UNIVERSITY OF JOENSUU
COMPUTER SCIENCE
DISSERTATIONS 20

ALEXEY PODLASOV

PROCESSING OF MAP IMAGES FOR IMPROVING QUALITY AND COMPRESSION

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 December 10th, 2007, at 13 noon.

UNIVERSITY OF JOENSUU
2007

Supervisor	Professor Pasi Fränti
	Department of Computer Science and Statistics
	University of Joensuu
	Joensuu, Finland
Reviewers	Professor Jukka Teuhola
	Department of Information Technology
	University of Turku
	Turku, Finland
	Associate professor Su Yang
	Department of Computer Science and Engineering
	Fudan University
	Shanghai, China
Opponent	Professor Pekka Toivanen
	Department of Computer Science
	University of Kuopio
	Kuopio, Finland

ISBN 978-952-219-069-7 (printed)
ISBN 978-952-219-070-3 (PDF)
ISSN 1796-8100 (printed)
ISSN 1796-8119 (PDF)
Computing Reviews (1998) Classification: I.4.2, I.4.3, I.4.10, C.3
Yliopistopaino
Joensuu 2007

Processing of map images for improving quality and compression

Alexey Podlasov

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

University of Joensuu, Computer Science, Dissertations 20
Joensuu, 2007, 93 pages

Abstract

The thesis is dedicated to processing of map images for improved filtering and compression. The main purpose of the study is the developing of optimized algorithms taking the properties of map imagery into account for improving performance of map processing techniques. The research consists of two major areas.

The first topic is layer-wise enhancement and compression of map images. Firstly we propose binary morphological restoration technique for semantic layers of the map. The proposed technique reconstructs layers corrupted by color overlapping and allows achieving better map image compression. Secondly, we study bit plane separation, predictive modeling and highly optimized context modeling for compression of natural and palette images. Extensive evaluation of standard and novel compression techniques is presented.

The second part of the thesis is dedicated to context tree modeling for filtering and compression of map images. Firstly, we propose generalized context tree based statistical filter for map images. The filter uses variable-size local probability estimator for effective detection of statistical inconsistencies and preserving the detailed areas of the image. Secondly, we propose the using of optimized free tree modeling and better color quantization for recently proposed progressive lossy-to-lossless compression algorithm. The new algorithm provides better compression and quality of the lossy progression. Thirdly, we propose a novel scheme for lossy compression of scanned map images based on color quantization and generalized context tree modeling. The new approach provides better lossy performance in sense of compression-quality tradeoff.

Keywords: Map images, lossless compression, lossy compression, reconstruction, filtering, context tree, bit plane separation, mathematical morphology.

Acknowledgments

I am very thankful to my supervisor Prof. Pasi Fränti for his support and guidance over the years I've spent in Speech and Image Processing Unit at the Department of Computer Science and Statistics, University of Joensuu.

I owe many thanks to Dr. Alexander Kolesnikov for his help, inspiring ideas and valuable advices. Working with you was a great pleasure for me.

I would like to express my gratitude to the staff of the department, especially to Merja Hyttinen, for her help and patience. I thank all my colleagues, former and present, especially Alexander Akimov, Victoria Yanulevskaya, Evgenia Chernenko, Ville Hautamäki and, of course, Alexey Andriyashin for making my working life more enjoyable and memorable. The supervisor of my master's thesis Eugene Ageenko deserves separate thanks for giving me a good start.

I am indebted to my reviewers Prof. Jukka Teuhola and Prof. Su Yang for their benevolence. I thank Prof. Pekka Toivanen for agreeing to be my opponent.

I sincerely wish to thank Prof. Chih–Cheng Hung and his wife May for hosting me on my trip to Atlanta, those days were filled with hospitality and warmth.

Also, I would like to express my gratitude to Tekniikan Edistämissäätiö for provided financial support.

I am deeply thankful to my beloved parents, Sergei Podlasov and Natalia Podlasova, who gave me their love and encouragement. I am proud to be surrounded by beautiful friends: Eugene Galiulin, Andrey Dyatlov, Konstantin ‘Foof’ Kozlov, Alexey Andriyashin, Ilia and Irina Lysenko and others, please forgive me for not mentioning your names =). Most of all, I wish to thank my dear Elena for her patience and sincerity.

Joensuu, November 2007

Alexey Podlasov

List of original publications

P1. Podlasov A., Ageenko E., Fränti P., Morphological reconstruction of semantic layers in map images. *Journal of Electronic Imaging*, 15(1), 013016, January–March 2006.

P2. Podlasov A., Fränti P., Lossless image compression via bit-plane separation and multi-layer context tree modeling, *Journal of Electronic Imaging*, 15(4), 043009, October–December 2006.

P3. Podlasov A., Fränti P., Merge-based color quantization and context tree modeling for compression of color-quantized images", *IEEE International Conference on Image Processing (ICIP'06)*, Atlanta, Georgia, USA, pp. 2277–2280, October 2006.

P4. Podlasov A., Kopylov P., Fränti P., Statistical filtering of raster map images using a context tree model, *Int. Conf. Signal-Image Technology & Internet-based Systems (IEEE-SITIS'07)*, Shanghai, China, December 2007. (to appear).

P5. Podlasov A., Kolesnikov A, Fränti P., Lossy compression of scanned map images, *17th International Conference on Computer Graphics and Vision (Graphicon'07)*, Moscow, Russia, pp. 79–83, June 2007.

Contents

1 Introduction

The use of *Geographical Information Systems* (GIS) to provide users with digital navigation information is widespread and becoming more and more popular. Examples of this are the personal car navigator and PDA-based digital topographic maps for foresters, geologists and engineers. Usually the architecture of such systems does not depend on the application area. A typical example is a system where the user's coordinates are obtained via a satellite positioning service, such as *Global Positioning System* (GPS), and geographical information about the current location is obtained from a local or remote map database.

Map images in a database can be stored in two principally different formats: *vector* and *raster*. The vector format assumes that the map is stored as a set of geometrical primitives (lines, symbols, curves, textures) describing the image content. Each primitive is described by a set of required parameters. For example, a straight line segment is described by four real numbers defining the end points. In order to be displayed, the data must be projected on a plane with the desired scale and rotation, and then drawn on the screen of the client device. However, some geographical data is still unavailable in vector form, and the only sources are traditional maps printed on paper sheets. Although vectorization is an actively developing technology [1][2], a universal non-supervised vectorization algorithm still does not exist. It is often too expensive to manually convert such data into vector format, and therefore storage in raster format can be a better solution.

Raster format assumes that the image is stored as an array of values that represents the rectangular matrix of pixels forming the picture. Depending on the application, the storage of one pixel requires one bit for binary images, one byte for gray-scale or indexed images, three bytes for true color, and even more in the case of a multi-spectral image. A natural advantage of raster format is that it does not require any additional processing for displaying the image. The image can be represented immediately after the data is received. A typical way of combining vector and raster format in the same system is to use the global database stored data as vectors and provide the user with a raster image converted from the vector original to represent the area needed.

The main drawback raster format is that it is not flexible when some transformation of the image is needed. For example, zooming, rotation and projection of the image are all impossible without degradation. The storage size needed is also a problem. In contrast to vectors, raster images store all pixels of the line instead of coordinates of the corresponding segment. In the case of geographical maps, the size of digitized images can be huge. For example, in maps from the *National Land Survey of Finland* (NLS) topographic database [3], a single map sheet of 10×10 km^2 1:20,000 scale is represented by a single image of 5000×5000 pixels, which requires about 70 Mbytes of memory to be stored. Another example is a map of A4 size scanned with 300 dpi in true color, which results in a 2500×3500 pixel image requiring about 25 Mbytes. The necessity for image compression is obvious since more efficient storage space utilization as well as faster map transmission is needed to make digital navigation services more usable, reliable and cheaper.

Features that are distinctive for map images can be characterized as follows. A map image contains only a few unique colors; in cases where the image is converted from a vector source the number of colors rarely grows higher than several tens. A map image also contains a lot of uniform areas representing particular regions like water, forests or background. The areas of the map are usually distinctly separated from each other. This makes a map image which contains sharp and easily localized edges. Smooth gradation is rarely present in map images. A typical map contains thin details and symbols, the presence of which is vital for the semantics of the map. Features of map images are illustrated in Figure 1.

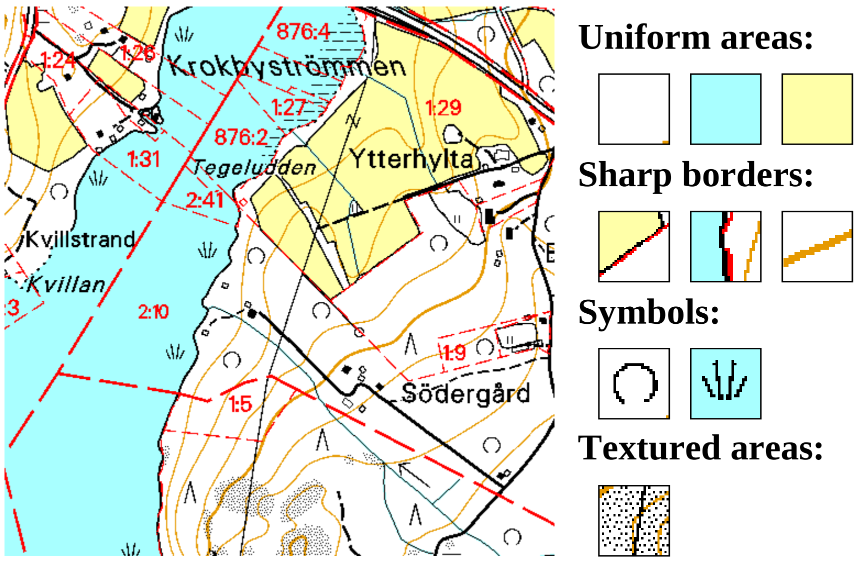


Figure 1. Features of a map image.

In addition to raster maps, which are converted from vector databases, there is another class of scanned map images. These images are produced by digitizing printed paper maps. Scanned map imagery unintentionally combines the properties of natural imagery and converted raster maps. Edges and details on a scanned map are smooth since the image is acquired with a physical sensor of the scanner.

Besides this, a scanned map image may have special patterns such as dithering. The number of paints available in typography is limited and color gradation is usually represented as a pattern of color dots. These structures are acquired by a scanner and appear in the scanned map image. Typical features of scanned map images are illustrated in Figure 2.

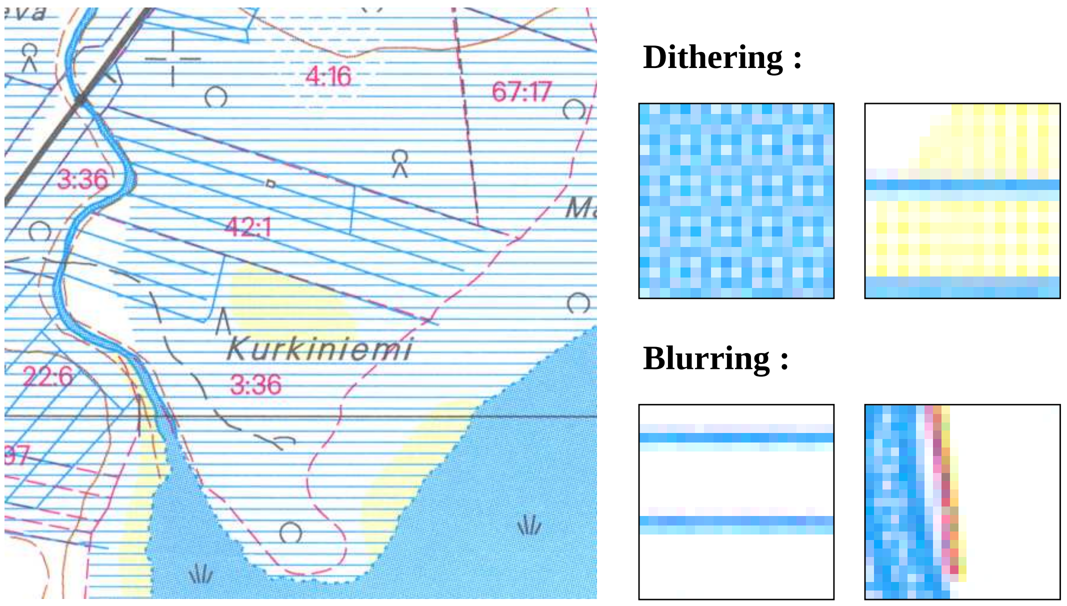


Figure 2. Features in a scanned map image.

A typical scanned map can contain hundreds of thousands of unique colors in contrast to the converted maps that contain only a few. Though these colors are visually grouped around the original colors in the distribution in the color space, they are far from being easily clustered. A visual example is presented in Figure 3 where the color distribution for a sample scanned map image in L*a*b* [4] color space is illustrated. One can see that the distribution does not contain clear centroids. For example, the water pixels, which are supposed to be grouped around a dominant blue color, are actually a mix of blue and white clouds due to the dithering effects. The same holds for the yellow fields. For the whole image this effect makes the distribution uniformly spread.

We consider map imagery as a class of images with distinctive properties separating them from photographic, computer generated or other classes of images. Digital map images (both scanned and converted) are widely used among a great variety of users worldwide. However, general purpose algorithms rarely take the properties of this kind of imagery into account. The work in this thesis is motivated by the fact that better understanding of the properties of map images together with designing and optimization of algorithms exploiting these features can make map image processing and compression algorithms more efficient.

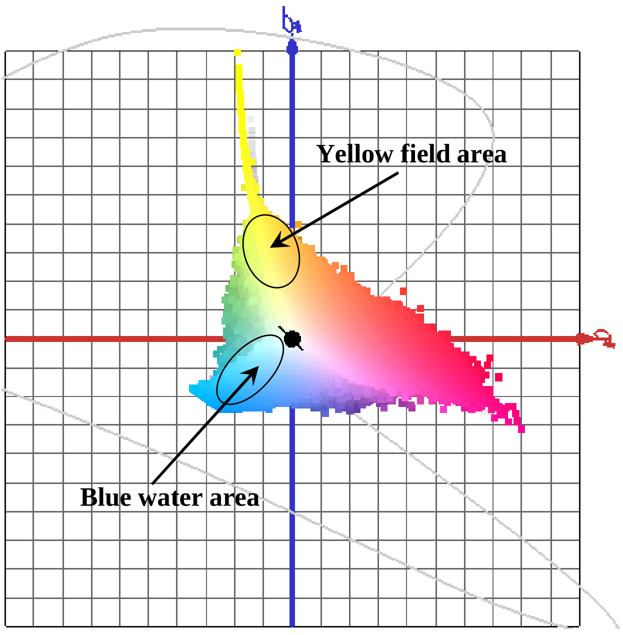


Figure 3. Distribution of colors in a sample scanned map image in L*a*b* color space

This thesis is aimed at two specific topics. The first topic of the thesis is layer-wise map image processing for reconstruction and compression. In paper **P1**, lossless compression is improved by trying to reconstruct the semantic layers of the map. In paper **P2**, bit plane separation and binary context-based compression of natural and palette images are studied.

The second topic is dedicated to context tree (CT) modeling. In paper **P3**, we apply highly optimized CT modeling for the progressive lossy-to-lossless compression of map images. In paper **P4**, we propose a CT filter for improving the quality of noisy map images. In paper **P5**, we generalize the method for lossy compression of scanned maps.

2 Image compression

Compression algorithms can be separated into two classes: *lossless* and *lossy* algorithms [5][6][7]. Lossless compression assumes that the data before and after the compression-decompression process is equal, *i.e.* no loss of information occurs. In contrast, lossy compression makes no such assumption, and allows distortion to happen. This is essential in those situations where some degradation of the data is tolerable for the benefit of better compression efficiency. The algorithms of the first type (lossless) are used in applications where information loss is not acceptable, *e.g.* compression of text, programs and executable code. In image compression, lossless compression can be used for compression of medical images, engineering drawings and circuits. The algorithms of the second type (lossy) are applied in photographic image, audio and video compression because minor degradation can be tolerated if it is visually not perceptible, and because lossless methods alone are inefficient for this type of data.

2.1 Lossless compression algorithms

Images as a class can be of very different natures, structures and contents. Therefore, any successful compression technique is usually adapted to be applied on a particular type of images. Lossless image compression algorithms can be organized into three groups: *continuous-tone*, *discrete-tone* and *universal* algorithms. The compression algorithms referred as continuous-tone are optimized to perform on *natural imagery*, usually photographic or other types of images obtained with a physical sensor. Discrete-tone algorithms are designed to perform on other types of images that contain fewer colors and less gradation, with more sharp edges and uniform areas. Images of this type are mostly of an artificial nature such as web graphics, engineering drawings, maps and circuits. Universal compression algorithms are usually applied when the type of the data is not predefined.

Popular universal compression techniques are based on various adaptations of a classical dictionary-based LZ77 or LZ78 [13][14] algorithm. For example, CompuServe *Graphics Interchange Format* (GIF) [8], which is widely used for the compression of palette images, uses LZC [9] improvement of LZW [10]. The *Portable Network Graphics* (PNG) algorithm [11], which was proposed as the replacement for the relatively old GIF, uses *DEFLATE* [12] algorithm. It uses a combination of LZ77 [13] and *Huffman coding* [15]. The *ITU Group 4* algorithm [16] incorporated in *Tagged Image File Format* (TIFF) [17] uses simple data compression techniques based on run-length coding, prefix coding and differential *relative address designate* (READ) coding to utilize line to line correlations.

However, universal compression algorithms suffer from the one-dimensional nature of the method, and thus present relatively low compression efficiency.

A natural way to increase the efficiency of the compression algorithm is to optimize the compressor for the particular class of images. This approach was realized in *Joint Bi-Level Image Experts Group* (JBIG) [18], which is an algorithm optimized for bi-level images containing pixels of two types: background and foreground. As originally proposed in [21][22], the algorithm is based on local probability estimation via context modeling followed by an arithmetic coding [19] performed by Q-Coder [20]. The JBIG standard was then expanded by JBIG2 [23][24].

Popular examples of lossless continuous-tone compressors are CALIC and JPEG-LS. *Context-based adaptive lossless image compression* (CALIC) [25][26] is based on *gradient-adjusted prediction* (GAP), which is adjusted via an error feedback loop. The residue of the predictor is entropy-coded based on eight estimated conditional probabilities in eight different contexts. JPEG-LS [27] is based on *Low Complexity Lossless Compression for Images* (LOCO-I) [28], which is also based on context-adaptive prediction and adaptive *Golomb-Rice coding* [29][30].

Discrete-tone images are of a different nature and prediction-based techniques usually fail to present high compression efficiency. Discrete-tone oriented compressors exploit different ways of removing the redundancy. For example, the *Piecewise-Constant Image Model* (PWC) compression algorithm [31] uses a two pass model to capture the characteristics of a discrete-tone image. During the first pass, boundaries between constant color areas are detected. The second pass then determines the color of the area. Encoding is performed in an object-oriented manner using the *PWC language* consisting of four decision possibilities.

Embedded Image-Domain Adaptive Compression (EIDAC) [32] compresses an input image as a sequence of bit planes starting from the most significant to the least significant bits allowing a progressive transmission. Coding is performed by inter-layer context modeling and an adaptive binary arithmetic coder providing high compression efficiency.

2.2 Context modeling

Context modeling is a well known tool which is widely used in image compression [22]. The main idea is to estimate the probability distribution of symbols in the input data using the knowledge of the context in which the unknown symbol appears. This concept is effective when there are statistical dependencies in the data, which holds true for most of the images. Usually, the encoder gathers the statistics of pixel occurrences taking into account the configuration of the already processed neighboring pixels. Using this information, variable length code words are assigned to pixels so that shorter codes are assigned to more probable pixels and vice versa.

We refer to a configuration of positions of neighboring pixels as a *context template*, *i.e.* the context template defines the shape of the neighborhood to be examined. The choice of the context template is essential for the compression performance. We refer a configuration of pixel values in the neighborhood as a *context*. The principles of context-based probability estimation are easy to illustrate for a binary case when only back- or foreground pixels are possible in the image. Sample 10-pixel context template used in JBIG compression standard [18], and sample contexts for a binary and four-color image together with the corresponding probabilities are shown in Figure 4.

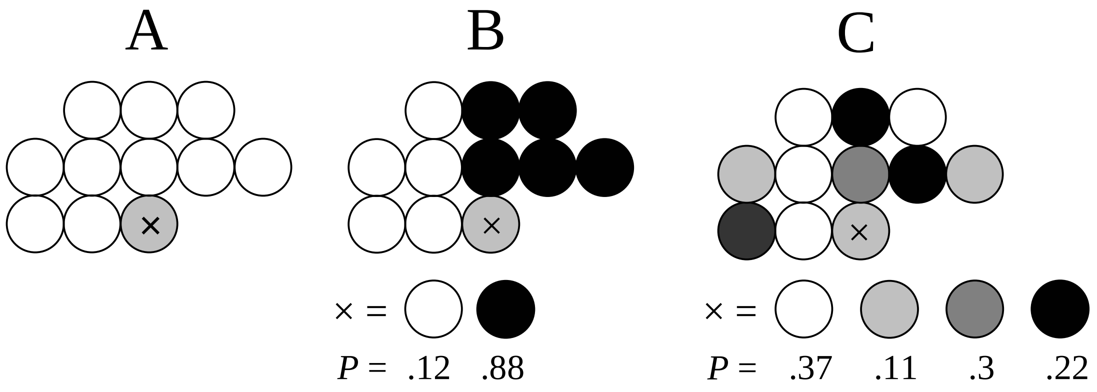


Figure 4. A sample 10-pixel context template (left); a binary 10-pixel context with the corresponding probability distribution (middle); a four-color 10-pixel context (right) with the corresponding probability distribution.

