UNIVERSITY OF JOENSUU
COMPUTER SCIENCE
DISSERTATIONS 15

ALEXANDER AKIMOV

COMPRESSION OF DIGITAL MAPS

ACADEMIC DISSERTATION

To be presented, with the permission of the Faculty of Science of the University of Joensuu, for public criticism in Louhela Auditorium of the Science Park, Länsikatu 15, Joensuu, on November 11th, 2006, at 12 o'clock.

UNIVERSITY OF JOENSUU
2006

Supervisors Professor Pasi Fränti
Department of Computer Science and Statistics
University of Joensuu
Joensuu, Finland

Dr. Alexander Kolesnikov
Department of Computer Science and Statistics
University of Joensuu
Joensuu, Finland

Reviewers Professor Ioan Tăbuş
Signal Processing Laboratory
Tampere University of Technology
Tampere, Finland

Professor Søren Forchhammer
Research Center COM
Technical University of Denmark
Lyngby, Denmark

Opponent Professor Xiaolin Wu
Department of Computer Science
McMaster University
Hamilton, Canada

ISBN 952-458-869-2 (printed)
ISSN 1238-6944 (printed)
ISBN 952-458-870-6 (PDF)
ISSN 1795-7931 (PDF)

Computing Reviews (1998) Classification: E.4, G.1.2, G.1.6, H.2.8, I.3.5, I.4.3

Yliopistopaino
Joensuu 2006

Compression of digital maps

Alexander Akimov

Department of Computer Science and Statistics
University of Joensuu
P.O. Box 111, FIN-80101 Joensuu, FINLAND
akimov@cs.joensuu.fi

University of Joensuu, Computer Science, Dissertations 15
Joensuu, 2006, 116 pages

ISBN 952-458-869-2 (printed)
ISSN 1238-6944 (printed)
ISBN 952-458-870-6 (PDF)
ISSN 1795-7931 (PDF)

Abstract

In this thesis, we study the compression of digital maps, which allows achieving compact storage size and fast transmission of them to clients. The thesis is composed of two main parts.

The first part is dedicated to the lossless compression of raster map images. We consider both dictionary-based and context-based statistical compression. The best compression performance is achieved by using the context-based compression. To prevent context dilution problem during the compression, we apply context-tree based compression, which operates by an incomplete *n*-ary context tree. The proposed algorithm outperforms all existing context-based methods on the set of test images used.

The second part considers compression of geographical maps in vector format. We study a variety of different methods of lossy compression of geographical vector data: compression of rasterized vector map, compression of map contours by chain codes, compression by coordinates quantization, and progressive encoding of the vector data.

In the raster-based compression, the vector map is first rasterized and then compressed by a raster image compression method. We consider to exploit the vector information to simplify the rasterized image, and in this way, to obtain better compression performance.

We also study the compression of map contours and use a chain code modeling approach for this purpose. This approach is good and efficient alternative to the

straightforward encoding of coordinates. We apply context-tree based compression, and the proposed algorithm provides better results than any of the competitive algorithms on the set of test data used.

Compression of vector data by coordinate quantization is also considered. We construct optimal product quantizer both in Cartesian and polar spaces. The proposed quantizer outperforms the heuristic ones in rate-distortion sense.

We consider also lossy compression of multiresolution vector data, based on coordinate quantization. We use the coordinates of lower resolution data to predict coordinate values of the higher resolutions. This approach narrows the set of prediction errors, which is used for constructing of the quantizer.

Keywords: map compression, image compression, context-tree modeling, coordinate quantization, chain code compression.

Acknowledgments

The work presented in this thesis was carried out at the Department of Computer Science, University of Joensuu, Finland, during the years 2002-2006.

I would like to express my sincere gratitude to my thesis supervisors, Professor Pasi Fränti and Dr. Alexander Kolesnikov, for their guidance, encouragement, and infinite support throughout the research process. I also owe thanks to Dr. Pavel Kopylov for co-operative work.

Also I would like to thank my colleagues, who worked with me and helped me in my professional and scientific growth.

I would like to express my gratitude to the Faculty of Science, and the East Finland Graduate School of Computer Science and Engineering (ECSE) for their financial support of my studies between 2003-2006. I would also like to express my gratitude to the National Technology Agency of Finland (Tekes), and all the companies involved in the Dynamap project in 2002: Arbonaut, Benefon, Kata-electronics and TikkaCommunications.

Joensuu, October 2006

Alexander Akimov

Contents

Publications

P1. A. Akimov, P. Kopylov and P. Fränti, Semi-adaptive dictionary based compression of map images, *Proceedings of International Conference on Computer Graphics and Vision* (*GraphiCon'2002*), pp. 218-224, Nizhny Novgorod, Russia, September 2002.

P2. A. Akimov, A. Kolesnikov and P. Fränti, Lossless compression of color map images by context tree modeling, *IEEE Transactions on Image Processing*. (in press)

P3. A. Akimov and P. Fränti, Symbol representation in map image compression, *Proceedings of ACM Symposium on Applied Computing* (*SAC'04*), vol. 1, pp. 29-34, Nicosia, Cyprus, March 2004.

P4. A. Akimov, A. Kolesnikov and P. Fränti, Lossless compression of map contours by context tree modeling of chain codes, *Pattern Recognition*. (in press)

P5. A. Akimov, A. Kolesnikov and P. Fränti, Coordinate quantization in vector map compression, *Proceedings of IASTED Conference on Visualization, Imaging, and Image Processing* (*VIIP'04*), pp. 748-753, Marbella, Spain, September 2004.

P6. A. Akimov, A. Kolesnikov and P. Fränti, Reference line approach for vector data compression, *Proceedings of IEEE International Conference on Image Processing* (*ICIP'04*), vol. 2, pp. 1891-1894, Singapore, October 2004.

1 Introduction

Maps are abstract objects used to represent real places and things through symbols. Digital maps are widely used in geographical information systems (GIS), which can be used for navigating a vehicle or in certain web-based services (see Fig. 1). The data from a digital map can be represented in vector or raster formats. Map data can also be represented in a hybrid format, where raster data is combined with vector layers.

Figure 1: Maps in mobile and in-car navigation systems.

Raster maps are abstractions where spatial data is represented in 2-D arrays of pixels, depicted as a schematic discrete-tone image. Raster-based GIS systems use the following procedure: after first receiving a request from a client, raster-based systems get a map from a database and convert it to a bitmap with predefined dimensions. Next, this bitmap is sent to the client. The main advantage of this procedure is its simplicity: the raster format does not require a large amount of

computational resources and can be represented in portable devices with low machine resources, like mobile phones.

Although the raster map format has many advantages, it also has several important disadvantages. The first disadvantage is the size of the map: increasing the map resolution immediately leads to a significant increase in file size. The second disadvantage is that raster maps are displayed as images and, accordingly, they do not contain any attributive information. The raster maps not having attributive information makes it impossible to process queries.

Whereas raster maps represent data as 2D pixels, vector maps represent data as geometrical objects. If, for example, in a vector map, a road was represented as a combination of lines, those lines would be the geometrical objects. A vector map consists of two main types of data: geometrical data and attributive data. Geometrical data includes the coordinates of points and the rules regarding how those points should be connected. Attributive data contains color, textual and other types of information related to the objects. Unlike raster maps, vector maps are more compact and are invariant to the zooming operation. The presence of attributive information makes it possible to process different queries. The main disadvantage of vector maps is their complex data structure, which can result in long display times the use of a significant amount of machine resources.

Because of the large size of digital maps, their data often needs to be compressed for map databases storage and transmission to remote users. The large size of digital maps, the limited bandwidth of wireless data transmission channels, and the low machine resources of mobile devices affect the efficiency of GIS and navigation systems.

The use of effective compression algorithms can reduce the storage space needed for map collection, which can increase the amount and quality of the geographic data that can be stored on clients' portable devices. Effective compression algorithms can

also accelerate the transmission of data through low-bandwidth channels, which will reduce the time needed to transfer geographic data to clients, which, in turn, will make map services cheaper and more reliable.

2. Compression of raster maps

Raster maps are *discrete-tone* schematic images with a limited number of colors. Unlike *continuous-tone* images where the intensities change smoothly, the intensities in discrete-tone images change abruptly through a limited number of values. Another important property of raster map images is that they composed solely of straight lines, text, and geometric objects.

Figure 2. An example of lossy compression in a map image. On the left is the original image with 6 colors, and on the right is the JPEG-compressed image with 646 colors.

Compression algorithms for raster maps must preserve a their structure and color information. Therefore, we concentrate our discussion here on lossless algorithms

for raster images. Lossy algorithms such as *JPEG* [Wal91] produce changes in the structure and the color palette that can significantly affect the quality of a decoded map (see Fig. 2).

2.1 Dictionary-based image compression algorithms

Dictionary-based compression algorithms replace a subsequence of the encoded message, by using pointers to a collection of strings of pixels called a *dictionary*. The best known dictionary-based methods are based on the algorithms proposed by Lempel and Ziv in 1977 [Ziv77] and 1978 [Ziv78], namely LZ77 and LZ78.

The *Portable network graphics* (PNG) format [PNG] and the *CompuServe Graphics Interchange Format* (GIF) [GIF] are the most widely used standards for lossless image compression. The PNG format is based on the *DEFLATE* [Deut96] compression algorithm, which is a modification of the LZ77 algorithm. The GIF format is based on the *Lempel Ziv Welch* (LZW) [Welch84] compression algorithm, which is a further development of LZ78. In publication **P1**, we consider the semi-adaptive modification of the LZW method in the case when an image is divided and then compressed into small, rectangular blocks.

The main drawback of these LZW algorithms is that they do not utilize the 2-D information of the image. Accordingly, GIF and PNG are less efficient than the newer compression algorithms.

