chain code. In other word, the general numbers of binary bits are increased largely.

F. Equal-Length Compressed Vertex Chain Code (EC_VCC)

To further decrease the chain length of the vertex chain code, in [23], another vertex chain code named equal-length compressed vertex chain code (EC_VCC) was proposed, which is developed based on E_VCC. The main idea of EC_VCC is to replace the minimum unit bit of other vertex chain code with byte. EC_VCC utilizes one byte (eight binary bits) for one code, and it divides the eight bits into two parts: two high bits and six low bits. The encodings of two high bits represent for four code values of EC_VCC, and the encodings of six low bits represent for the numbers of sequential codes indicated by the two high bits. For the six binary bits can encode

for 64 codes at most, if the numbers of sequential codes exceed 64, there will start a new byte.

Table VI shows the relationship between EC_VCC and original VCC. The four codes of the two high bits of a byte are “00”, “01”, “10”, and “11”, which represent code 0, 1, 2, 3 of E_VCC respectively. The six low bits are indicated by “b5b4b3b2b1b0”, in which “b” means bit and the number from 5 to 0 indicates the location of each bit of the byte. The binary code of the six low bits is from “00000” to “1111111”, which represents the decimal value from 0 to 63. So the decimal number 0 indicates the number of sequential codes is one, the decimal number 1 indicates the numbers of sequential codes are two, and so on, until the decimal number 63 indicates the numbers of sequential codes are 64. When the numbers of sequential codes exceed 64, there will start a new byte.

TABLE VI.

RELATIONSHIP BETWEEN EC_VCC AND ORIGINAL VCC

EC_VCC (eight binary bits)	E_VCC
00 b5b4b3b2b1b0	Code 0 and its sequential number/numbers
01b5b4b3b2b1b0	Code 1 and its sequential number/numbers
10 b5b4b3b2b1b0	Code 2 and its sequential number/numbers
11 b5b4b3b2b1b0	Code 3 and its sequential number/numbers

As an illustration, consider the shape of Figure 5, in the same conditions, when encoding by EC_VCC, the binary form of the chain code is

01000000 10000000 11000000 00000000 01000001 11000000 10000100 01000001 11000000 10000110 01000001 10000000 11000000 00000010 11000000 10000011 01000000 10000000 01000000 10000000 11000000 01000001 10000000 11000001 10000000 00000000 01000000 10000001.

As we can see from the above, the numbers of chain code of EC_VCC for the shape of Figure 5 are 28 which are less than the original VCC with 54 for the same reason with D_VCC that the codes of them apply the method of arithmetic coding, but the numbers of binary bit of it are 224 which are more than the D_VCC with 160 and the original VCC with 108 because of the same reason with D_VCC that the coding for the codes of them are also with equal length. Moreover, from the above description, we can see that EC_VCC uses 64 codes to represent code "1" of VCC which only needs two code in deed because the sequential numbers of code "1" are only two at most. It is obviously that the code values of EC_VCC generate more redundancy than the D_VCC.

It is obviously that the ultimate goal of EC_VCC is also to reduce the general numbers of chain code by overlooking the memory capability occupied by the chain code, which is as same as D_VCC. Each code of EC_VCC has eight binary bits, and the whole numbers of codes in EC_VCC are 256.

III. COMPARISON AND EVALUATION

A. Method of Evaluation for Chain Codes

In [21], a method for evaluating the efficiency of chain codes is proposed. The efficiency E of the chain code was defined as:

$$E = C / L \quad (1)$$

where C is the average expression ability per code, and L represents the average number of bits per code (the average code length). In other words, the efficiency of a chain code is proportional to the average expression ability per code, and with inverse ratio to the average number of bits per code. It means the average length of contours which represented by each binary bit.

The expression ability per code C refers to the average length of contours (or digital curves) which can be represented by every code of the chain code (measured in pixel units). When a code represents the relationship between two edge-adjacency pixels, such as the code “1”, “2” and “3” of E_VCC, the expression abilities of them become 1, and when a code represents the relationship between two corner-adjacency pixels, such as the code “0” of E_VCC, the expression ability of it becomes 2.