When using large context modeling, its extensive memory consumption is a major problem. With any increase of the context template size the number of possible contexts grows exponentially; for an N-pixel context there can be 2^N contexts in total. In a binary case, one must keep two counters for each context to track the probability distribution. Though this number is reasonable for $N \approx 10$, any further increase of the model size is problematic. Although K number of colors is possible in the image, the number of possible contexts raises to K^N making the storage of K counters for each context impossible.

Another problem with context modeling is *context dilution* [66]. Usually, a bigger context allows more accurate estimation of the probabilities. However, at some point the improvement will stop and any further increase becomes counter productive. Since the image is restricted by size, bigger contexts tend not to appear frequently enough to make an accurate estimate of the probability. Wrong estimation causes deterioration of the compression efficiency.

Both problems are usually overcome by considering *context tree* (CT) modeling [22]. The idea behind CT is that although the number of possible contexts is huge, the number of contexts actually appearing in an image is upper limited by the size of the image. Therefore, if memory is allocated only for really appearing contexts, a reliable estimation of the pixel probabilities becomes possible.

2.3 Context tree modeling

The storage of pixel counters in CT is organized in a tree structure, see Figure 5. The nodes of the tree represent the contexts appearing in the image. Symbol '×' denotes the unknown pixel within the particular context. The statistics *i.e.* the counters represent how many times the unknown pixel appeared as a particular color. In case of binary image, two counters are needed: N_W represents the number of white pixels which appeared, and N_B represents the number of black pixels.

The positions of the context pixels in the context template are arranged in a predefined order; that is, the construction of the tree starts from the root. For every pixel of the image, the algorithm sequentially examines its neighbors according to the defined context template. If the first pixel of the template appears as white, the transition to the left child node is made; otherwise the right transition is made. The process continues recursively: the algorithm examines the second neighbor position in a template and makes the transition to the next level of the tree corresponding to the value of the next context pixel. With every transition from node to node, the algorithm updates the pixel counters according to the value of the current (unknown) pixel. In cases where the current transition does not exist, the necessary node of the tree is created dynamically. Accordingly, only nodes corresponding to existing contexts are created in the tree and no memory is wasted for non-existing pixel combinations.

In order to prevent a context dilution problem the size of the model used must be restricted. In context modeling this is usually done by using a *context quantization* approach [67], which is referred as *tree pruning*. The simplest way is to require that every node has children only if the code size provided by the children is less than the one provided by their parent. This guarantees that all surplus nodes of the tree will be pruned and the undesired increase of the context model will be prevented. However, this greedy-style approach does not provide the optimal performance and more sophisticated pruning algorithms can be found in literature [68][72][73].

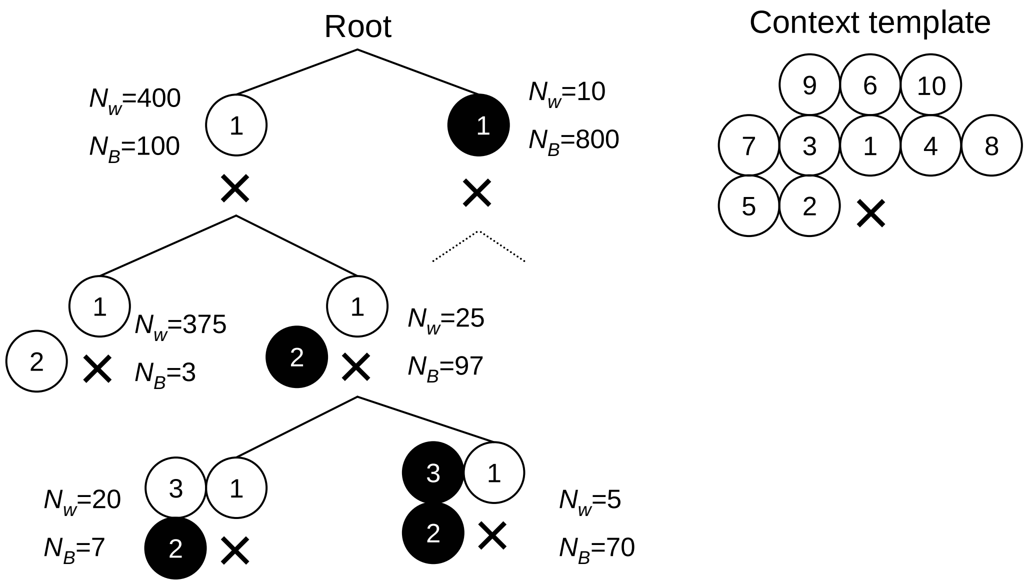


Figure 5. Construction of binary context tree

Although context tree modeling is not restricted to binary images, resource allocation problems limit the use of a more general approach for more than two colors. The principle of constructing a *generalized context tree* (GCT) as proposed in [72] is illustrated in Figure 6. In contrast to the binary case, the nodes of the tree have more than two children. Potentially there are as many children as there are colors in the image. Every node must track the appearance counters for all possible colors. Pruning is also more complicated in the generalized case. For K children, there are 2^K pruning configurations and each can provide different code size. The selection of the optimal combination by a full search is extremely slow and therefore impractical. In GCT, sub-optimal pruning by a *steepest descent search* algorithm was considered for solving the pruning problem in a reasonable time, and still providing performance close to the optimum.

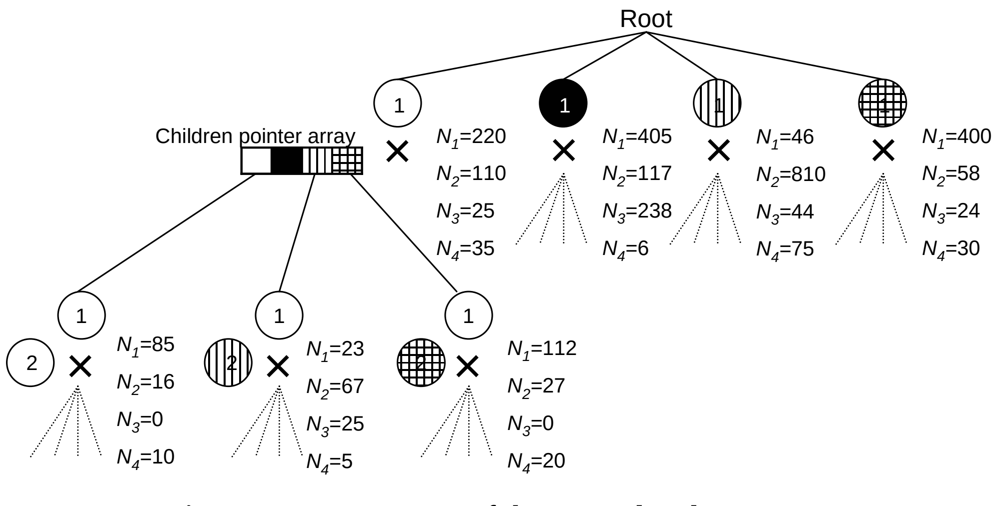


Figure 6. Construction of the generalized context tree

The order of pixel positions in a context template is also essential and can be a subject of optimization. A solution for the CT model is called *Free Tree* [73] and it has been considered both for binary images [70] and for gray-scale [72]. It provides better compression performance in comparing to a static order CT [68]. Sample contexts optimized by free tree are illustrated in Figure 7

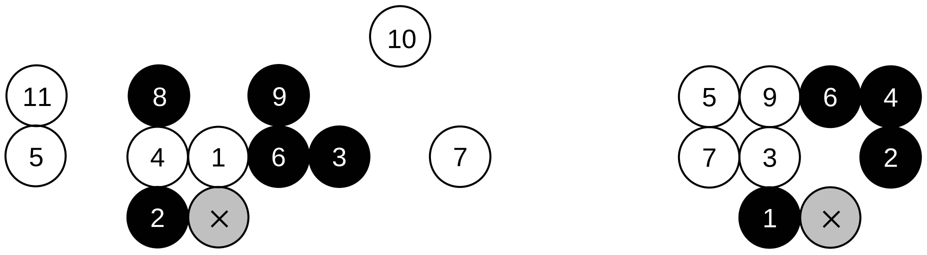


Figure 7. Two sample free tree contexts.

2.4 Lossy compression algorithms

Most popular lossy algorithms are used for compression of photographic imagery since the nature of the human eye's perception allows significant reduction of information in the image without any subjective loss of quality. However, in some applications the properties of the input imagery can differ significantly from natural photography, thus requiring different compression principles to be applied.

2.4.1 Existing methods

The classical examples of popular lossy compression algorithms are *Joint Photographic Expert Group* (JPEG) [33] and a more recent standard JPEG2000 [34]. These algorithms are based on image transforms: *discrete cosine transform* (DCT) [35] for JPEG and *wavelet transform* [36] for JPEG2000. The transform coefficients are rounded and quantized causing partial loss of information. These algorithms are optimized for compression of photographic images, which are mostly used in computer industry. There are also transform-based algorithms optimized for different tasks such as *Enhanced Compression Wavelets* (ECW) [37] and *Multiresolution Seamless Image Database* (MrSID) [38]. These are commercial solutions for the compression of aerial and satellite photos. DCT and especially wavelet based algorithms present excellent compression efficiency in terms of compression *vs.* degradation tradeoff for the class of images to which they were optimized.

The DjVu [39] algorithm was proposed for lossy compression of scanned imagery containing text and line drawings, especially scanned books. This algorithm utilizes the fact that scanned images of that type contain a lot of sharp edges and details, which are difficult to represent by DCT or wavelets. The algorithm therefore separates the image into two parts: text and background, and applies different compressors for each. The binary context-based algorithm JB2 is a variant of JBIG2 [23] standard and is applied for text. The low resolution wavelet-based IW44 is proposed to compress the background.

Lossy predictive coding is also used for the *near-lossless compression* when the degree of imposed degradation is limited. Lossy predictive coding assumes that the prediction error is not encoded precisely but quantized, thereby causing minor errors when the image sample is reconstructed. This technique is used in JPEG-LS [27] near-lossless mode, for example.

Quantization of signal can also be seen as an approach of lossy compression [40]. Reducing the number of unique colors (or gray scale gradations) in the image imposes distortion, and at the same time, reduces the informational content of the image, thus improving its compressibility. For example, the GIF standard operates only on indexed palette images requiring quantizing colors to a predefined palette (typically 256-color) before the compression. The impact of quantization on compression efficiency has been studied in several papers [41][42][43].

2.4.2 Lossy-to-Lossless approach

In some applications, it is not necessary to transfer the whole image data in one continuous transmission. It is often more important to have a schematic thumbnail of the image faster than the whole image. This requirement is typical for browsing and retrieval applications in restricted bandwidth transmitting channels, when one must decide whether the acquired image is relevant to the query.

This task is usually solved by designing the compression algorithm in a way that allows *lossy-to-lossless (progressive) decompression* [34][44]. The image is decompressed step-by-step so that the most important part of the information is decompressed first. Each step then updates the data finally giving the exact lossless reproduction of the image. An importance criterion is usually defined by minimizing the *mean squared error* (MSE) distance from the partially reconstructed image to the original. An example of progressive reconstruction is given in Figure 8 where JPEG quality progression is illustrated. Progressive decompression is a popular feature of existing compression standards such as JBIG [18], JPEG [33], JPEG2000 [34], GIF [8] and PNG [11].

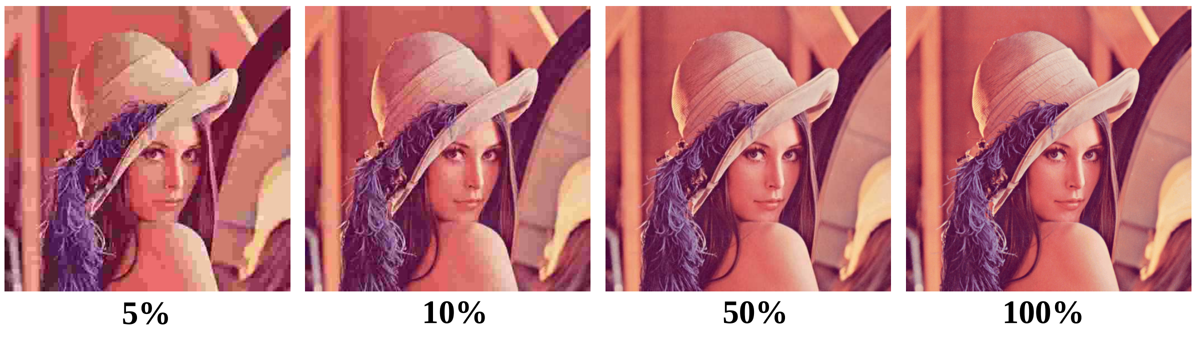


Figure 8. Sample JPEG quality progression.

2.4.3 Lossy compression by color quantization and GCT modeling

The class of scanned raster map images is commonly used in navigational applications in cases when vector map is not available. A scanned map image combines the properties of both the natural class and artificial imagery class. They originally contain only a few colors, sharp edges and small details. After the scanning, however, this image is corrupted by noise caused by the acquisition sensor imposing blurring and other inconsistencies. Therefore, neither traditional lossy image compression algorithms like JPEG and JPEG2000, nor lossless image compression techniques like PNG are well suited for scanned maps.

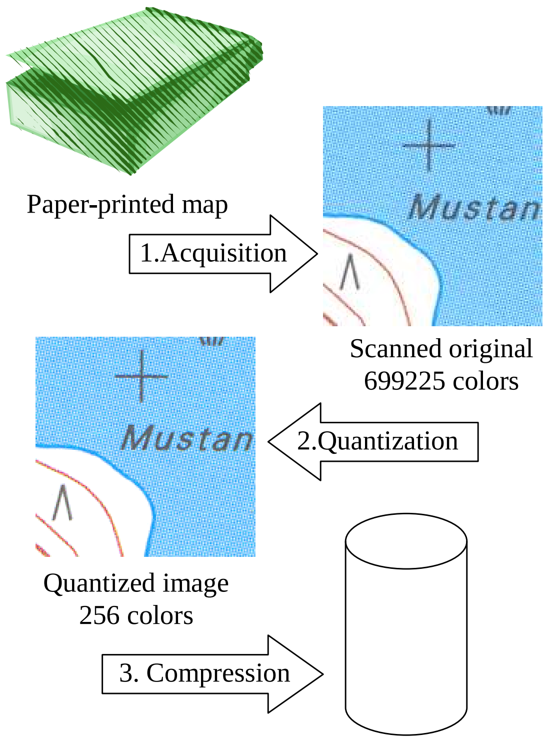


Figure 9. Overall scheme of the proposed lossy compression technique.

In **P5,** we propose an algorithm for lossy map image compression based on *Median Cut* [71] color quantization and generalized context tree modeling (GCT) [72]; see Figure 9 for the overall scheme. Samples representing the quality provided by the proposed technique are presented in Figure 10. The upper row represents 0.72 bit per pixel compression results, and the artifacts and blurring imposed by JPEG2000 along the edges are clearly seen. The corresponding image provided by the proposed algorithm is free from these artifacts. The lower row represents a higher quality level for the proposed technique using 256 colors. Although any difference with JPEG2000 is hardly visible, the objective measurement shows one advantage of the proposed algorithm. In general, when comparing images at a similar objective quality level the proposed algorithm provides up to 50% better compression efficiency than JPEG2000.

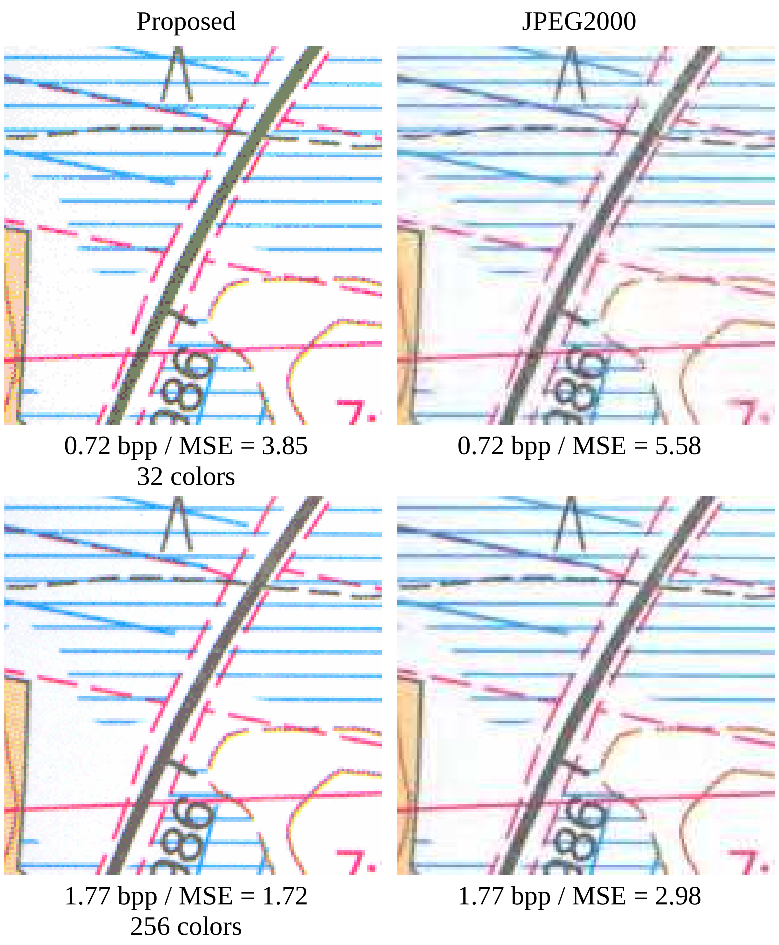


Figure 10. Visual comparison of JPEG2000 and the proposed lossy compression algorithms.

3 Image filtering

Image filtering aims at reconstructing the original image before degradation [45][46][47][48]. As a rule, the reconstruction involves a criterion for measuring the quality of the desired result. There are two different approaches for the quality measurement: *objective* and *subjective*. Objective quality measurement assumes that it is possible to establish an objective metric. The most common examples of these metrics are MSE and *peak signal-to-noise ratio* (PSNR). The objective measurement measures a 'distance' between the original image and the result of reconstruction. This is possible when the original image is available for measurement, which is not always the case.

Another approach is subjective quality estimation. In the case when the uncorrupted image is not available, one can estimate the restoration quality by subjective observations of the reconstructed image. This approach is less analytical than the first one and, therefore, less popular. Besides the two above mentioned approaches, different performance evaluation methods can be defined. For example, in **P1** we use image compressibility as a quality evaluation criterion.

3.1 Existing algorithms

Linear filtering is an approach that has been widely used since the beginning of the computer era. The filter replaces a pixel with a linear combination of its neighbors combining the simplicity of implementation with robustness to various tasks from smoothing to edge detection. Linear filters, however, are not well suited for filtering of map images since the imposed smoothing is not (always) tolerable. Linear filters homogeneously process all pixels, which is another drawback for a filtering of images consisting of complex structures.

Later, a great variety of more general *non-linear filtering* algorithms were considered. In the early sixties, the investigations of Matheron and Serra led to a new quantitative approach in image analysis, now known as *mathematical morphology* [49][50][51]. The central idea of mathematical morphology is to examine the geometrical structure of an image by matching it with small patterns at various locations in the image. By varying the size and shape of the matching pattern, called *structuring element*, one can obtain useful information about the shape of the different parts of the image and their interrelations. Flexibility of the concept allows various filters to be designed [52][53][54][55]. Mathematical morphology is widely applied in various disciplines such as mineralogy, medical diagnostics, machine vision, pattern recognition, granulometry and others [56].

There exists a great variety of heuristical filtering approaches, which exploit knowledge about the noise. For example, edge preserving filters are trying to smooth uniform areas while keeping the edges untouched. One of the most popular edge preserving filtering methods is *vector median filter* (VM) [57], which is a non-linear operator. The filter replaces the current pixel value with a value called *vector median* defined in a local neighborhood. An attempt to design a filter that would be invariant to the features of the particular image was made in [58]. The filter is called *rank-conditioned vector median filter* or *adaptive vector median filter* (AVM), and it uses a noise detector before applying VM. An overview of weighted median filters can be found in [59].