2.2 Predictive lossless compression of images

Predictive methods are the best for lossless encoding of photographic images. Algorithms such as *Fast and Efficient Lossless Image Compression Algorithm* (FELICS) [Hova93], *Context-Based, Adaptive Image Codec* (CALIC) [Wu97], *Low-Complexity Context-Based Lossless Image Compression Algorithm* (LOCO-I) [Wein96a] and *TMW* [Meye97] are based on the encoding of prediction errors,

where the prediction is based on the values of the neighborhood pixels. These algorithms give excellent results for continues-tone images, where colors change smoothly. The main disadvantage of predictive algorithms for compressing raster maps is that the prediction cannot be done properly, due to the discrete-tone nature of the map images. Therefore, the proposed algorithms are unable to predict a change of colors and are even less effective at prediction than dictionary-based algorithms. Due to this fact, we do not consider prediction-based algorithms any further here.

2.3 Context-based compression of images

Pixels in a map image form geometrical structures with appropriate spatial dependencies. Those dependencies can be localized to a limited neighborhood, and described by a *context-based statistical model* [LR81]. In this model, the pixel probability is conditional on the *context* C, which is defined as a distinct configuration of neighboring pixels. An example of a 4-pixel context template is demonstrated in Figure 3.

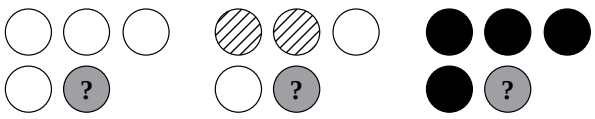

Figure 3: An example of a 4-pixel template and three sample contexts.

The probabilities of different contexts are assumed to be independent. Because of this assumption, the pixel probability for each context C_j can be found by calculating the counters ($n_k^{C_j}$) for all possible pixel values k appearing in that particular context in the entire image

$$p_m^{C_j} = \frac{n_m^{C_j}}{\sum_k n_k^{C_j}} \cdot \tag{1}$$

The corresponding probability of each context C_j is calculated by

$$p_{C_j} = \frac{\sum_k n_k^{C_j}}{\sum_i \sum_k n_k^{C_i}} \tag{2}$$

The *Entropy* of the context-based model is a weighted sum of the entropies of the individual contexts

$$H = -\sum_i p_{C_i} \cdot \left(\sum_k p_k^{C_i} \cdot \log_2 \left(p_k^{C_i} \right) \right). \tag{3}$$

A context with more skewed probability distribution has smaller information content and, therefore, smaller entropy, than with less skewed one.

Encoding symbols within each separate context are made by an entropy coder, such as *arithmetic* [Riss79] or *Huffman* [Huff52] coding.

In principle, a skewed probability distribution can be obtained by using a larger number of pixels in context. However, the overall number of contexts increases exponentially as the number of pixels included in each particular context decreases. This leads to the *context dilution* problem, which occurs when the count statistics are distributed over too many contexts, thus affecting the accuracy of the probability estimation.

2.4 Context tree modeling

In *variable-size context modeling*, the number of context pixels depends on the combination of the neighboring pixel values; context selection is done by traversing

the *context tree* instead of using a fixed size template [Riss83]. Each node in a tree represents a single context, and the children of a context correspond to the parent context augmented by one more pixel. The position of this pixel can be fixed in a predefined order or optimized within a limited search area relative to the compressed pixel position [Nohre94], [Mart98], namely *free tree* (see Fig. 4).

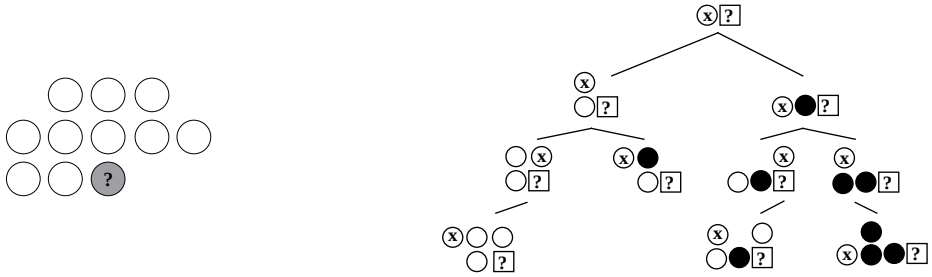

Figure 4: Locations of the context pixels: predefined (left) and optimized (right).

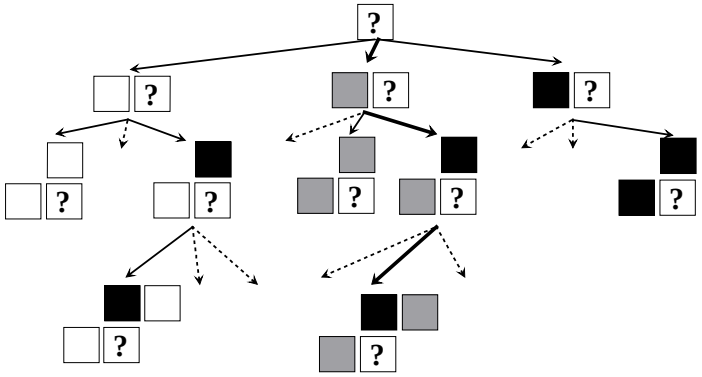

Figure 5: A small example of the incomplete trinary context tree. Instances of traversing are marked with a bold line.

The context tree is used for compression in the same way like fixed-size context templates, only the context selection is different. Context selection is done by traversing the context tree from the root to the *terminal node*, each time selecting the branch according to the corresponding neighbor pixel colors, as shown in Fig. 5. The traversing stops if it comes to a leaf or if there is no outgoing branch, corresponding to the next neighbor pixel color.

Context-tree-based compression consists of two phases: the construction of the context tree and image encoding. The tree can be used in a *static* manner, when the context tree is constructed for a training image, and then used for the compression of images with similar properties [Fränti99]. The context tree can also be optimized directly for the encoded image [Nohre94] [Mart98]. In this case it must be stored in the compressed file.

Context tree construction consists of two main phases: initialization of the context tree, and pruning of the constructed tree.

To construct an initial context tree for an input image, we need to process through the image data to collect statistics for all potential contexts, leaves and internal nodes. Each node stores information on the counts of each pixel value that appears in this particular context. Ragnar Nohre [Nohre94] introduced an exponential-memory algorithm, but this algorithm was not applicable to practical tasks due to its huge memory requirements. Another algorithm for constructing a context tree was proposed in [Helf98]. That algorithm has linear time and memory requirements in respect to the number of pixels in the image.

After collecting the statistics for all possible contexts, the context tree T must be pruned by comparing every node w against its children nodes $\{w_i\}$ to find the optimal combination of siblings. The number of bits required for describing each node of the context tree is shown below, where the size of the image palette is denoted as α

$$c(w) = \begin{cases} 1, & \text{if } T \text{ is full} \\ \alpha, & \text{if } T \text{ is incomplete.} \end{cases} \tag{4}$$

Let' denote the set of all terminal nodes of the tree T as $S(T)$. For each node $w \in S(T)$, the count of the color index i is denoted as $n_i(w)$. The estimated code length generated by a terminal node $w \in S(T)$ is calculated using the following expression

[Wein95, Mart98]

$$c_T\left(n_1(w),\ldots,n_\alpha(w)\right) = -\log_2 \frac{\prod_{i=1}^{\alpha}\prod_{j=0}^{n_i(w)-1}\left(j+\varepsilon\right)}{\prod_{j=0}^{n_1(w)+\ldots+n_\alpha(w)-1}\left(j+\alpha\cdot\varepsilon\right)}, \quad (5)$$

where ε is a constant. The aim of context tree pruning is to find a tree structure that will minimize the following function

$$L(T) = \sum_{w\in T} c(w) + \sum_{w\in S(T)} c_T\left(n_1(w), n_2(w),\ldots,n_\alpha(w)\right), \quad (6)$$

where the first term gives the storage cost of the tree structure, and the second term is the estimated number of bits produced during the compression of the image using this context tree.

A major difficulty in the pruning of the α-ary incomplete context tree is that the number of all possible variants of the prunings of each node is $O(2^{\alpha})$, as shown in Figure 6. Due to this fact, efficient construction of the optimal α-ary incomplete context tree is still an open problem [Mart04]. All existing algorithms for context tree pruning operate on full trees, where each node has α outgoing branches or does not have them at all. In publication **P2**, we propose a solution, which uses a two-stage algorithm, for the efficient construction of an α-ary incomplete context tree.

In our algorithm, pruning can be done in a *bottom-up* [Nohre94] or *top-down* manner [Furl91, Fränti99]. In the top-down approach, the context tree is constructed level-by-level. It compares the children nodes with their parent and prunes them out if the addition of the new children would increase (6). The process continues until a predefined depth is achieved, or when no new nodes are created during the process. The bottom-up approach constructs the full tree up to a predefined depth and then analyzes the tree from leaves to the root. The sub-trees of the nodes that increase (6)

are pruned from the tree, which is similar to the way that new subtrees are pruned in the top-down approach.

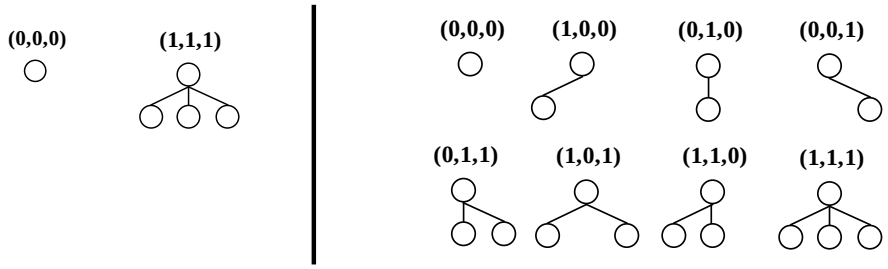

Figure 6: Possible variants of ternary tree pruning for a full tree (left) and an incomplete tree (right). The numbers of possible variants are 2 and 8, correspondingly.

2.5 Algorithms based on binary context modeling

JBIG is an ISO/ITU lossless binary image compression standard [ITU-T T.82], based on the context modeling and arithmetic coding of the QM-Coder [Penn88]. This coder was specially developed for the encoding of binary data and it provides a table-driven technique for updating a probability estimation. The context in JBIG is defined according to the combination of neighboring pixels in locations predefined by a context template (see Fig. 7).

