ARTICLE IN PRESS

Available online at www.sciencedirect.com
ScienceDirect

Pattern Recognition ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Lossless compression of map contours by context tree modeling of chain codes

Alexander Akimov*, Alexander Kolesnikov, Pasi Fränti

Department of Computer Science, University of Joensuu, P.O. Box 111, 80110 Joensuu, Finland

Received 24 October 2005; received in revised form 8 July 2006; accepted 7 August 2006

Abstract

We consider lossless compression of digital contours in map images. The problem is attacked by the use of context-based statistical modeling and entropy coding of the chain codes. We propose to generate an optimal *n*-ary incomplete context tree by first constructing a complete tree up to a predefined depth and creating the optimal tree by pruning out nodes that do not provide improvement in compression. We apply this method for both vector and raster maps. Experiments show that the proposed method gives lower bit rates than the existing methods of chain codes compression for the set of test data.

Keywords: Contour compression; Chain code encoding; Context tree

1. Introduction

Digital maps are usually stored as vector graphics in a database for retrieving the data using spatial location as the search key. The visual outlook of maps representing the same region varies depending on the type of the map (topographic or road map), and on the desired scale (local or regional map). Vector representation is convenient for zooming as the maps can be displayed in any resolution defined by the user. The maps can be converted to raster images for data transmission, distribution via web and visualization. Also map images can exist in raster format.

Chain coding is a common approach for representing different shapes such as line-drawings, planar curves and contours. We consider the compression of thin digital curves of one-pixel width, which are extracted from the vector data. The same time we consider the case when the contours information is obtained directly from the raster map.

Previous works in literature consider different schemes of encoding and chain code representation. The approaches, introduced by authors, are distinguished into several groups. Some of the works [1–4] consider the compression of chain codes with Huffman or arithmetic coders. Another authors used a context models of different orders: 1-order [5,6], 2-order [7,8], 3-order [9], 5-order [10] and up to 8 [11]. The problem of encoding of chain codes by prediction by partial matching (PPM) algorithm [12] has been considered in Refs. [13,14].

In principle, context-based compression can be improved by using of a larger number of neighboring symbols as a context. However, the growth of the context depth leads to the problem of context dilution, in which the statistics are distributed over too many contexts, and thus, affects the accuracy of the probability estimates. *Context tree* provides a more flexible approach for modeling the contexts so that a larger number of neighbor pixels can be taken into account without the context dilution problem [15]. The context tree algorithm was introduced in Ref. [16], and analyzed in Ref. [17]. Practical solutions for the binary context tree based compression algorithms for grayscale and bi-level images have been described in Refs. [18,15] correspondingly.

* Corresponding author. Tel.: +358 46 810 8544; fax: +358 13 251 7955.
E-mail addresses: akimov@cs.joensuu.fi (A. Akimov), koles@cs.joensuu.fi (A. Kolesnikov), franti@cs.joensuu.fi (P. Fränti).

0031-3203/$30.00
doi:10.1016/j.patcog.2006.08.005

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005

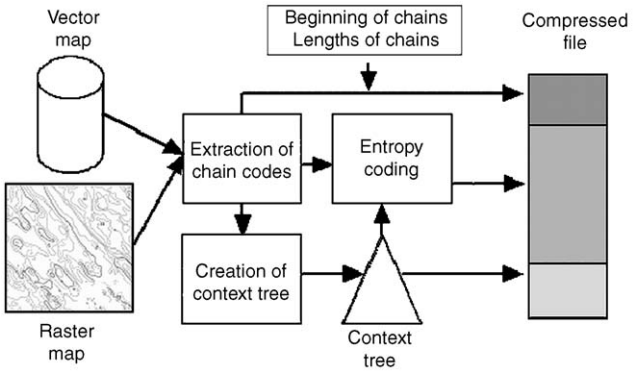

Fig. 1. Overall system diagram of the proposed method.

In this paper, we use the context tree approach for compression of chain codes. We provide algorithm for optimal n-ary incomplete context tree construction. The proposed algorithm constructs a context tree up to a predefined depth, and then prunes it according to the collected statistics in order to minimize the cost of the tree. The tree structure depends on the encoded data, and needs to be stored in the encoded file. Arithmetic coding is used for entropy encoding of the output data. We compare the proposed algorithm with existing compression methods of chain codes. Finally, we compare the compression of map contours encoded by the context tree to the same rasterized contours when encoded as bitmaps.

2. Overall system description

The overall scheme of the proposed compression method is as follows:

Step 1: Extract the chain codes from a vector or raster map. Store the information about the beginning of the chains (BOC) and their lengths into the output file.

Step 2: Create an optimal context tree for the chain codes. Store the context tree structure in the output file.

Step 3: Encode the chain codes using the resulting context tree and an entropy coder.

This scheme is shown in Fig. 1. The procedure of BOC storage is described below.