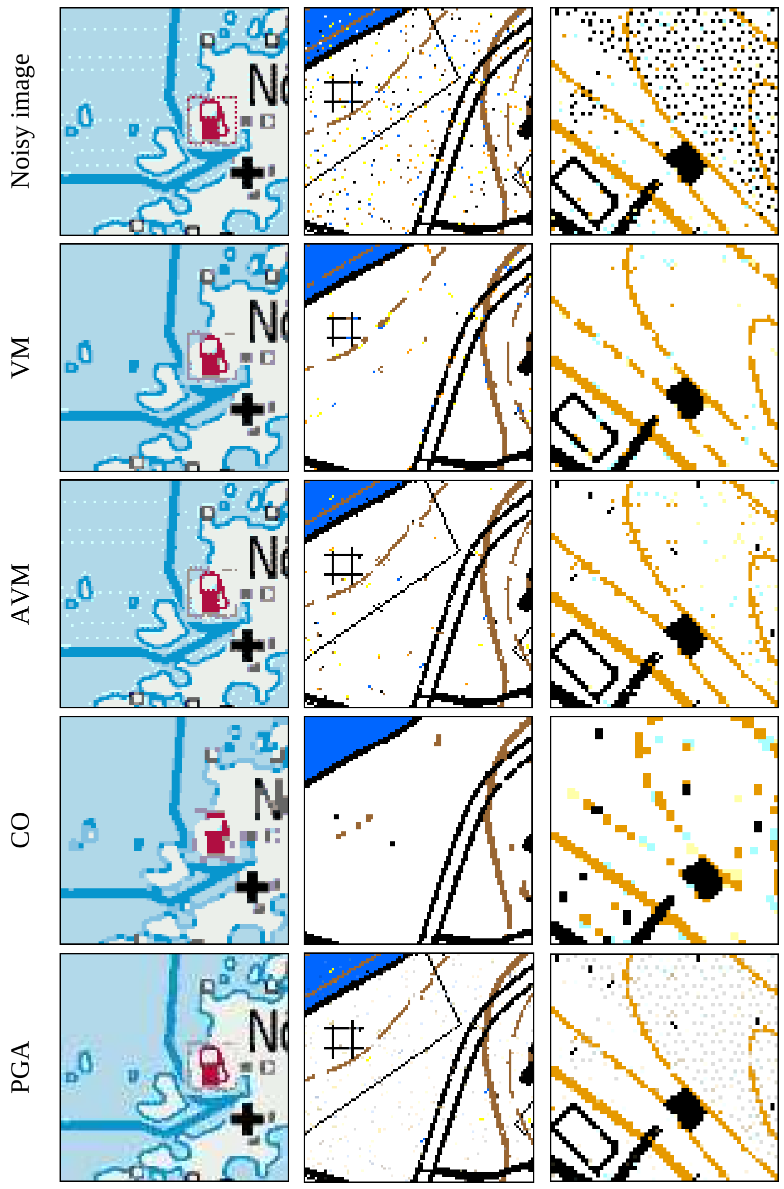


Figure 11. Application of vector median (VM), adaptive vector median (AVM), morphological Close-Open (CO) and peer-group analysis filters to sample map images.

Classical *Kuwahara filter* [60] examines the neighborhood for a sub-region with the smallest variance and replaces the current pixel with the mean of the region. A similar approach is used by *peer group analysis* (PGA) [61], which is an edge-preserving smoothing technique based on finding a group of pixels similar to the current one in a local neighborhood. When such a group is found, the current pixel is replaced with the average of the group. Statistical non-linear filters use local probability estimation for noise detection and correction. A *gain-loss filter* was proposed in [62] for improving the compression of binary images. Various vector-based filters are discussed in [63] and [64]. Application of selected VM, AVM, morphological *Close-Open* (CO), and PGA filters is illustrated in Figure 11.

3.2 GCT filtering

Map images are typically highly structured. The patterns are usually clearly defined and commonly repeated in the image. This makes context tree modelling an effective tool for statistical analysis and processing. Consider a map image which is slightly corrupted by impulsive or content-dependent noise. By *content-dependent* noise, we assume that the corruption occurs at the borders of the objects. The presence of noise corrupts the statistical consistency of the image and, therefore, statistical analysis is an appropriate tool for noise detection and removal.

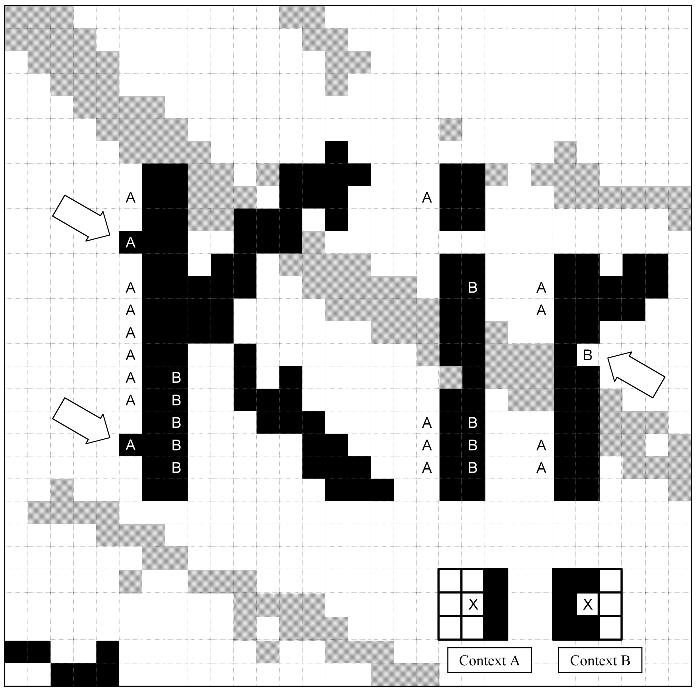


Figure 12. Principles of context-based filter.

In **P4,** we propose a statistical context tree based filter for map images basing this on the preliminary works published earlier [62][65]. The filter analyzes the statistical distribution of the colors within a local neighborhood using a generalized context tree model. Pixels are considered as noisy if their conditional probability falls below a predefined threshold. The size of the neighborhood is dynamically adapted using a tree pruning technique. The principle of the algorithm is illustrated in Figure 12, where two 3×3 contexts A and B are presented. One can see that black is much less probable than white in the context A, and vice versa; white is less probable than black in the context B. By replacing noisy pixels by the most probable ones, the filter is able to reconstruct the initial structure of the image. The proposed filtering is very sensitive to the original structure of the image and the amount of the corruption imposed is rather small. Sample corrupted and reconstructed images are presented in Figure 13.

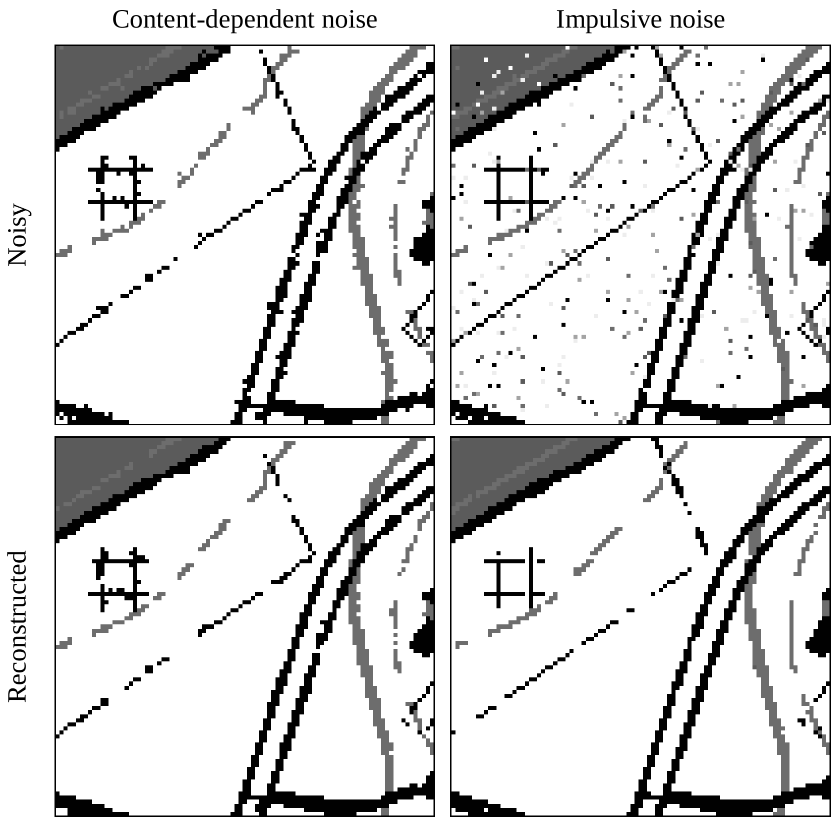


Figure 13. Sample noisy and reconstructed images.

4 Layerwise processing

There is a class of images consisting of a set of binary *layers*. Maps are a typical example of this kind of images since they consist of *semantic layers*: binary images representing geographical objects of similar nature and depicted with a particular color. For example forests are depicted in green and one can extract all green regions of the image into a binary image (a forest layer); see an example of layered image in Figure 14. In an atlas map, colors can represent a great variety of parameters such as density of population, pollution and temperature. Bit plane separation is another example of the layer decomposition approach. A gray-scale image can be decomposed into a set of binary layers according to the bits of each pixel value.

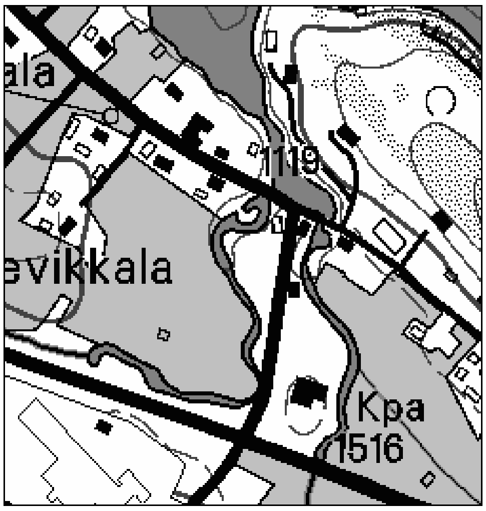


Figure 14. Illustration of a multilayer map image from the NLS topographic database [3].

If there is a strong correlation between layers, this can be utilized to improve the performance of the compression, filtering, or other processing algorithms. This correlation certainly exists in map images between their semantic layers [69]. In [70], inter-layer correlation was used to improve the performance of a two-layer context-based lossless compressor. This motivates us to research layerwise processing of map and gray scale images as a method of improving the performance of lossless compression algorithms.

4.1 Morphological reconstruction of semantic layers

When producing a raster map image, map layers of different semantic nature are combined together overlapping each other in a predefined order. This image is well suited for user observation but less appropriate for further processing since the layer structure has been corrupted as the raster map image was produced. The problem is that the overlapping introduces severe artifacts in places where the information on different layers overlap each other; see Figure 15, upper row. The holes on the face of the lake left by the overlapping letters are typical examples of the artifacts. The presence of these artifacts degrades the compressibility of the color map image, in comparison to the situation when the original semantic layers were available.

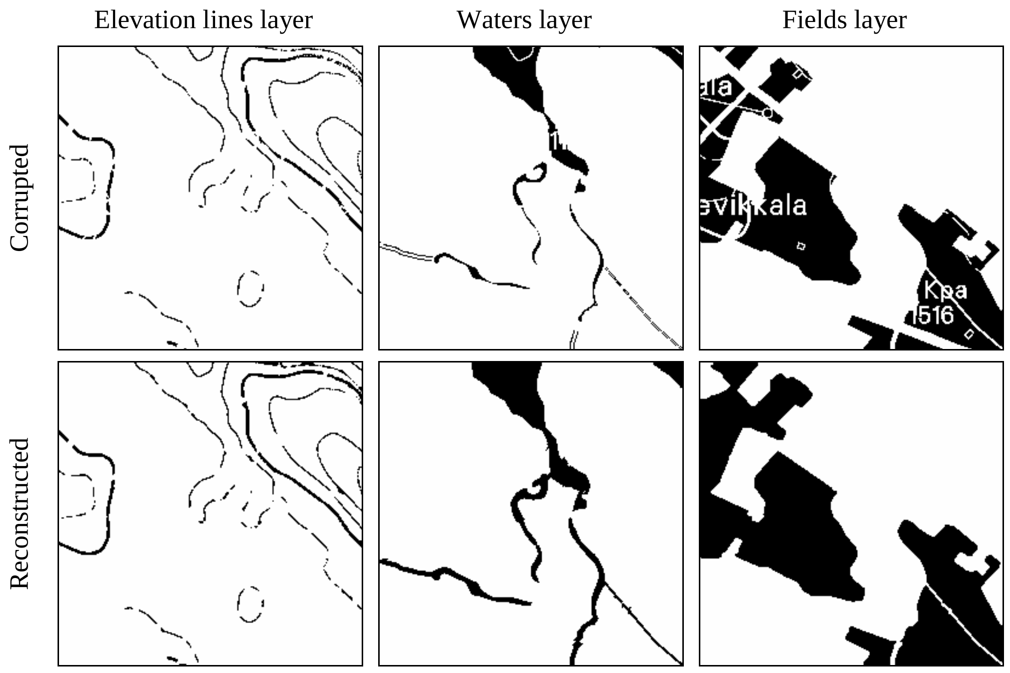


Figure 15. Semantic map layers: corrupted layers due to the color separation (upper row); reconstructed with the proposed algorithm (lower row).

This problem led us to develop an algorithm for the reconstruction of the corrupted layers of map images. The algorithm proposed in **P1** approximates the original layer structure existing before the color combination by repairing the corrupted layers as close as possible to the original ones. Since the converted raster map images are usually compressed by a lossless algorithm, we require that the color combination of the reconstructed layers must be equal to the originally received raster map image.

The results of the proposed reconstruction technique are presented in Figure 15. The removal of overlapping artifacts provides 30-50% better compression on standalone layers, and 5-10% better compressibility for 4-layer map images without any loss of quality. Besides that, the proposed technique can be used for the removal of unnecessary layers from the map.

4.2 Compression of gray scale images

Other types of images can also be treated as layered images via the use of bit plane separation. This assigns one bit from the binary representation of the pixel value into each bi-level layer, thus losslessly separating any gray scale image into eight layers. The overall scheme of this approach is shown in Figure 16. In the case where there is a correlation between the bit layers, it is possible to utilize this to gain better compression efficiency. For example, in context modeling, involving neighboring pixels from already processed binary images can improve the probability estimation and, therefore, the compression. Among existing implementations we can mention the EIDAC [32] lossless compression algorithm, which uses a binary multi-layer context model that operates on bit-layers of the image using both the actual bit values and their differential characteristics as context information. Two-layer context modeling with optimization of the order of layer processing was considered in [70].

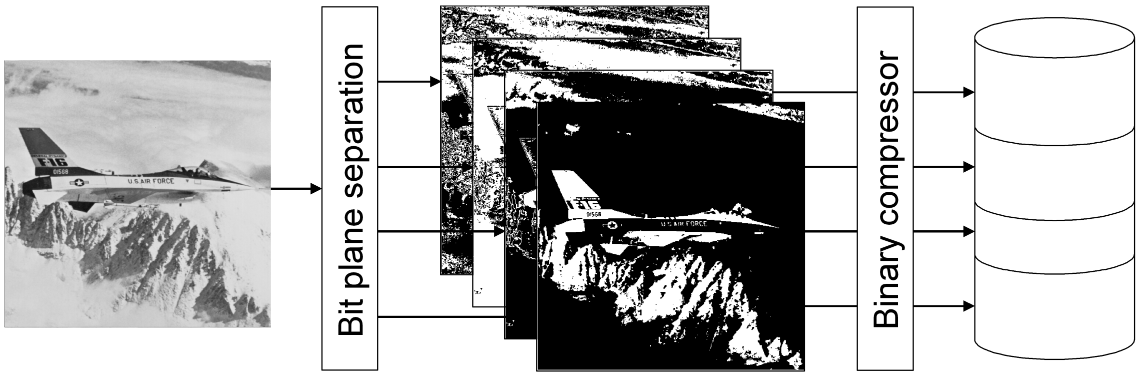


Figure 16. The overall scheme of bit-plane-based compression.

In **P2,** we study how well the bit-plane-based approach can work on natural and palette images. We consider four different bit plane separation schemes: straightforward bit plane separation, Gray-coded bit plane separation, bit plane separation of prediction errors and separation of Gray coded prediction errors. We use the highly optimized MCT context modeling method for lossless compression and, furthermore, extend the two-layer MCT model to a multi-layer context model for better utilization of cross-layer dependencies. In general, any previously compressed layer can be used to provide the contextual information for the next layer being compressed. An example of a multi-layer neighborhood used in **P2** is presented in Figure 17. We extensively evaluate the proposed combinations of the different bit plane separation and context modeling schemes, by applying them to natural and palette images. The efficiency of the bit-plane-based compression is compared to the existing compressors. Moreover, the dependency of the compression on the image content is studied by modeling the transition between natural and palette image classes.

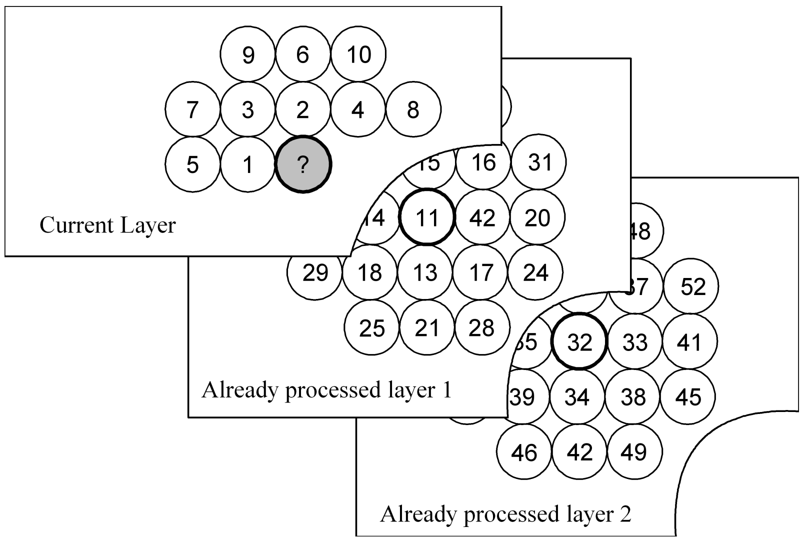


Figure 17. A sample multi-layer context template.

4.3 Progressive compression via binary layers

Binary context modeling is also used for progressive encoding of color quantized images in [74]. The algorithm uses binary tree representation of the color palette followed by a progressive binary context-based encoding. In [75], an improved version has been proposed. In **P3,** we continue the development of this approach by improving the quality of the color progression by using merge-based color clustering [76] instead of the original splitting-based approach. We also propose the use of binary free tree modeling instead of the static context model. The proposed improvements provide better subjective quality of the color progression (see Figure 18), and 10-20% better compression performance for the set of palette images.

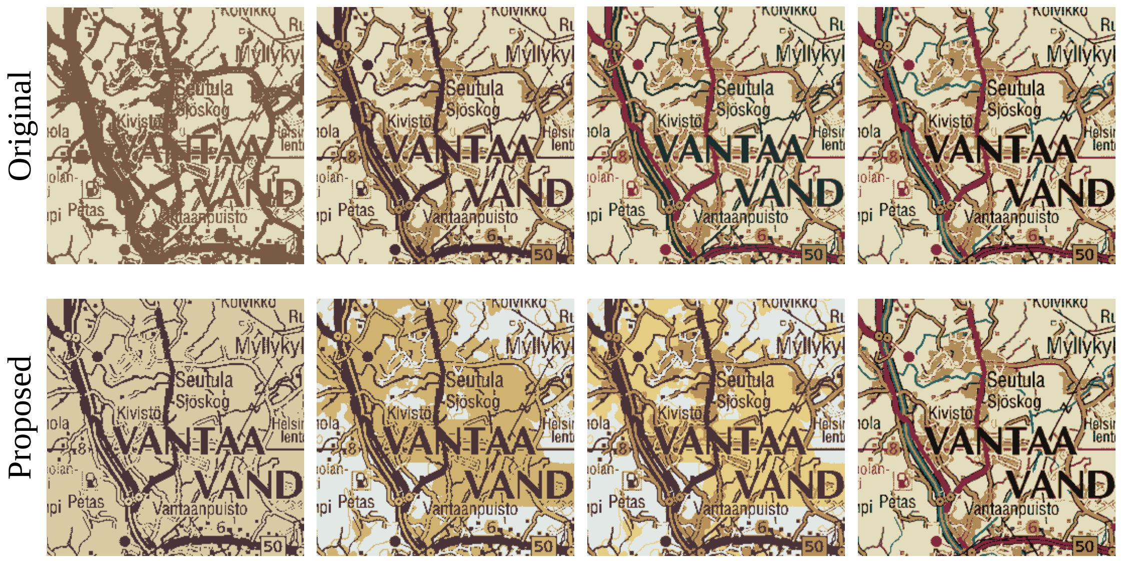


Figure 18. Color progressions provided by the original and proposed algorithms.

5 Summary of the publications

In the first paper (P1), we propose a technique for reconstruction of binary semantic layers of map images from the corruption imposed by overlapping of color layers in the map. Separation of the map into color layers and compressing them individually provides better compression performance than using standard techniques. However, color separation causes artifacts in areas where one layer overlaps another. The proposed algorithm approximates the original structure of the layer existing before the overlap by a sequential application of masked morphological operations. The image is processed so that the color map itself remains untouched, and only the underlying binary layers are modified. The proposed technique obtains up to 30-50% compression improvement for single layers, and improves the compression ratio of the whole map up to 5-10%.

In the second paper (P2), we explore the efficiency of binary-oriented compression algorithms applied to gray-scale and palette images. In contrast to map images, gray-scale imagery contains much more gradation and it is more difficult to exploit spatial dependencies via binary layers and color separation as in **P1** is not possible since it would lead into too many (256) layers. In this work, we consider four different bit-plane separation schemes using error prediction and Gray-coding. For prediction-based schemes we evaluate three different predictors. Bit-plane separation schemes are combined with two binary-oriented compressors: one known (MCT) and one novel referred as *N-layer Context Tree* (NCT).