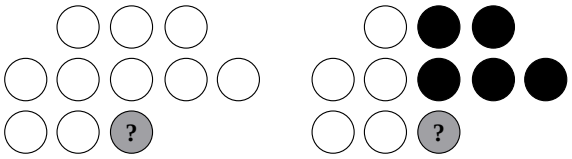

Figure 7: The default 10-pixel context template, which is used in JBIG (left). An example of a context uniquely defined by the pixel configuration: “0101010101” (right).

Although it has been designed primarily for the encoding of binary images, JBIG is capable of compressing color and grayscale images up to a reasonable depth (e.g. 8 bits per pixel). The compression of a 256-color image will be processed by separating the image into eight bit planes followed by separate JBIG. A grayscale image can be preprocessed with a Gray-coding algorithm [Gray53] to normalize the changes between adjacent byte values in image data. This process increases the efficiency of the JBIG encoder.

JBIG2 [Howa98] is a compression standard [ITU-T T.88] for binary images, which extends JBIG by incorporating two pattern matching strategies: *Pattern Matching and Substitution* (PM&S) and *Soft Pattern Matching* (SPM).

PM&S operates by first segmenting an image into blocks and then searching the dictionary in order to locate a previously coded block that matches the current block. If an acceptable match is found, the associated dictionary index and the position offset are encoded. If there is no acceptable match, the current pixel block is encoded and its index appended to the dictionary. This strategy allows a high level of lossy compression to be achieved.

SPM differs from PM&S in that, in addition to the dictionary index and position offset, the current block of pixel data, called *refinement data*, is encoded without losses. A two-layer coder makes use of previously coded pixels from a matched block employing a context template, consisting of two binary layers. Since these blocks match each other, the similarities between them allow the current block to be very efficiently compressed. The inclusion of the refinement data enable the original pixel to be losslessly reconstructed.

Context pixels are chosen using a 4-pixel template in the currently encoded block and 6 pixels in the pattern image, see Figure 8.

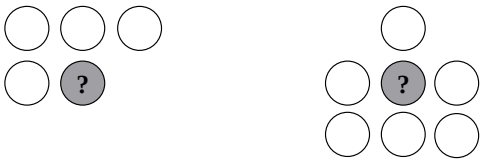

Figure 8: The 2-layer context template used in SPM encoding. The context pixels of the current block are on the left and the pixels from the matched block are on the right. The positions, shown in gray, are aligned.

The *Embedded Image-Domain Adaptive Compression* algorithm (EIDAC) was introduced in [Yoo98]. The given compression algorithm processes the bit-planes of the compressed image from the most significant bit (MSB) plane to the least significant bit (LSB) plane, see Figure 9. The bits are encoded through context modeling and arithmetic coding.

EIDAC uses a binary multilayer context. The context is defined by the neighboring values of the current bitplane, namely C_{intra}, and the previous bit planes, namely C_{intro}, see Figure 10. Note that this illustration shows only a simple context model. The configuration of the bits in C_{intra} and C_{intro} could be defined at will, depending on the particular properties of the image.

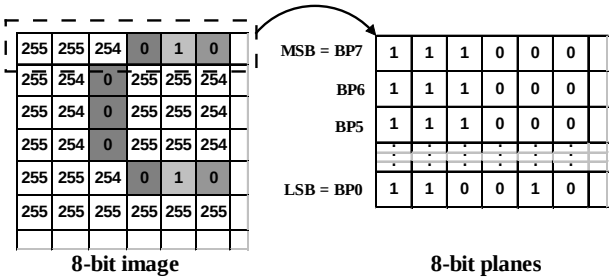

Figure 9: Bit-plane-oriented compression in the EIDAC algorithm [Yoo98].

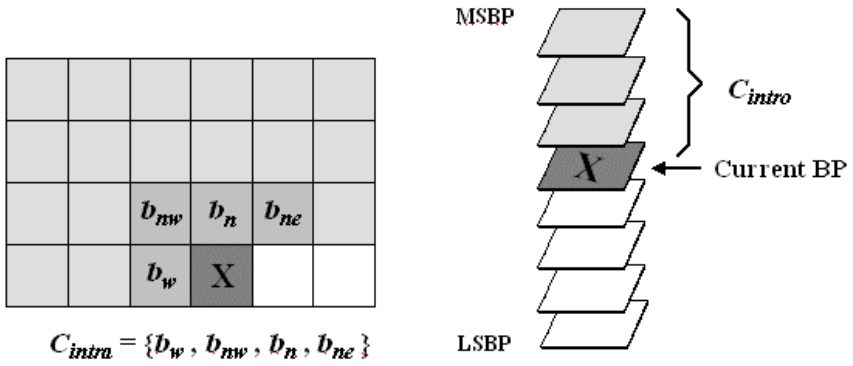

Figure 10: A context template used in the EIDAC algorithm [Yoo98]: C_{intra} (left) and C_{intro} (right).

Multi-Layered Context Tree (MCT) modeling for encoding digital map images was proposed in [Kopyl05]. The encoding was done by separating the map image into binary layers and then by using binary context tree modeling. The separation could be done through *color separation*, where each layer corresponds to one color in the encoded image, or through *semantic-separation*, where each layer corresponds to a semantic layer of the map, such as water and field layers. The second type of separation requires that the encoder have semantic information for the map beforehand.

The MCT algorithm optimizes the context template through the *free tree* technique. Improved compression is obtained by using a *reference layer*: for each currently encoded layer the pixels of the previously encoded layer are also used in the construction of the optimized context template. The authors proposed using *optimal ordering* of the binary layers to increase compression efficiency. For example, a coastline layer strongly depends on the water layer and so on. This problem is solved by constructing a *cost matrix* of the dependent compression, and finding the *minimum spanning tree* in the graph based on this cost matrix.

2.6 The runs of adaptive patterns algorithm

The *Runs of Adaptive Patterns* (RAPR) compression method was introduced in [Ratn98]. That algorithm is based on the context of *patterns*, not on the values of the individual pixels. A specialized *basic pattern* is determined for each pixel according to its four closest neighbors. A set of basic patterns consists of 15 possible ways of labeling these pixels with the four most common labels. Each basic pattern is defined by a string of four letters, identifying the *W*, *NW*, *N* and *NE* neighbors. The basic pattern is defined by the number of different colors in the neighbor pixels and their order. Denoted by the letters *A*, *B*, *C* and *D*, the set of basic patterns is: {*AAAA*, *AAAB*, *AABA*, *ABAA*, *ABBB*, *AABB*, *ABAB*, *ABBA*, *AABC*, *ABAC*, *ABCA*, *ABBC*, *ABCC*, *ABCD*}, see Figures 11 and 12. For instance: if the color of the *W*, *NW* and *NE* pixels are the same, and the color of the *N* pixel is different, then this case is defined by the pattern *AABA*.

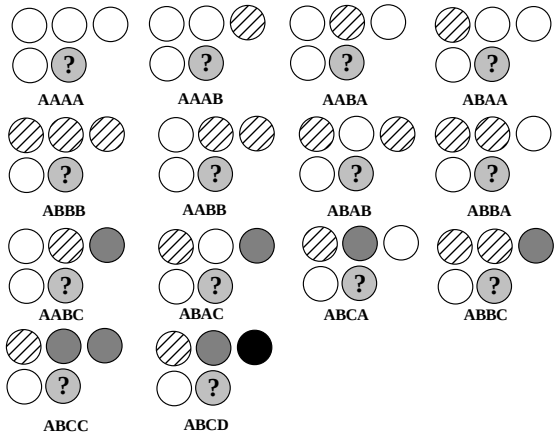

Figure 11: An example of the set of basic patterns in RAPR algorithm.

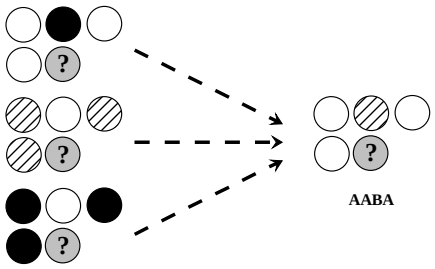

Figure 12: The correspondence between different combinations of colors and a basic pattern.

The set of RAPR patterns, or the *augmented patterns*, is based on the combination of the *basic patterns* and the uniform runs of colors along four directions (see Fig. 13). A pattern defines the adaptive prediction rules, which are used in a binary manner: the current pixel is the same as the most probable symbol of the current pattern. If the encoded pixel value is present in the current basic pattern, then its index is sent to the encoder. Otherwise, a special symbol *ANOMALY* is coded and the value of the pixel is sent as supplementary data. The encoding is done through variable-length encoding, such as arithmetic or Huffman coding. The supplementary data is encoded by the DEFLATE algorithm [Deut96a].

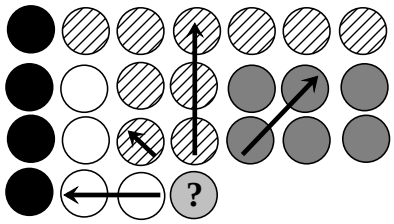

Figure 13: The augmented context describing the pattern {<A, B, B, C>, <2, 1, 3, 2>}.

2.7 Skip pixel coding

Jensen and Forchhammer [Forch02a] introduced content-layer compression of *layered images*. Layered images are composed of a number of layers, each representing one type of information, such as roads, buildings and text. Efficient

coding of layered images is achieved by utilizing inter-layer dependencies through two approaches: the first one uses pixels from the layers with higher priority in the context modeling; the second one uses *SKIP* pixel coding [Forch02a]. In a particular layer, if a given pixel has already been coded in a layer of higher priority, it does not need to be coded in the current layer or any of the other layers with lower priority. Encoding of the layers is done with JBIG2. It is done in a free tree manner for bi-level layers, and in an RAPR manner for multi-level layers.

2.8 The piecewise-constant image model

Piecewise-Constant Image Model (PWC) [Ausb00] is an algorithm for the compression of palette images. It establishes boundaries between constant color pieces and determines the domain colors using the following object-based language:

- D1: Is the color of the current pixel identical to a rectilinearly (horizontally or vertically) connected neighbor?
- D2: Is the color of the current pixel identical to a diagonally (diagonally left or diagonally right) connected neighbor?
- D3: Is the color of the current pixel identical to an estimated value?
- D4: What is the color of the current pixel?