2.1. Chain code representation

Freeman [20] proposed chain codes for description of digitized curves, contours and drawings. Chain codes represent the digital contour by a sequence of line segments of specified length and direction, see Fig. 2. We consider both four- and eight-connected chain coding schemes. The four-connected chain coding scheme is restricted by four-connectivity and needs more chain codes to represent a contour (see Fig. 3).

Another type of chain codes are *crack codes* [21,22], which are used to describe boundaries of objects and borders between regions in color raster images. Crack codes are defined in the same way as four-connected chain codes, but the contour is formed by traversing along the cracks between adjacent pixels with different colors, see Fig. 4.

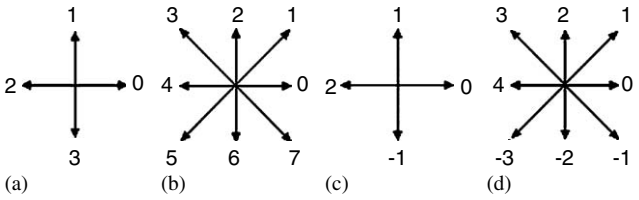

Fig. 2. The four- and eight-connected chain codes (a), (b) and the differential chain codes (c), (d).

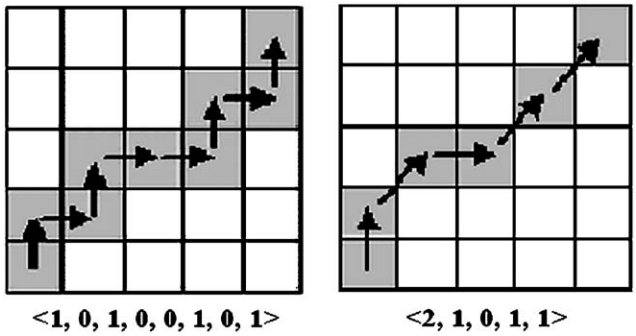

Fig. 3. An example of four- and eight-dconnected chain codes for four-connected (left) and eight-connected chains (right).

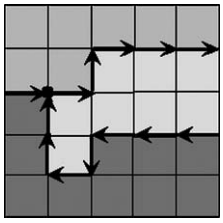

Fig. 4. An example of crack codes for non-binary image. There are three chains: ⟨0⟩, ⟨3, 3, 3, 2, 3, 1, 1⟩, ⟨0, 1, 0, 0, 0⟩.

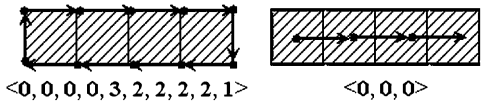

Fig. 5. An example of crack codes (left) and Freeman chain codes (right) for a one-pixel width object.

Crack codes are efficient in representing of the region borders. The using of crack codes for color images solves the problem of the border representation redundancy, which arises when Freeman codes are in use [23,24]. Freeman chain codes are more efficient for representation of one-pixel width lines, see Fig. 5. Here, for description of four-pixels shape we need ten crack codes, or three eight-connected Freeman chain codes.

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005

The more efficient way of the chain code representation is the *differential chain codes* [25]. Each chain code is replaced by its difference from the preceding chain code (see Fig. 2). We denote a chain code by c_i and the difference between the codes by $k_i = c_i - c_{i-1}$. The differential chain code x_i for four-connected and eight-connected schemes is calculated by Eqs. (1) and (2) correspondingly.

$$x_i = \begin{cases} k_i + 4 & \text{if } k_i < -1, \\ k_i - 4 & \text{if } k_i > 2, \\ k_i & \text{otherwise,} \end{cases} \quad (1)$$

$$x_i = \begin{cases} k_i + 8 & \text{if } k_i < -3, \\ k_i - 8 & \text{if } k_i > 4, \\ k_i & \text{otherwise.} \end{cases} \quad (2)$$

2.2. Extraction of chain codes from vector and raster data

In case of vector maps, the chain codes could be extracted directly from the vector data. The procedure consists of the following steps:

Step 1: Transformation of map vertices coordinates: convert the original vector coordinates into coordinates on raster. The transformed coordinates represent the start and end points of straight lines, which form contours and curves on the raster.

Step 2: Chain codes construction: sequentially, calculate the chain codes for each digital line segment using Bresenham's algorithm for digital line drawing [26] taking into account type of connectivity in use.

If the map is given as a raster image, we can extract the chain codes from the raster data either by line tracing using Freeman chain codes, or by border tracing using crack codes for binary [21,22] and color images [23,24]. In the latter case, crack code chains are traced and encoded from one junction point to another one; here junction (or branching) point is the point where the borders of three regions are met, see Fig. 4.

The eight-connected chain codes for an object contour can be constructed directly from the crack codes of the contour as alternative for shape representation (see Fig. 6). The rules described in Table 1 [27] perform the conversion of two crack codes c_i and c_{i-1}. The *null* value in the table means the impossible combination of two neighbor crack codes. The empty cell means that this combination of crack codes does not produce the eight-connected chain code.