We evaluate the proposed variants on natural and palette images. The variants providing the best compression are compared with six existing compression algorithms. We conclude that despite the high order optimization a binary-oriented compressor cannot outperform the best lossless gray-scale oriented algorithms due to the nature of the signal. For palette images we found out that highly optimized binary compression is able to provide compression performance close to the best existing compressors but at the cost of higher processing time.

In the third paper (P3), we improve a recently proposed layerwise lossless compression algorithm, which is based on binary tree representation of the colors and on context-based arithmetic coding. We considered two improvements for the algorithm: merge-based color quantization instead of the original split-based strategy, and a context tree modeling optimized for each layer separately. The improved algorithm is evaluated on natural and palette images. The proposed method provides better subjective quality of the color progression, and compression improvement of 12% in the case of color palette images.

In the fourth paper (P4), we propose a statistical filter for map images based on local probability estimation using a context tree. The estimation is followed by replacing less probable pixels by the most probable one. The size of the context is dynamically adjusted according to the proposed tree pruning procedure. The main feature of the proposed filter is that the use of the context tree allows investigating larger neighborhoods with reasonable time and memory consumption. The filter effectively reconstructs patterns of the image in the presence of moderate impulsive and content-dependent noise.

In the fifth paper (P5), we consider a novel lossy compression scheme for scanned map images. The proposed compression algorithm consists of two stages. First the number of colors in the original image is reduced by color quantization. The quantized image is then compressed with the lossless GCT compression algorithm. In this work, two improvements for the original GCT were considered: a fast pre-pruning method and an optimized memory allocation. Both improvements reduce the memory consumption as well as significantly reducing the processing time of the algorithm. The proposed compression scheme is evaluated on a set of scanned topographic maps. The evaluation shows that the algorithm provides a compression improvement of about 50% in comparison to the closest competitor, JPEG2000, at the similar objective quality level.

In paper **P1**, the author developed the principles of the algorithm, implemented and evaluated it. The other two authors took part in the problem formulation and editing of the article. In paper **P2**, the author implemented the N-layer context tree modeling, bit-plane separation schemes, predictors and performed all the experiments. In paper **P3**, the author implemented and tested the improved compression algorithm. Paper **P4** is based on a preliminary version published in a conference by the second and third authors. The contribution of the author includes the tree pruning procedure, a new implementation of the filter with significantly better memory consumption, performing new experiments and a broader evaluation. In paper **P5**, the author considered and implemented the improved GCT compressor and performed the experiments.

6 Conclusions

In this thesis, we have studied lossless and lossy compression of raster map images as well as layerwise processing algorithms for their improvement.

We have proposed a morphological algorithm for restoration of binary semantic layers of multi-layer map images from the corruption appearing in areas where semantic layers overlap each other. The proposed reconstruction allows improving the lossless compression of the layers up to 30-50% for standalone layers and in the total compression rate up to 5 to 10%, depending on the compression method applied.

We have studied the efficiency of highly-optimized binary-oriented compression algorithms to examine if it is possible to utilize their high performance, as presented for maps, for grayscale and palette images. We consider a set of binary layer separation schemes. We also consider two schemes for context modeling: one existing and one novel (NCT). The experiments show that statistical context modeling and arithmetic coding cannot outperform the best grayscale-oriented compressors. On the other hand, when applied to artificial palette-like imagery, the optimization of the model results in a compression performance which is close to the best existing algorithms and further improvement is possible.

We have proposed a novel filter for reconstruction of map images in the presence of noise. In contrast to the existing edge-preserving filters designed to preserve areas of high color variation, our filter aims at preserving the repetitive structures of the image which is an essential property for raster map images. The problems of the appropriate context size and resource allocation are solved. The proposed filter outperforms edge-preserving competitors both in objective and subjective comparisons.

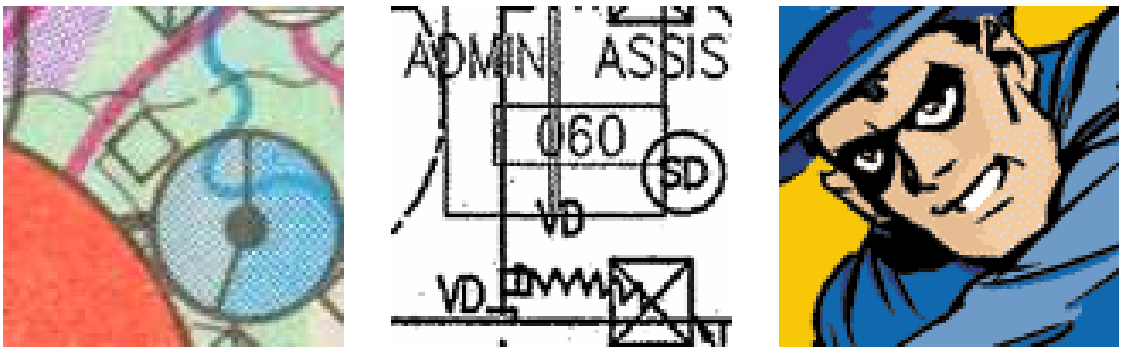
We have improved a recently proposed lossy-to-lossless compression algorithm based on layerwise progressive binary compression. The improved algorithm provids better visual quality of the lossy progression; 12% better compression is achieved for palette images.

We have proposed a novel scheme for lossy compression of scanned map images utilizing the common features of the map imagery. The novel scheme provides up to 50% better compression at the same quality level when compared to its closest competitor JPEG2000.

7 Future work

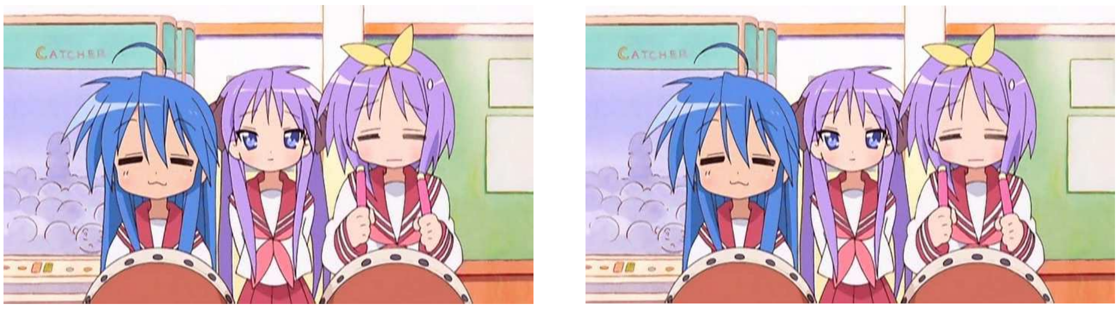
We believe that this thesis can be used as a basis for further research. In **P5** a pioneer work in lossy context-based compression has been done. Although the algorithm is applied on scanned topographic maps only, we expect the results to generalize to similar image classes, such as other types of maps, engineering drawings, schemes, comic books and similar art imagery. Images of this kind have properties similar to map imagery, see Figure 19. A similar lossy algorithm DjVu can be developed where background and textual information would be separated, and the textual part compressed by a GCT-based encoder, which is not restricted to work only for bi-level images as DjVu.

The potential of GCT-based compression is possible to extend to video compression. The method is expected to be efficient for video where features of the imagery are close to the ones illustrated in Figure 19, such as high quality cartoons and animation, see Figure 20. Since subsequent frames of the video are highly correlated, multi-layer GCT modeling is expected to be a very efficient compression tool.



Atlas Engineering drawing Comic book

Figure 19. Sample images with features similar to map imagery.



Anime frame 1 Anime frame 2

Figure 20. Two consequent anime frames.

8 Summary of the results

8.1 Publication 1

Table 1. Compression of topographic map images with different compressors. The average results (size, bits per pixel and compression improvement) presented for original, corrupted with color separation and reconstructed layers.

Compression algorithm	Original		Corrupted		Proposed reconstruction		
	Size	bpp	Size	bpp	Size	bpp	imp.
PNG	2 085 871	0.66	2 149 490	0.68	2 078 254	0.66	3.31%
TIFF	1 473 824	0.47	1 708 362	0.54	1 480 657	0.47	13.33%
JBIG	684 978	0.21	790 257	0.25	720 185	0.23	8.87%
AKF2	624 117	0.19	696 017	0.22	660 661	0.21	5.08%

8.2 Publication 2

Table 2. Compression results (bits per pixel) for the natural images.

Image	Proposed	Competitive							
	MCT-GCPES	JBIG-BPS	JBIG-GCS	CALIC	JPEG-LS	PWC-G	PWC-P	JPEG2K	PNG
Average	4.42	5.64	4.75	**4.11**	4.18	4.21	4.84	4.36	4.56

Table 3. Compression results (in bytes) for the palette images.

Image	Proposed	Competitive							
	NCT-BPS	JBIG-BPS	JBIG-GCS	EIDAC	CALIC	JPEG-LS	PWC-G	PWC-P	PNG
Total	175590	351913	211943	**140957**	226296	272555	198931	144344	245745

8.3 Publication 3

Table 4. Compression results of PNG, Chen's, Pinho's and the proposed algorithm as well as obtained compression improvement (comparing to the closest competitor) for natural and palette test sets.

Test set	PNG	Chen *et al.*	Pinho *et al.*	Proposed	Improvement
Natural	7261542	2521448	2426446	2399451	1%
Palette	712726	274700	257126	226469	12%

8.4 Publication 4

Table 3. The efficiency of mathematical morphology (MM), vector median (VM), adaptive vector median (AVM) and the proposed (CT) filters measured as Δ*E* distance to the original image for 20% content-dependent (CD) and 5% impulsive noise (I).

	Image 1		Image 2		Image 3		Image 4		Image 5		Image 6	
	CD	I	CD	I	CD	I	CD	I	CD	I	CD	I
MM	23.52	24.37	29.66	30.28	27.75	28.33	14.10	14.48	4.54	8.68	30.45	31.11
VM	3.16	2.51	8.50	7.73	8.58	7.37	3.27	2.46	1.99	1.66	7.81	6.67
AVM	2.51	1.70	4.60	2.46	5.05	3.12	2.18	1.18	1.33	1.15	5.07	3.10
PGA	2.51	1.50	5.48	3.71	5.76	3.79	2.24	1.32	1.75	1.56	5.90	4.02
CT	2.14	0.89	3.95	2.89	3.96	2.44	1.70	0.94	1.19	1.18	3.86	2.94

8.5 Publication 5

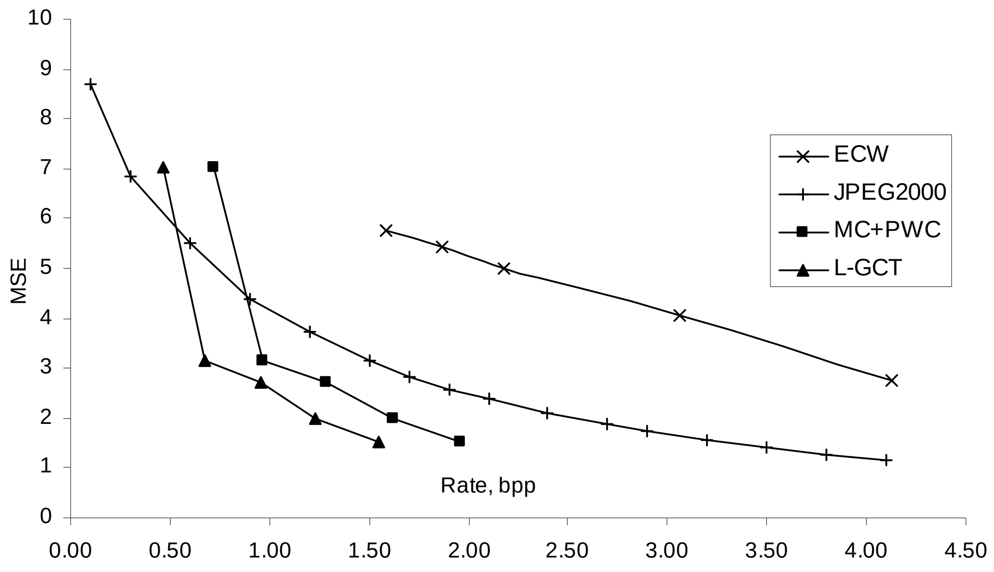


Figure 21. Operational rate-distortion function of the proposed algorithm (L-GCT) and its competitors.

Table 5. Compression performance of JPEG2000 and the proposed algorithm for similar objective quality level.

MSE distance	JPEG2000, Bpp	Proposed, Bpp	Improvement,%
1.52	3.20	1.55	51
1.99	2.40	1.23	48
2.71	1.70	0.95	44

References

[1] S. Ablameyko, T. Pridmore, *Machine Interpretation of Line Drawing Images*, Springer, 2000.

[2] A.K. Chhabra, D. Dori (eds.), Graphics Recognition – Recent Advances, Lecture Notes on Computer Science, vol. 1941, Springer, Verlag, 2000.

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

[4] CIE, *Colorimetry*, CIE Pub. No. 15.2, Centr. Bureau CIE, Vienna, Austria, 1986.

[5] K. Sayood, *Introduction to Data Compression*, 2nd edition, Academic Press, 2000.

[6] D. Salomon, *Data Compression: The complete reference*, 3rd Edition, Springer Verlag, 2004.

[7] I.H. Witten, A. Moffat, T.C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images. 2nd Edition*, Morgan Kaufmann, 1999.

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

[9] S. W. Thomas, J. Mckie, S. Davies, K. Turkowski, J. A. Woods, and J. W. Orost. *Compress (version 4.0) program and documentation*, 1985.

[10] T. Welch, "A technique for high-performance data compression", *Computer Magazine*, vol.17(6), pp. 8-19, 1984.

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

[12] P. Deutsch, "DEFLATE compressed data format specification", RFC1951, http://www.faqs.org/rfcs/rfc1951.html, 1991.

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

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

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

[16] International Telegraph and Telephone Consultative Committee (CCITT), "Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus," Recommendation T.6, 1984.

[17] TIFF Revision 6.0, Adobe Developers Association, Adobe System Incorporated. http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf, 1992.

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

[19] J. J. Rissanen, Langdon G. G., "Arithmetic coding", IBM Journal of Research, Development 23: 146-162, 1979.

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

[21] G. G. Langdon and J. Rissanen, “Compression of black-white images with arithmetic coding,” *IEEE Trans. on Communications*, vol. 29, no. 6, pp. 858-867, June 1981.

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

[23] ITU-T recommendation T.88, “Information technology – coded representation of picture and audio information – lossy/lossless coding of bi-level images”, 2000.

[24] P. Howard, F. Konssentini, B. Martins, S. Forchhammer, W. Rucklidge, “The emerging JBIG2 standard”, *IEEE Trans. On Circuits and Systems for video technology*, vol. 8(7), pp. 838-848, 1998.

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

[26] X.Wu, “L_∞–constrained high fidelity image compression via adaptive context modelling”, *IEEE Trans. on Image Processing*, 9, (2000) 536-542.

[27] ISO/IEC JTC1/SC29/WG1, “JPEG-LS part-2 WD.” ISO/IEC JTC1/SC29/WG1 N938, 1998.

[28] M. Weinberger, G. Seroussi, G. Shapiro, “The LOCO-I lossless image compression algorithm: principles and standartization into JPEG-LS”, *IEEE Trans. on Image Processing*, vol. 9 (8), pp. 1309–1324, August 2000.

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

[30] R. F. Rice, "Some Practical Universal Noiseless Coding Techniques", Jet Propulsion Laboratory, Pasadena, California, *JPL Publication*, pp. 79-22, March 1979.

[31] Ausbeck, P.J., Jr. “The piecewise-constant image model”, *Proceedings of the IEEE*, Vol. 88, Issue 11, pp. 1779–1789, Nov 2000.

[32] Y. Yoo, Y. G. Kwon, A. Ortega, “Embedded image-domain compression using context models”, *Proc. of IEEE Int. Conf. on Image Processing 1999 (ICIP 99)*, Kobe, Japan, vol. 1, pp. 477–481, 1999.

[33] W.B. Pennebaker, J.L. Mitchell, *JPEG: Still Image Data Compression Standard*, New York: Van Nostrand Reinhold, 1993.

[34] D. Taubman, M. Marcellin, *JPEG2000: Image Compression Fundamentals, Practice and Standards*, Kluwer Academic Publishers, 2001.

[35] R.J. Clarke, *Transform Coding of Images*, Academic Press, 1985.

[36] S.Mallat, *A Wavelet Tour of Signal Processing*, 2nd Edition, Academic Press, 1999.

[37] C. Ueffing, “Wavelet based ecw image compression”, *Photogrammetric Week 01*, Wichmann Verlag, Heidelberg, pp. 299–306, 2001.

[38] LizardTech, "MrSID Technology Primer", http://www.lizardtech.com/files/geo/techinfo/MrSID_Tech_Primer.pdf

[39] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, Y. Le Cun, "High Quality Document Image Compression with DjVu", *Journal of Electronic Imaging*, vol. 7 (3), pp 410-425, SPIE, 1998.

[40] A. Gersho, R.M. Gray, *Vector quantization and signal compression.* Norwell, MA: Kluwer, 1992.

[41] W. Ding, Y. Lu, F. Wu, S. Li, "Rate-distortion optimized color quantization for compound image compression," *Proc. SPIE Visual Communications and Image Processing (VCIP 2007)* (6508-98), San Jose, CA, USA, Feb. 2007.

[42] R. Balasubramanian, J.P. Allebach, "Sequantial scalar quantization of vectors: an analysis", *IEEE Trans. on Image Processing*, vol. 4 (9), September 1995.

[43] S.-C. Pei, C.-M. Cheng, "Dependent scalar quantization of color images", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 5(2), pp. 124-139, Apr 1995.

[44] Y.-Kheong Chee, "Survey of progressive image transmission methods", *International Journal of Imaging Systems and Technology, Special Issue: Image and Video Compression*, vol. 10(1), pp. 3-19.

[45] Gonzalez, R. C., Woods, R. E., *Digital Image Processing*, 2nd ed., Prentice Hall, Upper Saddle River, NJ, 2002.

[46] I. Pitas, *Digital Image Processing Algorithms and Applications*, New York: Wiley, 2000.

[47] Dougherty E.R., Astola J. (eds), *Nonlinear Filters for Image Processing*, SPIE Optical Engineering Press, 1997.

[48] Pitas, I., Venetsanopoulos A.N., *Nonlinear digital filters: principles and applications*, Boston, Mass.: Kluwer, 1990.

[49] Serra J., *Image Analysis and Mathematical morphology*, London: Academic Press, 1982.

[50] Matheron G. *Random Sets and Integral Geometry*, J. Wiley & Sons, New York, 1975.

[51] Heijmans H.J.A.M., *Morphological image operators*. Boston: Academic Press, 1994.

[52] Koskinen L., Astola J. "Soft morphological filters: A robust morphological filtering method". *Journal of Electronic Imaging*, vol. 3(1), pp. 60-70, 1994.

[53] M.L. Comer, E. J. Delp, Morphological operations for color image processing, *Journal of Electronic Imaging* 8(3), 279-289, July 1999.

[54] Dougherty E.R., "Optimal mean-square n-observation digital morphological filters. Part I: Optimal binary filters", *Computer Vision, Graphics, and Image Processing*, vol. 55, pp. 36-54, 1992.

[55] Ping Z., Lihui C., Alex K.C., "Text document filters using morphological and geometrical features of characters", *Proc. Int. Conf on Signal Processing-ICSP'00*, pp. 472-475, 2000.

[56] Doughberty E., Lotufo R., Hands-on morphological image processing. *SPIE Optical Engineering Press*, 2003.

[57] J. Astola, P. Haavisto, Y. Neuvo, “Vector median filters”, *Proc. of IEEE*, 78 (4), pp. 678-689, 1990.

[58] R. Lukac, “Adaptive vector median filtering”, *Pattern Recognition Letters*, 24, pp. 1889-1899, 2003.

[59] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, “Weighted median filters: a tutorial,” *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, issue 3, pp. 157-192, Mar. 1996.

[60] M. Kuwahara, K. Hachimura, S. Eiho, and K. Kinoshita, “Digital Processing of Biomedical Images”, *Plenum Press*, pp. 187-203, New York, NY, 1976.

[61] C. Kenney, Y. Deng, B.S. Manjunath, G. Hewer, “Peer group image enhancement”, *IEEE Trans. on Image Processing*, vol. 10 (2), 326-334, 2001.

[62] Ageenko E., Fränti P., "Context-based filtering of document images", *Pattern Recognition Letters*, 21 (6-7), 483-491, Elsevier Science, 2000.

[63] K. N. Plataniotis, D. Androutsos, and A. N. Venetsanopoulos, “Color image filters: The vector directional approach,” *Optical Engineering*, vol. 36, no. 9, pp. 2375–2383, 1997.

[64] R. Lukac, B. Smolka, K. Martin, K.N. Plataniotis, and A.N. Venetsanopoulos, “Vector filtering for color imaging”, *IEEE Signal Processing Magazine*, vol. 22 (1)74-86, January 2005.