Boundary information is represented with an *edge map*. This information is encoded with the context modeling scheme proposed by Tate [Tate92] followed by entropy coding.

The latest modification of PWC applies the *skip-innovation* technique. In this technique, the four closest neighbors are checked: if they are of the same color, then the skip-innovation case is considered. If the current pixel has the same color as its

neighbors, then the skip case happens, and the length of the run of the same color is output. Otherwise, it is called *innovation*, and a zero-length code with the value of the encoded pixel is output. The lengths of the skips are encoded by *Golomb-Rice codes* [Golo66] [Rice79].

2.9 The prediction by partial matching algorithm

Prediction by partial matching (PPM) [Clea84] is a fixed-order context-based statistical modeling technique, which blends together several fixed-order contexts to predict the next symbol from the input stream. The combination of contexts is achieved through the use of “escape" probabilities. The context with the largest depth is, by default, the one used for coding. However, if a novel symbol is encountered under this context, then the largest depth context is not used for encoding and an “*escape*" symbol is transmitted to give the decoder a signal that the model has been switched to the context with a smaller depth. This process continues until a model is reached in which the character is not novel, at which point it is encoded in respect to the distribution predicted by that model. To ensure that the process terminates, a model is assumed to be present below the lowest level containing all characters in the coding alphabet.

In [Forch02b] it was proposed that PPM with a 2-D context be used for compression of raster map images. The proposed scheme used content-layer compression with JBIG compression of bi-level text layers and 2-D PPM coding of the background. The authors have shown that the proposed method is more efficient that other object-oriented compression algorithms, such as PWC [Ausb00], for the compression of map images.

Summary

Context-based compression has worked well for the compression of raster maps. But the problem of context dilution, which arises during high-depth context modeling, decreases the performance of context-based compression. The obvious way to avoid this problem is to use variable depth context modeling or to use object based modeling. The context tree approach is the most often-used type of variable depth context modeling. Most of implemented context tree are based on a full tree. If a map image consists of many colors then the context dilution problem becomes a threat. In **P2** we describe a sub-optimal algorithm for incomplete context-tree construction. We show experimentally [P2] that the proposed scheme is at least as efficient as other algorithms. Object-based modeling is an efficient way to reduce the alphabet of encoding data. Such methods encode special object events instead of real pixel values. The combination of object-based modeling with *n*-ary context tree modeling seems to be very promising for further development of algorithms of lossless compression of raster maps and other discrete-tone images.

3. Compression of geographical vector data

A vector map is a type of digital map based on vector graphics. The main advantages of vectors graphics, compared to raster graphics, are the ability to support the zooming operation without loss of data representation quality, control of the level of details, and the ability to process spatial queries such as “What is the object?” or “Find this object” (see Fig. 14). Vector objects can also be placed on top of other objects so that the object below will show through. Because of these advantages, many different vector map formats have been developed over the past few years: RaveGeo™ [RaveGeo], MapTP™ [MapTP], etc.

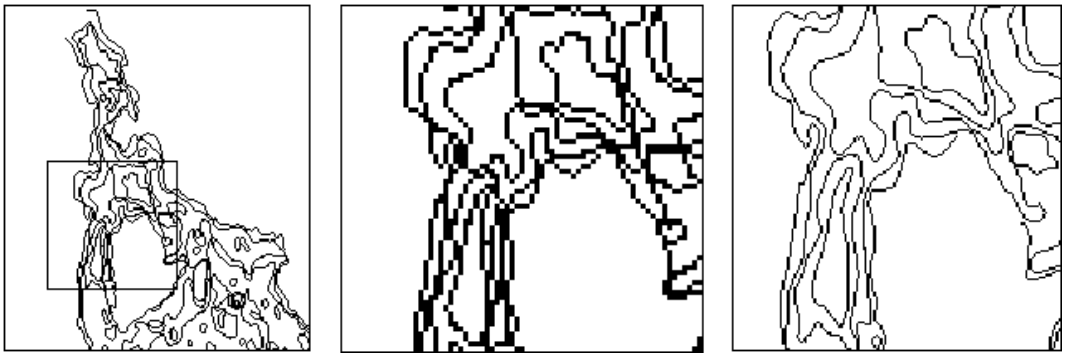

Figure 14: An example of zooming: original map (left), zooming in raster format (center), zooming in vector format (right).

In general, a geographical vector map consists of several semantic layers, which contain different types of geographical information – such as information about water and fields. The layers contain *geometry* and *associated information*. By *geometry*, we mean a series of coordinate pairs (x, y), which form curves and objects that describe the geographical information, and topological information about how

to connect them. The geometrical information keeps the data about coordinates of the objects and curves in the maps. The associated information keeps attributive, color, textual, style and other information associated with the map.

Geospatial data is usually quite large and methods to reduce their size need to be investigated in order to reduce the requirements of clients' portable devices, such as PDAs and mobiles, and to improve the efficiency of data transfer. Methods to reduce data size include *map generalization* [McMa98, Kast89, Zhou00] and data compression. In this thesis, we do not consider the problem of map generalization because it is beyond the scope of this work.

The compression of vector data can be performed in lossless and lossy manners. Lossless compression can be achieved by applying universal lossless compression algorithms, such as PPM [Clea84], LZW [Welch84] or GZip [Deut96b].

Lossy compression can be obtained by reducing the accuracy of the data. In this case, the processed data must satisfy certain accuracy constraints. These constraints could be defined by standards, like the *National Map Accuracy Standards* [NMAS], or by customers' requirements.

3.1 Raster-based compression

Raster-based GIS processes use the following scheme: they receive a request from a client, get a map from a database, and convert the map to a bitmap with predefined dimensions. This bitmap is then sent to the client. This scheme has two advantages: the raster format does not require many computational resources and it can be represented on portable devices with low machine resources, such as mobiles phones.

The process of converting vector graphics into a raster format is called *rasterizing*. It is performed by transforming the original coordinate values to the corresponding

position in the raster, and filling in all the pixels between the end-points of the vector object (in the case of straight line) or the pixels inside (in the case of closed polygons). The compression of raster-based systems can be based on the algorithms for lossless image compression described in Section 2. If a vector map is composed from several semantic layers, then the result of the rasterization can either be one image representing the overall map, or a set of separate layers of images, see Fig. 15.

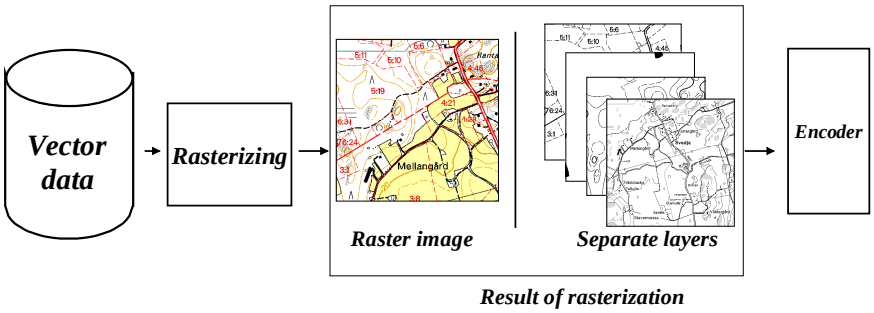

Figure 15: An overall scheme for raster-based compression.

Raster-based vector map encoding can use vector information to improve compression. This idea has been used in [Forch02b] and [Kopyl05].

In [Kopyl05], the authors dealt with the case when a map image is a result of rasterization. They worked with a vector map that consisted of several semantic layers representing different types of geographical information, such as information about water, fields and administrative borders. The semantic layers were represented as black and white images, which were compressed sequentially by using binary context-tree modeling. The context was built up from the pixels of the current and previous layers. They found that the correlation between some types of the data, like between the data from water areas and the data from coastal areas, could significantly improve the efficiency of the pixel prediction during encoding and, correspondingly, the performance of the encoding. The authors showed that the use of the semantic separation is much better than the use of simple color separation.

In [Forch02b], the authors formulated the object of compression as a composite image that is formed from the content layers according to composition rules, which are given beforehand. This approach works well when applied for raster-based compression of vector maps, which could also be composed of several semantic layers. In [Forch02b], layered information is used to prevent the multiple encoding of pixels. If pixels were already encoded in a higher layer, then the same corresponding location of pixels in the lower layers are skipped and not encoded. Thus, they reduce the amount of encoded pixels from layer to layer. This inter-layer dependency is also used for constructing the inter-layer contexts in statistical modeling, as in [Kopyl05].

In publication **P3**, we consider an approach where text and symbols are separated from the rest of the map. A separate symbol dictionary is constructed from the corresponding bitmaps and compressed in a way that it is compatible with the JBIG2 standard. The rest of the information is compressed as a sequence of separate binary layers.

3.2 Compression through chain code representation

An alternative to the coordinate representation of digital contours and shapes on a surface is the *chain code* approach. Chain codes, or the *Freeman code*, were introduced in [Free61]. They are used to represent connectivity between adjusted pixels on a bitmap. If the chain coding scheme is based on 8-connectivity, then a link can be represented by one of eight possible directions and can be denoted by eight different values: *East*, *North-East*, *North*, *North-West*, *West*, *South-West*, *South* and *South-East*.

Alternatively, we can describe a contour by using 4-connectivity, where only transitions in four directions are allowed: *East*, *North*, *West* and *South*. A straightforward encoding of the 8-connected chain codes requires 3 bits per code,

while the 4-connected chain code requires only 2 bits. However, the 4-connected chain code cannot describe diagonal moves efficiently; instead, it needs two codes to represent one diagonal move (see Fig. 16). Therefore, the 4-connected chain codes are 33% longer, on average, than the 8-connected codes [Rosen82].