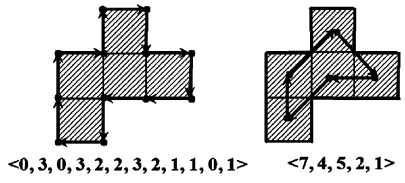

Fig. 6. A crack code (left) and the corresponding Freeman chain code.

Table 1
Rules for converting four-connected crack codes to eight-connected chain codes

	c_i				
c_{i-1}		0	1	2	3
	0	0	1		Null
	1	Null	2	3	
	2		Null	4	5
	3	7		Null	6

3. Compression of chain codes

3.1. Finite context modeling

We consider the compression of the chain codes, where the encoding is done according to their order in the input data. We denote the ith input symbol by x_i, and the string of the m previous symbols $x_{i-1}, \ldots, x_{i-m}$ by x^{i-m}. In *context-based modeling*, the probability of the current symbol x_i is conditioned on its *context* x^{i-m}. The probabilities of different contexts, as well as the probabilities of the symbols generated in a given context, are usually treated as being independent [15]. Thus, a model becomes a collection of independent sources of random variables. By the assumption of independence, it is simply to assign probabilities to each new symbol. We denote the cardinality of the encoded data alphabet by α and the counts of symbols generated under the given context x^{i-m} by $n_1(x^{i-m}), \ldots, n_\alpha(x^{i-m})$. The conditional probability of the event $x_i = k$, $k \in [1, \ldots, \alpha]$ is defined by

$$p(x_i = k | x^{i-m}) = \frac{n_k(x^{i-m})}{\sum_{j=1}^{\alpha} n_j(x^{i-m})}. \quad (3)$$

We consider the encoding of the given statistical model by an entropy encoder. The adaptive probability estimator of coder operates by the following formula:

$$p(x_i = k | x^{i-m}) = \frac{n_k(x^{i-m}) + \delta}{\sum_{j=1}^{\alpha} n_j(x^{i-m}) + \alpha \cdot \delta}. \quad (4)$$

The parameter δ here depends on different arithmetic coders, but it usually equals to $1/\alpha$ [28,29].

3.2. Context tree algorithm revisited

Context tree is applied for the compression in the same manner as the fixed size context, only the context selection is different. The context selection is made by traversing the context tree from the root to a terminal node, each time selecting the branch according to the corresponding previous symbol value. If the corresponding symbol points to a non-existing branch or if the current node is a leaf, then we came to a terminal node and stop descending. The terminal node points to the statistical model that is to be used.

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005

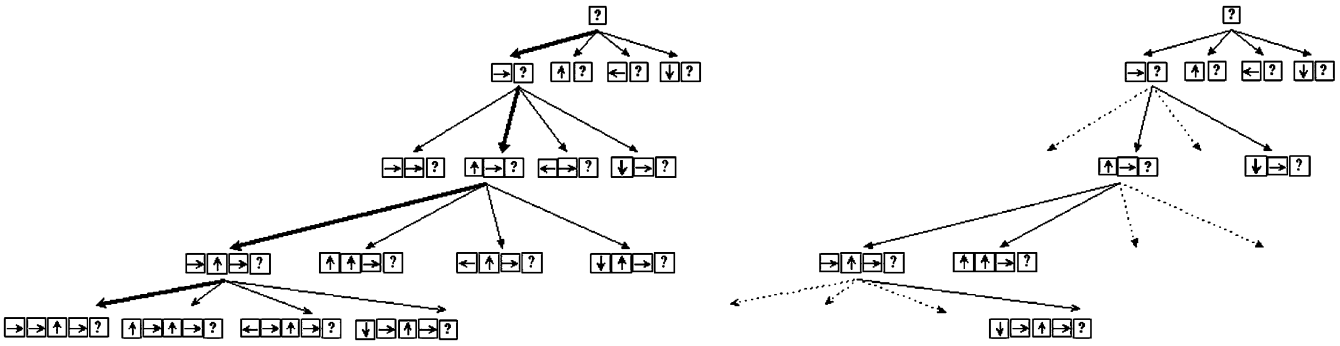

Fig. 7. The construction of the context tree with maximum depth 4: an initially full tree, constructed along the traversing path ⟨0, 1, 0, 0⟩, and the pruned incomplete one (right).

The context tree can be constructed beforehand (*static approach*) or optimized directly from the encoded data (*semi-adaptive approach*). In the second case, the tree structure must be stored in the compressed file.

The process of optimal tree construction consists of two main phases: the initialization of the context tree, and the pruning of the constructed tree [30], see Fig. 7. These phases will be described below.

3.3. Construction of an initial context tree

To construct an initial context tree for the input data, we need to process through the data to collect statistics for all potential contexts. Each node stores information of the counts of each code. The algorithm of the context tree construction by processing every chain code as follows:

Step 1: Create the root of the tree.

Step 2: For each symbol x_i, $i \in [1, \ldots, n]$:

- Traverse the tree along the path defined by the symbols in context x^{i-m}.
- If some node, visited along the path, does not have a consequent branch for transition to the next context symbol, then create the necessary child node and process it. Each new node has α counters, which are initially set to zero.
- In all visited nodes, increase the count of the current symbol x_i by 1.

