UNIVERSITY OF JOENSUU
COMPUTER SCIENCE
DISSERTATIONS 8

PAVEL KOPYLOV

PROCESSING AND COMPRESSION OF RASTER MAP IMAGES

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 October 20th, 2004, at 12 noon.

UNIVERSITY OF JOENSUU
2004

Supervisor Professor Pasi Fränti
Department of Computer Science
University of Joensuu
Joensuu, Finland

Reviewers Professor Jukka Teuhola
Department of Information Technology
University of Turku
Turku, Finland

Professor Pekka Toivanen
Department of Information Technology
Lappeenranta University of Technology
Lappeenranta, Finland

Opponent

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

ISBN 952-458-519-7
ISSN 1238-6944
Computing Reviews (1998) Classification: E.4, I.4.1, I.4.2, I.4.3, H.2.8
Yliopistopaino
Joensuu 2004

Processing and compression of raster map images

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

University of Joensuu, Computer Science, Dissertations 8
Joensuu, 2004, 132 pages
ISBN 952-458-519-7, ISSN 1238-6944

Abstract

The thesis is dedicated to study methods for processing raster map images. All these methods are aimed at reducing the total storage size of map images.

First we study the task of color quantization of map images distorted by scanning noise or artifacts, caused by lossy compression such as JPEG. We have developed heuristic methods for improving the visual quality of resulting map images and increasing processing speed of the color quantization algorithm.

Secondly we consider the problem of filtering map images. We develop a method based on context tree modeling. This is a two-pass method, where at the first pass we accumulate statistical information about image spatial structure into the context tree, and at the second pass the gathered statistical information is used in the selection of the noise pixels.

Thirdly we develop a method for compression of map images by multi-layer context tree modeling and arithmetic coding. We separate map images into binary layers by color or semantic separation. We utilize inter-layer dependencies for solving optimal

ordering of the layers. We construct an inter-layer dependency graph and solve optimal branching for this graph.

Finally, we develop a compact and flexible map image storage format, in which the maps are stored as hierarchically separated logical layers. The proposed method allows designing of a real-time map handling system for personal navigation systems.

Acknowledgments

The work of the presented thesis has been carried out at the Computer Science Department, University of Joensuu, Finland, during the years 2000-2004.

I owe many thanks to my supervisor Professor Pasi Fränti for his never-ending patience and guidance throughout my research. I also thank Dr. Alexander Kolesnikov for his comments and guidance. I am thankful to the Head of Department Professor Jussi Parkkinen, to Dr. Simo Juvaste, and all my colleagues for the wonderful atmosphere and working conditions.

I also thank Professor Jukka Teuhola and Professor Pekka Toivanen, the reviewers of this thesis, for their helpful comments and recommendations.

In addition, I would like to express my sincere thanks to Gaetano La Russa for providing Italian coffee, good moods and being involved in checking the language of this thesis.

I owe my thanks to Merja Hyttinen for being always conversable, to Marja-Liisa Makkonen, Viktor Veis, Alexander Akimov, Dr. Eugene Ageenko, Florian Berger, and Sami Gröhn for their infinite support.

Finally, I would like to express my gratitude to East Finland Graduate School in Computer Science (ECSE), Centre for International Mobility (CIMO), National Technology Agency of Finland (TEKES), Kata-Electronics Oy, Oy Arbonaut Ltd., Benefon Oyj and Tikka Communications Oy for their financial support of my studies.

I owe my special thanks to my beloved wife Anja and son Daniel for their unlimited emotional and psychological support.

Joensuu, August 2004

Pavel Kopylov

List of original publications

P1. E.I. Ageenko, P. Kopylov, P. Fränti, "On the size and shape of multi-level context templates for compression of map images", *IEEE International Conference on Image Processing (ICIP'01)*, Thessaloniki, Greece, vol.3, pp. 458-461, October 2001.

P2. P. Kopylov, P. Fränti, "Context tree compression of multi-component map images", *IEEE Proceedings of Data Compression Conference (DCC'02)*, Snowbird, Utah, USA, pp. 212-221, April 2002.

P3. P. Kopylov, P. Fränti, "Compression of map images by multi-layer context tree modeling", *IEEE Transactions on Image Processing*, accepted for publication.

P4. P. Kopylov, P. Fränti, "Filtering of color map images by context tree modeling", *IEEE International Conference on Image Processing (ICIP'04)*, Singapore, October 2004, (to appear).

P5. P. Kopylov, P. Fränti, "Color quantization of map images", *Visualization, Imaging, And Image Processing (VIIP 2004)*, Marbella, Spain, pp. 837-842, September 2004.

P6. P. Fränti, E. Ageenko, P. Kopylov, S. Gröhn, F. Berger, "Compression of map images for real-time applications", *Image and Vision Computing*, accepted for publication.

P7. P. Fränti, P. Kopylov, V. Veis, "Dynamic use of map images in mobile environment", *IEEE International Conference on Image Processing (ICIP'02)*, Rochester, New York, USA, vol. 3, pp. 917-920, September 2002.

Contents

1 Introduction 1

1.1 Map images 2

1.2 Motivation 3

1.3 Contribution of the thesis 5

1.4 Structure of the thesis 8

2 Color quantization of map images 9

2.1 Problem formulation 9

2.2 Creating a palette 10

2.3 Color space 12

2.4 Distortion measure 13

2.5 Number of colors 13

2.6 Preprocessing methods 14

2.7 Summary 15

3 Filtering of map images 16

3.1 Nonlinear filters 16

3.2 Color image filtering 19

3.3 Context tree filter 21

3.4 Summary 23

4 Compression of map images 24

4.1 Lossless compression methods 24

4.1.1 Statistical binary image compression 25

4.1.2 Statistical palette image compression 27

4.1.3 Statistical compression with prediction based context model 28

4.1.4 Object oriented context modeling 29

4.1.5 Multi-layer statistical compression 29

4.2 Optimizing the context template 32

4.3 Context tree modeling 34

4.4 Multi-layer context tree 36
4.5 Optimal layer ordering 36
4.6 Summary 37

5 Map handling 39
5.1 Existing mobile navigation systems 40
5.2 Map Image Storage System 40
5.2.1 Multi-scale representation 41
5.2.2 Different compression strategies 42
5.3 Dynamic map handling 42
5.4 Detailed file structure 43
5.5 Client-server communication 46
5.6 Summary 47

6 Conclusions 48

7 Comparison results of the presented methods 49

References 52

1 Introduction

In this thesis we study methods for map image processing and storing, which allow gaining compact storage size, as well as fast browsing and retrieving capabilities. Map images are computer versions of physical maps. Map images are usually digitized, processed and finally stored in databases, allowing users fast access and retrieval.

We consider the situation when map images are stored in a server side and users are accessing the maps remotely using a portable communication device. This device can be a laptop with wireless access, a PDA or even a cellular phone. The communication system has the following characteristics:

- The server side has enough computational power, and high storage volume.
- The communicational channel has a narrow bandwidth, like in native GSM network, where the bandwidth is limited by 9600 bps.
- The client device has reduced viewing capabilities, limited storage volume and low computational power.

The outlook of the entire system is as follows: map images are processed and stored on the server side. The client obtains the coordinates of its current location using global positioning service (GPS) [Kap96] or mobile positioning service (MPS) [DB99] and sends to the server the request for a map. The server receives the request, locates the proper map and sends it to the client. Finally the client device receives the requested map and displays it (see Fig. 1).

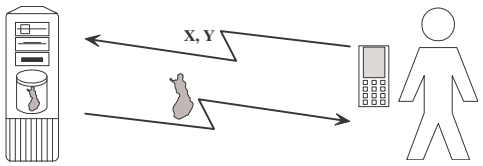


Fig. 1: The illustration of the communication process between client and server

1.1 Map images

Map images and their original hard-copies usually have a set of properties, which remark to the observer a clear distinction between a map images and any other electronic drawing diagram or photographic picture. These properties are:

- Map image usually consists of a limited set of colors.
- Each color used in a map image is used to represent some spatial property of the map object. For example, blue color is usually used to represent water areas, yellow color for fields, and so on.

The map image can be a scanned version of a map or manual digitalization obtained via the use of a pointing device. However, during the map image lifecycle the original color information can be distorted. This can happen because of the changing of the image resolution or because a lossy compression method has been applied. In any case the image can also include additional color artifacts, which can even be visually indistinguishable but still increasing the storage size (see Fig. 2).

Fig. 2: The 64×64 fragment of a map image containing 4 colors (left), and its JPEG compressed version with 1931 colors (right).

Hereinafter we address to the name JPEG as a synonym for the Sequential Baseline System of ISO/ITU compression standard [Wal91].

1.2 Motivation

In this thesis a system for handling the map images is described. In order to provide a comprehensive solution, we propose that the system should have four major parts:

1. *Color quantization* to reduce the amount of colors for true-color map image to a reasonable value.
2. *Filtering* to improve the visual quality and compression ability of the map image by noise removal.
3. *Compression* to reduce the storage size.
4. *Map handling*: map images are stored and transmitted in compressed form so that the system could be useful for mobile navigation applications.

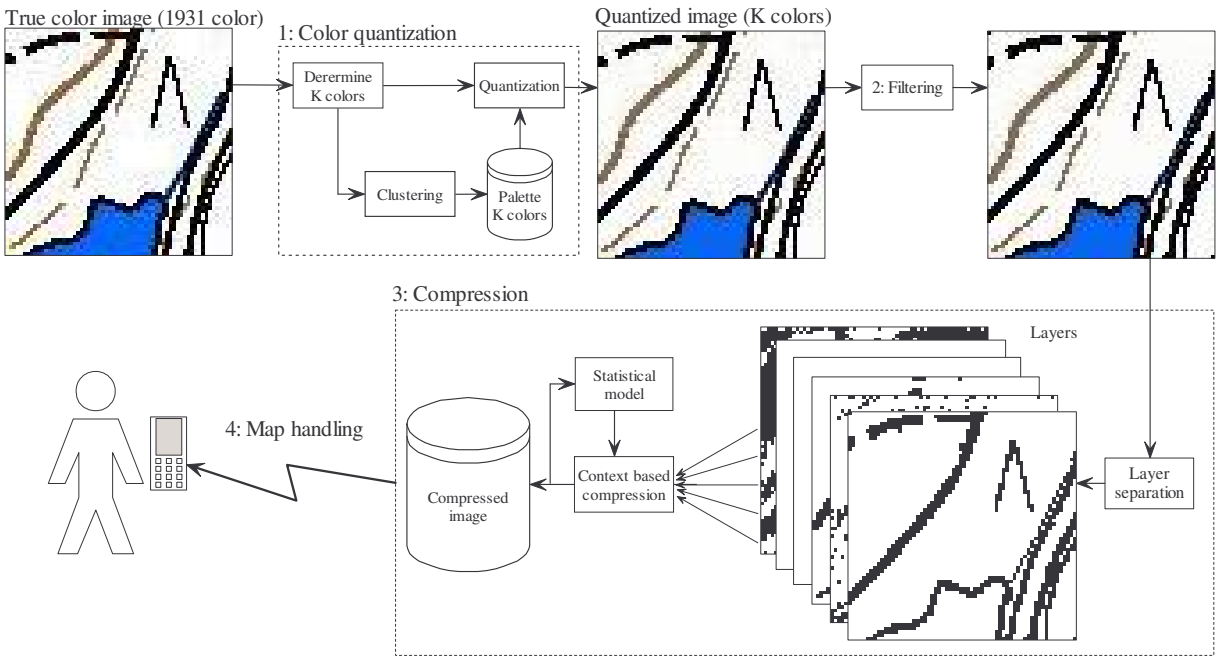


Fig. 3: Illustration of the system diagram.

We consider the color quantization to be the first step of the system. We assume to deal with the input of a distorted true color map image and want to produce, as the output, an image with a reduced palette of colors. Most color quantization methods consider as final palette the size of 256 colors. However, even if 256 color images can be

successfully processed without losses by the existing compression methods, this amount of colors is unacceptable for us because of the following reasons:

- Map images do not use so much color information, (usually there are no more than 16 colors).
- We employ context-based filtering method in which memory requirements grow exponentially with the number of used colors. This renders the use of high number of colors unacceptable in terms of memory consumption.
- The proposed compression method operates with binary layers, therefore the input map image must be first decomposed into binary layers. The use of a high number of colors would cause this method to become unacceptable in terms of processing time.

Filtering of the map image is the second step of the system. We assume that map images originally can contain noise, and we also suppose that some fraction of the noise might still remain after proceeding through the color quantization. Another cause of the image noise is the color quantization method itself. The filtering is usually applied in order to enhance the visual quality of the map image. There are a number of filtering methods that can be applied to the color images, however, the map image filtering requires additional requirements to be set:

- Preservation of the edge information on the map image.
- The filter should not introduce additional colors to the image, because in the compression stage we separate the color image into binary layers. Additional color information might also destroy existing correlations between the layers.

Additionally, removal of the noisy pixels from the image might also improve further the compression ratio.