[65] P. Kopylov, P. Fränti, “Filtering of color map images by context tree modeling”, *Proc. IEEE Int. Conf. on Image Processing (ICIP'04)*, Singapore, vol. 1, 267-270, October 2004.

[66] M. J.Weinberger, J. Rissanen, and R. B. Arps, “Applications of universal context modeling to lossless compression of gray-scale images,” *IEEE Trans. on Image Processing*, pp. 575–586, Apr. 1996.

[67] X. Wu, “Lossless Compression of Continuous-tone Images via Context Selection, Quantization and Modeling”, *IEEE Trans. on Image Processing*, no. 6, pp. 656-664, 1997.

[68] Ageenko E.I., Fränti P., “Compression of large binary images digital spatial libraries”, *Computers and Graphics*, vol. 24(1), pp. 91-98, 2000.

[69] S. Forchhammer and O. Jensen, “Content layer progressive coding of digital maps,” *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1349–1356, Dec. 2002.

[70] P. Kopylov, P. Fränti. “Compression of map images by multi-layer context tree modeling”, *IEEE Trans. on Image Processing*, vol. 14, pp. 1-11, Jan. 2005.

[71] P. Heckbert, “Color image quantization for frame buffer display”, *Comput. Graph.*, vol. 16, pp. 297-307, 1982.

[72] A. Akimov, A. Kolesnikov and P. Fränti, “Lossless compression of color map images by context tree modeling”, *IEEE Trans. on Image Processing*, vol. 16 (1), 114-120, 2007.

[73] B. Martins and S. Forchhammer, “Tree coding of bilevel images,” IEEE Trans. Image Process., vol. 7, no. 4, pp. 517-528, Apr. 1998.

[74] X. Chen, S. Kwong, and J.-F. Feng, “A new compression scheme for color-quantized images”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 904-908, Oct. 2002.

[75] A.J. Pinho and A.J.R. Neves, “A context adaptation model for the compression of images with a reduced number of colors”, *Proc. of the IEEE Int. Conf. on Image Processing (ICIP-2005)*, September 2005, Genova, Italy.

[76] P. Fränti, T. Kaukoranta, D.-F. Shen and K.-S. Chang, “Fast and memory efficient implementation of the exact PNN”, *IEEE Trans. Image Process.*, vol. 9, no. 4 pp. 773-777, May 2000.

Publication P1

Podlasov A., Ageenko E., Fränti P., Morphological reconstruction of semantic layers in map images, *Journal of Electronic Imaging*, 15(1), 013016, January-March 2006.

Morphological reconstruction of semantic layers in map images

Alexey Podlasov
Eugene Ageenko
Pasi Fränti
University of Joensuu
Department of Computer Science
Box 111
FIN-80101 Joensuu
Finland

Abstract. *Map images are composed of semantic layers depicted in arbitrary color. Color separation is often needed to divide the image into layers for storage and processing. Separation can result in severe artifacts because of the overlapping of the layers. In this work, we introduce a technique to restore the original semantic layers after the color separation. The proposed restoration technique improves compression performance of the reconstructed layers in comparison to the corrupted ones when compressed by lossless algorithms such as International Communication Unit (ITU) Group 4 (TIFF G4), Portable Network Graphics (PNG), Joint Bi-level Image experts Group (JBIG), and context tree method. The resulting technique also provides good visual quality of the reconstructed image layers, and can therefore be applied for selective layer removal/extraction in other map processing applications, e.g., area measurement.* © 2006 SPIE and IS&T. [DOI: 10.1117/1.2178188]

1 Introduction

Currently, there exist various services delivering map imagery content to the user. For example, real-time map imaging applications provide users with a view of a geographical map for the area surrounding the user's location. The location can be obtained using a global positioning service (GPS), mobile positioning service (MPS), or other analog services. It could also be weather, traffic, pollution, or any other kind of map. The imagery data are usually obtained from a digital spatial library,[1] and transmitted via the network to the user's device such as a pocket computer (PDA), mobile phone, or desk-top terminal.

A typical map image consists of a set of semantic layers, each containing data with distinct semantic content, each depicted with its own color, e.g., black roads, brown elevation lines, blue water areas, yellow fields, etc. Regardless of the semantic nature, typical maps need only a few color tones to represent the layers, but high spatial resolution for representing details. Let us call these images multilayer map images.

We consider topographic images from the National Land Survey of Finland (NLS) topographic database, in particular the basic map series 1:20,000.[2] The images consist of a set of semantic layers, each containing data with distinct semantic content, such as roads, elevation lines, infrastructures, state boundaries, and water areas. The layers are combined and displayed to the user as a generated color image, in which the data of each type are depicted using their own color. These images consist of the following semantic layers: basic (roads, contours, labels, and other topographic data), elevation lines (thin lines representing elevations levels), waters (solid regions and polylines representing water areas and ways), and fields (solid polygonal regions) (see Fig. 1).

The original map data are usually stored in vector format on a server-side database. Each semantic layer is stored separately. As the user's request arrives, the server prepares part of the data and transmits it to the user in raster format, since raster images are easier to handle on a client-side device. Using a vector format requires special software developed for vector map image processing, then the processing of raster images is a standard feature of almost any mobile terminal. Raster format is also often used for digital publishing on the web or CDs.

When producing a raster map image, map layers of different semantic nature are combined together overlapping each other in a predefined order. This image is well suited for user observation, but it is less appropriate for further processing, since the layer structure has been corrupted when the raster of the map image was produced. For ex-

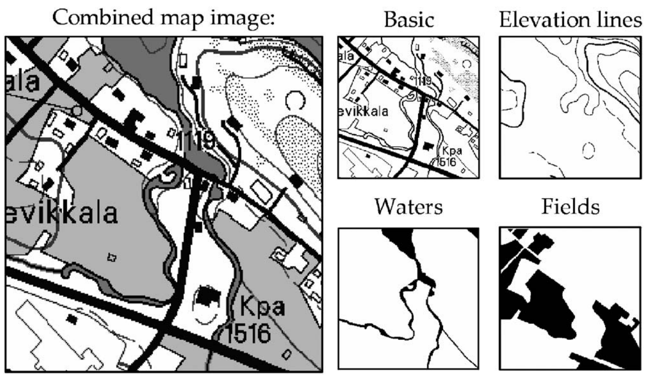


Fig. 1 Illustration of a multilayer map image from the NLS Topographic database.

Paper 05030R received Feb. 24, 2005; revised manuscript received Jun. 27, 2005; accepted for publication Jul. 6, 2005; published online Mar. 9, 2006.
1017-9909/2006/15(1)/013016/10/$22.00 © 2006 SPIE and IS&T.

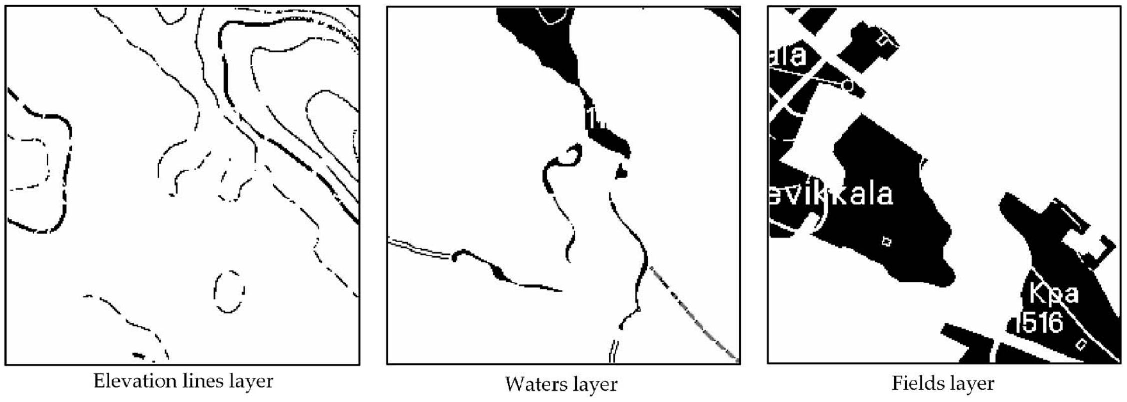


Fig. 2 Corrupted layers due to the color separation.

ample, when one needs to calculate the area of the fields, or the length of coastline, the layer must be extracted from the color raster image through color separation. During this process, the map image is divided into binary layers, each representing one color in the original image. The main problem of this approach is that the separation introduces severe artifacts in places where the information of different layers overlap each other (see Fig. 2). The holes on the fields caused by overlapping letters are a typical example of these artifacts. The presence of the artifacts can make the color separated layer useless for many map processing tasks.

Moreover, the problem also affects the compressibility of the images. Though the raster image could be compressed with any existing lossless compression algorithm, it has been shown that the best compression results can be achieved if the image is decomposed into binary semantic layers, which are consequently compressed by algorithms designed to handle binary data (e.g., JBIG).[3] The artifacts of the color separation, however, affect the statistical properties and consistency of the layers, and result in degraded compression performance in comparison to the original ones. This is apparent especially in applications requiring the use of mobile hardware such as mobile phones or pocket computers. For example, a single map sheet of 10×10 km^2 is represented by a single map image of 5000×5000 pixels. Larger image size also takes a longer time to transmit. For example, 10-sec transmission via a GPRS channel with bandwidth 45 kb/sec results in at most 54 kB of image data. This corresponds to only about 500×500 pixels for a four-layer map image.

The problems mentioned led us to develop an algorithm for the reconstruction of the corrupted layers of map images. The proposed algorithm approximates the original layer structure existing before the color combination by repairing the corrupted layers as close to the original ones as possible. A natural restriction for the reconstruction technique is that the color combination of the reconstructed layers should be equal to the originally received raster map image.

The goal of image restoration is to reconstruct the original image before degradation.[4] The reconstruction involves a criterion for measuring the quality of the desired result. For our problem, we consider two criteria: image quality and image compressibility. The first criterion measures how close the reconstructed layer is to the original semantic layer. This is important for applications where visual quality is essential, or the reconstructed layer is used for processing, e.g., measuring the area of the fields. The second criterion aims to modify the corrupted layer so that its compressibility will be improved as much as possible without causing any changes to the corresponding output color image.

The artifacts appearing on the layers could be treated as noise. If we guarantee that the color map remains untouched, noise removal could be considered as a tool for improving image quality for achieving better compressibility. There are many image enhancement methods in the literature,[4–7] and various reconstruction techniques have been considered.[8–11] Statistical modeling,[12] and specific data modeling and representation techniques[13–16] have also been considered. However, noise filtering and typical image enhancement algorithms are not suitable for solving our

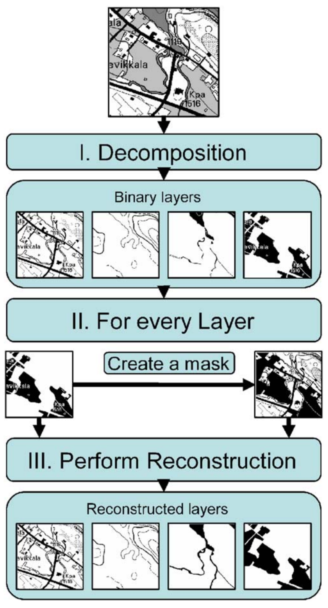


Fig. 3 Outline of the reconstruction algorithm.

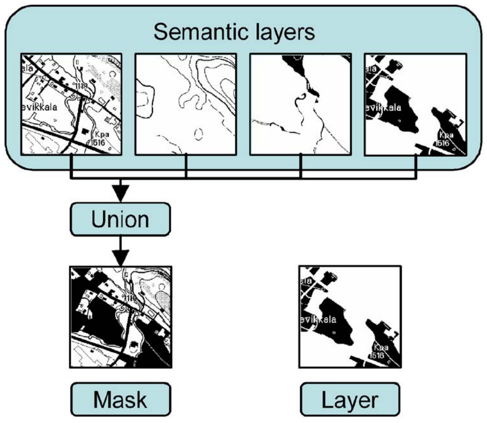


Fig. 4 Scheme for the mask creation.

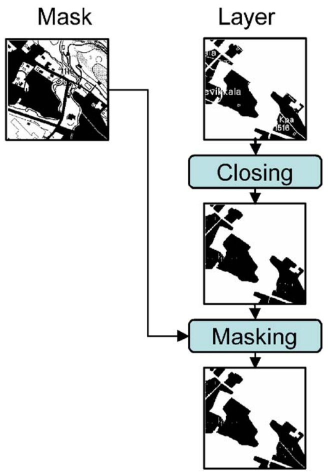


Fig. 6 The block diagram of CC algorithm.

problem, because, due to their local nature, they are not able to recognize larger structures and dependencies between layers.

Therefore, we introduce a new morphological filter for layer reconstruction. We chose mathematical morphology to be the tool, mostly due to the simplicity of implementation. Morphological operators do not require sufficient computational and memory resources to be applied, which is apparent for use on mobile terminals. The benefits of the proposed filter are its capability to reconstruct semantic information in a multilayer map image, and that the original color image can always be reconstructed exactly without any loss in the quality. The effect of the filter is therefore limited only to the binary layers. The method is applicable for extraction or removal of individual layers, and for lossless compression of the map images. The method is fast and simple to implement.

The rest of the work is organized as follows. Mathematical morphology is briefly introduced in Sec. 2. In Sec. 3, we introduce two variants of the new filtering method for layer extraction, and then apply it for layer removal in Sec. 4. Empirical results are reported in Sec. 5, and conclusions drawn in Sec. 6.

2 Mathematical Morphology

Mathematical morphology refers to a branch of nonlinear image processing and analysis originally introduced by Matheron[17] and Serra,[18] and currently continuing its development.[19] This chapter gives the basic morphological definitions. In discrete binary morphology, an image space E is usually defined as $E=\mathbb{Z}^2$ (the space of all possible image pixel locations), and a binary image X as a set $X \subseteq E$. For a given set A, the reflection (or the symmetric) of A with respect to the origin, denoted as $\tilde{A}$ or $-A$, is defined by $\tilde{A}=\{-a \mid a \in A\}$. The power set of E, or in other words, the set of all subsets of E, is denoted as $\mathcal{P}(E)$. One of the main fundamentals of mathematical morphology is to analyze the geometrical and topological structure of an image X by "probing" the image with another small set $A \subseteq E$ called a structuring element. The choice of the appropriate structuring element depends on the particular application.

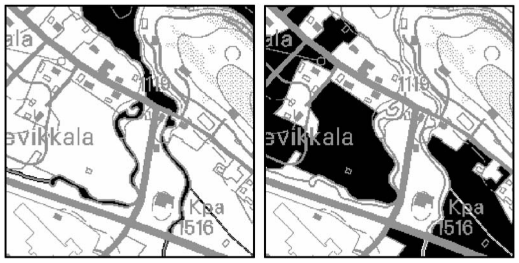


Fig. 5 Water and field layers with their masks. Object pixels from the layer are plotted by black, and the mask pixels by gray to differentiate which pixel belongs to the layer, and which pixel to the mask.

2.1 *Fundamental Morphological Operators*

The dilation of X by A, denoted as $\delta_A(X)$, is defined as the operator on $\mathcal{P}(E)$ given by:

$$\delta_A(x) = \bigcup_{a \in A} X_a = \{h \in E \mid \widetilde{A_h} \cap X \neq \emptyset\}.$$

The erosion of X by A, denoted by $\varepsilon_A(X)$, is

$$\varepsilon_A(X) = \bigcap_{a \in A} X_{-a} = \{h \in E \mid A_h \subseteq X\}.$$

The cardinality of set A, or the number of elements in A, is denoted by card(A). Let us also define the translation invariant operator $\rho_{A,n}$, called a rank operator, as follows:

$$\rho_{A,n}(X) = \{h \in E \mid \mathrm{card}(X \cap A_h) \geqslant n\}.$$

The operator $\rho_{A,n}(X)$ sets current pixels to be the foreground if the amount of foreground pixels in a neighborhood defined by the structuring element is greater than n. Otherwise, the pixel is defined as a background pixel. Since the rank operator performs similar to erosion or dilation depending on the value of the rank parameter, it is possible to treat the rank as a soft counterpart of classical erosion and dilation operators. In particular:

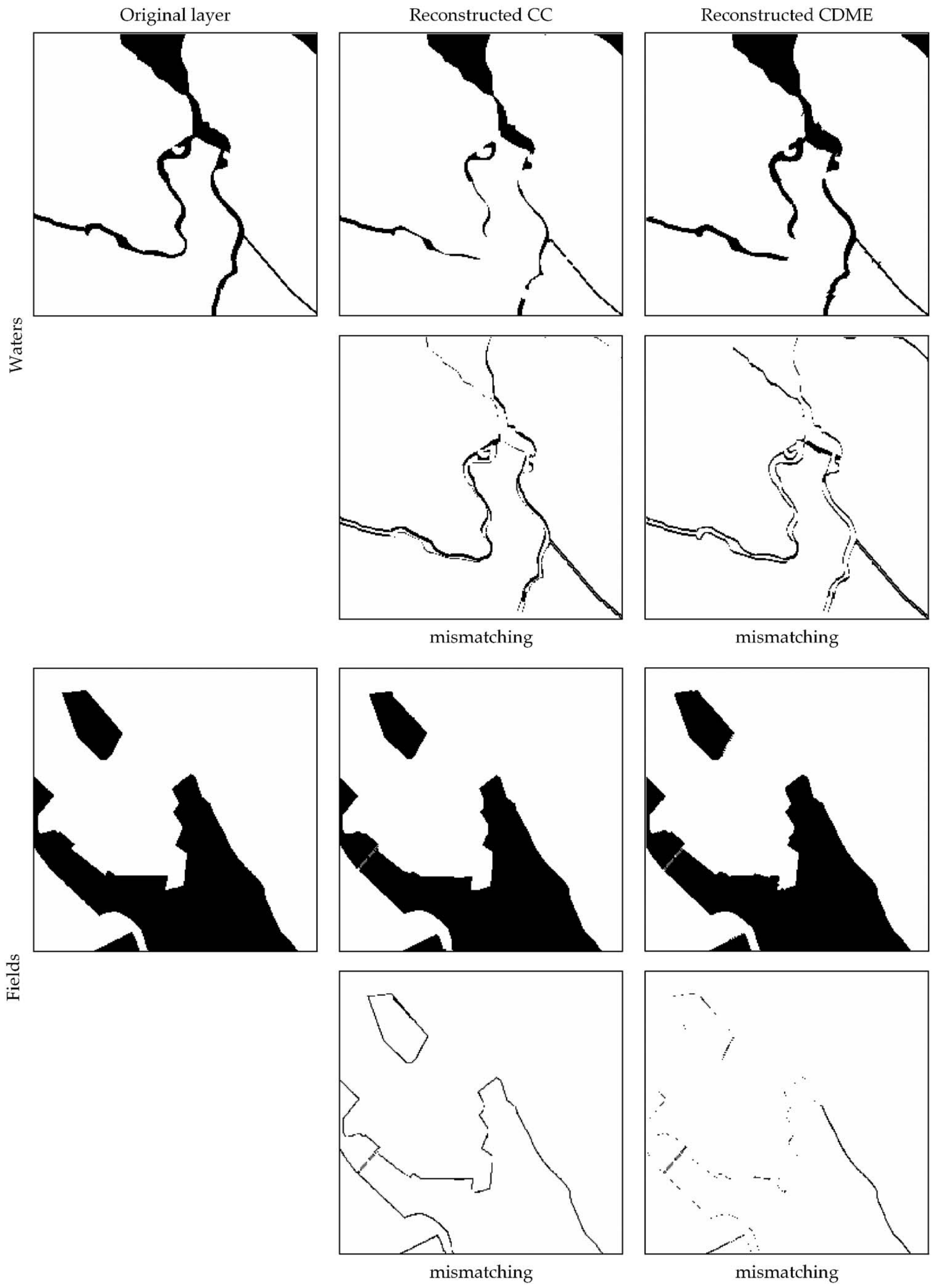


Fig. 7 Sample images for the water and field layers: original and reconstructed with CC and CDME algorithms as well as mismatching of reconstructed layer and the original.

$\delta_{\tilde{A}}(X)=\rho_{A,1}(X)$ and $\varepsilon_A(X)=\rho_{A\mathrm{card}(A)}(X)$.

The operator $\alpha_A(X)=\delta_A[\varepsilon_A(X)]$ is called the (structural) opening by A. Dually, the operator $\beta_A(X)=\varepsilon_A[\delta_A(X)]$, is called the (structural) closing by A.

2.2 *Conditional Operators*

If an image is, say, dilated by a structuring element containing the origin, it is expanded, and the manner of the expansion depends only on the shape of the structuring element. If the dilation is successively repeated, the original image grows without bounds. Sometimes it is important to restrict the growth. This can be accomplished by using conditional operators. A common form of conditioning restricts the translations to a superset of the input image: if image A is a subset of image T, then for any operator $\psi(A)$, the operator $\psi(A|T)$ is called $\psi(A)$ conditioned relative to T and is defined as:

$$\psi(A|T)=\psi(A)\cap T.$$

The image T is usually referred to as a mask image.

```
L ← Layer
M ← Mask
A ← Structuring element of dilation
B ← Structuring element of erosion
  REPEAT
      $L := \delta_A(L \mid M)$ ;
      $M := \varepsilon_B(M)$ ;
      $M := L \cup M$ ;
  UNTIL Iteration criterion met
```

Fig. 8 Outline of the CDME algorithm.