This completes the construction of the context tree for all possible contexts. The time complexity of the algorithm is $O(m \cdot n)$, where n is the number symbols in the data.

3.4. Constructing of an optimal context tree

The initial context tree is pruned by comparing every parent node against its children nodes for finding the optimal combination of siblings. We denote the overall tree structure by T and the nodes of the tree by $w \in T$. The number

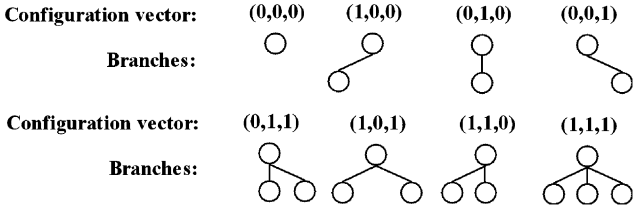

Fig. 8. Example of different configuration vectors.

of bits, required for description of each context tree node in the compressed file, is the size of the alphabet α.

We denote the set of nodes of the tree T by $S(T)$. We denote the count of the symbol i by $n_i(w)$, $w \in S(T)$. By the cost of node w here we understand the following expression [15]:

$$c_T(n_1(w), n_2(w), \ldots, n_\alpha(w)) = -\log_2 \frac{\prod_{i=1}^{\alpha} \prod_{j=0}^{n_i(w)-1} (j+\delta)}{\prod_{j=0}^{n_0(w)+n_1(w)+\cdots+n_\alpha(w)-1} (j+\alpha\cdot\delta)}. \quad (5)$$

By the cost of the context tree T, we will denote the following expression:

$$L(T) = \alpha \cdot |S(T)| + \sum_{w \in S(T)} c_T(n_1(w), n_2(w), \ldots, n_\alpha(w)). \quad (6)$$

The first term gives the number of bits, needed to store the tree, and the second term is an estimation of the compressed file size. The goal of the tree pruning is to modify the context tree structure in order to minimize (6).

3.5. Bottom-up algorithm

We use a bottom-up algorithm [31] to solve the problem of the optimal context tree pruning. The main principle of this algorithm is that the optimal tree consists of optimal sub-trees.

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005

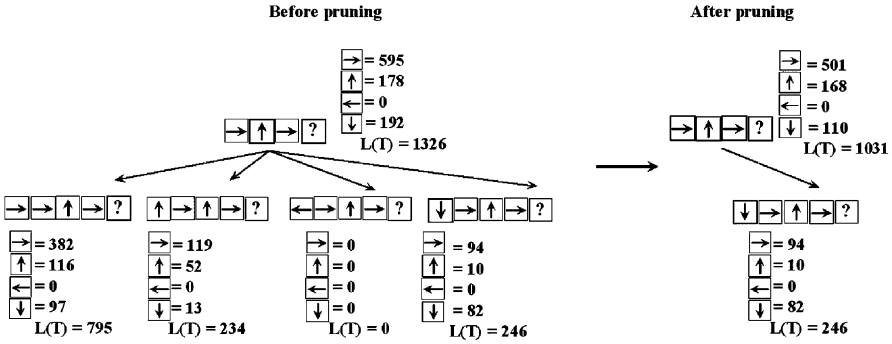

Fig. 9. Example of a single node pruning: resulted configuration is (0, 0, 0, 1).

For any node $w \in T$, we denote its child nodes by w_i. We denote the vector that describes the structure of node branches as the *node configuration vector*. The vector $v = (v_1, \ldots, v_\alpha)$ $v_i \in \{0, 1\}$ defines which of the branches will be pruned out after optimization: if $v_i = 0$, then the ith branch is pruned. An example of different configuration vectors is shown in Fig. 8.

The optimal cost $L_{opt}(T)$ for any given tree T can be expressed by the recursive equations (7) and (8):

$$L_{opt}(T) = \begin{cases} c_T(n(w)) + \alpha & \text{if } T \text{ is leaf,} \\ \min\limits_{v}\{L_v(T, v)\} & \text{otherwise,} \end{cases} \tag{7}$$

where

$$L_v(T, v) = c_T\left(n(w) - v \circ \left(\sum_i n(w_i)\right)\right) + \sum_i (v_i \cdot L_{opt}(T_i)) + \alpha. \tag{8}$$

The tree T_i is a sub-tree of T, starting from the node w_i.

The calculation of the cost of the optimal context tree T is done according to:

Step 1: If T has no child nodes, then return the accumulated code length of its root according to formula (7).

Step 2: For all sub-trees $T_i \subset T$, starting from the child nodes of T root, calculate their optimal costs $L_{opt}(T_i)$.

Step 3: According to the found $L_{opt}(T_i)$, the vectors of counts $n(w), n(w_1), \ldots, n(w_\alpha)$, find the optimal configuration vector $v_{min} = \text{argmin}\{L_v(T, v)\}$.

Step 4: Prune out the children sub-trees according the vector v_{min}.

Step 5: Return the value $L_v(T, v_{min})$.