B. Comparison of Chain Codes

Figure 7 shows ten sample shapes we used to apply vertex chain codes, as taken from black-and-white raster images, whose sizes and numbers of pixels are shown in Table VII. We compared them from two aspects with the six vertex chain codes we described above. One is from the way of experiment results; the other is from the way of theoretic analysis. All these contours were counterclockwise oriented in the process of experiments.

Figure 7. Sample shapes utilized to apply vertex chain codes.

TABLE VII.

SIZE AND NUMBER OF PIXELS OF EACH SAMPLE SHAPE

Shape	Size	Pixels
Tiger	133×129	17157
Dragon	127×143	18161
Gun	557×258	143706
Mouse	508×466	236728
Flower	484×478	231352
Handwriting	352×335	117920
Angel wings	654×315	206010
Moon cake	504×463	233352
Table	364×400	145600
Cuttlefish	396×376	148896

1) Comparison on Experiment Results

First, we calculated the chain lengths of each contour shape in terms of the numbers of chain code with the six

vertex chain codes described as above, which are showed in Table VIII.

As it can be seen in Table VIII, the longest chain length comes from VCC and V_VCC with 29600 codes of total length following by E_VCC with 24124. On the other hand, the shortest chain length comes from EC_VCC with 14241 codes of total length following by D_VCC with 22310 and then VC_VCC with 22265 which is very near to the D_VCC.

Then, we calculated the memory capability of each contour shape in terms of the numbers of binary bit with the six vertex chain codes described as above, which are showed in Table IX.

TABLE VIII.

CHAIN LENGTH, IN NUMBERS OF CHAIN CODE, FOR THE SAMPLE SHAPES

	VCC	E_VCC	V_VCC	VC_VCC	D_VCC	EC_VCC
Tiger	868	698	868	648	666	424
Dragon	1134	939	1134	843	876	608
Gun	1884	1462	1884	1401	1346	766
Mouse	4140	3235	4140	2992	3334	2085
Flower	4602	3801	4602	3468	3599	2419
Handwriting	4112	3389	4112	3175	3414	2344
Angel wings	3454	2760	3454	2485	2780	1745
Moon cake	2026	1640	2026	1476	1532	956
Table	3288	3031	3288	2913	1336	819
Cuttlefish	4092	3169	4092	2864	3427	2075
Total	29600	24124	29600	22265	22310	14241

TABLE IX.

MEMORY CAPABILITY, IN NUMBERS OF BINARY BIT, FOR THE SAMPLE SHAPES

	VCC	E_VCC	V_VCC	VC_VCC	D_VCC	EC_VCC
Tiger	1736	1396	1364	1120	2664	3392
Dragon	2268	1878	1784	1481	3504	4864
Gun	3768	2924	2884	2131	5384	6128
Mouse	8280	6470	6642	5183	13336	16680
Flower	9204	7602	7206	6149	14396	19352
Handwriting	8224	6778	6606	6378	13656	18752
Angel wings	6908	5520	5566	4413	11120	13960
Moon cake	4052	3280	3172	2410	6128	7648
Table	6576	6062	4100	3626	5344	6552
Cuttlefish	8184	6338	6700	5109	13708	16600
Total	59200	48248	46024	38000	89240	113928

As it can be seen in Table IX, the most number of binary bits produced by EC_VCC with 113928 bits of total memory capability following by D_VCC with 89240 bits of total memory capability. On the other hand, the least number of binary bits produced by VC_VCC with 38000 bits of total memory capability following by V_VCC with 46024 bits of total memory capability.

And then, based on the chain lengths of each contour shape shown in Table VIII and the memory capability of each contour shape shown in Table IX, we calculated the average length in terms of bits per symbol with the six vertex chain codes described as above, which are showed in Table X. In Table X, the numbers of symbols of each vertex chain code are also listed.

TABLE X.

AVERAGE LENGTH OF EACH CODE, IN NUMBERS OF BITS PER SYMBOL

	VCC	E_VCC	V_VCC	VC_VCC	D_VCC	EC_VCC
Numbers of symbols	3	4	3	5	10	256
Bits/symbol	2	2	1.55	1.71	4	8

As it can be seen in Table X, the longest average length in bits per symbol is produced by EC_VCC with 8 bits per symbol following by D_VCC with 4 bits per symbol. On the other hand, the shortest average length in bits per symbol is produced by V_VCC with 1.55 bits per symbol following by VC_VCC with 1.77 bits per symbol.