The third step is compression. Efficient compression has always been an issue for storing and transmission of data. Here we consider a map image as composite image consisting of several binary layers representing semantic or color information about the

map image. The number of efficient lossless image coding methods could be applied for the binary image layers. However, the independent layer consideration will lose the correlation between the information layers; an example of correlation is that water areas on a map are always outlined by ground. Another issue is that most of the existing methods do not take into account the optimal order of binary layers, which should be considered in order to achieve the good compression ratio.

Map handling is the final issue of the proposed system. It can be considered as a part of a compression method where we construct the final storage for the map images. However, the storage system in order to be used in mobile communication environment must comply with specific requirements, such as:

- Compact storage size. This option is crucial for the mobile devices, which have only limited memory resources.
- Direct access. Necessary both for server and for client-side applications in order to speed-up the locating of information without complete search though a database.
- Fast transmission and decompression time.
- Dynamic map handling. The system should have flexible storage structure that allows addition of new information without complete rebuilding of the storage structure.

1.3 Contribution of the thesis

In the first paper (P1) we study a method for estimating optimal context templates, which are used for conditioning the pixel probabilities in context-based image compression. We present an algorithm for estimating an optimal context template for map images containing four binary layers: basic, contours, fields and water. The context templates are trained on map image containing most common spatial properties. The algorithm was applied to estimate the multi-layer context templates. The optimized template improves the compression of field layers by about 12% on average. On the

other hand, the use of multi-layer context template improves the compression for fields and water layers up to 50%.

In the second paper (P2) we extent the idea proposed in **P1** by applying context tree modeling instead of static context template. We optimize the context tree for the individual binary layers, and then apply statistical modeling with the optimized context tree, and arithmetic coding. For estimating inter-layer dependencies we construct the multi-layer context tree, in which the context pixels are located in the current layer, and in the previous layer that has been already coded. The obtained inter-layer dependency information is used to create an ordered compression sequence of the layers. The proposed technique achieves improvements of about 25% over a static context template as in **P1** and 15% over similar single-level context tree.

In the third paper (P3) we propose a method for solving the optimal layer order for the method presented in **P2**. The inter-layer dependencies are acquired by optimizing the context tree for every pair of layers, and put the corresponding file sizes into a cost matrix, which is considered as a graph. The problem of solving the optimal order of the layers is the Minimum Spanning Tree problem for directed graph. We apply Edmond's algorithm [Edm67] for solving optimal branching. We achieve the improvement of 50% over a static context template as in **P1**, and 25% over single-level context tree.

In the fourth paper (**P4**) we focus on the filtering of the map images. The filtering should eliminate noise without destroying map image crucial information. We present a filtering method based on context tree modeling. This is a two-pass method, where at the first pass filter accumulates the spatial information about the image into a context tree, and at the second pass this information is used for conditioning pixel probabilities to decide whether the pixel is noise or not. The proposed context tree filter outperforms vector median filter until about 12% noisy pixels and until about 20% noisy pixels for multi-iteration variant of the context tree filter.

In the fifth paper (P5) we address the problem of color quantization of map images. We consider the map images that have color information distorted by false contouring

or by color artifacts. To approximate the number of colors originally used in the image we employ variance ratio F-Test. We also present methods for increasing the quantization quality and processing speed, which can be used with color quantization for map images.

In the sixth paper (P6) we propose a compact and flexible storage system for the raster map images. It supports partial decomposition of the image, and smooth transitions between various scales. The storage format is constructed as hierarchical tree structure out of logically separated layers, representing the structure of the map image, allowing fast access to the various map image information.

In the seventh paper (P7) we propose a method for handling huge map images on mobile devices having limited memory capacity, small computational power and narrow connection channel. The map images are constructed out of non-intersecting rectangular blocks and stored as compressed rasters in the server, using the system proposed in **P6**, and requested by the client device by demand only. This means that without storing the whole map image in the client device at once, only those blocks that are actually needed and their neighboring blocks are stored. For compression we use the method proposed in **P1**. The result reports that this method is useable for real-time map imaging application for portable devices.

In papers **P1**-**P3,** the author has developed and implemented algorithms for optimizing the context templates, constructing and optimizing the context tree, and implemented the algorithms for optimal branching in **P3**. In papers **P2** and **P3** the author is the principal author. In papers **P4**-**P5** the author has developed and implemented algorithms and also is the principal author of the papers. In paper **P6** the author participated in developing the application, performing the tests and was involved in preparing the publication. In paper **P7** the author, cooperatively with V.Veis, redesigned and developed the system proposed in **P6**, was responsible for performing the tests and was involved in preparing the publication.

1.4 Structure of the thesis

In Chapter 2 we study the problem of color quantization of map images. We consider the situation when the number of colors in the image is unknown, or the image color information has already been distorted. We determine the amount of colors in the image and, as well, we present methods for improving the quality and speed of existing color quantization algorithms.

In Chapter 3 we consider the problem of filtering map images. An essential property of the filter is that it should prevent corrupting the edges by smoothing, and try to retain edge information untouched as much as possible. In the chapter we introduce a context tree filter, which constructs the statistical information about the image and uses this information during the filtering, determining pixel probabilities.

Chapter 4 is dedicated to the problem of compression of map images. We consider map images as compound of several binary layers, representing the separate color or map image semantic information. The method utilizes the inter-layer dependencies, and solves optimal ordering of layers by an algorithm for finding optimal branching.

In Chapter 5 we introduce hierarchical image storage system based on compressed raster format. This system is designed for real-time handling of map images on portable devices with low memory resources and small computational power.

In Chapter 6 we present the conclusions.

Finally, in Chapter 7 we show summary of the results collected in papers P1-P4.

2 Color quantization of map images

True color images might consist of 16 million (2^{24}) colors using 24 bits to represent each color sample, or one byte for each of the primary colors: red, green and blue. It is enough to display such images on the CRT displays. However, the amount of memory used for storing and transmitting these images is very high. It is also known, that the human eye can only distinguish about 300.000 different colors [MM01]. Moreover mobile devices like PDA or mobile phones are often having displays which can show only a limited set of the colors (256 or 65535) at the same time.

2.1 Problem formulation

Color quantization is the process of representing true color images with smaller set of colors while at the same time aiming at maintaining the quality of the image. In general, color quantization algorithms consist of two phases:

1 Creating a color palette (colormap) in which the best possible set of color representatives for an image is selected. This consists of:

- Selecting the clustering algorithm.
- Selecting the color space.
- Selecting the distortion measure.

2 Mapping the original colors to the closest representative from the palette.

The color quantization technique is a special case of a more general class of clustering techniques. Clustering is considered as combinatorial optimization problem, aimed at partitioning a set of data objects into groups, where the objects with similar features should be grouped together and objects with different features placed in separate groups.

In color imaging, the information about color data point is usually represented as three-component vector, each vector component contains a value of according color primary.

Thus, given three-dimensional data there is no efficient solutions known for *k*-means problem, since it is proven to be NP-hard [Bru77].

The clustering problem is defined as follows: Given a set of *N* data objects (x_i), partition the data set into *K* disjoint subsets S_k, called clusters, where similar objects are grouped together and objects with different features belong to different groups, so that the total distortion measure function *E* will be minimized. Usually, as the total distortion function the *MSE* function is used.

$$E = \frac{1}{N}\sum_{i=1}^{N}\left\|x_i, c_{p(i)}\right\|^2,$$

where x_i is the one of image colors, c_k is the representative or palette color and $p(i)$ denotes the partition, or the mapping function $P=\{p_1, \ldots, p_N\}$, which maps original code vectors into constructed clusters. In other words, cluster S_a is defined as the set of data vectors that belong to the same partition *a*.

$$S_a = \{x_i | p_i = a\}$$

2.2 Creating a palette

Hierarchical methods

Bottom-up approach is known as *Pairwise Neighbor Search* (PNN) or *Agglomerative clustering* [Equ89], [War63], [VFK01], [XJ94]. The idea here is as follows: in the beginning, every data-vector represents a single cluster by its own. Then clusters are merged together according to the selected criteria, until the desired number of clusters is reached. The merged clusters are chosen on the basis of inter-cluster distances. The main advantage of these methods is that they are conceptually simple and provide relatively good results [VFK01].

In the top-down approach the idea is as follows: In the beginning all data vectors are placed into a single cluster, which is recursively split until the desired number of clusters is obtained. The clusters are split using specified heuristic rules to minimize the

distortion measure. This process creates a binary tree representing the splitting. Each split is done along a splitting plane separating the cluster into two subclusters. The axis orthogonal to this plane is called splitting axis.

The key issues of top-down clustering approaches are to:

- Select next cluster to split.
- Select the splitting axis.
- Determine the position of the splitting plane along the splitting axis.

The *median-cut* method by Heckbert [Hec82] recursively splits the RGB color space into subsets of equal color population along the longest axis. The *tree-structured color quantizer* by Orchard and Bouman [OB91] determines nodes to split in an attempt to minimize the total distortion measure. The method reported by Wu and Zhang [WZ91] selects the cluster with greatest total variance and perform the split along the greatest variance axis, minimizing the total distortion measure. The *sequential scalar quantizer* method by Balasubramanian, Bouman and Allebach [BBA94] performs the quantization separately on each principle axis. The method by Wu [Wu92] performs the cuts by the halfplanes normal to the principal axis of the data and proceeds with the constrained global optimization of the cutting halfplanes.

Iterative optimization methods

The *k-means* algorithm [McQ67] also known as *generalized Lloyd algorithm* (GLA) [Llo82] or *Linde-Buzo-Gray* (LBG) algorithm [LBG80] starts with an initial solution and iteratively improves it until a local minimum is reached. The *Randomized Local Search* (RLS) [FK00] method is a trial-and-error approach constructed over the *k-means*. *Tabu search* (TS) method [FKN98] is a variant of traditional local search, which prevents an algorithm from getting stuck in local minima by making modifications into a current solution. *Genetic algorithms* (GA) are based on stochastic search, which simulates the biological model of evolution [Gol89]. At each iteration the algorithm generates a set of new solutions by genetic operations such as *crossover* and *mutation.*

Other methods

Methods based on Self Organizing Maps [Koh95], [Dek94], [PAS02] usually aim to represent an input dataset as some kind of an ordered entity, such as two-dimensional array, where similar data-vectors are close to each other. This is achieved by competitive learning. Greedy algorithms [Har75] might be used for creating the initial solution for optimization methods, because greedy algorithms itself tend to produce rather weak solutions. In [PAS02], spatial characteristics of the image are utilized in the clustering in addition to the color values.

2.3 Color space

The choice of distance measure $\|x_i, c_k\|$ between two color samples x_i and c_k is critical for a color quantization algorithm. It completely relies on the choice of the color space. It is desirable for the color space to be perceptually uniform, in which $\|x_0, x_1\| = \|x_1, x_2\|$ if x_0 and x_1 differ as much as x_1 and x_2 in visual sense.

The RGB color space is one of the most popular and widely used color spaces. Its format is the most common for digital images and is compatible to computer devices. It is also easy for computer users to understand and manipulate the primary color components. The advantage of the RGB color space, that it allows the integer calculations, which makes it useful for fast algorithm implementations. The major drawback of the RGB color space and its linear derivates is that it is not perceptually uniform [Equ89], [Hec82].

The YIQ model is used in U.S. commercial color television broadcasting and is a recoding instead of RGB for transmission efficiency and for compatibility with black-and-white television. This compatibility stays in the fact that black-and-white televisions pay attention only to the Y-component of the transmission, which contains relative luminance information. This component gets the majority of the bandwidth in television broadcasting because the human visual system is more sensitive to changes in

luminance than to changes in hue or saturation. However, YIQ is not uniform color space [Wu96].

The CIE $L^*a^*b^*$ and CIE $L^*u^*v^*$ [CIE86] color spaces have been designed to represent differences in color by the Euclidean metric in a psychophysically meaningful way. The uniformity assumption for $L^*a^*b^*$ is based on color matching experiments conducted with relatively large color patches and thus was only ensured for low spatial frequencies. The $L^*a^*b^*$ and $L^*u^*v^*$ color spaces are only approximately uniform because the human perception is also context sensitive and depends on the color values of neighboring pixels. However, the $L^*a^*b^*$ and $L^*u^*v^*$ color spaces are not device-dependent, in contradiction to the RGB and YIQ color spaces [OB91], [Wu92].

The artificially created color spaces are often used to improve quantization heuristics [BB97].

2.4 Distortion measure

Usually the MSE function is used as a total distortion measure of clustering algorithms. However, the weighted distortion measures are proposed to add a specific behavior to the color quantization methods. The weighted distortion measures proposed in [OB91] are tailored to perceive small intensity changes, and to identify the problematic regions where the false contouring may appear. The distortion measure, introduced in [CTM94], takes into account the higher sensitivity of the *Human Visual System* (HVS) to the errors in low activity areas compared to errors in high activity areas. The activity-weighted distortion measure based on the color visual sensitivity and absorbance of the HVS according to each color component in the local region of color image was proposed in [KLL96]. Simplified model of human perception, which incorporates spatial and contextual information, was presented in [PHK00].