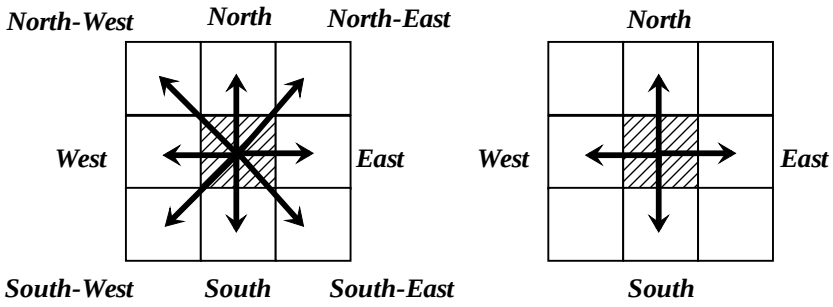

Figure 16: 8-connected (left) and 4-connected (right) chain codes.

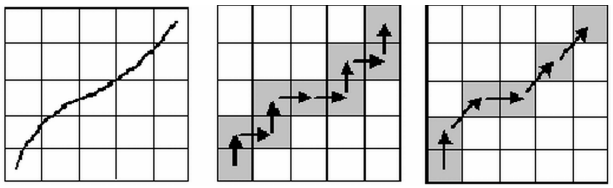

Figure 17: A digital contour (left), described with 4-connected (center) and 8-connected (right) chain codes.

An alternative method of chain code representation is to use differential chain codes, as proposed in [Free74]. In differential chain codes, each chain code is replaced by its difference from the preceding chain code. If a chain code is denoted as c_i and the difference between the codes as $k_i = c_i - c_{i-1}$, then the formulas for obtaining a differential chain code for 4-connected and 8-connected chain codes, respectively, are

$$x_i = \begin{cases} k_i + 4 \text{ if } k_i < -1 \\ k_i - 4 \text{ if } k_i > 2 \\ k_i \text{ otherwise} \end{cases} \quad (7)$$

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

The chain extraction consists of

Step 1: The transformation of coordinates.

Step 2: Chain code extraction.

In the first step, the original vector coordinates are converted to raster coordinates. The dimensions of the raster are predefined. The transformed coordinates represent the starting and ending points of straight lines, which form contours and curves on the raster. In the second step, the chain codes are extracted pixel-by-pixel for each line segment by using Bresenham's algorithm [Bres62]. The resulting sequence of the codes forms the final chain code.

Like raster-based compression, the compression of chain coding is lossy in the sense that it depends on the dimensions of the bitmap, which are used in the final representation of the map.

Encoding of the chain codes consists of two steps: encoding of the chain codes and encoding of the *beginning of chains* (BOC).

3.3 Compression of chain codes

Variable-length chain coding has been proposed in [Liu05]. The authors proposed using fixed Huffman coding to encode differential chain codes. The Huffman codes

are predefined by a code table, which was defined after analyses of more than 1000 curves, contour patterns and various shapes found randomly on the Web [Liu05]. The code table assigns shorter code words to chain codes with smaller turn angles and longer code words to chain codes with larger turn angles.

Kaneko and Okudaira [Kanek85] have proposed an algorithm for the encoding of smooth contours. In their algorithm, first the contours are divided into straight segments, each of which is represented by a sequence of one or two adjacent-directions chain codes (see Fig. 18 [Kanek85]. Within a segment, each chain code must be in one or two adjacent directions, requiring, therefore, only one bit for encoding. The length and the main directions for each segment must also be encoded and transmitted. Thus, the algorithm is more efficient for smooth curves with long segments, but less efficient for rough curves with shorter segments.

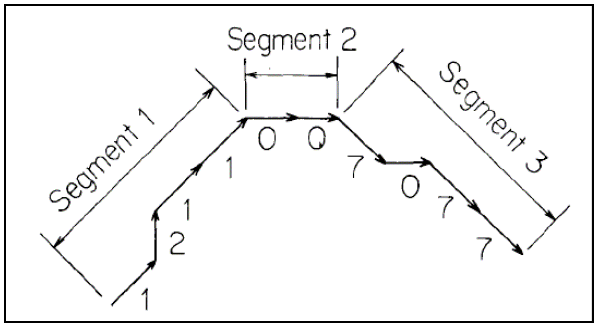

Figure 18: The contour segmentation proposed in [Kanek85]

Lakhani's [Lakh05] suggested encoding chain codes by describing image edges. The author proposed using the LZW algorithm [Welch84] for encoding chain codes, which was an improvement over an arithmetic encoder.

Eden and Kocher [Eden85] suggested encoding 4-connected chain codes based on the analysis of a single code history. They introduced three symbols: turn left, turn right and straight ahead, which were used to represent 4-connected chain codes.

They also mentioned that, except in the case of very short curves, a right turn is never followed immediately by another right turn, and the same applies to left turns. In this way, they reduced the bit rate per code to $\log_2\left(1+\sqrt{2}\right)$.

Lu and Dunham [Lu91] proposed using *context modeling* [Salo00] to predict the probabilities of chain codes. By using Huffman coding, based on the second-order context model, they achieved bit rates similar to that of [Kanek85] but with a significantly simpler implementation. The authors also showed that better results can be obtained if arithmetic coding is used instead of Huffman coding.

Estes and Algazi in [Estes95] proposed using higher order context modeling for the encoding of *crack codes* [Wils90], which are a modification of 4-connected chain codes. The authors extended the context modeling up to the order 8, and used *QM encoder* [Penn88] as the entropy coder. They showed that the use of a high order model leads to better compression performance. A further increase in the context depth, however, leads to the context dilution problem, when small probabilities are distributed over too many contexts and, correspondingly, the efficiency of compression is decreased.

Extending the context-based approach to adaptive context modeling, chain codes can be efficiently encoded with *prediction by partial matching* (PPM) [Clear84], as proposed in [Egger96].

In publication **P4**, we use context tree modeling for lossless compression of chain codes.

3.4 Clustering-based compression

Clustering-based compression (CBC) was introduced in [Shekh02]. The authors proposed using *open-loop fixed-rate quantization* [Gersho91] of the relative

coordinates for lossy compression of vector data. Two different quantization schemes were considered: quantization with a static codebook, and vector quantization. The first quantization scheme uses a static codebook called *Fibonacci, Huffman, Markov* (FHM) [Salo00] (see Fig. 20). In the second scheme, the codebook was generated by the *k-means* algorithm [Linde80], also known as the *Generalized Lloyd Algorithm* (GLA) or the *Linde, Buzo and Gray algorithm* (LBG) [Gray98], see Figures 19 and 20. The main result of [Shekh02], which was presented in a lemma, was that for a given size of codebook, the FHM quantization is looser, in terms of the average distortion, than the k-means algorithm.

In publication **P5**, we perform optimal product scalar quantization, and then fine-tune the result through vector quantization. Optimal quantizer of 1-dimensional (scalar) data can be constructed by Dynamic Programming algorithm [Bruce65] of complexity $O(MN^2)$. Wu [Wu91] reduced complexity of the algorithm to $O(MN)$, using Monge-property of the quantization error.

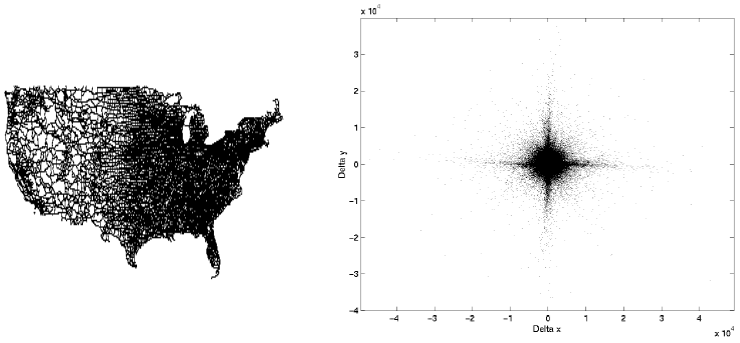

Figure 19: The original map (left) and the corresponding set of relative coordinates (right) [Shekh02].

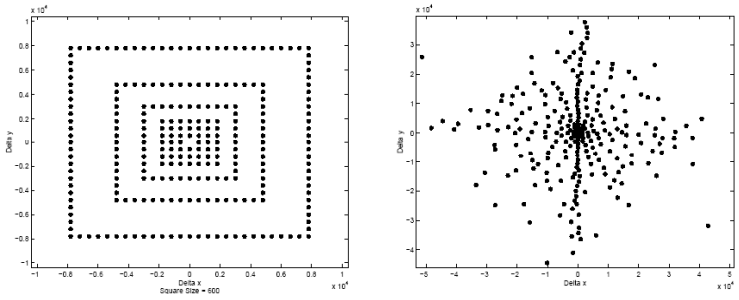

Figure 20: The CBC codebooks [Shekh02]: the FHM codebook (left) and the *k*-means codebook adapted to the encoded data (right).

3.5 Compression through polygonal approximation

Polygonal approximation refers to the approximation of a 2-D piecewise curve through the use of another coarser curve [Koles03a] (see Fig. 21). Polygonal approximation is the simplest geometrical approximation, because only the positions of vertices need to be encoded. Approximation algorithms using curves of higher order are not considered here.

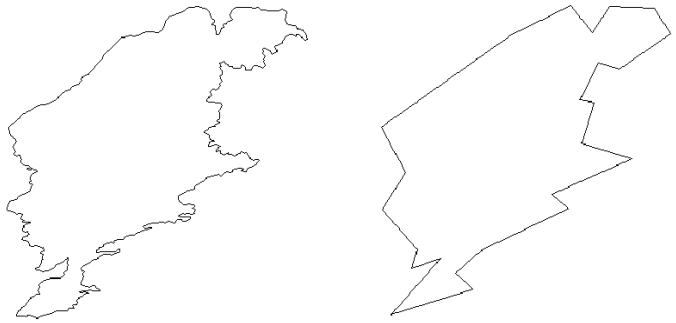

Figure 21: A digital contour and its polygonal approximation.

Compression using polygonal approximation is achieved through the reduction of the number of points. The reduction can be done by any algorithm that minimizes the *approximation error E*(*P*) of an approximated curve *P*.

There are two different types of polygonal approximation:

- *Min-ε problem*: Given a polygonal curve *P*, approximate it by another polygonal curve *Q* with a given number of line segments *M* so that the approximation error *E*(*P*) is minimized.
- *Min-# problem*: Given a polygonal curve *P*, approximate it by another polygonal curve *Q* with the minimum number of segments *M* so that the approximation error *E*(*P*) does not exceed a given maximum tolerance ε.