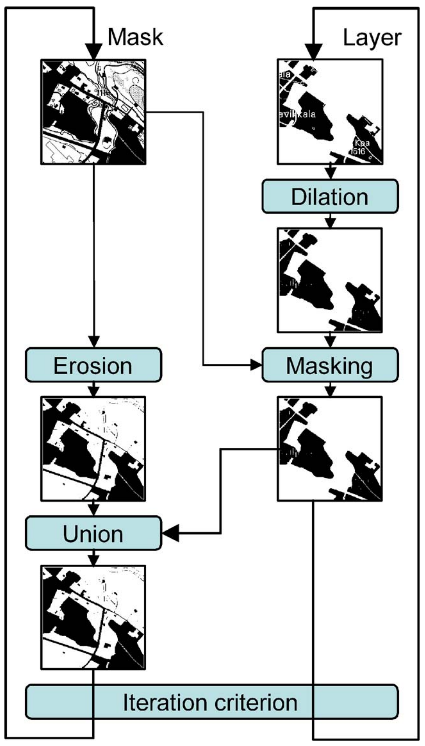


Fig. 9 Block diagram of the CDME algorithm.

3 Layer Reconstruction Technique

We consider two approaches. The first approach aims at maximal compression improvement for the reconstructed layers, and the second at more accurate restoration of the original semantic layers.[20] In the first approach, we simply try to minimize the storage size of the layers, which is essential for map storage systems. In the second approach, we try to produce layers that are as close to the original semantic layers as possible. The resulting layers can then be used for additional map processing and analysis. Following the underlying principles behind the previous two approaches, we have designed two reconstruction algorithms referred to further as conditional closing (CC) and conditional dilation with mask erosion (CDME).

Both algorithms have the same structure, consisting of three main steps as outlined in Fig. 3. At the first step, the color map (scanned or obtained from the third party source) is decomposed into a set of binary layers by a color separation process. This is done so that each layer represents one color in the original image.[3] Then, according to the predefined layer order, a conditioning mask is created for every layer for restricting the reconstruction of the layers to be equal to the original color image. Finally, the actual reconstruction is performed for every layer with respect to its conditioning mask.

3.1 *Conditioning Mask*

Further, we denote a layer image as L; when we talk about some particular layer, we denote it as L_k, $1 \leq k \leq N$, where N is the total number of layers in a map image. The requirement that the composition of reconstructed layers should be identical to the initial color map can be met by conditioning the operator $\psi(L_k)$ on the mask M_k, which defines the region where changes of the layer content are allowed. The requirement of keeping the reconstructed color map identical to the original one leads to the fact that the restoration operator must not remove pixels that are already present in the corrupted layer. It can only add pixels to a layer, so that the condition

$$L_k \subseteq \psi(L_k | M_k),$$

is met. The conditioning mask defines the set of pixels that are allowed to change value in the restoration, so that the combination of the restored layers would be kept untouched. Since we assume that the order of layer overlapping is predefined, the mask for every layer is computed as the union of all upper-laying layers,

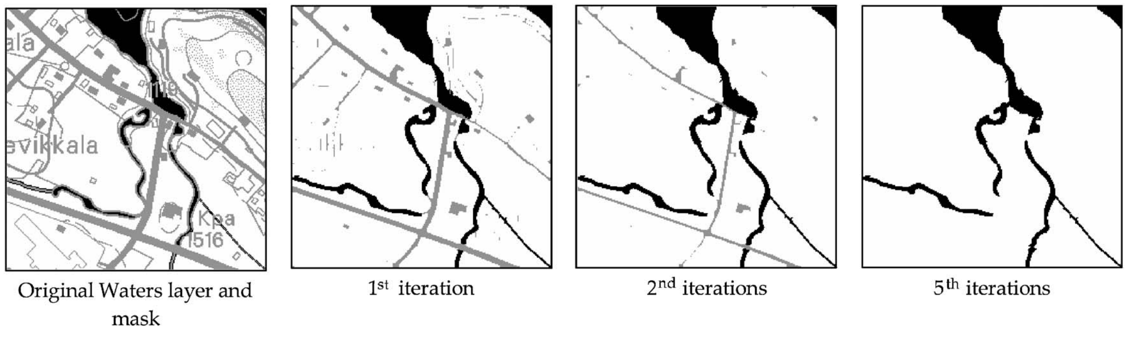


Fig. 10 Step-by-step illustration of the dilation with mask erosion. Object pixels from the layer are plotted by black, and the mask pixels by gray to differentiate which pixel belongs to the layer, and which pixel to the mask.

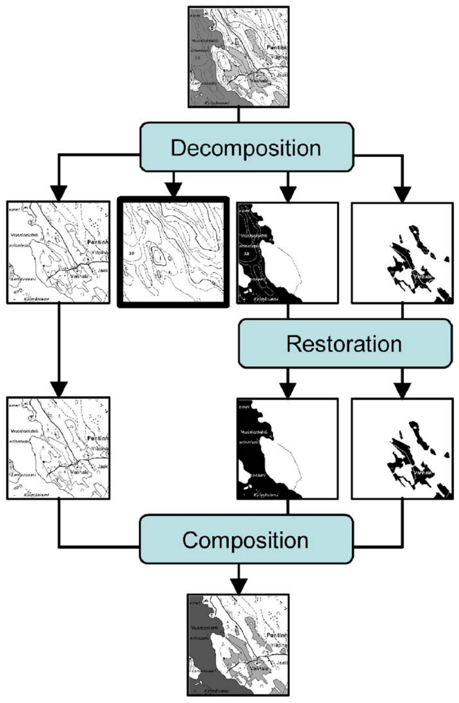


Fig. 11 Block diagram of the layer removal algorithm. Elevation lines layer to be removed is outlined with a black frame.

$$M_k := \bigcup_{j=1}^{k} L_k,$$

(see Figs. 4 and 5).

3.2 *Layer Reconstruction*

3.2.1 *Conditional closing*

Having the compression objective in mind, let us consider using a simple and effective conditional closing (CC) operator $L := \beta_A(L|M)$ to perform reconstruction. The algorithm is outlined in Fig. 6. The quality of the reconstruction in terms of compressibility strongly depends on the applied structuring element. In our experiments, we have tried out several alternatives and found that square block provides the best compression improvement. The size of the block depends on the size of the artifacts, and, for our test set 7×7 has been selected. Once applied successfully, the closing fills artifacts inside the objects, leaving the borders almost untouched (see Fig. 7). The main characteristics of this approach are its simplicity of implementation, and its positive effect on compression.

3.2.2 *Conditional dilation with mask erosion*

Although efficient in terms of compression, the CC algorithm is not as effective in approximating (restoring) the original layers. The method expands the lower layers too conservatively, whereas the color layer is typically a reduced version of the original semantic layer, due to overlapping. Therefore, we propose another algorithm, which we call conditional dilation with mask erosion (CDME), using more aggressive expansion based on dilation, and thus, aiming at a more accurate approximation of the original semantic layers.

The idea in general is to spread objects step by step and shrink the mask, too. The process is iterative: first, the spreading is performed by the dilation operator $\delta_A(X)$, and then the mask shrinking is performed by the erosion operator $\varepsilon_A(X)$. The pseudocode of the algorithm is shown in Fig. 8, and outlined in Fig. 9. The stepwise process of the iterations is illustrated in Fig. 10.

The iterative process is controlled by a stopping criterion. We have investigated two approaches: iterate until stability and iterate fixed amount of times. The first approach assumes that the iterative process will continue until the layer (and mask) converges. The convergence is guaranteed, because the erosion sequentially decreases the mask (see Fig. 10). We can therefore perform the iterations until the mask equals the layer itself.

Examination if the mask and layer are equal could be a time-consuming operation, especially if the image size is big. To avoid this, we consider the second approach by assuming that most of the artifacts are of limited size, which can be determined within the first few iterations. We therefore restrict the amount of iterations by a fixed number. For example, if we suppose that the size of an artifact is four pixels, on average, three dilations with a 3×3 block are enough for the restoration.

As with the conditional closing, an important question is the choice of an appropriate structuring element. There are

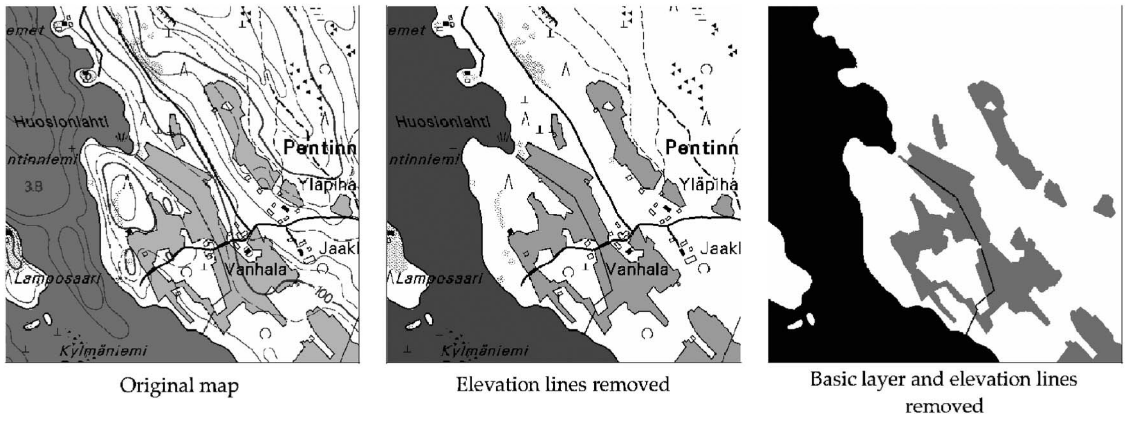


Fig. 12 Example of the layer removal.

Table 1 Restoration of elevation layer.

Compression algorithm	Semantic layers	Corrupted layers	CDME		CC	
			Size	Imp.	Size	Imp.
PNG	825 510	808 126	811 958	−0.47%	805 774	4.30%
TIFF	460 811	481 338	464 934	3.41%	461 482	0.29%
JBIG	236 210	269 423	259 139	3.82%	253 606	6.24%
AKF2	223 555	261 386	250 820	4.04%	243 870	6.70%

two structuring elements used in the algorithm: in the object dilation and in the mask erosion. By varying the first element, we can control how fast the object expands over the mask, while varying the second element controls how fast the mask shrinks. An essential matter is the relation between the speeds of the dilation and erosion. Let A be the structuring element of dilation and B be the structuring element of erosion. We use two structuring elements: square is the 3×3 block $\{[-1,0,1]\times[-1,0,1]\}$ and cross $\{(0,-1),(1,0),(0,0),(-1,0),(0,1)\}$. We have tested three cases: objects dilating faster than mask eroding (A =square, B=cross), objects dilating slower than mask eroding: (A=cross, B=square), and the case of equal speed (A =square, B=square or A=cross, B=cross).

The speed of dilation and shrinking could also be controlled if dilation and erosion operators used in a restoration technique are replaced with a rank operator as their “soft” counterpart.[17] The rank operator is equal to the dilation operator when the rank parameter n is set to 1, and to the erosion operator when the rank parameter is equal to the cardinal number of the structuring element $[n=\text{card}(A)]$. Rank operators with rank parameters lying between these two values behave approximately like dilation or erosion operators. In other words, a rank parameter could be used to regulate the “strength” of erosion or dilation, or how fast objects shrink or expand. The case when a rank operator equals $\text{card}(A)/2$ is called a median operator.

The performance of the restoration strongly depends on the morphological structure of the layer under reconstruction. To choose the variant of the algorithm for evaluation, we examined different structuring elements and parameter values. The modification gaining the best performance is described and evaluated in Sec. 5.

4 Layer Extraction/Removal Technique

The task of layer restoration arises if there is a need for layer extraction or removal. Layer extraction is needed when one wishes to perform some specific processing over the layer, e.g., to calculate the area of the fields. Naturally, a corrupted layer could not be accepted as accurate input. A similar task is layer removal when less important layers are not needed by the map user, e.g., user driving a car does not need elevation lines, as such layers can limit map readability. To remove a layer, the restoration technique of Sec. 3 is first applied to all layers, and the color image is composed of the restored layers except for the one to be removed (see Fig. 11).

The most important feature here is the quality of the restoration, i.e., how closely the corrupted layer approximates the original layer. Moreover, in user interactive applications, the visual appearance of the reconstructed layer becomes essential. Figure 12 illustrates the effect of the removal of the basic and elevation layers.

5 Evaluation

The restoration techniques have been evaluated on a set of topographic color-palette map images. These images were decomposed into binary layers with distinctive semantic meaning identified by the pixel color on the map. The restoration algorithms have been applied for reconstruction of these semantic layers after the map decomposition process. Both the combined color map images and the binary semantic layers composing these color map images were originally available for testing. This allowed us to compare the restored images with their original undistorted counterparts.

Table 2 Restoration of water layer.

Compression algorithm	Semantic layers	Corrupted layers	CDME		CC	
			Size	Imp.	Size	Imp.
PNG	381 608	425 862	384 766	9.65%	378 484	11.13%
TIFF	167 361	357 164	168 630	52.79%	171 673	51.93%
JBIG	81 334	137 258	93 230	32.08%	95 520	30.41%
AKF2	49 230	73 107	57 370	21.53%	54 695	25.18%

Table 3 Restoration of field layer.

Compression algorithm	Semantic layers	Corrupted layers	CDME		CC	
			Size	Imp.	Size	Imp.
PNG	309 712	456 710	320 821	29.75%	313 486	31.36%
TIFF	99 622	196 456	105 388	46.36%	119 306	39.27%
JBIG	49 409	113 977	50 936	55.31%	56 950	50.03%
AKF2	5917.5	16110.5	7056.25	56.20%	6 212	61.44%

The performance of the proposed restoration techniques was evaluated according to two measures: the improvement of compression performance and the quality of the reconstruction. The first measure is relevant when dealing with map image storage, and concerns only the improvement in compressibility, regardless of how exact the reconstruction is. The second measure is relevant to applications where the reconstruction is expected to approximate the original as close as possible.

The test set consists of five randomly chosen images from the NLS Basic Map Series 1:2000, corresponding to the map sheets 431306, 201401, 263112, and 431204. Each image is of 5000×5000 pixels and consists of four binary layers. The layer names are the following:

- basic—topographic image, supplemented with communications networks, buildings, protected sites, benchmarks, and administrative boundaries
- elevation—elevation lines
- water—lakes, rivers, swamps, and water streams
- fields—agricultural areas.

5.1 *Compression Performance*

The evaluation examines the compression performance of the map images constructed on reconstructed layers in comparison to semantic layers (not affected by the layer separation process) and corrupted layers (result of the layer separation). The proposed algorithms were evaluated using four compression techniques: LZ (PNG), ITU Group 4 (TIFF), JBIG, and AKF2[21] (context-based compression with optimized context size and shape). For each of these compression methods, we have measured the compressed data size for the original semantic layers, for the corrupted binary layers after decomposition, and for the reconstructed layers with the two reconstruction algorithms (CC and CDME). The structuring elements in CC are 7×7 blocks; the CDME uses soft erosion and dilation with rank parameters 2 and 8, respectively.

5.1.1 *Results for stand-alone layers*

Tables 1–3 give the average compressed sizes of the restored elevation lines, water, and field layers, respectively. The results are the average compressed file size (size) improvement in compression ratio (imp) for semantic, corrupted, and reconstructed layers.

Evaluating the performance for the elevation-line layer we conclude that neither reconstruction technique is effective in improving the compression performance. The structure of the layer does not allow for remarkable increase, and only about 5% improvement was achieved. On the other hand, significant compression improvement is gained for water and field layers (20 to 50%, depending on the compression technique), since these layers contain a lot of closed solid regions. The holes left by letters and other artifacts were successfully filled by both algorithms. The tradeoff between the algorithms is in the computational complexity (the CC is simpler) and the quality of the restoration (the CDME has significantly better visual appeal, as shown further).

5.1.2 *Overall results*

In a real application, however, one cannot consider compression improvement for independent layers, but we must evaluate the compression performance altogether for all

Table 4 The average compression performance of the topographic map images based on semantic layers, color layer separation, and reconstructed color layer separation with CDME and CC restoration algorithms.

Compresson algorithm	Semantic layers		Corrupted layers		Reconstructed with CDME			Reconstructed with CC		
	Size	bpp	Size	bpp	Size	bpp	imp.	Size	bpp	imp.
PNG	2 085 871	0.66	2 149 490	0.68	2 078 254	0.66	3.31%	2 063 955	0.66	3.98%
TIFF	1 473 824	0.47	1 708 362	0.54	1 480 657	0.47	13.33%	1 483 727	0.47	13.15%
JBIG	684 978	0.21	790 257	0.25	720 185	0.23	8.87%	718 446	0.22	9.09%
AKF2	624 117	0.19	696 017	0.22	660 661	0.21	5.08%	650 191	0.20	6.58%

layers forming a map image. Table 4 illustrates the average compression performance of the test images based on semantic layers, color layer separation, and reconstructed color layer separation with CDME and CC restoration algorithms. The results are average compressed file sizes (size) computed as the sum of all compressed layers, bit rate (bpp), and improvement ratio (imp). We conclude that the proposed restoration technique achieves almost the same degree of compression of the map images as if the original semantic layer decomposition was available. Relatively low compression improvement is caused by the dominant size of the nonrestorable top-level layer basic, or the hardly restorable elevation-line layers.

5.2 *Restoration Quality*

This section evaluates the restoration performance of the proposed technique. By restoration quality, we mean how close the original and reconstructed layers are, with respect to some distance measure. In this work, we use normalized mean absolute error (NMAE), i.e., Hamming distance, which measures the average number of different pixel values in the original semantic layers, and in the reconstructed layers.

$$\mathrm{NMAE}(X,Y) = \frac{\sum_{j=1}^{\mathbb{H}} \sum_{i=1}^{\mathbb{W}} |x_{i,j} - y_{i,j}|}{\mathbb{H} \cdot \mathbb{W}},$$

where H and W are image dimensions.

The compression evaluation showed that the elevation layer is hardly restorable. Therefore, we do not consider it in the quality evaluation. We measured the NMAE difference between the original layers of water and field, and their reconstructed counterpart, with both CC and CDME. The same was done for the corrupted layers with respect to the original ones. These results show that the reconstructed layers are closer to the original layers than the corrupted ones. In Fig. 13, we present the total NMAE differences within the test set for each layer separately. The CDME algorithm showed better reconstruction comparing CC both for water and fields.

We evaluated the performance of the restoration by applying it to the task of area measurement. We compared the area measured over the original layer with one measured over the reconstructed and corrupted layer. The results are presented for water and field layers separately on average within the whole test set (see Table 5). Since CDME approximates layers better, its area measurements are also much closer to the original than the CC results. CDME reconstruction reduces the error of the area measurement from 15 to 20% to just about 1%.

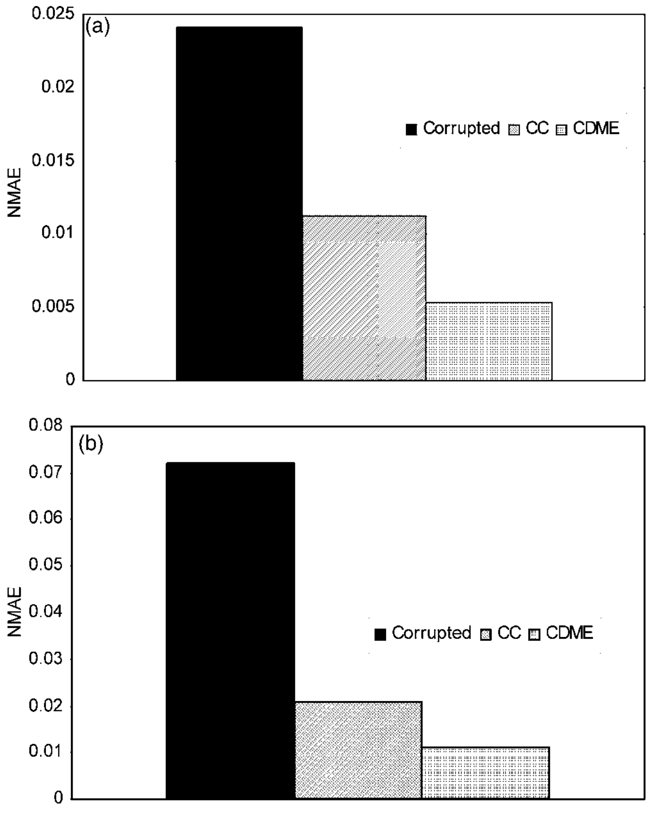


Fig. 13 The average NMAE difference with the original measured for reconstruction with CC and CDME field (a) and water (b) layers compared to corrupted ones.

6 Conclusion

We propose a technique for the restoration of binary semantic layers of map images from the corruption caused by the decomposition of the image using a color separation process. The performance of the proposed method is evaluated by improvement in compression performance and in quality of the restoration. It allows us to obtain up to 30 to 50% compression improvement for stand-alone layers and improves the total compression rate (calculated for the sum of the layers) up to 5 to 10%, depending on the compression method. Low total improvement rates are caused by the presence of non- or hardly restorable layers, such as basics and elevation.