2.5 Number of colors

The number of quantized colors is a parameter of color quantization process, it can be a value based on user observation, or value based on empirical data. In our research we

rely on F-Test variance ratio, which is based on statistical ANOVA test procedure [Ito80]. The F-Test variance ratio is useful in estimation of codebook size, which also relies on the geometrical structure of the input data. It is calculated as the ratio of the total within-groups variance against the total between-group variance.

$$F = \frac{k \cdot \sum_{i=1}^{N} \left\| x_i - c_{p(i)} \right\|^2}{\sum_{j=1}^{K} n_j \left\| c_j - \bar{x} \right\|^2} = \frac{k \cdot MSE}{\sigma(X) - MSE},$$

where n_j is the size of the cluster j, $\bar{x}$ is the mean vector of a training set, k is the size of palette, and $\sigma(X)$ is defined as the total variance of the training set and can be decomposed into the sum of within-group variance and between-groups variance as:

$$\sigma(X) = \sum_{i=1}^{N} \left\| x_i - c_{p(i)} \right\|^2 + \sum_{j=1}^{K} n_j \left\| c_j - \bar{x} \right\|^2 .$$

It was shown in [Xu04] that the F-Test variance function can be used in clustering as the total distortion measure.

Thus we define an algorithm for estimating the number of clusters K as follows: We use F-Test variance ratio as the total distortion measure in the clustering algorithm and look for minimal value of evaluation function over all possible K. When the minimal value of the evaluation function is found, the appropriate value of K, and resulting codebook C, are reported.

2.6 Preprocessing methods

False contouring and blurring may cause situation, in which artifact colors are concentrated around the original colors in the color space. Moreover, many color quantization methods do not take into account the spatial and contextual information of the processed image. Thus in **P5** we construct two heuristic preprocessing methods *Color Filtering* and *Activity Thresholding* for improving the quality of color quantization and increasing processing speed:

- In color filtering we address to the situation when in the color space the original color is surrounded by artificial colors caused by false contouring and blurring. We iteratively locate the color peaks and filter the color samples, which fall into a Just Noticeable Difference (JND) [WS82] interval around the color peak.
- In the activity thresholding we make an attempt to utilize the image texture information by calculating for every input pixel a *color activity function* as the color difference between the input pixel color value and the mean color in a local neighborhood. At the second step, when we are reading the image color content, we exclude the pixels, whose color activity value falls into JND interval. In other words, we exclude from the color palette those colors that are different from the background but only by an amount that is not distinguishable in terms of JND.

2.7 Summary

We have presented a method for color quantization of map images. The main set-up for the method is that we do not have any knowledge about the number of real colors used in the image. The method performs the quantization in Uniform CIE L*a*b* color space. It uses F-Test variance ratio as a distortion measure to approximate the number of real colors. We also use color filtering and activity thresholding to improve the quality of the color quantization algorithm and speed-up the process.

3 Filtering of map images

The quality of map images may degrade during the life cycle and digitization process, since the noise introduces artificial unnecessary details to the images. Filtering is usually applied to the image to improve the visual quality of the image or to outline the specific features like edges. The filter is considered as an operator, which computes the pixels for the output image as a function of several pixels in the original image, where the location of these pixels is defined by some neighborhood area.

Traditionally the image filtering is done by *linear filters* [Jai89]. Linear filters are very popular because of their mathematical simplicity and their efficiency in presence of additive Gaussian noise. A mean filter is the optimal filter for the Gaussian noise in the sense of mean square error. Linear filters, however, tend to blur sharp edges, destroy lines and other fine details and perform poorly in presence of signal-dependent noise [HK01].

3.1 Nonlinear filters

Morphological filters [Mat75], [Ser82] were introduced by Matheron. Morphological filters perform a transformation of an image using a set, known as structuring element, which acts as a probe sensitive to geometrical information. Geometrical structures of the image that are similar in shape and size to the structuring element are preserved, while other features are extracted or suppressed. Morphology has been used to perform noise suppression, edge detection, shape analysis, skeletonization for applications in many areas.

Median filter originally were introduced by Tukey [Tuk77]. It performs the filtering by first sorting all the pixel values from a surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. The median filter has two distinctive properties: it efficiently suppress impulsive noise and preserve the edges (see Fig. 4). The *weighted median filter* [Jus81] is a successor of a median

filter. It assigns specific weights to each position in the window to give more flexibility to the filter. Originally median filter performed on grayscale images.

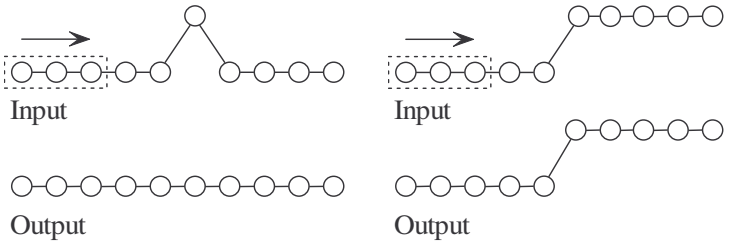


Fig. 4 : Example of median filter behavior in presence of impulsive noise (left); Preservation of the edges (right).

Stack filters were first introduced by Wendt *et al*. in [WCG86] to work on grayscale images. The principle of the stack filter is first to separate the input signal into binary layers using the *threshold decomposition*, this step is performed to divide the design the filtering analysis into smaller and simpler parts. Second, the binary filtering is performed on each binary layer separately, and finally the output is reconstructed back from the binary layers after filtering.

Consider a vector $x = (x_1, x_2, ..., x_N)$, where $x_i \in \{0,1,..., M-1\}$. The threshold decomposition of x is a set of M–1 binary vectors $x^1, x^2, ..., x^{M-1}$, according to the rule

$$x_n^m = T_m(x_n) = \begin{cases} 0, & \text{if } x_n < m, \\ 1, & \text{otherwise} \end{cases}$$

So, an element x_n^k of a binary vector x^k takes the value 1 whenever the element of the input vector x_n is greater than or equal to k. The original value can be reconstructed from binary vectors according to the rule:

$$x_n = \sum_{m=1}^{M-1} x_n^m$$

An example of the stack filter is given in the Fig. 5.

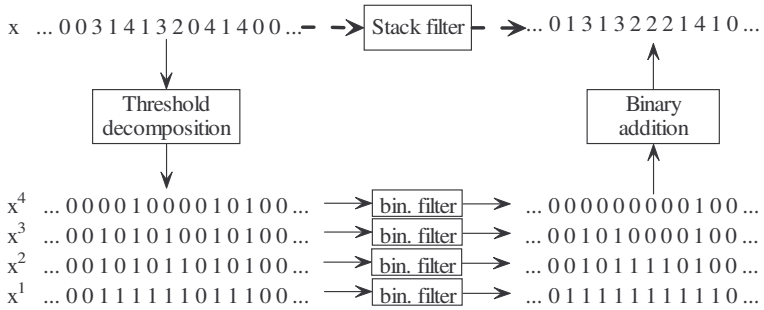


Fig. 5: Illustration of stack filter performance.

Context-based filter and its derivatives [AF00] use context template to calculate the probability of the pixel of being filtered according to its local neighborhood. The principle of context-based filtering corresponds to the adaptive arithmetic coders [LR81], which use context model to accurately estimate the probability of the upcoming symbol. The combination of colors within a local neighborhood template uniquely defines a *context* (see Fig. 6).

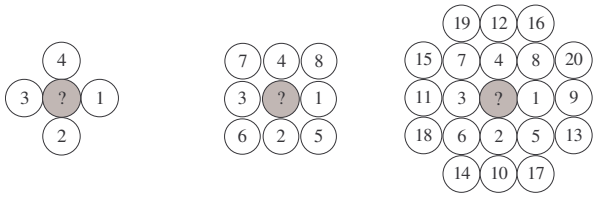
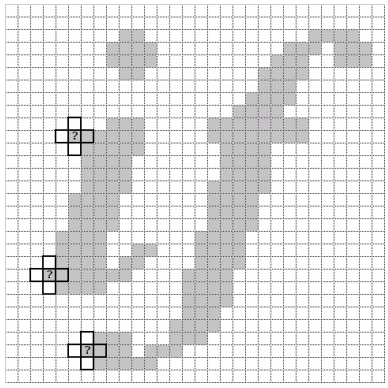


Fig. 6: Context templates: 4 pixel clairvoyant template (left), 8 pixel clairvoyant template (center), and 20 pixel clairvoyant template (right).

The context filter takes two passes over the image. The first pass of filtering process is counting the number of times each color appears in every context, and on the basis of these counters conditional probabilities are estimated. As a result, we have an array of the size equal to the number of all possible contexts filled with information of how many times different colors appear in each context. In the example given in Fig. 7, N_b and N_w are relative counters for appearance of black or white pixel in particular context and P_b and P_w are accordingly their probabilities. P_c is the probability of the context to appear. In practice, only counters N_b and N_w are needed, the other information can be calculated on their base. The selected context on the image corresponds to the highlighted row in the table. At the second pass, for every image pixel we set its context, and if the conditional probability of the pixel is less than a predefined threshold

level, then its color value is changed to the one in that particular context, which delivers the maximum conditional probability.



	Context	N_w	N_b	P_c	P_w	P_b		Context	N_w	N_b	P_c	P_w	P_b
1		562	0	62.%	100%	0%	9		18	0	2%	100%	0%
2		25	3	3.1%	89.3%	10.7%	10		5	6	1.2%	45.4%	54.6%
3		19	0	2.1%	100%	0	11		Not present				
4		18	20	4.2%	47.3%	52.7%	12		0	22	2.4%	0%	100%
5		28	1	3.2%	96.5%	3.5%	13		17	21	4.2%	44.7%	55.3%
6		0	3	0.3%	0%	100%	14		1	8	1%	11.1%	88.9%
7		2	7	1%	22.2%	77.8%	15		1	22	2.6%	4.3%	95.7%
8		2	8	1.1%	20%	80%	16		0	81	9%	0%	100%

Fig. 7: A context model for the "if" binary image.

3.2 Color image filtering

Mostly, map-images appear as color images, thus we have to consider the filtering methods that can deal with color images. Many filtering methods that perform on grayscale images could be easily transformed for color images, performing filtration on each primary color channel separately. However, the information in color channels is generally correlated, and if each component is processed separately, this correlation is not utilized. In the example on the Fig. 8 we show the filtration process of the color signal consisting of two channels, red and green separately, with median filter of length

5. The impulse in the red component is removed, but this causes the edge to be moved by one sample. This is known as *edge jitter* effect [AHN90].

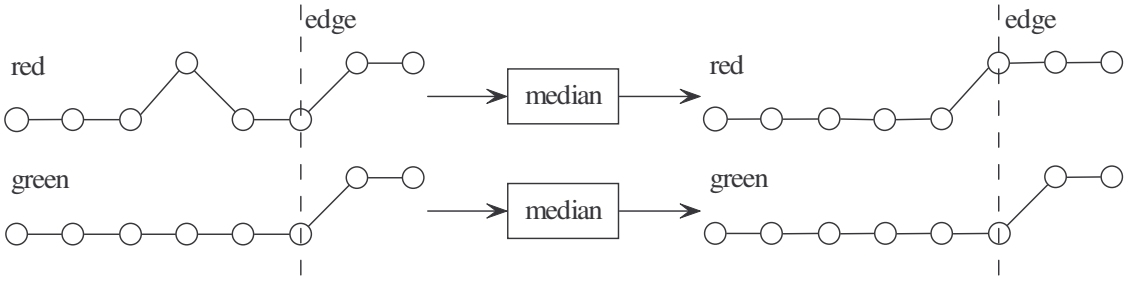


Fig. 8: Example of performing median filter of length 5 for two color channels separately.

The edge jitter effect is unacceptable when filtering map-images, because it will lead to destroy information on the map image. An alternative is to treat the color at each pixel as a vector in three-dimensional space. The general issue of filtering of three-dimensional data is to create the ordering scheme, according to which the values within the filtering window can be sorted. *Vector Median filter* [AHN90] uses the norm of vector in Euclidean space and *Vector Directional filter* [TV93] considers the angle between vectors as ordering criterion. *Rank-Conditioned Vector Median* filter [Luk03] uses specified threshold value to detect noisy pixels. It selects the resulting value from the reduced set of vector ordered statistics. However, the filtering parameters, such as the threshold and the length of reduced set, have to be optimized for every image before the filtering.

The context filter, however, does not need to define the ordering criterion. It can simply be adjusted to process the color images by increasing the array size, which is used for storing the relative counters. The length of the array is defined as $N = K^{M+1}$, where K is the number of colors used in the image and M is the size of the context template. In the case of filtering binary image where the number of colors is 2, with context size of 4, we will have 2^4=16 different contexts, plus for each context we have to allocate space for storing the actual counters. To sum up, we till have two-dimensional array with 2×16 cells.

3.3 Context tree filter

The context filter has one major drawback: it has to allocate memory for storing conditional pixel counters for all possible contexts. But it could happen that the particular context would not be present in the image because of the image spatial properties. Considering the example in Fig. 7, we see that one context did not appear on the image. Moreover if we use the 8-pixel context template from Fig. 6 for the same image, we find that 187 out of 256 contexts are not used at all.

We have proposed to use the tree structure to store the information about filtering statistics. We refer to this structure as context tree. It is constructed as follows:

- The information about contexts is stored in the leaves
- Every tree node has as many branches as there are colors in the image in that particular context
- The children of a node correspond to their parent by adding one more pixel at the position defined by context template (see Fig. 6)
- The context selection is made by traversing the context tree from root to leaf, each time selecting the branch according to the value of the pixel in the corresponding position within context template (see Fig. 9).