At the same time, it can be seen in Table X, the numbers of symbols of EC_VCC are the most with 256 symbols following by D_VCC with 10 symbols, and the numbers of symbols of VCC and V_VCC are the least with 3 symbols following by E_VCC with 4 symbols and then the VC_VCC with 5 symbols.

The results of chain codes of D_VCC and EC_VCC above show that the more the numbers of symbols, the more bits per symbol of the chain code. And the results of chain codes of V_VCC and VC_VCC above show that the use of Huffman encoding concept can decrease the bits per symbol by contrast them to the chain codes of VCC and E_VCC.

The next, we calculated the average expression ability. Analyzing the above six vertex chain codes, for VCC and V_VCC, the expression ability of each code is 1 because each code represents the relationship between two edge-

adjacency pixels. Namely, the average expression abilities of VCC and V_VCC are 1.

For E_VCC and VC_VCC, the expression ability of some codes is 1 because they represent the relationship between two edge-adjacency pixels such as code 1, 2 and 3 of E_VCC and code c1, c4 and c5 of VC_VCC, and the expression ability of the other codes is 2 because they represent the relationship between two corner-adjacency pixels such as code 0 of E_VCC and code c2 and c3 of VC_VCC. So, we calculated the probabilities of the codes with expression ability of 1 and 2 respectively, and then calculated the average expression abilities of E_VCC and VC_VCC.

For D_VCC and EC_VCC, some codes represent the relationship between two edge-adjacency pixels such as code 0, 1 and 9 of D_VCC and code 01000000, 10000000, and 11000000 in binary form of EC_VCC, and the others represent the relationship between two corner-adjacency pixels such as code 00000000 in binary form of EC_VCC. So, some codes have the expression ability of 1 and the others have the expression ability of 2. At the same time, because some codes are generated based on calculation, they represent the relationship not only between two pixels but a number. For instance, the expression ability of code 2 to code 8 of D_VCC is from 2 to 8, because they represent the sequential numbers of code 2 are from 2 to 8; the expression ability of some codes of EC_VCC such as code “01000001”, “01000010”, and so on, until code “01111111” in binary form, and code “10000001”, “10000010”, and so on, until code “10111111” in binary form, and code “11000001”, “11000010”, and so on, until code “11111111” in binary form can be from 2, 3 up to 64 respectively; and the expression ability of some codes of EC_VCC such as code “00000001”, “00000010”, and so on, until code “00111111” in binary form can be from 4, 6, up to 128 respectively. To sum up, we calculated the probabilities of different codes with different expression ability respectively, and then calculated the average expression abilities of D_VCC and EC_VCC.

Table XI shows the number of chain symbols, different chain symbols with same expression ability and probabilities for each, and the average expression ability of each vertex chain codes calculated by applying to the above ten sample shapes. In Table XI , the chain symbols of EC_VCC are represented in decimal form, namely the symbols from 0 to 255 are represented for the codes from “00000000” to “11111111” in binary form of one byte respectively.

For the last row of Table XI, it only provided the average expression ability of EC_VCC. Considering that the symbols of EC_VCC are very large (from 0 up to 256), and the expression ability for each symbol is vary (from 1 up to 128), it did not list different chain symbols with same expression ability and probabilities respectively, but only represent the total probabilities of 1.000 for all symbols. Nevertheless, the result of average expression ability of EC_VCC is also calculated based on the experiments on the same ten sample shapes with the same method applied to other vertex chain codes, the difference lies in the amount of calculation is very large.

TABLE XI.