The most practical error measures in use are based on distance between vertices of the input curve and the approximation of linear segments, see Figure 22.

The additive error measure L_p for a segment $\{p_i, p_j\}$ of the curve $P = \{p_1, \ldots, p_i, \ldots, p_j, \ldots p_N\}$ is defined by the sum of distances $d_k(i, j)$, $k \in (i, \ldots, j)$, for all vertices in the segment

$$e_p(i, j) = \sum_{k=i+1}^{j-1} d_k^p(i, j). \tag{9}$$

For L_∞ the approximation error is defined as

$$e_\infty(i, j) = \max_{i<k<j}\{d_k(i, j)\}. \tag{10}$$

The approximation error of the whole curve in L_p and L_∞ metrics is defined by (9) and (10), respectively.

$$E_p(P) = \sum_{m=1}^{M} e_p(i, j), \tag{11}$$

$$E_\infty(P) = \max_k\{d_k(i, j)\}. \tag{12}$$

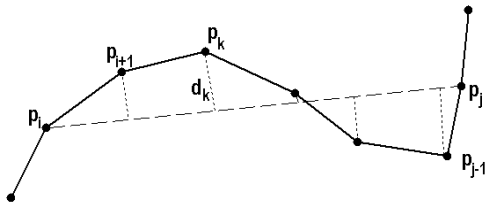

Figure 22: Distances between the points of the curve and the approximating segment $\{p_i,p_j\}$.

A large number of heuristic, sub-optimal and optimal algorithms for polygonal approximation have been developed over the last 30 years [Koles03a].

The most widely used algorithm is a heuristic method called the *Douglas-Peucker algorithm* [Dougl73]. The first approximation is given by only one segment between the end points of the contour. This approximation is then recursively refined by inserting new vertexes on the contour that is furthest away from its current approximation. The procedure stops when the approximation meets a given error constraint. The time complexity of the algorithm is $O(N^2)$ in the worst case, and $O(N\log(N))$ on average.

The optimal algorithm for the *min-* ε approximation was introduced by Perez and Vidal [Perez94]. The proposed algorithm is based on the *dynamic programming* method [Bell57] for solving an optimization task. The complexity of the algorithm with error measure L_2 is $O(NM^2)$.

The optimal solution for the *min-#* problem was published in [Dunh86]. The author proposed using dynamic programming for optimal approximation with the least number of segments for error measure L_∞. The complexity of the algorithm is $O(N^3)$ in the worst case.

Heuristic algorithms are fast, but they are inferior to optimum algorithms. The *Iterative reduced search* approach for *min-*ε and *min-#* was introduced in [Koles02b,

Koles03b] and [Koles02a], respectively. In the proposed approach, the initial approximation for a given error tolerance ε is obtained with any other *min-ε* or *min-#* algorithm, then a bounding corridor is constructed along the reference path, and finally the solution is found through dynamic programming within the bounding corridor. The time complexity of the algorithm is between $O(N)$ and $O(N^2)$, which is similar to the complexity of fast heuristic algorithms.

In publication **P6**, polygonal approximation is used for generating a coarse approximation of the input curve. This reference curve is then used for predictive compression of the original points in a lossless manner.

3.6 Compression using multiresolution information

An important property of modern vector maps formats is that their data can be progressively transmitted. This streaming technique allows objects to be downloaded in lower resolution first and then be displayed while downloading the same objects in higher resolution. This technique minimizes the waiting time for data versus viewing data ratio. On the other hand, the multiresolution approach requires additional information for vector data indexing of the resolution level.

The principles of the progressive encoding of digital curves were first described in [LeBuh97]. Progressive encoding consists of first transmitting the raw polygonal approximation obtained at the upper level, and then transmitting its successive refinements. A series of vertex positions defining the first polygon must be sent first. For refinement polygons, information about the number of child vertices along the coarse polygon edges must be transmitted, so that the decoder can correctly produce the ordered list of vertices. The positions of these refinement vertices can be encoded relative to their parent edge. The order of transmission, for an image containing M approximated contours [LeBuh97], is illustrated in Figure 23.

In encoding and decoding operations, geometrical knowledge about coarse resolutions can be used. In [LeBuh97], the authors proposed using direct encoding on the upper coarse level and letting the more expensive lower levels be encoded relative to their parent approximation. The possible locations of the lower level points are placed in Δ-tunnel along the segments of the coarse approximation, where Δ is the maximum deviation of the coarse approximation, see Figure 24. This information is used by the encoder and, consequently, by the decoder.

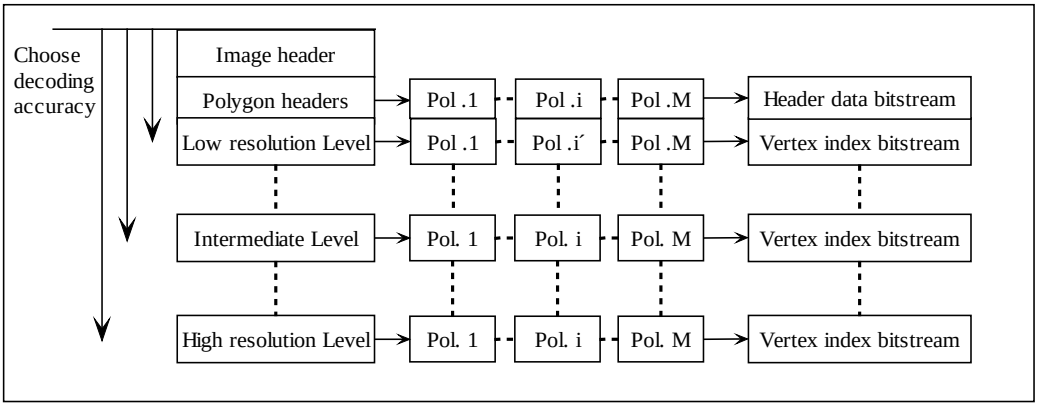

Figure 23: The scheme of progressive encoding of the vector data [LeBuh97].

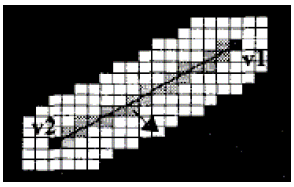

Figure 24: The Δ-tunnel along the segment of the coarse approximation [LeBuh97].

Summary

We consider different approaches of vector map compression. In **P3** we rasterize the vector information and apply context-based compression to the raster data utilizing repetition of symbols. In **P4** we use chain codes to represent the vector data and apply context tree based compression.

In **P5** we develop an optimal product scalar quantizer and use it for lossy compression of vector maps by coordinates quantization. In **P6** we study the problem of lossy compression of multi-resolution vector data. We use the coordinates from lower resolution levels to predict the coordinates of higher resolution layers.

4. Summary of the publications

In the first paper (P1), we propose a LZW based semi-adaptive algorithm of map image compression. The original LZW algorithm is an effective tool for lossless compression of different data. We study the case when the map is compressed beforehand and then transmitted to user. The LZW algorithm is symmetric by its resource demands: the decoder requires same machine resources as the encoder. This could be a bottleneck for user's devices with tiny machine resources, and therefore, we develop a semi-adaptive modification of the algorithm. The semi-adaptive approach is asymmetric: decoding requires fewer resources than encoding. The method also allows user to divide the whole image into rectangular blocks and process and transfer the coded blocks separately from each other. The proposed method gives compression ratio similar to that of GIF or PNG algorithms.

In the second paper (P2), we propose an algorithm for lossless image compression based on the *context tree modeling* [Riss81, Wein95] with incomplete tree structure, namely the *generalized context tree* (GCT) [Mart04]. The context tree is constructed according to the two-pass scheme [Nohre94]: collecting the contexts statistics and pruning of the context tree in order to minimize the *tree costs*. We introduce a near-optimal fast pruning algorithm, which makes the GCT algorithm suitable for on-line processing of map images. We also propose *hybrid-tree*, based on the *free-tree* [Nohre94], which optimizes the context pixels locations according to the encoded image. The proposed algorithm shows improvement over the competitive algorithms by 20% on the set of test images used.

In the third paper (P3), we study the problem of raster-based compression of vector maps. Rasterization is done by converting semantic layers of the vector map into a set of binary images followed by the compression by *JBIG* [ITU-T T.82]. In order to simplify the binary layers, we provide rasterization of the maps without textual information. The associated textual information and their locations are compressed separately. The proposed scheme improves the compression of a single binary layer by 12%, on average, and the whole map by 5%.

In the fourth paper (P4), we focus on lossless compression of *chain codes* [Free61], which are widely used for describing digital contours and raster objects. We extract chain codes from vector and raster maps. The resulting chain code description of the digital contours is compressed by the context-based method developed in **P2**. For vector data, this approach cannot be called lossless because the chain codes are a result of precision reduction of the data in order to fit the vector data to the raster image with some predefined dimensions. If the dimensions are large enough, then the losses will be small or even no loss at all. The proposed approach gives the best compression over the competitive methods on the test dataset. The improvement is about 2-3% in comparison to the *Prediction by Partial Matching* (PPM) algorithm [Clea84], and 40% in comparison to Huffman coding [Liu05].

In the fifth paper (P5), we study coordinate quantization by product quantizer in Cartesian and polar spaces, namely product quantizer (PQ) and strictly polar quantizer (SPQ). The previously introduced algorithms use the Max-Lloyd algorithm [Vora94], which cannot guarantee optimality. Other authors [Buck79, Moo98a, Moo98b, Pearl79, Peric02, Vasil99] use numerical analysis for constructing the quantizer. The numerical analyses are made under the assumption of uniform phase distribution. This assumption provides optimal solution only for the case of 2-D Gaussian distribution of the signal. We propose to use optimal scalar

quantization for constructing the scalar quantizers for PQ and SPQ. This approach is applied to the compression of relative coordinates of vector map and it gives better rate-distortion performance.