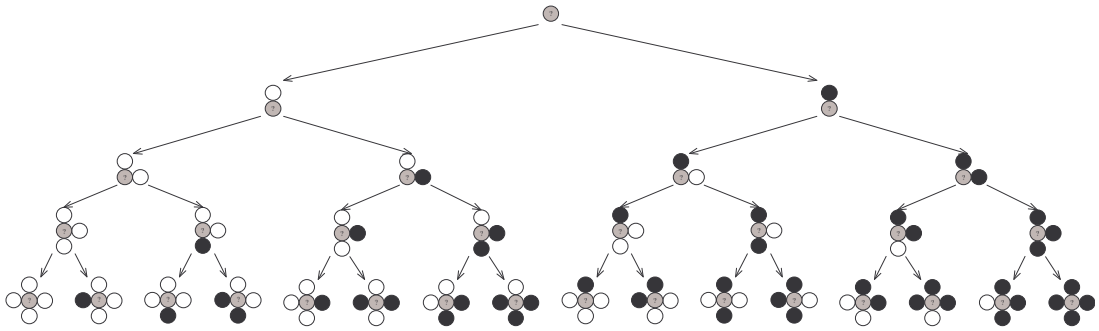


Fig. 9: An example of a context tree with depth of 4.

Generally, the context tree filter differs from context filter only by the data structure used to store the filtering statistics. Also the filtering process has two phases, as in context filter. In the first phase we create the tree and populate it with statistics. In the

second phase we proceed with actual filtering on the basis of the statistics of the context in the same way as done in context filter, with the difference that the context selection is made by traversing the context tree instead of straight-forward context mapping.

We consider two strategies for constructing the context tree: breadth-first and depth-first. In the breadth-first strategy the one-more level of the tree is constructed and filtering statistics for the new constructed level are calculated. Then the process is repeated until the predefined depth of the tree, which is defined by the length of context template, is reached (see left of Fig. 10). The drawback of this strategy is that it is quite slow because at each construction level we have to make pass over the filtering image to calculate the filtering statistics for the current tree level. Also, using this strategy it is impossible to calculate the number of used contexts before the tree is constructed. Considering for example a 16 color image, with a dimension of 1024×1024 pixels, and using 20 pixel clairvoyant template, we expect to have 16^{20} contexts in the tree.

On the other hand, the physical dimensions of the image limit the number of available contexts so that we can have at most 1024×1024 different contexts. Thus, in the depth-first strategy the tree is constructed up to a depth defined by context template while traversing the image in raster scan order. For every pixel in the image we select a corresponding context and update its color counters, if during the selection process an appropriate branch in the tree was not found, we create it (see right of Fig. 10).

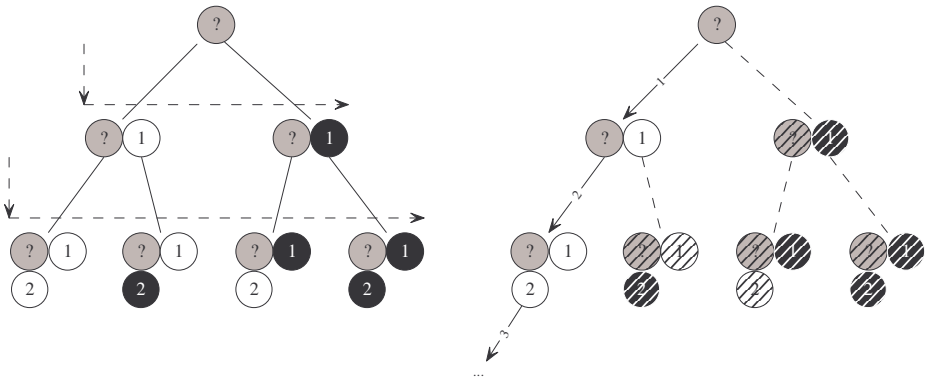


Fig. 10: The example of breadth first (left) and depth first (right) tree construction strategy.

The main disadvantage of the context tree filter is the extensive memory consumption due to storing the filtering statistics and tree structure overhead. This problem is especially noticeable when filtering color images, which have large dimensions. However, the impact of this problem could be reduced when using pre-quantizing step, so that each node, except the leaf nodes, will have some reduced amount of children nodes. Also tree pruning methods could be involved to eliminate the nodes, which use does not give us any difference considering the parent nodes.

3.4 Summary

Statistical filtering method based on context tree structure was presented. The method has wide application area because without any modifications it can be applied to binary, grayscale or color images. It is a two pass filter. At the first pass the image statistics are calculated and stored in the tree structure and at the second pass the actual filtering is performed on the basis of previously collected statistics.

4 Compression of map images

Map images are considered here as *discrete-tone* images with a limited number of colors, having large spatial dimensions. In discrete-tone images pixel intensities do not vary smoothly as in *continuous-tone* images, but change within a small set of values. We also consider map images as composite image consist of several binary layers representing semantic or color information. Map images cannot be compressed well using lossy color compression, such as JPEG [Wal91], because they are not suited for them. Moreover, compressing a map image using JPEG to the same compression level that is achieved with GIF we would have a barely readable picture see Fig. 11. for a comparison.

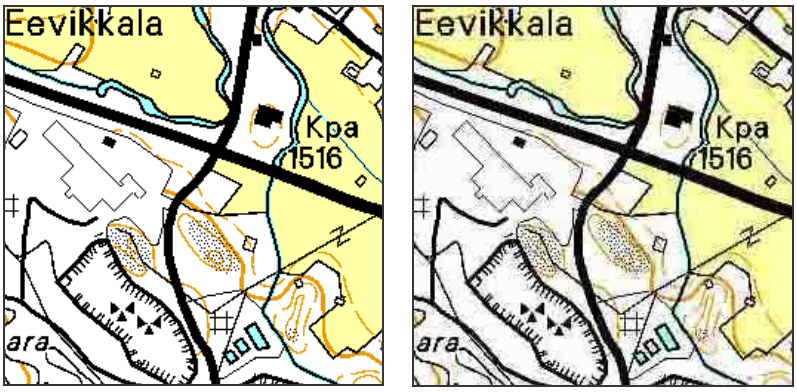


Fig. 11: A comparison example of a map image fragment compressed with GIF (left), using 8055 bytes, and JPEG (right), using 44160 bytes.

4.1 Lossless compression methods

Graphics Interchange Format (GIF) is lossless compression method suited for palette images. It can hold multiple bitmaps of up to 256 colors each. For reducing the size of compressed raster data it uses LZW dictionary compressor [Wel84]. GIF and 'Graphics Interchange Format' are trademarks of CompuServe Inc and the format is publicly available without royalties or licensing restrictions. However, the LZW compressor is patented by Unisys Corporation.

Portable Network Graphics (PNG) provides a patent-free replacement for the GIF. It is based on *deflate* algorithm [Deu96], which is a combination of LZ77 dictionary compression [LZ77] and *Huffman coding*.

CCITT Group 3 (G3) and Group 4 (G4) are the most common methods for binary image compression and facsimile algorithms can be applied to the map image separated into the binary layers. G3 has two realizations: G3-1D, which is a one dimensional algorithm and where the scanline is encoded as a set of runs, representing a number of white or black pixels. The runs further are encoded by the Huffman coding. G3-2D is a modified READ algorithm [CCITT T.4]. G4 is known as modified-modified READ algorithm [CCITT T.6].

4.1.1 Statistical binary image compression

Statistical image compression consists of two distinct phases: *statistical modeling* and *coding* [RL81]. In the modeling phase the probability distribution of the symbols to be compressed is estimated. The coding process assigns variable length code words to the symbols according to the probability model so that shorter codes are assigned to more probable symbols, and vice versa. The coding can be performed using *arithmetic coding*, which provides optimal coding for the given probability model [RL79].

A binary image can be considered as a message generated by an information source. The idea of statistical modeling is to describe the message symbols (pixels) according to the probability distribution of the source alphabet. Shannon has shown in [Sha48] that the information content of a single symbol (pixel) in the message (image) can be measured by its *entropy*.

The pixels in an image form geometrical structures with appropriate spatial dependencies that can be described by *context-based statistical model* [LR81]. The probability of a pixel is conditioned on a context C, which is defined as the black-white configuration of the neighboring pixels within a local template. The entropy of an N-level context model is the weighted sum of entropies of individual contexts:

$$H_N = -\sum_{j=1}^{N} p(C_j) \cdot \left(p_W^{C_j} \cdot \log_2 p_W^{C_j} + p_B^{C_j} \cdot \log_2 p_B^{C_j} \right)$$

where $p(C_j)$ is the probability of the context C_j; $p_W^{C_j}$ and $p_B^{C_j}$ are respectively the probabilities of the white and black pixel in the context C_j.

JBIG is the ISO/ITU lossless binary image compression standard [ITU-T T.82]. It is based on statistical context modeling and arithmetic coding. The QM-Coder [PM88] used in the JBIG standard provides an approximate arithmetic coder and table-driven technique for updating the probability estimate for each pixel's context after the pixel is coded. JBIG encodes the image in raster-scan order. For each coded pixel it locates its context consisting of neighboring pixels, defined by a context template (see Fig. 12). Then the number of the selected context and the pixel to be coded are sent to the QM-coder.

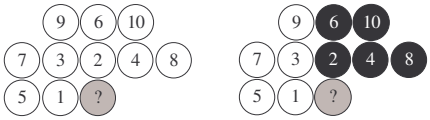


Fig. 12: The default 10-pixel context template, which is used in JBIG (left). Example of context which have unique number defined by the configuration of the pixels within the template: "$010101010101_{(bin)}$"=$341_{(dec)}$

Although designed primarily as a method for compressing binary image data, JBIG is capable of compressing color or grayscale images with a depth of up to 255 bits per pixel. Such multi-bit pixel images are compressed by bitplane rather than by pixel. For example, an 8-bit image compressed using JBIG would be encoded into eight separate bitplanes.

It is recommended before separating the grayscale image into set of bitplanes, to preprocess the grayscale image with a gray-coding algorithm to normalize the changes between adjacent byte values in the image data. This process increases the efficiency of the JBIG encoder.

JBIG images may be encoded sequentially or progressively. Sequentially encoded images are stored in a single layer at full resolution and without other lower resolution

images being stored in the same data stream. Progressively encoded images are started with the lowest resolution image first and end with the highest (see Fig. 13). The high-resolution image is stored in a separate layer and is then used to produce a lower resolution image, also stored in its own layer.

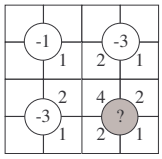


Fig. 13: The weight mask used for resolution reduction used in JBIG. Pixels from a new lower resolution layer are shown with circles.

There is no limit to the number of resolution layers that may be encoded. For example, a 1200-dpi image can be encoded as one layer (1200 dpi), three layers (1200, 600, and 300 dpi) or five layers (1200, 600, 300, 150, and 75 dpi). The lowest resolution is determined by whatever is considered useful.

Progressive encoding does not add much more data to a JBIG data stream than does sequential encoding but it does have greater memory requirements. Because a lower resolution image is encoded from data of the next higher resolution image (and vice versa when decoding), a frame buffer must be used to store image data that is being used as a reference.

4.1.2 Statistical palette image compression

Prediction by Partial Matching (PPM) data compression scheme [CW84] uses adaptive context determination scheme. At each coding step the longest previously encountered context is used to predict the next character. If the symbol is novel to that context and cannot be modeled by that context, an escape symbol is transmitted and context is shortened by dropping one symbol. This process of transmitting an escape code and then shortening the context will continue until the symbol is successfully transmitted. If the current symbol is novel even to the zero order context then a final escape will be transmitted and the symbol will be encoded as is. The adaptive model can then add the current symbol to all applicable contexts. The actual coding of a symbol, given its

predicted probability, is performed by arithmetic coding. The PPM compression scheme has also been adapted to the compression of map images [FS02].

4.1.3 Statistical compression with prediction based context model

In principle, better probability estimation can be achieved using a larger context template. The use of large template, however, does not always result in compression improvement. The number of contexts grows exponentially with the size of template; adding one more pixel to the template doubles the size of the model. This leads also to the context dilution problem, in which the statistics are distributed over too many contexts, and thus affecting the accuracy of the probability estimates. The methods considered here use the combination of prediction techniques with context modeling in order to decrease the size of context model.

JPEG-LS is a lossless/near lossless compression standard [ITU-T T.87] for continuous-tone images based on LOCO-I algorithm (Low Complexity Lossless Compression for Images) [WSS00]. The algorithm uses non-linear predictor with simple edge-detector, able to predict horizontal, vertical and -45° edge orientations. The context model is determined from four local gradient calculations and further quantized according to their distance from the current pixel to reduce the number of free parameters. Coding is done by Golomb-Rice codes with adaptively estimating the skewness parameter. In uniform image regions the compressor operates in *run-length* mode. The compressor also has near-lossless mode, where the compression level depends on a specified maximum reconstruction error. The main advantage of JPEG-LS compression method is that its performance is comparable to the compression schemes based on arithmetic coding, with only a fraction of their complexity.