The algorithm recursively prunes out all unnecessary sub-trees, and finally gets the optimal structure of the context tree, see Fig. 9. The optimal configuration vector for each node is found by full search algorithm.

4. Encoding of BOC

The description of contours consists of chain codes and the coordinates of the chain beginnings. In case of open chains we need to encode the lengths and start positions. In case of closed ones we encode only coordinates of chain beginnings.

We encode BOCs by one of the following methods. According to the first method we directly store the coordinates of BOCs in the compressed file. According to the second method we represent all BOCs by black points on a white background and encode the resulting image as a binary one. We choose that method which produces better compression.

The description of BOCs of crack codes, extracted from non-binary images, also includes two numbers, which define the colors of cracked regions.

5. Experiments

We tested the proposed algorithm for different types of data as shown in Fig. 10. The first five images from the test set are the vector data, transformed into chain codes. The vector images #1–4 are used after data precision reduction in order to fit the data for the image with dimensions 5000×5000 pixels. The vector image #5 was taken as it is. These images are contours of geographical objects and elevation lines. The next six maps are raster images. The images #6–8 represent schemes of fields and forest stands: image #6 is binary and images #7 and 8 are grayscales with 256 colors. The last three binary images are semantic layers of different maps from NLS [31]: water, elevation lines and administrative information. Properties of the test images and statistics of extracted chain codes are shown in Table 2.

We have provided two series of experiments. The first series illustrates the efficiency of the optimal context tree encoding of the chain codes. The second one studies the ability of the chain encoding to increase the compression performance of the map image compression in general. The used entropy coder is a range-coder [29].

We divided the algorithms into two groups: encoding of four-connected differential chain codes (CC4) and encoding of eight-connected differential chain codes (CC8). By CC4 we understand four-connected Freeman chain codes resulted from vector data and the crack codes, obtained from

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005

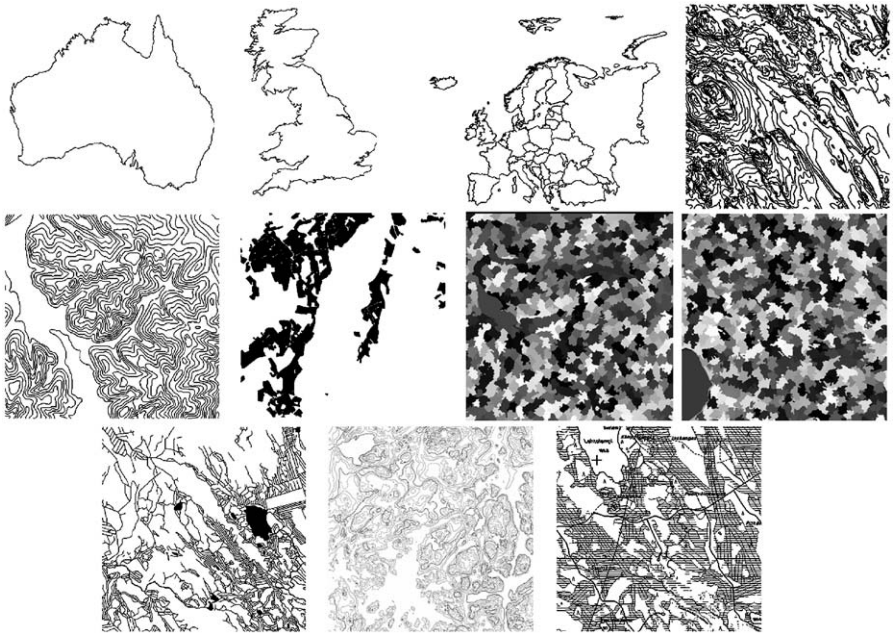

Fig. 10. The test set of images; Images #4–6 and #9–11 are represented by fragments.

Table 2
Chain code statistics for the tested images

	Type of the map	Image dimensions	Number of chains	CC4	CC8
Image #1	Vector	5000 × 5000	1	37 888	27 306
Image #2	Vector	5000 × 5000	1	86 216	64 220
Image #3	Vector	5000 × 5000	142	208 205	160 749
Image #4	Vector	5000 × 5000	537	519 316	356 839
Image #5	Vector	4391 × 4053	2551	1 708 522	1 222 485
Image #6	Raster	5000 × 5000	341	149 546	106 762
Image #7	Raster	1024 × 1024	1859	83 199	58 069
Image #8	Raster	1024 × 1024	1893	85 574	59 824
Image #9	Raster	5000 × 5000	3167	1 610 382	1 100 952
Image #10	Raster	5000 × 5000	18 498	2 565 670	1 786 825
Image #11	Raster	5000 × 5000	60 039	3 970 510	3 186 423

Table 3
Comparison of bit rates for different algorithms operating with CC4 (bits per code)