In the sixth paper (P6), we consider quantization based lossy compression of vector data. The main idea of the proposed approach is to use the approximation to reduce redundancy of input vector data. For each digital curve, we construct a coarse approximation, which is used then for improved prediction of the coordinates. The residual vectors are encoded by vector quantization. The proposed approach slightly loses to the DPCM algorithm. However, the proposed method allows to store a rough representation for low-resolution mode at the cost of a small increase in the bit rate.

The contributions of the author in these publications can be briefly summarized as follows. In **P2** and **P4**, the author is responsible for the development, implementation, running of the experiments, and writing of the paper. In **P1**, **P3**, **P5** and **P6,** the author took part in the development of the algorithm, took significant part in implementation, running experiments and writing of the papers.

5. Conclusions

We have studied the compression of raster and vector maps. We proposed several approaches in the framework of lossless compression of raster map images, and lossy compression of vector maps.

Semi-adaptive LZW algorithm was developed in order to minimize machine requirements of the decoder. The original LZW algorithm is adaptive, which means that the encoder and decoder require equal amount of memory. In the case of semi-adaptive modification, the generation of the dictionary is done during the encoding only. The decoder does not need to generate it again, which requires less memory and processing time. The proposed approach shows good compression performance in comparison to other dictionary-based algorithms.

We have studied the context-based compression of map images. The main problem of the context-based compression is context dilution, when the number of context models became too big. To prevent this happen we use the context tree modeling. All existing algorithms use context tree modeling with full tree structure. This approach makes the context tree inefficient in case of encoding of images with large number of colors, because it leads to the same problem of context dilution. Because of this, recent works have considered the case of binary tree only. We have proposed the compression algorithm based on an incomplete *n*-ary context tree. The main design problem of the context tree modeling with incomplete tree structure is the pruning algorithm, which can be very time-consuming. We have solved this problem by introducing a recursive-search pruning algorithm. The proposed algorithm

showed excellent compression performance for map images in comparison with other algorithms. A future extension of this work should consider the combination of the incomplete context tree modeling with the object-based modeling.

For lossy compression of vector maps, we have considered raster-based compression, compression by quantization of coordinates, and multi-resolution compression.

First, we use the vector information to simplify the rasterized vector map in raster-based compression. We considered compression by JBIG algorithm, where binary layers represent semantic layers of the encoded map. The simplification was obtained by extracting of textual information, which was encoded separately from the rest. Experiments showed good performance of the proposed scheme.

We have proposed an optimal algorithm for constructing product quantizer in Cartesian and in polar spaces. We used the dynamic programming approach to construct scalar quantizer instead of heuristic ones, like Max-Lloyd. The proposed optimal product quantizer was applied for the compression of vector maps.

We studied compression of vector map contours by chain coding, which are widely used for describing digital contours and raster objects. The existing methods of chain codes compression use the statistical correlation between the currently encoded chain code and the history of previously encoded codes, also known as the context modeling. We proposed to use the context tree modeling in order to prevent the problem of context dilution.

For lossy vector map coding, we proposed to generate a coarse approximation and used it as a reference line for coordinate transformation. We also proposed several methods for prediction of the coordinates by using the values of the coarse approximation. The improved prediction let us narrow the quantized set of prediction errors and improve the compression performance in rate-distortion sense.

Future extension would use coordinate quantization combined with polygonal approximation and entropy encoding of the relative coordinates. The approximation could be made on vertices of near-optimal distortion-constrained scalar quantizer.

References

[NMAS] http://nationalmap.gov/gio/standards/

[Ausb00] P. Ausbeck, “The piecewise-constant image model”, *Proceedings of the IEEE*, vol. 88 (11), pp. 1779-1789, 2000.

[Bell57] R. Bellman, *Dynamic Programming*, Princeton University Press, New Jersey, 1957.

[Bres62] J. Bresenham, “Algorithm for computer control of a digital plotter”, *IBM Systems Journal*, vol. 4(1), pp. 25–30, 1965.

[Bruce65] J. Bruce, “*Optimum quantization*”, Technical Report, MIT Research Laboratory of Electronics, 1965.

[Buck79] J. Bucklew, N. Gallagher, “Two-dimensional quantization of bivariate circularly symmetric densities”, *IEEE Transaction on Information Theory*, vol. 25(6), pp. 667-671, 1979.

[Chou89] P. Chou, T. Lookabaugh, R. Gray, “Entropy-constrained vector quantization”, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 37(1), pp. 31–42, 1989.

[Clea84] J. Cleary, I. Witten, “Data compression using adaptive coding and partial string matching”, *IEEE Trans. on Communications*, vol. 32(4), pp. 396–402, 1984.

[Deut96a] P. Deutsch, “DEFLATE compressed data format specification”, RFC1951, http://www.faqs.org/rfcs/rfc1951.html, 1996.

[Deut96b] P. Deutsch, “GZIP file format specification”, RFC1952, http://www.faqs.org/rfcs/rfc1952.html, 1996.

[Dougl73] D. Douglas, T. Peucker, “Algorithm for the reduction of the

number of points required to represent a line or its caricature", *The Canadian Cartographer*, vol. 10 (2), pp. 112–122, 1973.

[Dunh86] J. Dunham, "Optimum uniform piecewise linear approximation of planar curves", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8(1), pp. 67–75, 1986.

[Eden85] M. Eden, M Kocher, "On the performance of a contour coding algorithm in the context of image coding part I: Contour segment coding", *Signal Processing*, vol. 8(10), pp. 381–386, 1985.

[Egger96] O. Egger, F. Boseen, T. Ebrahimi, "Region based coding scheme with scalability features", Proceedings of the VIII European Signal Processing Conference, vol. 2, pp. 747–750, 1996.

[Estes95] R. Estes, R. Algazi, Efficient error free encoding of binary documents, Proceedings of Data Compression Conference, pp. 122–131, 1995.

[Forch02a] S. Forchhammer, O. Jensen, "Content layer progressive coding of digital maps", *IEEE Trans. on Image Processing*, vol. 11(12), pp. 1349–1356, 2002.

[Forch02b] S. Forchhammer, J. Salinas, "Progressive coding of palette images and digital maps", *IEEE Proceedings of Data Compression Conference*, pp. 362–371, 2002.

[Fred60] E. Fredkin. "Trie Memory", *Communication of the ACM*, vol. 3(9) pp. 490–499 1960

[Free61] H. Freeman, "On encoding of arbitrary geometric configuration", *IRE Trans. on Electronic Computers*, EC-10, pp. 260–268, 1961.

[Free74] H. Freeman, "Computer processing of line drawing images", *ACM Computing Surveys*, vol. 6, March 1974, pp. 57–59.

[Fränti99] P. Fränti, E. Ageenko, “On the use of context tree for binary image compression”, *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 752–756, 1999.

[Furl91] G. Furlan, ”An enhancement to universal modeling algorithm context for real-time applications to image compression”, *International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 2777–2780, 1991.

[Gall78] R. Gallager, “Variations on a theme by Huffman”, *IEEE Trans. on Information Theory*, vol. 24(6), pp. 668–674, 1978.

[Gersho91] A. Gersho, R. Gray, *Vector Quantization and Signal Compression*, Boston, MA: Kluwer, 1991.

[GIF] Graphics Interchange Format(sm), Version 89a, http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/GIF89a.txt, 1990.

[Golo66] S. Golomb, “Run-length encodings” *IEEE Trans. on Information Theory*, vol. 12(3), pp. 399–401, 1966.

[Gray98] R.Gray, D. Neuhoff, ”Quantization”, IEEE Trans. on Information Theory, vol. 44(6), pp. 2325–2383, 1998.

[Grøn95] H. Grønning, T. Ramstad, "2-dimensional scalar quantization with performance close to vector quantization", *Proceedings of the Norvegian Signal Processing Symposium*, Stavanger, Norway, pp. 199–204, 1995.

[Helf98] H. Helfgott, M. Cohn, "Linear-time construction of optimal context trees," *Proceedings of the Data Compression Conference*, pp. 369–371, 1998.

[Howa82a] P. Howard, J. Vitter, ”Practical Implementations of Arithmetic

Coding," *Image and Text Compression*, J. A. Storer, ed., Kluwer Academic Publishers, pp. 85–112, 1992.

[Howa82b] P. Howard, J. Vitter, ”Analysis of arithmetic coding for data compression", *Information Processing and Management*, vol. 28, pp. 749–763, 1992.

[Howa93] P. Howard, J. Vitter, “Fast and efficient lossless image compression”, *Proceedings of the Data Compression Conference*, pp.: 351-360, 1993.

[Howa98] P. Howard, F. Kossentini, B. Martins, S. Forchhammer, W. Rucklidge, “The emerging JBIG2 standard”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 8(7), Issue 7, pp. 838–848, 1998.

[Huff52] D. Huffman, "A method for the construction of minimum-redundancy codes", *Proceedings of the I.R.E.*, 1952, pp. 1098–1102, 1952.

[Hui01] D. Hui, D. Neuhoff, “Asymptotic analysis of optimal fixed-rate uniform scalar quantization”, *IEEE Trans. on Information Theory*, vol. 47(3), March 2001, pp. 957–977, 2001.

[ITU-T T.82] ITU-T recommendation T.82, “Information technology – coded representation of picture and audio information – progressive bi-level image compression”, 1993.

[ITU-T T.87] ITU-T recommendation T.87, “Information technology – lossless and near-lossless compression of continuous-tone still images”, 1999.

[ITU-T T.88] ITU-T recommendation T.88, “Information technology – coded representation of picture and audio information – lossy/lossless

coding of bi-level images", 2000.

[Kanek85] T. Kaneko, M. Okudaira, "Encoding of arbitrary curves based on chain code representation", *IEEE Trans. on Communications*, vol. 33, pp. 697–707, 1985.

[Kast89] R. Kasturi, R. Fernandez, M. Amlani, W. Feng, "Map data processing in geographical information systems", *Computer*, vol. 22(12), pp. 10–21, 1989.

[Knuth75] D. Knuth, *Fundamental Algorithms: Sorting and Searching*, New York: Addison-Wesley, 1975.

[Koles02a] A. Kolesnikov, P. Fränti, "A fast near-optimal *min-#* polygonal approximation of digitized curves", *Proc. of the IASTED International Conference on Automation, Control and Information Technology*, pp. 418–422, 2002.