Context-Based, Adaptive, Lossless Image Coding (CALIC) [WM97] is operating in two different modes: binary and continuous tone. The binary mode is used in situations where the input pixel and its neighborhood, specified by a template, have no more than two distinct intensity levels. Context-based adaptive arithmetic coder is used to code three symbols: 0, 1 or an escape symbol, which is used to escape the binary mode. In

continuous-tone mode the system operates as follows: first, the so-called *Gradient-Adjusted Prediction* (GAP) is calculated. It is a simple, adaptive, nonlinear prediction used to detect the magnitude and orientation of edges in the image. Second, the context error modeling of this initial prediction is used to further update the prediction via one step delay feedback, using the error energy estimator (Δ). This Δ is further quantized into 8 levels. Third, the local spatial context is determined and quantized into an eight bit integer. The quantized error energy and quantized texture context form a compound context, which is finally used to select the statistical model for coding the prediction errors by arithmetic coding.

4.1.4 Object oriented context modeling

Piecewise Constant Model (PWC) [AUS00] is a technique designed for lossless compression of palette images. It is a two pass method. At the first pass the method uses special classification to establish the boundaries between constant color pieces in the image, which consists of four following classes:

Q1: Whether the pixel color equals to its left neighbor, and top neighbor.

Q2: Whether the pixel color equals to its top-left neighbor, and top-right neighbor.

Q3: Is the pixel color equals to a value that occurred previously in the same context. This classification is repeated until a guess is correct or the guesses are exhausted.

Q4: Report the current pixel color.

At the second pass the decisions are coded by binary arithmetic coder. The method takes also advantage of the uniform regions, where the same context appears repeatedly. For uniform regions an extra class is reserved, so when a uniform region is encountered, its length is determined and a decision is coded as to whether or not it can be skipped entirely.

4.1.5 Multi-layer statistical compression

Two-dimensional context modeling is capable of capturing spatial dependencies in the image. On the other hand, map images often originate from the data as separated to

binary layers such as roads, text, buildings and water areas. Most of the compression methods simply ignore the fact, that the water areas on the map image are usually outlined by the border. Thus, for the compression method suitable for map images it is essential to capture the existing inter-layer dependencies. This idea of utilizing inter-layer dependencies is used in the described following methods.

JBIG2 [HKM98] is a compression standard [ITU-T T.88] for binary images which can also be used for compressing grayscale images in the same way as it was implemented in JBIG. JBIG2 extends JBIG by incorporating two pattern matching strategies: *Pattern Matching and Substitution* (PM&S) and *Soft Pattern Matching* (SPM). These strategies differ from each other in the way how they encode pixel blocks. In PM&S, the image first is segmented into pixel blocks. Second, the dictionary is searched in order to locate the previously coded block that matches the current pixel block. If an acceptable match is found, the associated dictionary index and 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 high lossy compression level.

SPM differs from PM&S in the fact that, in addition to the dictionary index and position offset, the current pixel block, called *refinement data*, is losslessly encoded using two-layer coder, making use of previously coded pixels from the matched block employing the two-layer context template, (see Fig. 14). Since these blocks match to each other, high similarities between them allow compressing the current block very efficiently. Including refinement data allows reconstruction of the original pixel blocks, which allows lossless decompression.

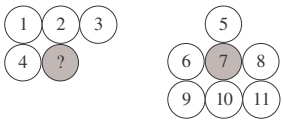


Fig. 14: The context template used for encoding the pixel block. The context pixels in the current pixel block are chosen using 4-pixel template (left), the pixels from the matched pixel block are chosen using 7-pixel template (right). The positions shown in gray are aligned.

DjVu compression technique [BHH98] is tailored for compression of high-resolution, high-quality images of scanned documents in color. It proceeds as follows: first, the

image is separated into three image components using color clustering technique based on LBG [LBG80] clustering algorithm: the foreground image that contains the color of the text and line drawings, the background image, and the mask image, which indicates whether the corresponding pixel in the image is chosen to be foreground or background. This component contains usually the text and the high contrast drawings. Second, the foreground and background images are downscaled to 100 dpi using an assumption that this resolution is sufficient to display pictures and enough to preserve the readability of the color document, except for the tiniest lines and fonts. Third, the foreground and background images are encoded using a wavelet-based compression algorithm. The mask image is coded at 300 dpi using a variation of JBIG2 compressor.

Embedded Image-Domain Adaptive Compression of Simple Images (EIDAC) method [YKO98] is tailored for compression of grayscale images. The image is separated into bitplanes, and each bitplane is compressed separately from the most significant bitplane to the least significant bit plane using three dimensional context modeling. Four context pixels are selected from the current bitplane, and one from each previous bitplanes that have been already processed (see Fig. 15). The encoding is performed by the binary arithmetic coder.

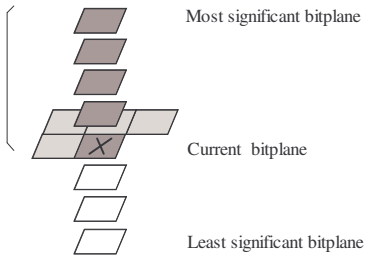


Fig. 15: The context template used by the EIDAC method.

SKIP pixel coding [FJ02] is a lossless context-based method for compressing limited bits/pixel images such as maps. It starts by color separation of the original image into binary layers and process them separately. However, it involves a simple modeling rule: if a given pixel in a particular layer has already been coded in a layer of a higher priority, it does not need to be coded again in the current layer or in any of the lower layers (see Fig. 16). Thus the coding of a large amount of redundant information around

blank areas can be “skipped”. The encoding component of this method is the binary arithmetic coder.

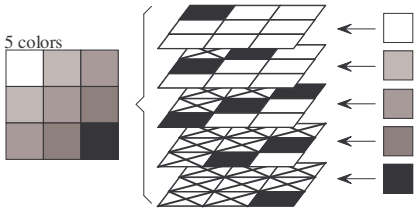


Fig. 16: The example of the SKIP modeling rule. The black pixels at each binary layer represent the pixels set with this color. The pixels marked with cross will be “skipped”.

4.2 Optimizing the context template

The location of the template pixels, if properly designed, may greatly improve the accuracy of the context model. It is therefore desirable to optimize the location of the template pixels for the compressed images. Usually the template pixels are distributed in the neighborhood using the principle of minimal distance to the current pixel. Standard 1-norm or 2-norm distance functions define two different templates shown in Fig. 17. These templates are well suited for mixed type of images. However, they are not necessary the best choices for map images. The map images consist of several binary layers with different semantic content. Each layer consists of geometrical structures that do not necessarily match to the structures of another layer.

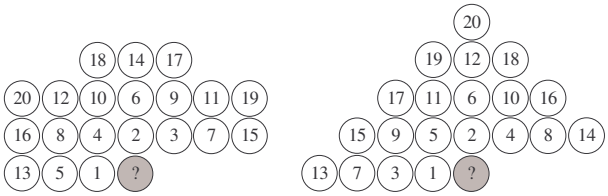


Fig. 17: The standard 1-norm (left) and 2-norm (right) context templates.

In **P1** we present a method for optimizing the context template for a given image. The method optimizes the location of the template pixels within a limited neighborhood area shown in Fig. 18 (left) and produces the ordered template as a result.

The optimal context template could be solved for a given template size k by compressing the image using all possible templates, and then selecting the one with the

best compression performance. However, this is computationally not possible as there is a huge number of different template configurations to be tested. Therefore, we take a more practical approach and construct the template stepwise similarly as in [DAN99], optimizing the location of one pixel at a time.

To optimize multi-layer context templates we modify the base algorithm in such a way that we consider context pixel locations within the combined neighborhood area of 77 pixels shown in Fig. 18. The current layer pixels are selected within 40-pixel neighborhood area, and in the reference layer pixels are selected from 37-pixel neighborhood area. An example of optimized context templates is shown in Fig. 19.

The optimized template can be used with JBIG-alike compressor. However, the template must be known both by encoder and decoder, and thus must be sent with compressed image. Another option would be to use static approach and train the template using a representative training image. This is possible because of the similarity of the context templates for images of similar type.

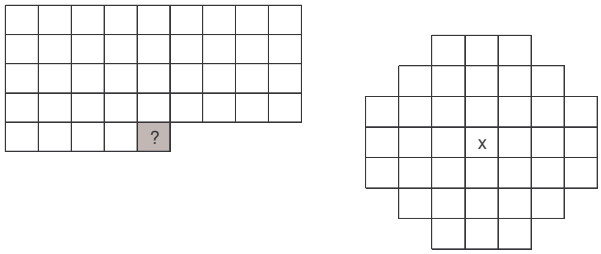


Fig. 18: The neighborhood area for optimizing the location of the template pixels: for the current layer (left) and for the reference layer (right).

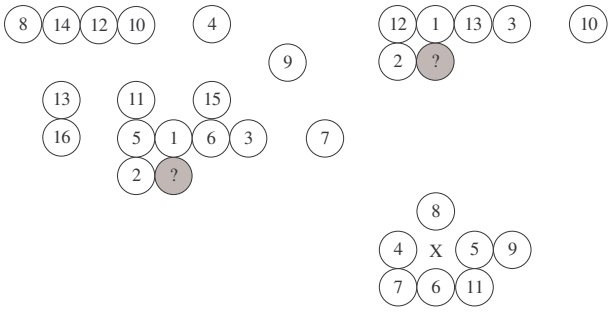


Fig. 19: An example of optimized templates for the field layer. Optimized 1-layer template (left) and optimized 2-layer template (right), where basic layer is chosen as reference layer.

4.3 Context tree modeling

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 [MF98]. The contexts are represented by a binary tree, in which the context is constructed pixel by pixel. The context selection is deterministic and only the leaves of the tree are used. The location of the next neighbor pixels and the depth of the individual branches of the tree depend on the combination of the already coded neighbor pixel values. Once the tree has been created, it is fully static and can be used in the compression as any other fixed-size template.

Context tree is applied in the compression in a similar manner as the fixed-size context templates; only the context selection is different. The context selection is made by traversing the context tree from the root to leaf, each time selecting the branch according to the corresponding neighbor pixel value. The leaf has a pointer (index) to the statistical model that is to be used. Each node in the tree represents a single context. The two children of a context correspond to the parent context increased 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.

The tree can be optimized beforehand using a training image (static approach) [FA99], or optimized directly to the image to be compressed (semi-adaptive approach) [MF98]. In the latter case an additional pass over the image is required to collect the statistics, and the tree must also be stored in the compressed file. The cost of storing the tree structure is one bit per node. The static approach is possible because of the similarity of the trees with images of the same type. On the negative side: the resulting tree would be more dependent on the choice of the training image.

To construct a context tree, the image must be processed and statistics should be calculated for potential contexts in the tree including the internal nodes. The tree must then be pruned by comparing the parent node and its two sub trees at every level. If compression gain is not achieved by using the two sub trees instead of the parent node,

the sub trees should be removed and the parent node would be a leaf node. The compression gain is calculated as:

$$Gain(C) = l(C) - l(C_{left}) - l(C_{right}) - SplitCost \tag{1}$$

where C is the parent node, and C_{left} and C_{right} are the two sub trees, see Fig. 20 for an example. The code length l denotes the total number of output bits from the pixels coded using the context in the particular node. The cost of storing the tree is integrated into the *SplitCost*. The code length can be calculated by summing up the self-entropies of the pixels as they occur in the image:

$$l(C) = -\sum_{t} \log_2 p^t(C) \tag{2}$$

where $p^t(C)$ is the probability of upcoming symbol within context C at time moment t. The probability of the pixel is calculated on the basis of the observed frequencies using a Bayesian sequential estimator:

$$p^t(C) = \begin{cases} p_W^t(C) = \dfrac{n_W^t(C) + \delta}{n_W^t(C) + n_B^t(C) + 2\delta}, & \text{if } t^{\text{th}} \text{ pixel is } white \\ p_B^t(C) = 1 - p_W^t(C), & \text{if } t^{\text{th}} \text{ pixel is } black \end{cases}$$

where n_W^t and n_B^t are time-dependent frequencies, and p_W^t and p_B^t are respectively the probabilities for white and black colors, and $\delta = 0.45$, as in JBIG.

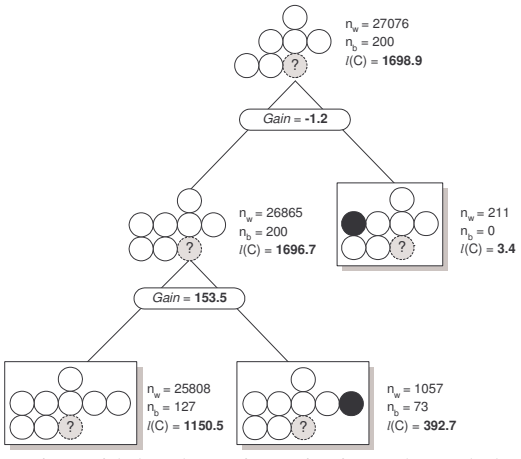


Fig. 20: Example of tree pruning with local pruning criterion. The code length is adaptively calculated according to Eq. (2).