Quality evaluation shows that restoration efficiently approximates corrupted layers to the original. The properly tuned algorithm reduces error in such applications as area measurement from 15 to 20% to about 1%. The color map

Table 5 The area (in pixels) measured over the original, corrupted, and reconstructed with CC and CDME elevation, water, and field layers.

Compression algorithm	Semantic layers	Corrupted layers		CDME		CC	
	Area	Area	Error, %	Area	Error, %	Area	Error, %
Water	10 480 893	8 678 605	17.20%	10 389 501	0.87%	9 996 454	4.62%
Field	4 267 983	3 663 960	14.15%	4 262 378	0.13%	4 057 253	4.94%

image resulting from the combination of the reconstructed layers remains identical to the original image, because all changes to the layer content are performed only within those areas that will be overlapped during composition. The method therefore affects only the separated layers, not the original color image.

References

1. "Introduction to digital libraries," in *Commun. ACM* **38**(4), 22–28, E. A. Fox, R. Akscyn, R. Furuta, and J. Leggett, Guest Eds. (1995).
2. National Land Survey of Finland, NLS, Opastinsilta 12 C, P.O. Box 84, 00521 Helsinki, Finland, see http://www.nls.fi/index_e.html.
3. P. Fränti, E. Ageenko, P. Kopyloy, S. Gröhn, and F. Berger, "Compression of map images for real-time applications," *Image Vis. Comput.* **22**(13), 1105–1115 (Nov. 2004).
4. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed., Addison-Wesley, New York (2002).
5. I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters: Principles and Applications*, Kluwer, Boston, MA (1990).
6. E. R. Dougherty, "Optimal mean-square n-observation digital morphological filters. Part 1: Optimal binary filters," *Comput. Vis. Graph. Image Process.* **55**, 36–54 (1992).
7. *Nonlinear Filters for Image Processing*, E. R. Dougherty and J. Astola, Eds., SPIE Optical Engineering Press, Bellingham, WA (1997).
8. D. Ting and B. Prasada, "Digital processing techniques for encoding of graphics," *Proc. IEEE* **68**(7), 757–769 (1980).
9. F. M. Wah, "A binary image preprocessor for document quality improvement and data reduction," *Proc. Intl. Conf. Acoustic, Speech, Signal Process. ICASSP'86*, pp. 2459–2462 (1986).
10. Q. Zhang and J. M. Danskin, "Bitmap reconstruction for document image compression," *Proc. SPIE* **2916**, 188–199 (1996).
11. L. Koskinen and J. Astola, "Soft morphological filters: A robust morphological filtering method," *J. Electron. Imaging* **3**, 60–70 (1994).
12. E. Ageenko and P. Fränti, "Context-based filtering of document images," *Pattern Recogn. Lett.* **21**(6,7), 483–491 (2000).
13. Z. Ping, C. Lihui, and K. C. Alex, "Text document filters using morphological and geometrical features of characters," *Proc. Intl. Conf. Signal Process. ICSP'00*, pp. 472–475 (2000).
14. T. R. Randolph and M. J. T. Smith, "Enhancement of fax documents using a binary angular representation," *Proc. Intl. Symp. Intell. Multimedia, Video Signal Process.*, pp. 125–128 (2001).
15. Q. Zheng and T. Kanungo, "Morphological degradation models and their use in document image restoration," Univ. Maryland Technical Report, LAMP-TR-065 CAR-TR-962 N660010028910/IIS9987944 (2001).
16. P. Fränti, E. Ageenko, S. Kukkonen, and H. Kälviäinen, "Using Hough transform for context-based image compression in hybrid raster/vector applications," *J. Electron. Imaging* **11**(2), 236–245 (2002).
17. G. Matheron, *Random Sets and Integral Geometry*, Wiley and Sons, New York (1975).
18. J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London (1982).
19. H. J. A. M. Heijmans, *Morphological Image Operators*, Academic Press, Boston (1994).
20. E. Ageenko and A. Podlasov, "On the restoration of semantic data in raster map images," *IADIS Informatics Int. Conf. (I2005), IADIS Virtual Multi Conf. Computer Sci. Info. Sys. (MCCSIS 2005)*, pp. 334–342.
21. E. Ageenko, P. Kopylov, and P. Fränti, "On the size and shape of multi-level context templates for compression of map images," *Proc. Intl. Conf. Image Process. ICIP'01* 3, 458–461 (2001).

Alexey Podlasov received his MSc degree in applied mathematics from Saint-Petersburg State University, Russia, in 2002, and his MSc degree in computer science from the University of Joensuu, Finland, in 2004. Currently, he is a doctoral student in computer science at the University of Joensuu. His research topics include processing and compression of map images.

Eugene Ageenko received his MSc degree in applied mathematics and software engineering in 1995 from the Moscow State University, Russia, and his PhD degree in computer science in 2000 from the University of Joensuu, Finland. Currently, he is a senior assistant for the Computer Science Department of the University of Joensuu, Finland. His research interests include line-art image processing and compression, mobile imaging, and location-based systems. He is a member of IEEE and ISO/IEC JTC 1/SC 29/WG1 (JBIG).

Pasi Fränti received his MSc and PhD degrees in computer science in 1991 and 1994, respectively, from the University of Turku, Finland. From 1996 to 1999 he was a postdoctoral researcher of the Academy of Finland. He has been a professor at the University of Joensuu, Finland since 2000. His primary research interests are in image compression, clustering, and speech technology.

Publication P2

Podlasov A., Fränti P., Lossless image compression via bit-plane separation and multi-layer context tree modeling, *Journal of Electronic Imaging*, 15(4), 043009, October–December 2006.

Lossless image compression via bit-plane separation and multilayer context tree modeling

Alexey Podlasov
Pasi Fränti
University of Joensuu
Speech and Image Processing Unit
Department of Computer Science
P.O. Box 111
80101 Joensuu, Finland
E-mail: apodla@cs.joensuu.fi

Abstract. *Color separation and highly optimized context tree modeling for binary layers have provided the best compression results for color map images that consist of highly complex spatial structures but only a relatively few number of colors. We explore whether this kind of approach works on photographic and palette images as well. The main difficulty is that these images can have a much higher number of colors, and it is therefore much more difficult to exploit spatial dependencies via binary layers. The original contributions of this work include: 1. the application of context-tree-based compression (previously designed for map images) to natural and color palette images; 2. the consideration of four different methods for bit-plane separation; and 3. Extension of the two-layer context to a multilayer context for better utilization of the crosslayer correlations. The proposed combination is extensively compared to state of the art lossless image compression methods. © 2006 SPIE and IS&T.* [DOI: 10.1117/1.2388255]

1 Introduction

Lossless image compression is needed for applications that cannot tolerate any degradation of original imagery data, e.g., medical applications such as mammography, angiography, and x-rays. It is essential that the decompressed image does not contain any degradation in quality, since it could lead to misdiagnosis and health injury. Satellite or geographical map images are another case where distortion caused by compression cannot be tolerated.

The earliest lossless compression methods used either dictionary-based methods or run-length encoding.[1] However, these techniques do not exploit 2-D correlations in the image, and they are not very efficient for natural images that contain smooth color variations but do not have repeating patterns. *Predictive modeling*, on the other hand, exploits spatial correlations by predicting the value of the current pixel by a function of its already coded neighboring pixels. The difference between the actual and predicted value, called *prediction error*, is then encoded.[1] A simple *linear prediction* is used in the lossless mode of the JPEG still compression standard and a nonlinear predictor in the newer *JPEG-LS* standard.[2] Despite their apparent simplicity, prediction-based techniques are quite effective and used in state of the art compression methods.

Another approach is to use *context modeling* followed by arithmetic coding.[3] In context-based models, every distinctive pixel combination of the neighborhood is considered as its own coding context. The probability distribution of the pixel values is estimated for each context separately based on past samples. In grayscale images, however, the number of possible pixel combinations is huge and only a small neighborhood can be used. The number of contexts must therefore be reduced by *context quantization*.[4] This approach, combined with predictive modeling, has been used in the context-based adaptive lossless image compression (CALIC) algorithm.[5] The recent JPEG2000[6] compression is based on wavelet transform, and although this algorithm is aimed at lossy compression, it also includes a lossless variant.

The efficiency of the prediction scheme also depends on the type of image. For example, CALIC is efficient on photographic images (see Fig. 1) but not so good on images that contain smaller amounts of color gradation (see Fig. 2), e.g., color palette images, web graphics, geographical maps, schemes, and diagrams. On the other hand, a method called the *piecewise-constant model* (PWC)[7] has been optimized for this type of image. The algorithm is a two-pass method. In the first pass, it uses special classification to establish boundaries between constant color pieces in the image. In the second pass, the decisions are coded by a binary arithmetic coder. The method also takes advantage of uniform regions where the same context repeatedly appears.

One approach for exploiting spatial correlations efficiently is to decompose the image into a set of binary layers, as demonstrated in Fig. 3, and then compress the layers by a binary image compression method such as JBIG.[8] The advantage of this approach is that a much larger neighborhood can be applied in the context model than when operating on the grayscale values. The decompression process is reversed: the compressed file is decompressed into a set of layers, which are then combined back into the grayscale image.

Unfortunately, JBIG is not very efficient when applied to bit-plane separated layers, as it is on images that are binary

Paper 05204RR received Nov. 23, 2005; revised manuscript received Jun. 22, 2006; accepted for publication Aug. 2, 2006; published online Nov. 22, 2006.
1017-9909/2006/15(4)/043009/11/$22.00 © 2006 SPIE and IS&T.

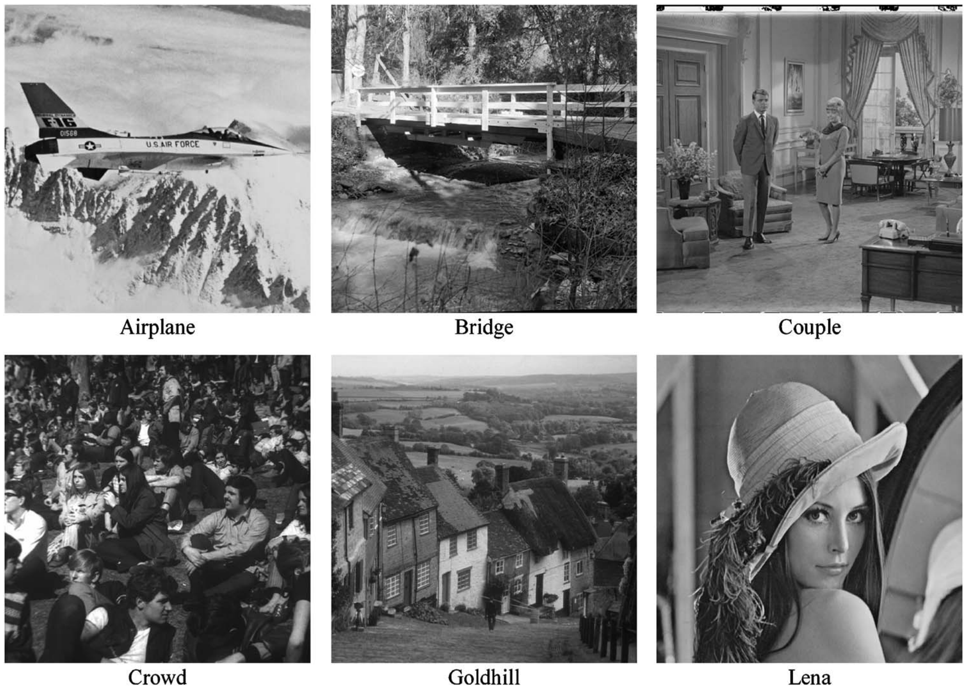


Fig. 1 Test set of natural images.

by their origin. Typically, the bit layers (especially less significant bits) lack predictable structure to be compressed well. This is because the bit-plane separation destroys the gray-level correlations of the original image, making the compressor unable to exploit them when coding the bit planes separately. In fact, interlayer dependencies are stronger than spatial dependencies within the layers. *Embedded image-domain adaptive compression of simple images* (EIDAC)[9] therefore uses a 3-D context model, where context pixels are selected not only from the current bit plane but also from the already processed layers.

Another way to improve compression performance is to increase the size of the context template. A larger context can be achieved by a selective context expansion using *context tree* (CT),[10] which allocates memory only for contexts that are really present in the image. The size as well as the ordering of the pixels within the context can be optimized.[11] An attempt to spread the optimized context tree modeling to a multilayer case called *multilayer context tree* (MCT) modeling has been made in the case of multilayer geographical map images.[12] Optimal ordering of the layers was shown to give additional improvement.[13]

In general, the efficiency of the particular compression method depends on the utilization of *color* and *spatial* dependencies (see Fig. 4). Prediction-based algorithms concentrate mainly on color dependencies, since they are looking for correlation between gray values in a relatively small spatial neighborhood. On the other hand, binary image compression algorithms concentrate more on utilizing spatial dependency than color dependencies. Binary nature of the input data makes it possible to use a larger spatial context template, but when applied to bit-plane separated data, the compression efficiency is low, since there are more interlayer (color) dependencies than spatial dependencies among the neighboring bits. A successful compression method should utilize both types of dependencies.

We study how well the bit-plane-based approach can work on natural and palette images. We apply the MCT method presented in Ref. 13, but instead of the color separation, we perform bit-plane separation because of a higher number of colors in the images. We consider four different methods: a straightforward bit-plane separation as such, gray coding, a separate prediction step, as well as the combination of the last two. Furthermore, we extend the two-layer context model to a multilayer context model for better utilization of the cross-layer dependencies. In general, one can use any previously compressed layer as the reference layers. The first layer is compressed as such, the second

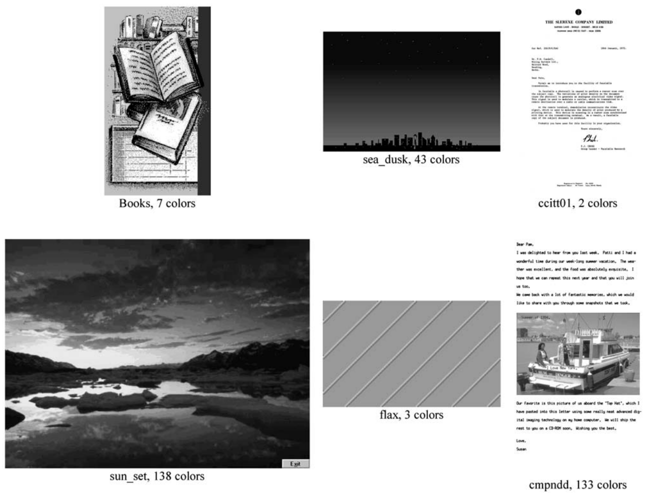


Fig. 2 Test set of simple images.

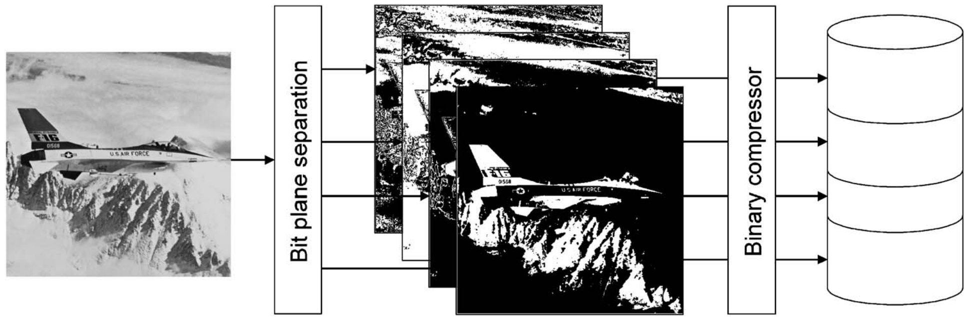


Fig. 3 Lossless compression of grayscale images by a binary-image-oriented compression.

layer can use the first one as the reference layer, and the process continues so that the last layer can use all previous layers. We denote this extension as an *N-layer context tree* modeling (NCT).

The rest of the work is organized as follows. The aspects concerning context modeling, context tree modeling, and multilayer context trees are described in Sec. 2. Different alternatives for bitplane decomposition are studied in Sec. 3. The performance of the proposed schemes is evaluated in Sec. 4 against the most competitive algorithms both for natural and palette images. Finally, conclusions are drawn in Sec. 5.

2 Multilayer Context Tree Modeling

Statistical image compression consists of two phases: *modeling* and *coding*. In the modeling phase, the probability distribution of the pixels to be compressed is adaptively estimated. The coding process assigns variable length code words to the pixels according to the probability model, so that shorter codes are assigned to more probable pixels and vice versa. The coding is performed by *arithmetic coding*[14] using implementation known as a QM-coder,[15] which was originally introduced for the JBIG standard.

2.1 *Context Modeling*

The probability of a pixel is conditioned on a *context*, which is defined as the black-white configuration of the neighboring pixels within a local template (see Fig. 5). The index of the selected context and the pixel to be coded are then sent to the arithmetic coder. In principle, better probability estimation can be achieved using a larger context template. However, it does not always result in compression improvement, because the number of contexts grows exponentially with the size of the template. This leads to the *context dilution* problem,[16] in which the statistics are distributed over too many contexts, and thus affects the accuracy of the probability estimates.

2.2 *Context Tree*

The *context tree* (CT) concept[10] 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. The contexts in CT 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 pixel values.

The tree can be constructed beforehand using a training image (static approach),[17] or optimized directly to the image to be compressed (semiadaptive approach).[10] We use the latter approach because it optimizes the structure and size of the tree directly to the input image without any parameter tuning or prior training. The structure of the tree must then be stored in the compressed file, and it takes 1 bit per node. In the case of our test sets (see Sec. 4), this corresponds to a 10 to 25% proportion of the compressed file.

A variant called *free tree*[10] optimizes the location of the template pixels adaptively at each step of the tree construction. When a new child node is created, every possible

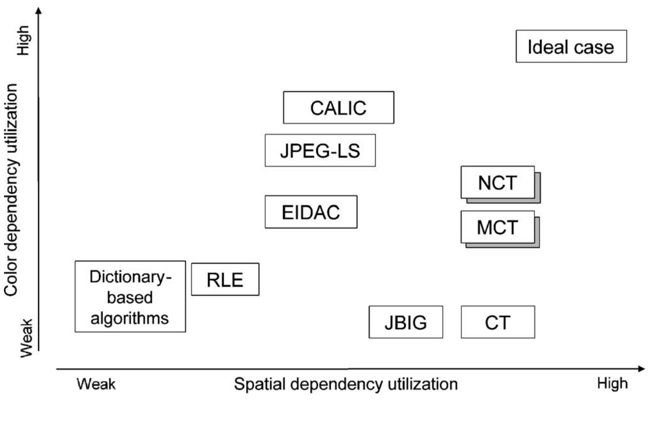


Fig. 4 Spatial and color dependency diagram. The algorithms considered in this work are emphasized by shadow.

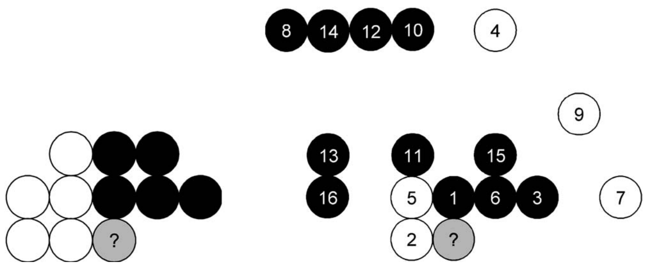


Fig. 5 Sample contexts defined by JBIG 10-pixel template (left), and the template optimized for a geographical image (right). The numbers refer to the order in which the pixels have been selected for this particular context.

location for the next context pixel is considered within a predefined search area and the compression efficiency is estimated by the entropy of the current context model H_N. The entropy is calculated as the sum of entropies of individual contexts:

$$H_N = -\sum_{j=1}^{N} p(C_j)(p_w^{C_j} \cdot \log_2 p_w^{C_j} + p_b^{C_j} \cdot \log_2 p_b^{C_j}),$$

where $p(C_j)$ is the probability of the context C_j, $p_w^{C_j}$ and $p_b^{C_j}$ are the probabilities of the white and black pixels in the context C_j, and N is the total number of contexts. The probabilities $p_w^{C_j}$ and $p_b^{C_j}$ are calculated on the basis of observed frequencies.[10] The position providing the best estimated-compression gain is included into the context rtemplate. The optimization, however, comes at the cost of additional computation time and increase in tree storage size. A sample context optimized by the free tree is demonstrated in Fig. 5.

2.3 *Two-Layer Context Tree*

The CT modeling can be extended to the multilayer case, called *MCT*, by defining a context template where pixels from previously coded layers can also be included. In this way, information from other bit layers, called *reference layers*, can compensate the loss of color correlation caused by the bit-layer separation. A two-layer model was considered in Ref. 12 using a search area consisting of 40 pixels from the current layer, and 37 from the reference layer. The pixels in the current layer can be located in the neighborhood area including already coded pixels, but the pixels in the reference layer can be located anywhere, since they are already known by the decoder, as the reference layer is always coded before the current one.

Further optimization exploits the fact that the efficiency of the compression of any particular layer strongly depends on the choice of the reference layer. In general, we can select any predefined order on the basis of known (or assumed) dependencies. When image source is not known beforehand, the optimal order of the layers can be solved as a *directed minimum spanning tree* problem[13] for maximal utilization of the interlayer dependencies. Again, the optimization comes at the price of a remarkable increase in the processing time.