AVERAGE EXPRESSION ABILITY CALCULATED BASED ON THE SAMPLE SHAPES

	Number of symbols	Chain symbols	Expression ability	probabilities	Average expression ability
VCC	3	1,2,3	1	1.000	1.00
V_VCC	3	1,2,3	1	1.000	1.00
E_VCC	4	1,2,3	1	0.769	1.23
		0	2	0.231	
VC_VCC	5	c1,c4,c5	1	0.664	1.34
		c2,c3	2	0.336	
D_VCC	10	0,1,9	1	0.869	1.40
		2	2	0.047	
		3	3	0.028	
		4	4	0.015	
		5	5	0.008	
		6	6	0.004	
		7	7	0.004	
		8	8	0.025	
EC_VCC	256	0~255	1~128	1.000	2.08

Synthesizing the above analysis and data, we calculated the efficiency E of the six vertex chain codes based on the average code length L obtained from the above Table X and the average expression ability per code C obtained from the above Table XI, as shown in Table XII.

TABLE XII.

EFFICIENCY OF VERTEX CHAIN CODES

	C	L	E
VCC	1	2.00	0.50
E_VCC	1.23	2.00	0.62
V_VCC	1	1.55	0.65
VC_VCC	1.34	1.71	0.78
D_VCC	1.40	4.00	0.35
EC_VCC	2.08	8.00	0.26

As it can be seen in Table XII, the highest efficiency is produced by VC_VCC with 0.78 following by V_VCC with 0.65 and then E_VCC with 0.62. On the other hand,

the lowest efficiency is produced by EC_VCC with 0.26 following by D_VCC with 0.35 and then VCC with 0.50.

2) Comparison on Theoretic Analysis

Analyzing the reason of the generation of different efficiencies, the two highest efficiency produced by the two vertex chain codes with not much symbols of chain code and variable length produced by Huffman coding. Namely, the numbers of bits per symbol is varying with optimum coding. On the other hand, the two lowest efficiency produced by the two vertex chain codes with equal length and much symbols. Even so, D_VCC and EC_VCC apply the idea of arithmetic encoding which can be referenced in producing new methods of chain code. Although D_VCC and EC_VCC produced the lowest efficiency, but for the reason that there are redundancy codes in them, and the number of symbols is too large with unnecessarily.

To enhance the efficiency of vertex chain code, we can consider from the selection of numbers of symbols, the coding of chain code symbols with variable-length encoding and arithmetic encoding, etc.

IV. Conclusion

The methods of vertex chain code discussed thus far are by no means the all. In the paper, we described the techniques of six vertex chain codes. The techniques are discussed from two aspects: the description of techniques of vertex chain code and the comparison and evaluation of them.

The main reason for the popularity of vertex chain codes is their compression capabilities with more and more economical in their uses of memory capacity. We hope our research in the paper can provide convenience and reference for the chain code researchers and users. Although not mentioned in this paper, there have been other approaches of vertex chain code. The goal of the research related to pattern recognition and image processing, but there are some problems such as extending some new algorithm about vertex chain code, which are the work we need to do in the future.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (60675008) and the Natural Science Foundation of Liaoning Province (201102042).

References

[1] Y.K. Liu, B. Zalik, “An Efficient Chain Code with Huffman Coding,” *Pattern Recognition*, 2005, 38 (4):553–557.

[2] S. Zahir, K. Dhou, “A new chain coding based method for binary image compression and reconstruction,” *PCS, Lisbon, Portugal*, 2007, pp. 1321–1324.

[3] H. Sánchez-Cruz, M.A. Rodríguez-Díaz, “Coding Long Contour Shapes of Binary Objects,” *14th Iberoamerican Congress on Pattern Recognition, CIARP*, 2009, pp. 45–52.

[4] H. Sánchez-Cruz, “Proposing a new code by considering pieces of discrete straight lines in contour shapes,” *J. Vis. Commun. Image R.* 2010, 21: 311–324.

[5] S. Priyadarshini1, G. Sahoo, “A new lossless chain code compression scheme based on substitution,” *International Journal of Signal and Imaging Systems Engineering*, 2011, 4 (1):50–56.

[6] H. Sun, J.Y. Yang, M.W. Ren, “A fast watershed algorithm based on chain code and its application in image segmentation,” *Pattern Recognition Letters*, 2005, 26(9): 1266–1274.

[7] F. Arrebola, F. Sandoval, “Corner detection and curve segmentation by multiresolution chain-code linking,” *Pattern Recognition*, 2005, 38(10): 1596–1614.

[8] S. Zhao, Y. Xu, H. Li, H. Yang, “A Comparison of Lossless Compression Methods for Palmprint Images”, *Journal of Software*, 2012, 7(3): 594–598.