4.4 Multi-layer context tree

In **P2** we construct the multi-layer context tree in order to utilize the existing inter-layer dependencies between the map image layers as follows. The tree starts from scratch and the branches are expanded one pixel at a time. The location of the template pixels are optimized and fixed beforehand and then applied for every branch. Another approach is to optimize the location of the context pixels separately for every branch (Free Tree approach). The context pixels are selected from the same joint 77 pixels neighborhood as in the method for optimizing multi-layer context template.

4.5 Optimal layer ordering

The captured inter-layer dependencies show us the correlation between the information content of the map image. Like in case of the NLS map images [NLS] there are strong correlation between *basic* layer and the two other layers (*fields* and *water*), whereas some layers do not seem to have any particular correlation to any other layers.

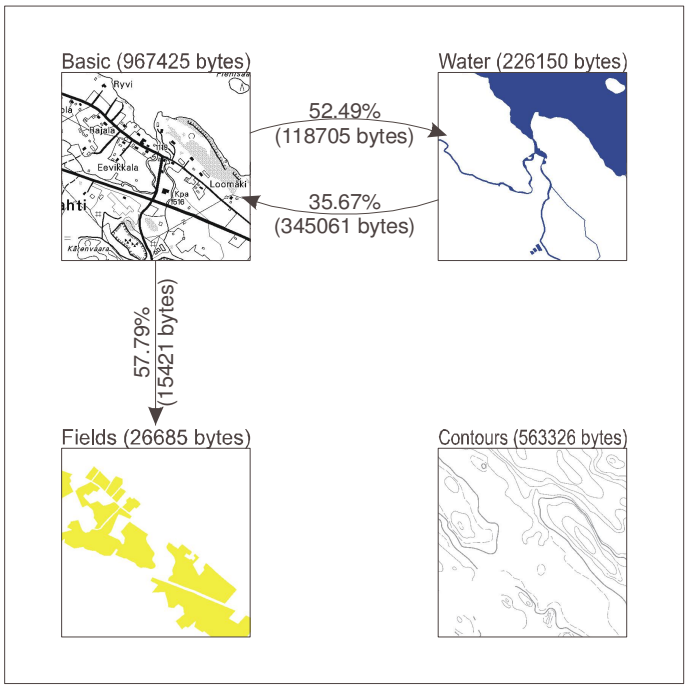


Fig. 21: The arrows show the inter-layer dependencies and the number of saved bits when compressing the second layer using the first one as a reference.

We can see in Fig. 21 that it is impossible to utilize all the existing dependencies as the order of processing restricts which layers can be used as the reference layer.

In general, we can select any predefined order on the basis of known or assumed dependencies, which can hold for the map images of the same type. However, if the source of map images is unknown we can still optimize the order in which the layers should be processed. The selected order must also be known by the decoder, so it must be sent with the compressed image as well.

In **P3** we consider the layer ordering optimization problem as a problem of solving directed *Minimum Spanning Tree* (MST) [CL90]. We consider an example of inter-layer dependencies given in Fig. 21 as a graph where layers are the nodes and the arrows represent the edges. The entering edges have weights in the graph, which show independent layer compression. The weights in the graph represent the saving of the bytes. We employ an algorithm of solving directed MST [Edm67] to maximize the utilization of inter-layer dependencies and thus improve the compression ratio.

In the case of color separated layers we can improve the compression by completely removing one of the layers and considering it as the background color. The background color is usually white but this is not necessarily always the case. In fact, we can set any layer as the background color. The advantage is that the compression of the chosen layer is avoided.

4.6 Summary

Two different techniques for optimizing the modeling phase in statistical compression methods were presented.

The first one optimizes the position of the context pixel within the neighborhood template. This technique allows more precise model to be constructed, which gives an improvement in the compression ratio. The technique can be extended to gather inter-layer dependencies between image layers.

The second one uses the context tree to construct the statistical model. This technique is also useful for obtaining inter-layer dependencies, which take place in the map images.

On the basis of captured inter-layer dependencies we are able to solve the optimal layer ordering. This provides us the sequence in which the layers must be processed in order to maximize the utilization of the inter-layer correlation providing the best possible result.

5 Map handling

Digital maps are usually obtained from spatial databases where they are stored in vector format [Sam89]. With vector format is possible to reproduce the view of a map at any zooming factor maintaining the finest details and quality. However, due to physical restrictions, this format is not handy for mobile devices. Most mobile devices, except for portable PC's, have very limited display capabilities and low storage size. For example, taking into account an average PDA device: the screen resolution has 240×320 pixel display and the memory is capacity is of 64 Mb RAM, which is shared between running programs and personal storage. The basic map operations like scrolling and zooming requires that the whole vector map must be stored in the client device, and a raster image, representing visual outlook, generated. Depending on the complexity of a map, its physical size might be large. However, vector map can be used to generate and transfer the part of the original vector map, which matches the requirements of the client device. But in this case a heavy computation on the server side is required, so the actual transfer might be delayed.

Another alternative is to generate a raster image representing the outlook of a map, store it in a server side and transfer it to the client device in compressed form. One of the compression standards, like GIF or JPEG, can be used. They are the de facto the standards used by millions of Internet users around the world for compressing images on the Web. However, to perform scrolling operations on map, the whole raster image must be transmitted to the client device. Moreover, to have an ability to perform zooming operations, we need to have separate raster images for different zoom factors because in the re-scaling the raster image might degrade the image quality. It is also impossible to generate part of an image without completely decompressing the whole image. It is also impossible to see only specific particular information of a map, like roads, without having the complete image generated for this purpose. However, the compression/decompression process itself can be made very fast.

5.1 Existing mobile navigation systems

The existing mobile navigation systems can be categorized into two different classes:

- Online systems, which use online internet connection to acquire the map for the desired location.
- Offline systems, which use maps existing in the navigational device by default.

Online systems are tailored for devices, which have online connections to the internet, such as portable PC's, PDA and SmartPhones. Nowadays this kind of systems have become more popular and it is quite easy to use such systems to retrieve maps of city's central areas or any major towns. Online systems, in general, have a set of very detailed vector maps of fixed regions and provide to users the outlook of desired areas in a raster form on demand, according to position, scale and detail level.

Most of the offline systems are provided by manufacturers of GPS devices such as *Garmin* (Garmin International Inc.) or *Magellan* (Thales Navigation Inc.). In general they have one thing in common: some sets of maps are already stored in the device memory; this can be the case of the general maps of Europe, USA or Russia. However, to get a new map into a device the user needs to have a subscription with the manufacturer or with the side companies that provide map solutions, such as *MapInfo* (MapInfo Corporation). For the offline navigation systems the maps are usually provided on CD in encrypted compressed raster images.

5.2 Map Image Storage System

Map Image Storage System (MISS) format, developed in **P6,** is based on compressed raster format. For client-server applications the maps are stored in MISS format in the server-side database. Spatial views are generated for the client-side and sent without decompression over connecting channels to the client. The system does not depend on any particular database or vector format as digitized raster maps can be easily generated and reproduced from any source format, including paper maps. Another advantage of

this system is that it requires only a modest memory and computing resources in order to be operational in a real-time environment.

5.2.1 Multi-scale representation

One positive feature of vector maps is that views of any zooming factor can be generated without having any problem. With vector maps it is also easy to control the amount of shown details: changing the view resolution from the smallest detail to the more general, where some information on the map can be omitted. With raster maps it is impossible to maintain the quality of zoomed images with factors that are not a multiple or dividend of 2 (in/out). The interpolation schemes, such as *bilinear interpolation*, can also be employed with zooming-in operation for increasing visual quality of the zoomed image. However, we do not use them because they tend to blur the image content and thus produce artificial colors. Instead, we simply copy the original pixel.

Therefore, for convenient zooming operations we propose to store the different scales of the same location, as the different pages of the same map image. The intermediate scales with the same detail level can be generated on fly by magnifying or stretching the page image by the factor of 2. However, the images with scales different from the origin by a factor different than 2 are put into MISS as different page (see Fig. 22).

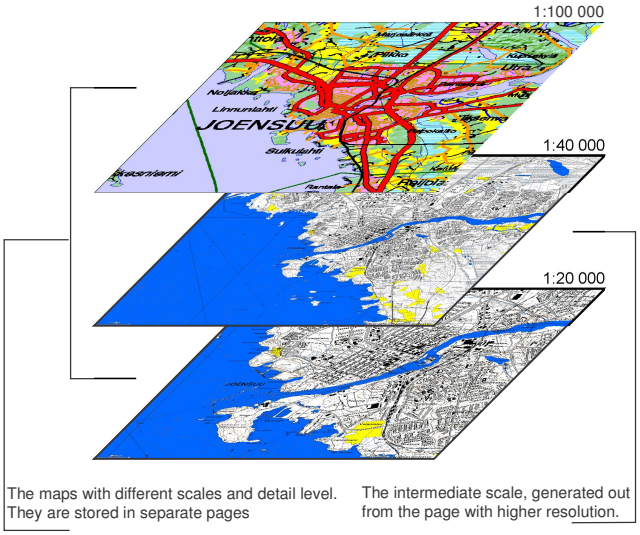


Fig. 22: Representation of the map as a collection of map images (1:100 000 and 1:20 000). The intermediate scales are generated by zooming

5.2.2 Different compression strategies

In the chapter 3 we have overviewed different compression methods: JPEG-LS [ITU-T T.82], CALIC [WM97], PWC [Aus00] and DjVu [BHHSBL98]. These methods have one thing in common: they may behave differently according to the type of information they compress.

We propose to construct the page out of different layers. The layers are compressed separately and represent different types of information on the page. Binary layers could be used for representing color or semantic information and grayscale layers for pictures with smooth color transitions. Binary layers can be successfully compressed with compression technique based on arithmetic coding. However, this compression technique might not be the best for grayscale layers. Thus we propose that each layer might have separate compression technique in use. Moreover, we do not restrict the application to use only a specific compression technique. For this motive we reserve in the MISS structure the space where the compressor might store the information required to proceed with the decompression.

5.3 Dynamic map handling

One of the purposes of the personal navigation devices is to track the current position of an observer on a map. In order to speed-up the map displaying process, map images must be stored in the memory of a client device. Usually, the whole map image is stored in different files according to the scale, type and covering region. The map images are updated from the remote server on user request, according to the subscription details. The decision to delete map images is entirely dependent on the user. We refer to this map handling scenario as *static map handling*.

In *dynamic map handling* the user has no direct control on the maps **P7**. The idea is that map images just appear when needed. The current user location is determined in the proper map found in the device memory, decompressed and displayed. In case that the map would not be found in the memory it would be automatically requested from the

server. The deletion of maps could be done automatically, depending on the set of relevance preferences such as time of the last access, or also manually on user decision.

It might happen that even if a whole map is stored in the memory, some particular places of this map will never be visited even once. Also, the display capabilities of most mobile devices, like PDA, GPS, and mobile phones, do not exceed dimensions of 240×320 pixels. Thus for reducing the storage size in the client device and the cost of transmitting the map image over the communication channel, we propose to split the original map image in the server side into rectangular non-overlapping blocks, compress them separately and send on demand in compressed form to the client (see Fig. 23). Every time before performing scrolling or zooming operations, the application checks the blocks that are in the current view, determines those that must be changed, perform the operation and check for changed blocks in the device memory, and if the desired information is not located then it is requested from the remote server.

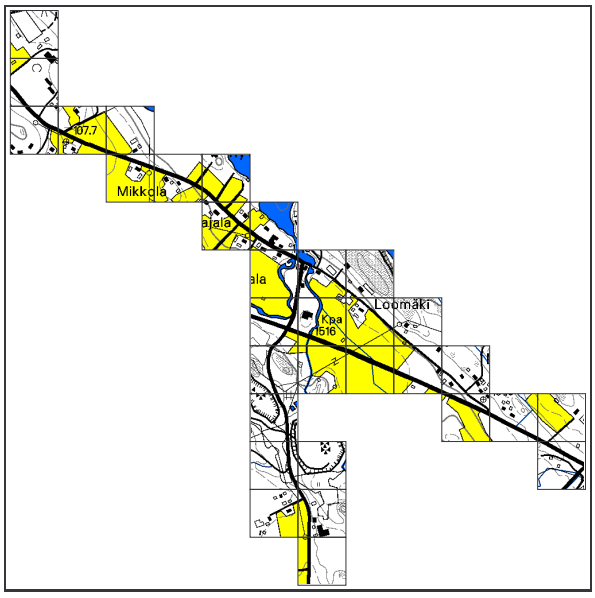


Fig. 23: Example of a map image constructed in the client device using dynamic map handling.

5.4 Detailed file structure

The developed MISS structure is compatible with dynamic map handling as it has the following properties:

- The map image is stored as one file, representing the fixed area on a map.
- The file has hierarchical structure, allowing direct access to the specific features without complete search over the file, and multi-scale representation (see Fig. 24).
- The map image is a set of digital maps (pages). Pages can be used to represent maps of the same area with different scales, or just different maps of the same area.
- The page is constructed out of binary or grayscale layers, representing the semantic or color information about the image.
- The layers are separated to non-overlapping rectangular blocks, which are compressed independently.
- As a compression part, we use the JBIG based compressor. However the system supports embedding of any suitable compression technique, like GIF or PNG. It is also possible to employ different compression techniques to compress different layers. For example, to compress grayscale and binary layers with different compressors, with the only restriction that both compressor and decompressor must be capable of handling the commonly treated data.