[Koles02b] A. Kolesnikov and P. Fränti, "A fast near-optimal algorithm for approximation of polygonal curves", *Int. Conf. on Pattern Recognition* (*ICPR'02*), vol. 4, 335–338, Québec, Canada, August 2002.

[Koles03a] A. Kolesnikov, "*Efficient algorithms for vectorization and polygonal approximation*", PhD Thesis, University of Joensuu, 2003**.**

[Koles03b] A. Kolesnikov, P. Fränti, "Reduced-search dynamic programming for approximation of polygonal curves", *Pattern Recognition Letters*, vol. 24(14), pp.: 2243-2254, 2003.

[Koles05] A. Kolesnikov, "Optimal encoding of vector data with polygonal approximation and vertex quantization", Lecture Notes on Computer Science, vol. 3540, pp. 1186-1195, Springer Verlag,

Berlin 2005.

[Kopyl05] P. Kopylov, P. Fränti, “Compression of map images by multilayer context tree modeling”, *IEEE Trans. on Image Processing*, vol. 11(1), pp. 1-11, 2005.

[Lakh05] G. Lakhani, “Encoding image edges as a curve collection”, *IEEE International Conference on Image Processing*, vol. 2, pp. 734–737, 2005.

[Lang81] G. Langdon, J. Rissanen, “Compression of black-white images with arithmetic coding”, *IEEE Trans. on Communications*, vol. 29(6), pp. 858-867, 1981.

[LeBu97] C. Le Buhan, T. Ebrahimi, “Progressive polygon encoding of shape contours”, *Proceeding of International Conference on Image Processing and Its Applications*, vol. 1, pp. 17–21, 1997.

[Linde80] Y. Linde, A. Buzo, R. Gray, “An algorithm for vector quantizer design”, *IEEE Trans. on Communications*, vol. COM-28, pp. 84–95, 1980.

[Lu91] C. Lu, J. Dunham, ”Highly efficient coding schemes for contour lines based on chain code representation”, *IEEE Trans. on Communications*, vol. 39(10), pp. 1511–1514, 1991.

[Liu05] Y. Liu, B. Žalik, “An efficient chain code with Huffman coding”, *Pattern Recognition*, vol. 38(4), pp. 553–557, 2005.

[Mall98] S. Mallat, *A Wavelet tour of signal processing*, Academic Press, 1999.

[MapTP] http://www.mapsolute.com

[Mart04] A. Martin, G. Seroussi, M. Weinberger, “Linear time universal

coding and time reversal of tree sources via FSM closure" *IEEE Trans. on Information Theory*, vol. 50(7), pp. 1442–1468, 2004.

[Mart79] G. Martin, "An algorithm for removing redundancy from a digitized message", Presented at: *Video and Data Recording Conference*, 1979.

[Mart98] B. Martins, S. Forchhammer, "Tree coding of bi-level images", *IEEE Trans. on Image Processing*, vol. 7(4), pp. 517–528, 1998.

[McMa92] R. McMaster, K. Shea. "Generalization in Cartography", *Association of American Geographers*, Washington, D. C., 1992.

[Meye97] B. Meyer, P. Tischer, "TMW—a new method for lossless image compression", *Proceedings of International Picture Coding Symposium*, 1997.

[Meyer93] Y. Meyer, *Wavelets: Algorithms and Applications*, Society for Industrial and Applied Mathematics, Philadelphia.

[Moo98a] P. Moo, "*Asymptotic Analysis of Lattice-Based Quantization*", PhD Thesis, University of Michigan, 1998.

[Moo98b] P. Moo, D. Neuhoff, "Uniform polar quantization revisited", *Proceedings of IEEE International Symposium on Information Theory*, p. 100, 1998.

[Moo98c] P. Moo, D. Neuhoff, "Polar quantization revisited", submitted to *IEEE Trans. on Information Theory*, 1998.

[Nohre94] R. Nohre, *Topics in descriptive complexity*, PhD Thesis, University of Lingköping, Sweden, 1994.

[Pearl79] W. Pearlman, "Polar quantization of a complex Gaussian random variable", *IEEE Transactions on Communications*, vol. 27(6), pp.

892–899, 1979.

[Pei95] S. Pei, C. Cheng, “Dependent scalar quantization of color images”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 5(2), pp. 124–139, 1995.

[Penn88] W. Pennebaker, J. Mitchell, “Probability estimation for the Q-coder”, *IBM Journal of Research and Development*, vol. 32(6), pp. 737–759, 1988.

[Perez94] J. Perez, E. Vidal, “Optimum polygonal approximation of digitized curves”, *Pattern Recognition Letters*, vol. 15, pp. 743–750, 1994.

[Peric02] Z. Peric, M. Stefanovic, “Asymptotic analysis of optimal uniform polar quantization”, *International Journal of Electronics and Communications*, vol. 56(5), pp. 345–347 2002.

[Pers04] J. Persson, “Streaming of compressed multi-resolution geographic vector data”, *Proc. 12th Int. Conf. on Geoinformatics Geospatial Information Research*, pp. 65–774, 2004.

[PNG] T. Boutell, "PNG (Portable Network Graphics) specification", ftp://ftp.uu.net/graphics/png/documents/

[Ratn98] V. Ratnakar, ”RAPP: Lossless image compression with runs of adaptive pixel patterns”, *Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1251–1255, 1998.

[RaveGeo] http://www.idevio.com/ravegeoinfo.com

[Rice79] R. Rice, "Some Practical Universal Noiseless Coding Techniques", *Jet Propulsion Laboratory Publication* , pp. 79–22, 1979.

[Riss76] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM Journal of Research, Development*, vol. 20(3), pp. 198–203,

1976.

[Riss79] J. Rissanen, G. Langdon, “Arithmetic coding”, *IBM Journal of Research, Development*, vol. 23, pp. 146–168, 1979.

[Riss81] J. Rissanen, G. Langdon, “Universal modeling and coding”, *IEEE Trans. on Information Theory*, vol. IT-27(1) pp. 12–23, 1981.

[Riss83] J. Rissanen, “A universal data compression system”, *IEEE Trans. on Information Theory*, vol. 29(5), pp. 656–664, 1983.

[Rosen82] A. Rosenfeld, A. Kak, *Digital Picture Processing*, vol. 2, Academic Press, New York, USA, 2nd edition, 1982.

[Salo00] D. Salomon, *Data Compression: The Complete Reference*, 2nd Edition, Springer Verlag, 2000.

[Shan48] C. Shannon, “A mathematical theory of communication”, *Bell Syst. Tech. Journal*, vol. 27, pp. 398–403, 1948.

[Shekh02] S. Shekhar, Y. Huang, J. Djugash, “Dictionary design for vector map compression”, *University of Minnesota, Computer Science and Engineering, Technical Report*, TR-02-001, 2002.

[Stor82] J. Storer, T. Szymanski, “*Data compression via textual substitution*”, *Journal of the Association for Computing Machinery (ACM)*, vol. 19(4), pp. 928–951, 1982.

[TIFF] TIFF Revision 6.0, Adobe Developers Association, Adobe System Incorporated. http://www.adobe.com/Support/TechNotes.html, 1992.

[Vasil99] A. Vasilache, I. Tabus, J. Astola, ”A polar quantizer for spherically symmetric sources”, *Proc. Int. Conf. Information Systems Analysis and Synthesis*, vol. 6, pp. 263–270, 1999.

[Wein95] M. Weinberger, J. Rissanen, “A universal finite memory source”, *IEEE Trans. on Information Theory*, vol. 41(3), pp. 643–652, 1995.

[Wein96a] M. Weinberger, G. Seroussi, G. Sapiro, “LOCO-I: a low complexity, context-based, lossless image compression algorithm”, *Proceedings of the Data Compression Conference*, pp. 140–149, 1996.

[Wein96b] M. Weinberger, J. Rissanen, R. Arps, “Application of universal context modeling to lossless compression of gray-scale images”, *IEEE Trans. on Image Processing*, vol. 5(4), pp. 575–586, 1996.

[Welch84] T. Welch, “A Technique for high-performance data compression”, *Computer Magazine*, vol.17(6), pp. 8–19, 1984.

[Wils90] G. Wilson, B. Batchelor, “Algorithm for forming relations between objects in a scene”, *IEE Proceedings Computers and Digital Techniques*, vol. 137(2), pp. 151–153, 1990.

[Wils97] G. Wilson, “Properties of contour codes”, *IEE Proceedings-Vision Image and Signal Processing*, vol. 144(3), pp. 145–149, 1997.

[Witt91] I. Witten, T. Bell, “The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression”, *IEEE Trans. on Information Theory*, vol. 37 (4), pp. 1085–1094, 1991.

[Vora94] S. Voran, L. Scharf, “Polar coordinate quantizers that minimize mean-squared error”, *IEEE Trans. on Signal Processing*, vol. 42, pp. 1559–1563, 1994.

[Wu91] X. Wu, “Optimal quantization by matrix searching”, *Journal of Algorithms*, 12, pp. 663-673, 1991.

[Wu97] X. Wu, N. Memon, ”Context-based, adaptive, lossless image

coding", *IEEE Trans. on Communications*, vol. 45(4), pp. 437–444, 1997.

[Yoo98] Y. Yoo, Y. Kwon, A. Ortega, "Embedded image-domain adaptive compression of simple images", *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers*, vol. 2, pp. 1256–1260, 1998.

[Zhou00] X. Zhou, Y. Zhang, S. Lu, G. Chen, "On spatial information retrieval and database generalization" *International Conference on Digital Libraries: Research and Practice*, pp. 328–334, 2000.

[ZIP] http://www.pkware.com/business_and_developers/developer/appnote/

[Ziv77] J. Ziv, A. Lempel, "A universal algorithm for sequential data compression", *IEEE Trans. on Information Theory*, vol. 23(6), pp. 337–343, 1977.

[Ziv78] J. Ziv, A. Lempel, "Compression of individual sequences via variable-rate coding", *IEEE Trans. on Information Theory*, vol. 24(5), pp. 530–536, 1978.