	Estes and Algazi	PPM4	CTC4
Image #1	0.761	0.726	0.729
Image #2	1.052	1.032	1.012
Image #3	1.524	1.567	1.494
Image #4	0.597	0.586	0.565
Image #5	0.704	0.690	0.670
Image #6	0.851	0.825	0.824
Image #7	1.066	1.068	1.025
Image #8	1.098	1.094	1.054
Image #9	0.613	0.615	0.574
Image #10	0.868	0.851	0.815
Image #11	0.732	0.730	0.688
Average	0.897	0.889	0.859

Table 4
Comparison of bit rates for different algorithms operating with CC8 (bits per code)

	Liu and Žalik	Lu and Dunham	PPM8	CTC8
Image #1	1.809	1.106	0.996	1.004
Image #2	1.949	1.486	1.441	1.400
Image #3	2.726	2.338	2.396	2.290
Image #4	1.827	0.971	0.810	0.813
Image #5	1.684	1.050	0.952	0.943
Image #6	1.812	1.250	1.179	1.153
Image #7	1.778	1.520	1.585	1.463
Image #8	1.795	1.564	1.620	1.506
Image #9	1.838	1.022	0.839	0.798
Image #10	1.880	1.329	1.195	1.177
Image #11	1.550	1.044	0.882	0.831
Average	1.877	1.335	1.263	1.216

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005

Table 5
Comparison of different compression schemes for raster images (bytes). Images #7, 8 are compressed by GIF; all other images are compressed by JBIG

	JBIG/GIF	Liu and Žalik	Lu and Dunham	Estes and Algazi	PPM4	PPM8	CTC4	CTC8
Image #1	5719	6185	3784	3615	3450	3410	3467	3437
Image #2	16 027	15 680	11 947	11 348	11 131	11 596	10 910	11 263
Image #3	41 789	55 169	46 977	40 071	41 183	48 545	39 291	46 416
Image #4	71 128	82 550	43 331	39 858	39 114	37 204	37 740	37 336
Image #5	181 065	270 870	174 078	163 893	160 935	159 051	156 583	148 531
Image #6	20 509	24 973	16 684	16 693	16 215	16 527	16 197	16 198
Image #7	99 304	18 454	16 575	16 633	16 653	17 051	16 175	16 156
Image #8	99 849	19 057	17 331	17 374	17 334	17 746	16 907	16 893
Image #9	117 436	258 878	140 715	129 493	129 797	121 461	121 570	115 826
Image #10	227 637	446 442	296 888	304 922	299 405	294 059	287 849	289 349
Image #11	252 740	674 051	415 958	419 744	418 888	407 882	397 986	387 626
Average	103 018	170 210	107 661	105 786	104 919	103 139	100 425	99 003

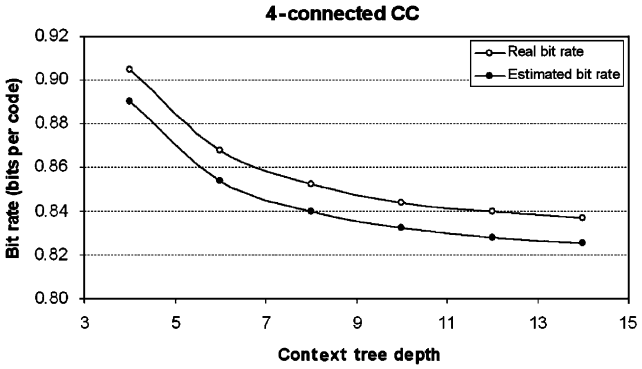

Fig. 11. Dependency between average bit rate and context tree depth in CTC4.

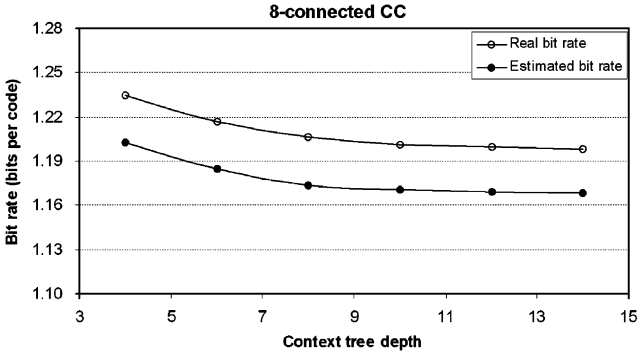

Fig. 12. Dependency between average bit rate and context tree depth in CTC8.

raster images. We compared the following algorithms of CC4 compression:

- *Estes and Algazi*: Encoding by n-order context model up to 10 [11];
- *PPM4*: PPM algorithm applied for CC4 compression [12];
- *CTC4*: The proposed context tree encoding of CC4.