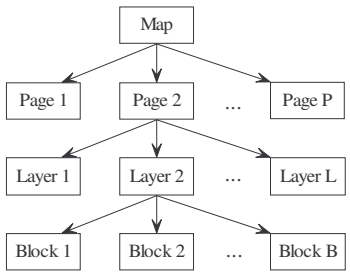


Fig. 24: The hierarchical structure of MISS file.

On the one hand, dynamic map handling also means fast and simple addition of image elements such as page, layer or block without complete reordering of the storage file.

This could be done by adding the elements at the end of the file. However, to access an element we will have to perform complete search over the file structure.

The access to the map image elements should remain direct. This could be done by an index table at the *parent* level, directing to the positions of the *children*. However, it means that at the moment of creating of a new page we need to know the number of layers. On the client device this information is often not available. Moreover, we cannot change the size of the index table in order to add more children.

To overcome these difficulties we use a *Continuous Index Table* (CIT) [Vei01] (see Fig. 25). CIT is a special index table with the possibility of varying the number of bytes reserved for each index and continuing the table by appending the new table at the end of file. Each index is an offset in the file to the location where the children data are stored.

Fig. 25: The structure of CIT.

The first parameter is used to specify the number of bytes used for the addressing index. Varying this parameter, we can modify the addressing length. The second parameter represents the number of indexes available in the current CIT block. Last four bytes of each CIT block are reserved for offset to the next CIT block.

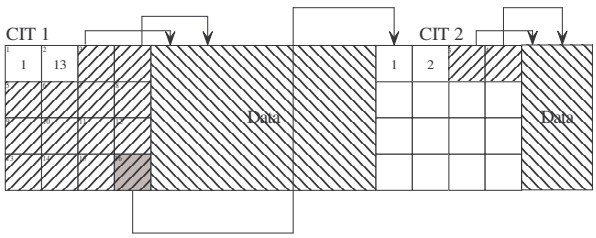


Fig. 26: Example of the use of CIT.

For example, using CIT in Fig. 26 and wanting to add one more data block to the file, the program must do the following steps:

1. Check the number of occupied blocks with a predefined value in CIT 1.
2. Go to the CIT 2, using the link at the end of CIT1.
3. Check the number of occupied blocks with a predefined value in CIT 2.

4. Go to the first unoccupied cell.
5. Write data at the end of file.
6. Put the corresponding address into a cell.

The CIT is used in MISS for indexing pages, layers and blocks, so that the structure of file, page and block headers will be as it shown in Fig. 27.

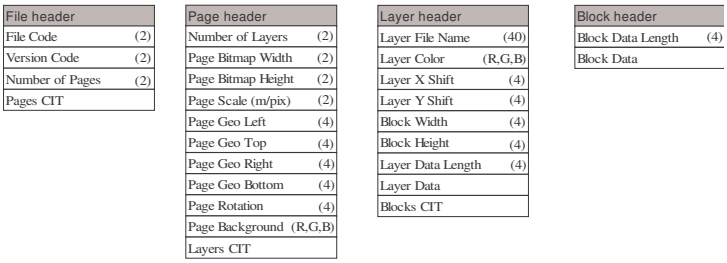


Fig. 27: The structure of the File, Page, Layer and Block headers.

5.5 Client-server communication

Using the proposed storage structure, the client-server communication dialog could be formed on the basis of four different request-response pairs:

- *Map request* is used in a situation when a map with particular coordinates is not found in the client memory. In *Map response* the server sends to the client the MISS File header. Automatically the set of 'Page requests' is generated for all existing pages in this Map.
- *Page request* is used to acquire from the server a Page with specific scale. In *Page response* the server sends to the client appropriate MISS Page header. Recursively the set of 'Layer requests' is generated and sent to the server to get the headers of all layers of this Page.
- *Layer request* is used for forming the layer data and determining the compression strategy for the layer. In *Layer Response* the server sends to the client appropriate MISS Layer header.
- *Block request* is used for acquiring from the server the actual data that will be displayed. Block Request if formed in the case when the appropriate block is not

found in the MISS file in the client memory. In *Block Response* the server sends to the client the actual block data.

5.6 Summary

We have introduced a new Map Image Storage System (MISS), which was developed for real-time applications in portable devices with low memory and computing resources. The system architecture is designed to minimize the storage size, transmission time and memory requirements for the client device. The MISS file can have several different maps stored in the same file. Different compression techniques could be involved. The MISS format allows direct access to the data and at the same time permits a flexible file structure. The detailed structure of the MISS format was outlined.

6 Conclusions

We have proposed a method for color quantization of map images where the number of output colors is unknown. We consider map images to have artificial colors caused by blurring and by compression using lossy techniques such as JPEG. To approximate the number of output colors we use F-Test variance ratio. We perform the color quantization in uniform L*a*b* color space as well as in standard RGB color space. Empirical results show that the proposed method can be useful to automatically determine the number of colors for the map images.

We have proposed a novel method for filtering raster map images by context tree modeling. The main advantage of the filter is that it does not destroy the object borders. It outperforms Vector Median filter until a level of 25% of corrupted pixels, which can be considered as severe noise.

We have proposed a method for compressing map images by multi-layer context tree modeling and by optimizing the order of the processing of the binary layers. Solutions are given for the context modeling, utilization of the multi-layer dependencies and for the optimal ordering of the layers. The optimal order of processing the layers was considered as directed spanning tree problem and solved with an algorithm based on the Edmond's algorithm for optimum branching and by the optimal selection and removal of the background color. The proposed method gives 50% better results than JBIG, and 25% better than a single layer context tree modeling.

We have developed a storage system for the map images which is tailored for real-time applications that use portable devices with low memory and computing resources. The system supports compact storage size, decompression of partial image, smooth transitions between various scales and small transmission time. This minimizes the transmission time and the memory requirements in the user device are minimized.

7 Comparison results of the presented methods

In this chapter, we have collected results from the experiments on the compression methods presented in **P1**, **P2** and **P3,** and also the performance comparison of the filtering methods presented in **P4**.

Firstly, we show the comparison between compression methods presented in **P1**, **P2** and **P3**. Following abbreviations are used:

- JBIG: baseline JBIG with 10-pixel context template.
- OSL: Optimized single layer context template. (introduced in **P1**)
- CT: Single layer context tree modeling.
- OTL: Optimized two-layer context template. (introduced in **P1**)
- MCT: Multi-layer context tree modeling (**P2**) and optimizing the order of layers. (introduced in **P3**)

Table 1: Comparison of the results presented in P1, P2 and P3. The compressed file sizes are given in kilobytes

	JBIG	OSL	CT	OTL	MCT
Image #1	159	136	130	126	106
Image #2	846	644	611	620	528
Image #3	315	262	246	239	223
Image #4	662	564	544	459	443
Total	1982	1606	1531	1444	1300

Secondly, we show the comparison of filtering methods presented in **P4**, (see Fig. 28, Fig. 29, and Fig. 30). Rank-conditioned vector median filter [Luk03] is added for comparison. Following abbreviations are used:

- VM: Vector median filter.
- RCVM: Rank-conditioned vector median filter.
- CT: Context tree filter.

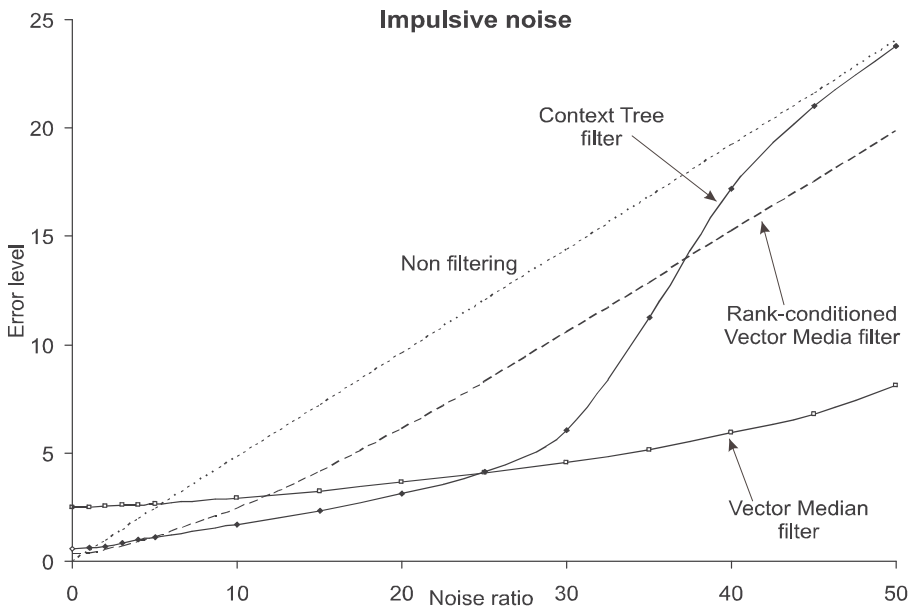


Fig. 28: The comparison of VM, RCVM and CT filters on images distorted with content-dependent noise for image #1 (left).

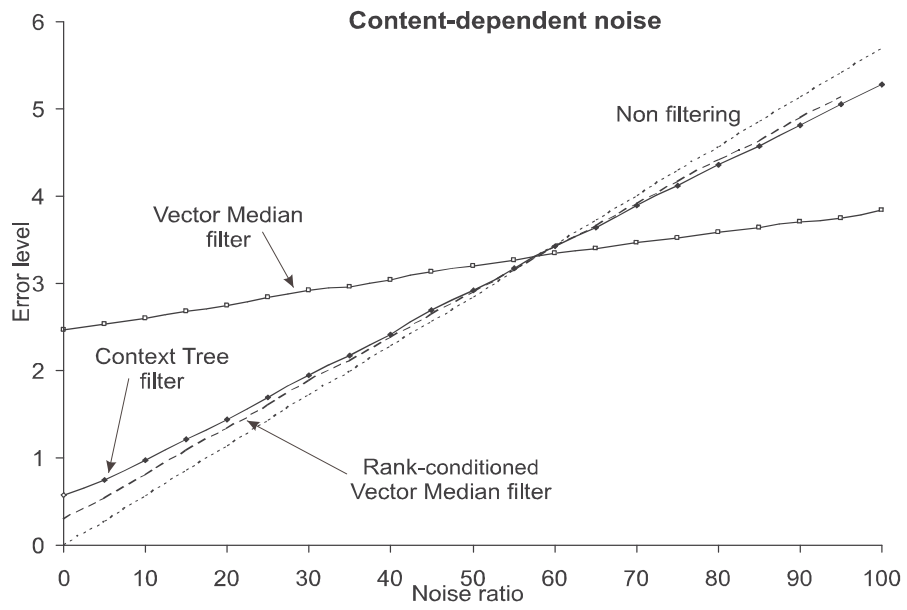


Fig. 29: The comparison of VM, RCVM and CT filters on images distorted with content-dependent noise for image #1. (right)

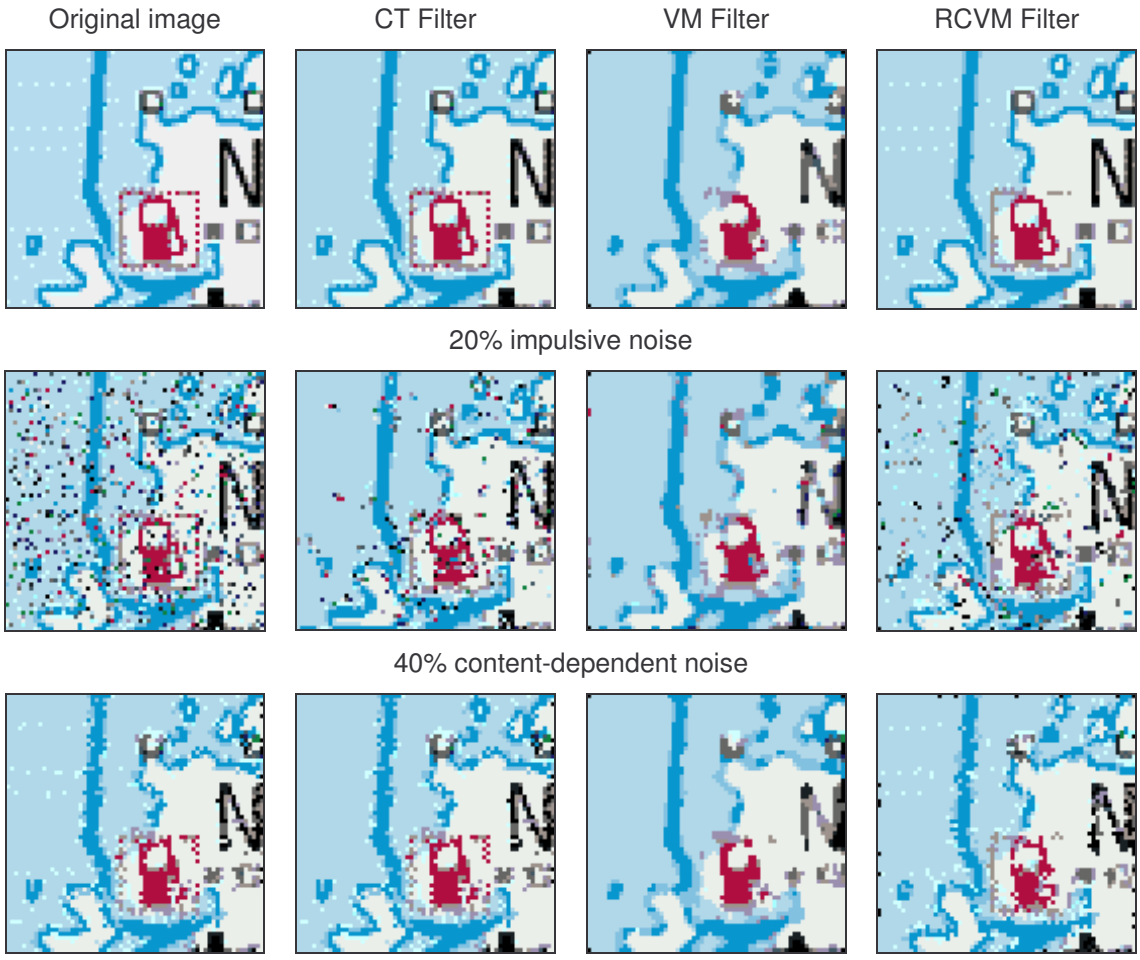


Fig. 30: Visual comparison of VM, RCVM and CT Filters on 64×64 pixel fragment of image #1

References

[AF00] E. Ageenko, P. Fränti, "Context-based filtering of document images", in *Pattern Recognition Letters 21*, pp. 483-491, 2000.