[9] Z. Zeng, W. Yang, “Design Patent Image Retrieval Based on Shape and Color Features”, *Journal of Software*, 2012, 7(6): 1179-1186.

[10] R.K. Gupta, B. Gurumoorthy, “Automatic extraction of free-form surface features (FFSFs),” *Computer-Aided Design*, 2010, 44(2): 99–112.

[11] X.L. Yan, L.P. Bu, L.M. Wang, “A Flame Apex Angle Recognition Arithmetic Based on Chain Code,” *Advances in Intelligent and Soft Computing*, 2012, 116: 29–35.

[12] I.K.G.D. Putra, M.A. Sentosa, “Hand Geometry Verification based on Chain Code and Dynamic Time Warping,” *International Journal of Computer Applications*, 2012, 38(12): 17–22.

[13] E. Bribiesca, “A chain code for representing 3D curves,” *Pattern Recognition*, 2000, 33(5): 755–765.

[14] H. Sánchez-Cruza, E. Bribiescab, R.M. Rodríguez-Dagnino, “Efficiency of chain codes to represent binary objects,” *Pattern Recognition*, 2007, 40(6): 1660–1674.

[15] M. Maitre, M.N. Do, “Depth and depth–color coding using shape-adaptive wavelets,” *Journal of Visual Communication and Image Representation*, 2010, 21(5): 513–522.

[16] E. Bribiesca, “A new chain code,” *Pattern Recognition*, 1999, 32(2): 235–251.

[17] M. Salem, A. Sewisy, U. Elyan, “A vertex chain code approach for image recognition,” *ICGST International Journal on Graphics, Vision and Image Processing*, 2005, 5(3): 1–8.

[18] S.H. Subri, H. Haron, R. Sallehuddin, “eural network corner detection of vertex chain code,” *AIML Journal*, 2006, 6(1): 37–43.

[19] S.X. Zhang, W. Zhang, G.Q. Li, G.Q. Gu, “Skew detection for form document using Vertex-chain-code,” *Journal of East China Normal University: Natural Science*, 2004, 9(3): 54–58.(In Chinese)

[20] D. Q. Wang, W. Zhang, G. Q. Gu, “Calculating the compact and posture ratio of an image region based on VCC,” *Journal of East China Normal University: Natural Science*, 2005, 3(1): 59–62. (In Chinese)

[21] Y. K. Liu, W. Wei, P. J. Wang, B. Žalik, “Compressed vertex chain codes,” *Pattern Recognition*, 2007, 40(11): 2908–2913.

[22] P. Y. Chen, C. P. Chang, “The segmented vertex chain code,” *Journal of the Chinese Institute of Engineers*, 2011, 34(6): 40–44.

[23] G. F. Yu, L. Wang, “Research on compression-type vertex chain code,” *Journal of Image and Graphics*, 2010, 15(10): 1465–1470. (In Chinese)

Linghua Li female, was born in Liaoning China in February 1975, a lecturer in Computer Science Department at Dalian Nationalities University, Dalian, China. She received her BSc in application of electronic technique from Liaoning Normal University in 1998. She received her MSc in physics from Liaoning Normal University in 2001. Her areas of interest are pattern recognition and image processing & recognition.

Yining Liu female, was born in Liaoning China in January 1991, a junior in College of Communication Engineering, Jilin University, Changchun, China.

Yongkui Liu male, was born in Liaoning China in May 1961, a professor in Computer Science Department at Dalian Nationalities University, Dalian, China. He has his BSc in computer science from Jilin University in 1982. He has his MSc in computer application from Shenyang University of Technology in 1987. He has his PhD in CAD and computer graphics from Zhejiang University in 1999. His research interests lie in computer Graphics, pattern recognition and image processing.

Borut Žalik is a professor of Computer Science at University of Maribor, Slovenia. He obtained PhD in computer science in 1993 from the University of Maribor, Slovenia. He is the head of Laboratory for Geometric modeling and multimedia algorithms. His research interests include computational geometry, geometric data compression, scientific visualization and geographic information systems. He is an author or co-author of more than 70 journal and 90 conference papers.