Table 6
The proportion of different parts in the compressed file

	BOC (%)	CT (%)	Chain codes (%)
Image #1	0.3	1.5	98.2
Image #2	0.1	0.6	99.3
Image #3	1.0	0.2	98.8
Image #4	2.8	0.7	96.4
Image #5	9.0	0.5	90.5
Image #6	4.9	0.8	94.3
Image #7	34.2	0.2	65.6
Image #8	33.3	0.1	66.6
Image #9	4.9	0.5	94.6
Image #10	9.2	0.6	90.2
Image #11	14.2	0.9	84.9
Average	10.4	0.6	89.0

Table 7
Three most used context for Image #4 in CTC4

Context	→	↑	←	↓	Total
→↓→↓→↓→↓?	10 577	0	2	1943	12 522
↑←↑←↑←↑?	4	1355	7924	0	9283
↓↓→↓↓→↓↓?	5904	0	8	1910	7822

The compression of CC8 was compared on the following algorithms:

- *Liu and Žalik*: Encoding by fixed Huffman codes [1];
- *Lu and Dunham*: Encoding by second-order context model [8];
- *PPM8*: PPM algorithm applied for CC8 compression [12];
- *CTC8*: The proposed context tree encoding of CC8.

We used PPM with maximum depth of the context 8. The usage of context of higher order in PPM algorithm leads to the context dilution problem and, consequently, to decreasing of the PPM compression performance.

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005

Table 8
Three most used context for Image #4 in CTC8

Context	→	↗	↑	↖	←	↙	↓	↘	Total
↘↓↘↓↘?	2	0	1	0	3	560	6473	854	7893
←←←←←?	0	0	6	681	4883	477	8	0	6055
↓↓↓↓↓↓?	45	2	0	0	0	0	4879	994	5920

The compression results for CC4 and CC8 are summarized at the Tables 3 and 4; correspondingly. The results for context tree encoding are calculated with maximum tree depth 14. The proposed context tree encoding algorithm outperforms all competitive methods in terms of compression performance. The best competitive algorithm, PPM [12], has quite close results, but it has a drawback. The efficiency of the PPM algorithm (like any other fixed depth context-based algorithm) strongly depends on correct choice of the context depth. The using of too big depths for such algorithms leads to context dilution problem. Due this, in some cases, other competitive algorithms, which are using smaller context depth, outperform the PPM (for instance, see results for Image #7 in Table 4). The used context tree based approach is not affected by the context dilution and, therefore, is able to use bigger context depths and achieve better compression results.

The Figs. 11 and 12 show the dependency between the maximum context depth and the compression performance in context tree compression of CC4 and CC8 for estimated and real bit rates. The estimated bit rate is obtained by formula (7), and the real bit rate is resulted after entropy encoding. We see that the compression performance only increases with growth of the context tree depth. The difference between estimated and practical bit rates is negligible, about 1–2%.

Table 5 shows the overall compression of map images. We compared different techniques of chain codes compression with raster image compression algorithm, JBIG [19] for binary images and GIF for non-binary ones. The results show that for the images with comparably low saturation of graphics details the chain code compression is better then the raster-based one. The JBIG algorithm obtained better results for the map images with large number of tiny graphical objects. This can be explained by the fact that the images include a lot of graphical patterns. For instance, the sands are defined by areas of dots, swamps and melioration systems by areas of horizontal lines and etc. The attempt of straightforward converting all of this small objects into chain codes leads to enormous number of chains and makes the map compression by chain codes less efficient then a simple raster-based compression. The chain codes are mostly efficient for encoding of the region borders for planar subdivision maps (Images #6–8).

Table 6 represents the structure of the compressed file in CTC4 compression: the percentage of all three part of data in the file: the BOC, structure of the context tree (CT) and the encoded chain codes.

The most used contexts in Image #4 for CTC4 and CTC8 compression are shown in Tables 7 and 8. We show that the best correlation have been achieved on chains, describing straight lines.

6. Conclusions

We considered the problem the lossless compression of the chain codes. We applied the context tree based approach to the problem and provide optimal algorithm for n-ary incomplete context tree construction.

The suggested approach showed the best performance for chain codes compression. It outperforms the PPM algorithm by 3–4%, 5–7% over Markov model-based methods and about 40% over simple Huffman encoding of differential chain codes. We showed the ability to use chain codes technique for compression of map images.

References

[1] Y. Liu, B. Žalik, An efficient chain code with Huffman coding, Pattern Recognition 38 (4) (2005) 553–557.
[2] B. Mealy, Region boundary generation and compression, Conference Record of the 35th Asilomar Conference on Signals, Systems Comput. 1 (2001) 205–209.
[3] R. Redondo, G. Cristobal, Lossless chain coder for gray edge images, in: Proceedings of IEEE International Conference on Image Processing, vol. 2, 2003, pp. 201–204.
[4] T. Pavlidis, Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, MD, 1982.
[5] J. Chung, J. Moon, Conditional differential chain coding for lossless representation of object contours, Electr. Lett. 34 (1) (1998) 55–56.
[6] L. Labelle, D. Lauzon, J. Konrad, E. Dubois, Arithmetic coding of a lossless contour-based representation of label images, International Conference on Image Processing, vol. 1, 1998, pp. 261–265.
[7] T. Kaneko, M. Okudaira, Encoding of arbitrary curves based on chain code representation, IEEE Trans. Commun. 33 (1985) 697–707.
[8] C. Lu, G. Dunham, Highly efficient coding schemes for contour lines based on chain code representations, IEEE Trans. Commun. 39 (10) (1991) 1511–1514.
[9] Y. Chan, W. Siu, Highly efficient coding schemes for contour line drawings, in: Proceedings of IEEE International Conference on Image Processing, vol. 3, 1995, pp. 424–427.
[10] M. Turner, N. Wiseman, Efficient lossless image contour coding, Comput. Graphics Forum 15 (2) (1996) 107–118.
[11] R. Estes, R. Algazi, Efficient error free encoding of binary documents, in: Proceedings of Data Compression Conference, 1995, pp. 122–131.