[AHN90] J. Astola, P. Haavisto, Y. Neuvo, "Vector median filters", *Proceedings of IEEE*, 78(4), pp. 678-689, 1990.

[Aus00] P. Ausbeck, "The piecewise-constant image model", *Proceedings of the IEEE*, 88 (11), pp. 1779-1789, 2000.

[BB97] J. P. Braquelarie and L. Brun, "Comparison and optimization of methods of color image quantization", *IEEE Transactions on Image Processing*, 6(7), pp. 1048-1052, 1997.

[BBA94] R. Balasubramanian, C. Bouman, and J. Allebach, "Sequential scalar quantization of color images", *Journal of Electronic Imaging*, 3(1), pp. 45-59, 1994.

[BHH98] L. Bottou, P. Haffner, P. Howard, P. Simard, Y. Bengio, Y. LeCun, "High quality document image compression with "DjVu"", *Journal of Electronic Imaging*, 7(3), pp.410-425, 1998.

[Bru77] P. Brucker, "On the complexity of clustering problems", in: R. Henn, B. Korte and W. Oettli (Edts.), *Optimizations and Operations Research*, Springer, pp. 45-54, 1977.

[CCITT T.4] CCITT Recommendation T.4, *standardization of group 3 facsimile apparatus for document transmission*, 1980.

[CCITT T.6] CCITT Recommendation T.6, *Facsimile coding schemes and coding control functions for group 4 facsimile apparatus*, 1984.

[CIE86] CIE, *Colorimetry*, *CIE Pub*. No. 15.2, Central Bureau CIE, Vienna, Austria, 1986.

[CL90] T. Cormen, C. Leiserson, R. Rivest, *Introduction to algorithms*, MIT Press, Cambridge, Massachusetts, 1990.

[CTM94] N. Chaddha, W. Tan, T. Meng, "Color quantization of images based on human vision perception", *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Adelaide, South Australia, 5, 89-92, 1994.

[CW84] J. Cleary, I. Witten, "Data compression using adaptive coding and partial string matching", *IEEE Transactions on Communications*, 32(4), pp. 396-402, 1984.

[DB99] S. Dye, S. Buckingham, Mobile Positioning, *Mobile Lifestreams*, 1999.

[Dek94] A. Dekker, "Kohonen neural networks for optimal color quantization", *Network: Computation in Neural Systems* 5, pp.351–367, 1994.

[DAN99] K. Denecker, S. Assche, P. Neve, I. Lemahieu, "Context-based lossless halftone image compression", *Journal of Electronic Imaging*, 8(4), pp.404-414, 1999.

[Deu96] P. Deutsch, "DEFLATE compressed data format specification," rfc1951, http://www.cis.ohio-state.edu/htbin/rfc/rfc1951.html, May 1996.

[Edm67] J. Edmonds, "Optimum branchings", *Journal of Research of the National Bureau of Standards*, 71B: 133-240, 1967.

[Equ89] W. Equitz, "A new vector quantization clustering algorithm", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(10), 1568-1575, 1989.

[FA99] P. Fränti and E.I. Ageenko, “On the use of context tree for binary image compression”, *Proceedings of IEEE International Conference on Image Processing* (ICIP’99), Kobe, Japan, vol. 3, pp. 752-756, 1999.

[FJ02] S. Forchhammer and O.R. Jensen, “Content layer progressive coding of digital maps”, *IEEE Transactions on Image Processing* 11(12): 1349-1356, 2002.

[FK00] P. Fränti and J. Kivijärvi, "Randomized local search algorithm for the clustering problem", *Pattern Analysis and Applications*, 3(4), pp. 358-369, 2000.

[FKN98] P. Fränti, J. Kivijärvi and O. Nevalainen: "Tabu search algorithm for codebook generation in VQ", *Pattern Recognition*, 31(8), pp. 1139-1148, 1998.

[FS02] S. Forchhammer and J.M. Salinas, “Progressive coding of palette images and digital maps”, *IEEE Proceedings Data Compression Conference*, Snowbird, Utah, USA, pp. 362-371, 2002.

[Gol89] D. Goldberg, “Genetic algorithms in search”, *Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[Har75] J. A. Hartigan, *Clustering Algorithms*, John Wiley & Sons, New York, USA, 1975.

[Hec82] P. Heckbert, “Color quantization for color displays”, *Proceedings of the SIGGRAPH’82*, Boston, MA, pp. 297-307, 1982.

[HK01] A. Hamza, H. Krim, “Image Denoising: A nonlinear robust statistical approach”, *IEEE Transactions on Signal Processing*, 49(12), pp. 3045-3054, 2001.

[HKMF98] P. Howard, F. Kossentini, B. Martins, S. Forchhammer, W. Rucklidge, "The emerging JBIG2 standard", *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7), pp. 838-848, 1998.

[Ito80] P. Ito "Robustness of ANOVA and MANOVA test procedures", in: Krishnaiah PR (Ed.), *Handbook of Statistics 1: Analysis of Variance*. North-Holland Publishing Company, pp. 199-236, 1980.

[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] ISO/IEC 14495-1, ITU 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.

[Jai89] K. Jain, *Fundamentals of digital image processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.

[Jus81] B. Justusson, "Median filtering: statistical properties", in *Two Dimensional Digital Signal Processing II* (T. Huang, Ed.). Berlin: Springer-Verlag, 1981.

[Kap96] E. Kaplan (ed.), "Understanding GPS: principles and applications", *Artech House Telecommunication Library*, March, 1996.

[KLL96] K. Kim, C. Lee, E. Lee, Y. Ha, "Color image quantization using weighted distortion measure of HVS color activity", *IEEE International Conference on Image Processing (ICIP'96), Lausanne, Switzerland,* vol. 3, 1035-1039, 1996.

[Koh95] T. Kohonen, *Self-organizing maps*, Springer-Verlag, Berlin, Germany, 1995.

[LBG80] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design", *IEEE Transactions on Communications*, 28(1), pp. 84-95, 1980.

[Llo82] S. Lloyd, "Least squares quantization in PCM", Memorandum, Bell Laboratories, Murray Hill, USA, 1957; published in *IEEE Transactions on Information Theory*, 28(2), 129-137, 1982.

[LR81] G. Langdon, J. Rissanen, "Compression of black-white images with arithmetic coding", *IEEE Transactions on Communications* 29(6), pp. 858-867, 1981.

[Luk03] R. Lukac, "Adaptive vector median filtering", in *Pattern Recognition Letters 24*, pp. 1889-1899, 2003.

[LZ77] A. Lempel, J. Ziv, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, 23(3), pp. 337-343, May 1977.

[Mat75] G. Matheron, *Random sets and integral geometry*, Wiley, New York, 1975.

[McQ67] J. B. McQueen, "Some methods of classification and analysis of multivariate observations", *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, USA, 281-297, 1967.

[MF98] B. Martins, S. Forchhammer, "Tree coding of bilevel images", *IEEE Transactions on Image Processing* 7(4), pp. 517-528, 1998.

[MM01] C. Morris, A. Maisto, "*Psycology: an introduction*", Prentice Hall, 2001.

[NLS] National Land Survey of Finland, Opastinsilta 12 C, P.O.Box 84, 00521 Helsinki, Finland. (http://www.nls.fi/index_e.html)

[OB91] M. T. Orchard, C. A. Bouman, “Color quantization of images”, *IEEE Transactions on Signal Processing*, 39(12), pp. 2677-2690, 1991.

[PAS02] N. Papamarkos, A. E. Atsalakis, and C. P. Strouthopoulos, “Adaptive color reduction”, *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(1), pp.44-56, 2002.

[PHK00] J. Puzicha, M. Held, J. Ketterer, J. Buchmann, and D. W. Feller, “On spatial quantization of color images”, *IEEE Transactions on Image Processing*, 9(4), pp. 666-682, 2000.

[PM88] W. Pennebaker, J. Mitchell, “Probability estimation for the Q-coder”, *IBM Journal of Research, Development* 32(6), pp.737-759, 1988.

[RL79] J. Rissanen, G. Langdon, “Arithmetic coding”, *IBM Journal of Research, Development* 23: 146-162, 1979.

[RL81] J. Rissanen, G. Langdon, “Universal modeling and coding”, *IEEE Transactions on Information Theory* IT 27: pp. 12-23, 1981.

[Sam89] H. Samet, “Applications of spatial data structures”, *Computer Graphics, Image Processing, GIS*. MA: Addison-Wesley, Reading, 1989.

[Sha48] C. Shanon, “A mathematical theory of communication”, *Bell Systems Technology Journal* 27: pp. 398-403, 1948.

[Ser82] J. Serra, “Image analysis and mathematical morphology”, London: Academic Press, 1982.

[Tat97] S. Tate, “Band ordering in lossless compression of multispectral images”, *IEEE Transactions on Computers*, 46(4), pp. 477-483, April 1997.

[Tuk77] J. Tukey, *Exploratory Data Analysis*, Meno Park, CA: Addison-Wesley, 1971, 1977.

[TV93] P. Trahanias, A. Venetsanopulos, “Vector directional filters – a new class of multichannel image processing filters”, *IEEE Transactions on Image Processing* 2, pp. 528-534, 1993.

[Vei01] V. Veis, “Representation of digital maps”, MSc Thesis, University of Joensuu, Finland, 2001.

[VFK01] O. Virmajoki, P. Fränti, T. Kaukoranta, “Practical methods for speeding-up the pairwise nearest neighbor method”, *Optical Engineering*, 40(11), pp. 2495-2504, 2001.

[Wal91] G. Wallace, "The JPEG still picture compression standard", *Communications of the ACM*, 34(4), pp. 40-44, 1991.

[War63] J. Ward, “Hierarchical grouping to optimize an objective function”, Journal of the American Statistical Association, 58(301), pp. 236-244, 1963.

[WCG86] P.Wendt, E. Coyle, N. Gallanger, “Stack filters”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 898-911, 1986.

[Wel84] T. Welch, “A technique for high-performance data compression”, *Computer*, 17(6), pp. 8-19, June 1984.

[WSS00] M. Weinberger, G. Seroussi, G. Shapiro, “The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS”, *IEEE Transaction on Image Processing*, 9(8), pp. 1309-1324, 2000.

[WS82] G. Wyszecki, W. Stiles, *Color science: concepts and methods, quantitative data and formulae*, John Wiley & Sons, New York, 1982.

[WM97] X. Wu, N. Memon, “Context-based, adaptive, lossless image coding”, *IEEE Transactions on Communications*, 45(4), pp. 437-444, 1997.

[Wu92] X. Wu, “Color quantization by dynamic programming and principal analysis”, *ACM Transactions on Graphics*, 11(4), pp. 348-372, 1992.

[Wu96] X. Wu, “YIQ vector quantization in a new color palette architecture”, *IEEE Transactions on Image Processing*, 5(2), pp. 321-329, 1996.

[WZ91] X. Wu, K. Zang, “A better tree-structures vector quantizer”, *In Proceedings of the IEEE Data Compression Conference*, Snowbird, Utah, USA, pp. 392-401, 1991.

[XJ94] Z. Xiang, G. Joy, “Color image quantization by agglomerative clustering”, *IEEE Computer Graphics Applications*, 14(3), pp. 44-48, 1994.

[Xu04] M. Xu, "Delta-MSE dissimilarity in GLA-based vector quantization", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, (ICASSP'04), Montreal, Canada. 2004. (to appear)

[YKO98] Y. Yoo, Y. Kwon and A. Ortega, “Embedded image-domain adaptive compression of simple images,” *32nd Asilomar Conference on Signals, Systems and Computers*, 1998.