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005

[12] J. Cleary, I. Witten, Data compression using adaptive coding and partial string matching, IEEE Trans. Commun. 32 (4) (1984) 396–402.
[13] F. Bossen, T. Ebrahimi, Region shape coding, Technical Report M0318, ISO/IEC JTC1/SC29/WG11, 1995.
[14] O. Egger, F. Boseen, T. Ebrahimi, Region based coding scheme with scalability features, in: Proceedings of the VIII European Signal Processing Conference, vol. 2, 1996, pp. 747–750.
[15] B. Martins, S. Forchhammer, Tree coding of bi-level images, IEEE Trans. Image Process. 7 (4) (1998) 517–528.
[16] J. Rissanen, A universal data compression system, IEEE Trans. Inf. Theory 29 (5) (1983) 656–664.
[17] M. Weinberger, J. Rissanen, A universal finite memory source, IEEE Trans. Inf. Theory 41 (3) (1995) 643–652.
[18] M. Weinberger, J. Rissanen, R. Arps, Application of universal context modeling to lossless compression of grey-scale images, IEEE Trans. Image Process. 5 (1996) 575–586.
[19] JBIG, Progressive bi-level image compression, ISO/IEC International Standard 11544, 1993.
[20] H. Freeman, Computer processing of line drawing images, ACM Comput. Surveys 6 (1974) 57–59.
[21] L. Cederberg, Chain-link coding and segmentation for raster scan devices, Comput. Graphics Image Process. 10 (1979) 224–234.
[22] G. Wilson, B. Batchelor, Algorithm for forming relations between objects in a scene, IEE Proc. Comput. Digital Tech. 137 (2) (1990) 151–153.
[23] S. Blackburn, Extraction of color region boundaries, IAPR Workshop on Machine Vision Applications, 1992, pp. 63–66.
[24] P. Zingaretti, M. Gasparroni, L. Vecci, Fast chain coding of region boundaries, IEEE Trans. Pattern Anal. Mach. Intell. 20 (4) (1998) 407–415.
[25] H. Freeman, Application of the generalized chain coding scheme to map data processing, in: Proceedings of IEEE Pattern Recognition and Image Processing, 1978, pp. 220–226.
[26] J. Bresenham, Algorithm for computer control of a digital plotter, IBM Systems J. 4 (1) (1965) 25–30.
[27] G. Wilson, Properties of contour codes, IEE Proc. Vision Image Signal Process. 144 (3) (1997) 145–149.
[28] P. Howard, J. Vitter, Analyses of arithmetic coding for data compression, in: Proceedings of the Data Compression Conference, 1991, pp. 3–12.
[29] G. Martin, An algorithm for removing redundancy from a digitized message, Presented at: Video and Data Recording Conference, 1979.
[30] R. Norhe, Topics in descriptive complexity, Ph.D. Thesis, University of Lingköping, Sweden, 1994.
[31] National Land Survey of Finland, Opastinsilta 12 C, P.O. Box 84, 00521 Helsinki, Finland ⟨http://www.nls.fi/index_e.html⟩.

About the author—ALEXANDER AKIMOV received the M.Sc. degree in Applied Mathematics in 2000 from the Saint Petersburg State University, Russia, and another M.Sc. degree in Computer Science in 2001 from the University of Joensuu, Finland. Currently he is a Ph.D. Student in Department of Computer Science of University of Joensuu. His main research areas are the compression of raster and vector map images.

About the author—ALEXANDER KOLESNIKOV received the M.Sc. degree in Physics in 1976 from the Novosibirsk State University, USSR, and the Ph.D. degree in Computer Science in 2003 from the University of Joensuu, Finland. From 1976 to 2003 he was a Senior Research Fellow with the Institute of Automation and Electrometry, Russian Academy of Sciences, Novosibirsk, Russia. In 2003, he joined the Department of Computer Science at the University of Joensuu, Joensuu, Finland. His main research areas are in signal and image processing, vector map processing and compression.

About the author—PASI FRÄNTI received his M.Sc. and Ph.D. degrees in Computer Science in 1991 and 1994, respectively, from the University of Turku, Finland. From 1996 to 1999 he was a postdoctoral researcher of the Academy of Finland. Since 2000, he has been a professor in the University of Joensuu, Finland. His primary research interests are in image compression, clusterization and speech technology.

Please cite this article as: A. Akimov, et al., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006), doi: 10.1016/j.patcog.2006.08.005