2.4 *N-Layer Context Tree*

In this work we generalize the idea by considering the *N-layer context tree*, further referred as NCT. We consider all previously compressed layers as reference layers. When the first layer is compressed, the free-tree context template involves only already processed pixels of the current layer. After being compressed, this layer becomes a reference layer for the second one. Figure 6 illustrates the search area used for the compression of the third layer. It consists of 52 pixel positions, of which ten are from the current layer and 42 are from the reference layers. Each template position is examined for the provided compression gain, and the most efficient position is included in the template at each step. The process then continues as long as further improvement will be achieved. A sample context is illustrated in Fig. 7.

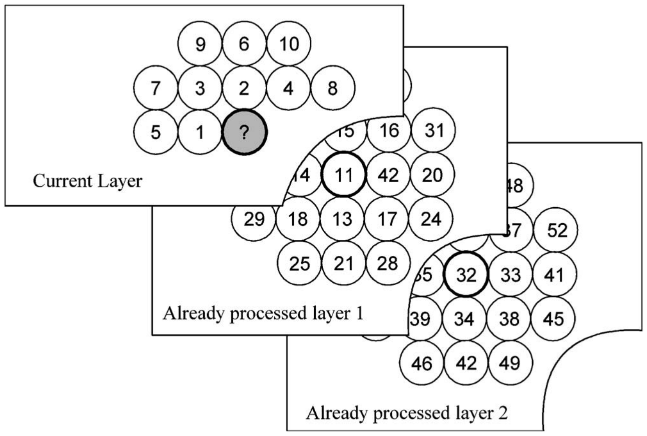


Fig. 6 Joint 52-pixel three-layer search area. The position of the current pixel is marked with "?" and the corresponding positions on the reference layers are emphasized with bold circles.

The ordering of the layers affects the compression performance in NCT in the same way as in MCT. For example, when test image Airplane is compressed starting from the least significant bit (LSB) toward the most significant bit (MSB), the obtained total code size would be 148,388 bytes. On the other hand, when compressed with reversed ordering (from MSB toward LSB), the code size is 136,185 bytes. In Ref. 13, the optimal ordering was solved as a directed minimum spanning tree problem, which was possible because only one previous layer was used as a reference layer. In the case of an N-layer context tree, similar formulation would lead to a traveling salesman problem. In this case, the optimal solution would take $O(n!)$, and an even faster heuristic would influence the processing time significantly because of a larger search area. Fortunately, the optimal ordering is not as critical as in the MCT, and therefore, we used a fixed order starting from MSB to LSB.

A common property of the context-based techniques is that in the case when the statistical dependencies of the source are extremely weak, the code size produced by the

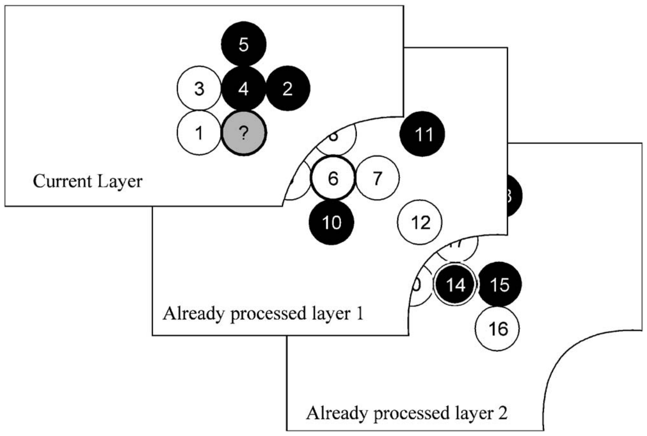


Fig. 7 A sample three-layer context constructed by the free-tree approach using the search area presented in Fig. 6. The black color represents 1 bit and white color represents 0 bits in the corresponding bit layer. The current pixels position is emphasized with a bolder circle.

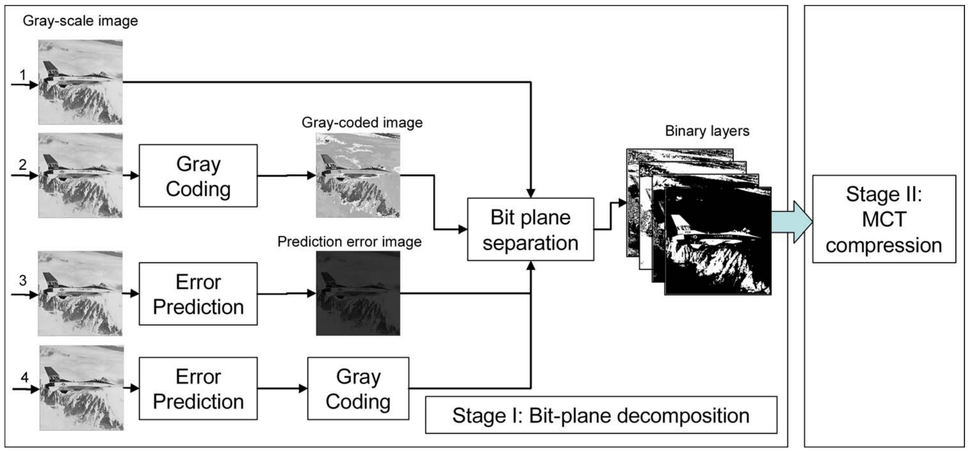


Fig. 8 Overall compression algorithm according to the different bit-plane decomposition schemes.

compressor could be even greater than the size of the uncompressed file. This issue is especially essential for the compression of the less significant bit layers, which are quite noisy. In this situation, we transmit the uncompressed bit layer as such.

3 Methods for Bit-Plane Separation

The proposed grayscale compression scheme consists of two independent lossless stages as shown in Fig. 8. In the first stage, the grayscale image is decomposed into a set of binary images (layers). In the second stage, the MCT or NCT compression method is applied. The decompression is performed in reverse order: first, an archive file is decompressed into a set of binary layers, which are then combined into a grayscale image. We consider the following four decomposition methods:

- bit-plane separation (BPS)
- gray code separation (GCS)
- Prediction error separation (PES)
- gray code prediction error separation (GCPES).

The first scheme is a straightforward bit-plane separation (scheme 1 in Fig. 8), which is a classical method for creating bit planes where each pixel value corresponds to a particular bit of the original grayscale image. The second scheme is a gray-code separation (scheme 2 in Fig. 8),[18] which codes the pixel intensities so that the change of pixel value by +1 or −1 causes the change of only 1 bit value in the corresponding bit layers. This transform is defined as

$$x \rightarrow G(x) = x \oplus (x \gg 1),$$

where $\oplus$ indicates the "exclusive-or" function, and » indicates the "binary shift-right" operation (i.e., $m \gg n = m/2^n$). For example, when the gray code is not applied, increasing value 127 (01111111b) by 1 gives 128 (10000000b), which causes changes in all eight bit layers. On the other hand, the gray code for 127 is 64 (01000000b), and for 128 it is 192 (11000000b), which differ in 1 bit only. Gray coding has turned out to be an efficient preprocessing technique for improving compression performance.[19]

The third scheme uses a separate prediction step followed by bit plane separation[20,21] (scheme 3 in Fig. 8). The idea is to encode the prediction error, i.e., the difference between the predicted and the actual value of a pixel, instead of the original gray value. Error prediction is a lossless transformation converting a grayscale image of gray values varying from 0 to 255 into a so-called *prediction error image*, where every pixel represents the prediction error varying from −255 to +255. Therefore, when using this scheme, the grayscale image is decomposed into nine binary layers instead of eight as in the first two schemes. When the predictor is effective, the prediction error values are mostly concentrated around zero. Therefore, after bit-plane decomposition, more significant bit planes contain a very small amount of variation, thus having low entropy and resulting in high compression ratio.

The fourth scheme (scheme 4 in Fig. 8) employs gray coding of the prediction error image with the following bit-plane separation. The bit layers produced by the four different bit plane separation schemes for the image Airplane are illustrated in Fig. 9.

An important design question is the choice of prediction technique. In this works, we considered three popular predictors in order to choose the most efficient for further use. The first scheme is a simple linear predictor defined as:

$$p(x,y) = \frac{(x,y-1)}{2} + \frac{(x-1,y)}{2} + \frac{(x+1,y-1)}{4} + \frac{(x+1,y+1)}{4},$$

where (x, y) is the pixel value at coordinates x and y. This is referred to further as *linear*. The second technique is a slightly more complicated prediction method employed in the JPEG-LS compressor,[2] which we refer to here as a *median* predictor. Finally, for the third scheme we have chosen the gradient-adjusted prediction (GAP) algorithm used in CALIC,[5] which is the most complicated of the three predictors considered. This predictor is referred here to as *GAP*.

4 Experiments

We used two test image sets to evaluate the algorithms. The first set consists of five classical test images: Airplane,

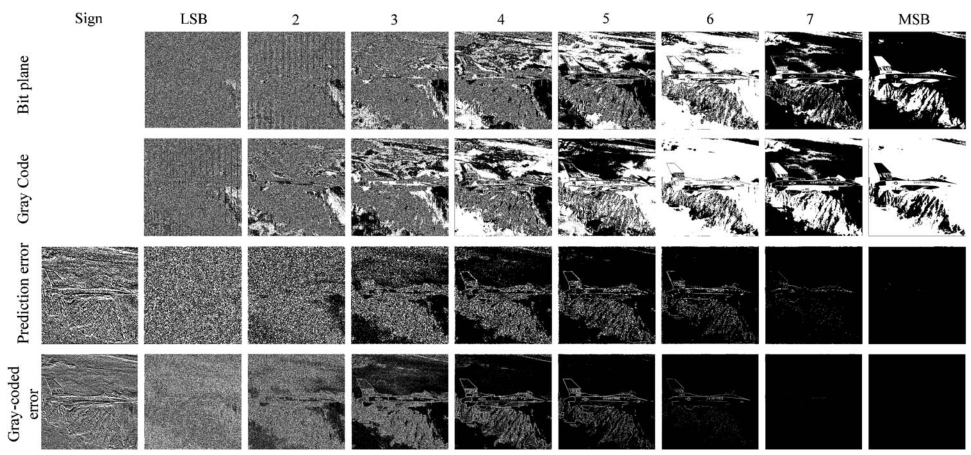


Fig. 9 Four bit-plane decomposition schemes applied to the image Airplane. Columns correspond to the bit planes starting with the sign bit, and continuing from the least significant bit to the most significant bit. Rows are different bit-plane decompositions.

Couple, Crowd, Goldhill, and Lena (see Fig. 1). All of them are 8-bit grayscale images of size 512×512 pixels. This test set represents a class of natural images that are typically photographic images of smooth color gradation. The second test set represents a class of palette images (see Fig. 2), where the number of colors is much smaller than the amount of pixels in the image. Such images can be Web graphics, schemes, maps, slides and engineering drawings, for example. This test set consists of eight images used in Ref. 7, Benjerry, Books, Ccit01, Cmpndd, Flax, Gate, Sea_dusk, and Sunset.

First, we evaluate the performance of the three prediction techniques: linear, median, and GAP predictors. Then we evaluate six variants of the proposed algorithms produced by the combination of the two context modeling schemes (MCT and NCT), and the three bit-plane decomposition schemes (BPS, GCS, and the best prediction-based scheme). Finally, we compare the best variants with the existing compressors. The competitive algorithms are:

- JBIG-GCS: JBIG applied to gray code separated layers[18,19]
- JBIG-PES: JBIG applied to prediction error separated layers
- EIDAC[9]
- CALIC[5]
- JPEG-LS[2]
- PWC-G: piecewise constant model optimized for grayscale images[7]
- PWC-P: piecewise constant model optimized for palette images[7]
- JPEG2000[6]
- PNG.

Results for EIDAC are taken from Ref. 9 and appear only for the set of palette images. The rest of the results are reported using publicly available implementations. All tests have been performed on a Pentium III 996-MHz computer with 384-MB memory and a Windows XP operating system.

4.1 *Choice of the Predictor*

We tested the performance of the three prediction techniques to choose the best for further comparison. Tables 1 and 2 summarize the overall compression performance of the PES and GCPES variants depending on the choice of predictor. In the case of natural images, the GAP predictor provides the best compression performance with all variants. The best performance (4.42 bpp) is obtained by the MCT-GCPES variant. In the case of palette images, the median predictor works better and the best result is obtained by MCT-PES (210,718 bytes). In the rest of the work, we apply the GAP predictor for natural images and the median predictor for palette images.

Table 1 Average compression results (bits per pixel) depending on the choice of predictor for the natural images.

	MCT		NCT	
Predictor	PES	GCPES	PES	CGPES
Linear	4.53	4.49	4.52	4.50
Median	4.49	4.48	4.50	4.48
GAP	4.45	**4.42**	4.44	4.43

Table 2 Total compression results (in bytes) depending on the choice of predictor for the palette images.

	MCT		NCT	
Predictor	PES	GCPES	PES	CGPES
Linear	282,280	274,525	296,877	290,815
Median	**210,718**	211,686	221,546	221,921
GAP	279,924	277,994	288,161	286,051

Table 3 Compression results (bits per pixel) for the natural images.

	MCT			NCT		
Image	BPS	GCS	GCPES	BPS	GCS	GCPES
Airplane	4.60	4.21	**4.05**	4.13	4.16	4.08
Couple	5.22	4.68	**4.54**	4.78	4.72	**4.54**
Crowd	4.76	4.38	**4.09**	4.13	4.14	4.24
Goldhill	5.68	5.10	4.96	5.02	5.08	**4.88**
Lena	5.25	4.62	4.46	4.58	4.59	**4.41**
Average	5.10	4.59	**4.42**	4.53	4.54	4.43

4.2 *Comparison of the Proposed Variants*

Here we evaluate the proposed algorithms over two test sets separately and choose the most efficient variants for further comparison. Table 3 presents the compression results for the natural image test set. The best result (4.42 bpp, on average) was obtained by MCT using both the prediction and gray coding (GCPES), but the difference from the corresponding variant of NCT is only marginal. The results also show that the choice of the bit-plane separation method is important when using MCT, as the best bit rate (4.42 bpp) is significantly better than if neither prediction nor gray coding were used (5.10 bpp). In the case of NCT, the choice of the bit-plane separation is less significant. This is because NTC can use all previous layers as references, and thus it exploits the interlayer dependencies better than MCT, which is limited to only one reference layer.

Table 4 presents results for the palette image test set. The best results (in total) are obtained by NCT without any prediction (BPS) or by using gray-coding (GCS). From this, we make three main observations. First, NCT performs better than MCT and is therefore the recommended variant for palette images. Second, the prediction-based bitplane separation is extremely inefficient. Third, a noticeable exception is the simplest three-color image (flax), for which the MCT provides significantly better results. In this test set, the larger image files dominate the results. However, if we were to compress a large number of small images with very simple structure, then the preferred variant should be MCT.

4.3 *Comparison with Existing Methods*

The best variant of the proposed method (MCT-GCPES) is compared against existing methods in Table 5. As expected, the proposed algorithm outperforms the standard JBIG applied for separated binary layers. It also gives better results than PWC-P, which is a palette-image-oriented technique, and PNG, which is a dictionary-based method. However, the MCT fails to compete with the best grayscale oriented methods such as CALIC, JPEG-LS, and JPEG2000, as well

Table 4 Compression results (in bytes) for the palette images.

	MCT			NCT		
Image	BPS	GCS	PES	BPS	GCS	PES
Benjerry	4236	4173	6204	**2988**	3135	5071
Books	8749	10,145	14,610	**7948**	8486	15,041
Ccitt01	12,046	11,827	18,312	12,055	**11,990**	27,993
Cmpndd	68,215	62,330	70,645	**57,229**	59,080	70,710
Flax	82	**81**	142	156	146	213
Gate	24,381	22,512	25,144	**18,954**	19,937	25,082
Sea_dusk	**739**	748	1047	992	859	1203
Sunset	83,011	75,673	74,614	75,268	**73,914**	76,233
Total	201,459	187,452	210,718	**175,590**	177,547	221,546

Table 5 Compression results (bits per pixel) for the natural images.

	Proposed	Competitive							
Image	MCT-GCPES	JBIG-BPS	JBIG-GCS	CALIC	JPEG-LS	PWC-G	PWC-P	JPEG2K	PNG
Airplane	4.05	5.23	4.38	3.74	3.81	3.84	4.40	4.01	4.28
Couple	4.54	5.82	4.83	4.25	4.26	4.27	5.02	4.49	4.50
Crowd	4.09	5.35	4.57	3.77	3.91	3.93	4.46	4.19	4.52
Goldhill	4.96	6.17	5.26	4.64	4.71	4.71	5.33	4.81	4.88
Lena	4.46	5.66	4.72	4.11	4.23	4.33	4.96	4.28	4.60
Average	4.42	5.64	4.75	4.11	4.18	4.21	4.84	4.36	4.56

Table 6 Compression results (in bytes) for the palette images.

	Proposed	Competitive							
Image	NCT-BPS	JBIG-BPS	JBIG-GCS	EIDAC	CALIC	JPEG-LS	PWC-G	PWC-P	PNG
Benjerry	2988	7209	7104	**2659**	5939	6707	3960	3120	4846
Books	**7948**	23,277	14,927	8517	22,299	39,859	14,878	8972	15,019
Ccitt01	12,055	103,864	13,549	**5471**	22,547	35,840	20,619	14,056	46,772
Cmpndd	57,229	89,822	67,244	**48,305**	71,917	71,469	66,090	50,026	72,695
flax	156	1208	1143	**71**	760	3411	3485	1380	420
gate	18,954	31,020	26,198	16,662	25,038	27,656	23,127	**16,242**	24,922
Sea_dusk	992	2444	2344	**870**	1219	4061	941	1292	1986
Sunset	75,268	93,069	79,434	58,402	76,577	83,552	65,831	**49,256**	79,085
Total	175,590	351,913	211,943	**140,957**	226,296	272,555	198,931	144,344	245,745

Table 7 Compression results (bits per pixel) for Bridge image. Algorithms where prediction error modeling is used provide the worst results.

	Proposed			Competitive					
Image	MCT-BPS	MCT-GCS	MCT-PES	JBIG-BPS	JBIG-GCS	CALIC	JPEG-LS	PWC-G	PWC-P
Bridge	4.40	4.93	5.80	5.37	5.20	5.37	5.50	**4.09**	4.16

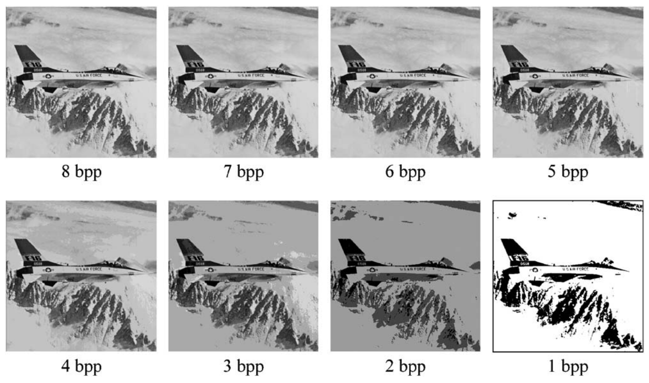


Fig. 10 Image Airplane with sequentially reduced graydepth.

as the PWC-G, which is also optimized for grayscale images. We conclude that the proposed method is most efficient when comparing to binary, dictionary-based, and palette-oriented compression algorithms, but the best grayscale-oriented techniques cannot be outperformed.

Similar comparisons for palette images are shown in Table 6. Again, the best variant of the proposed method (NCT-BPS) outperforms JBIG and all grayscale-oriented methods: CALIC, JPEG-LS, and PWC-G, as well as the dictionary-based PNG. Results for wavelet-based JPEG2000 are not presented, since this algorithm demonstrated extremely weak performance. On the other hand, the best palette-oriented algorithms such as EIDAC and PWC-P are more efficient.

Although error prediction applied to natural images is efficient in general, one can find an image where it fails to improve the compression performance. The Bridge image (see Fig. 1) is an example of such an image, as illustrated in Table 7. Note that MCT-PES failed, presenting the worst bit rate. The same holds for all competitive algorithms in which error prediction is used—CALIC and JPEG-LS. This example shows that even for a continuous-tone image case, there can be found counterexample where prediction error modeling fails to improve the compression.

4.4 *Grayscale Versus Palette Compression*

Competitive algorithms are designed to be applied to particular classes of images, either palette or photographic. These classes can be considered as images with the opposite characteristics: typical photographic images contain a lot of unique colors and have smooth color gradation, while palette images have only few colors and have sharp edges. We next study how the efficiency of the compression algorithms depends on how close the given image is to the class for which the algorithm is tuned to.

We designed an artificial test set to fill the gap between photographic and palette images by sequentially decreasing the gray depth of the original 8-bpp images. For each five images, we produced eight images with sequentially reduced gray depth, giving a set of 40 images in total. The process is illustrated in Fig. 10 for the image Airplane. We then compressed the images with two variants of the proposed algorithm: MCT (GCPES variant, the best variant for natural images) and NCT (BPS variant, the best variant for palette images), and compared them with CALIC, PWC-P, and PNG.

The results presented in Table 8 and illustrated in Fig. 11 show that, as expected, the best results for the 8-bpp images are obtained by CALIC and the worst by palette-optimized PWC-P. For images with 1 and 2 bpp, the situation inverts and the best results are shown by PWC-P and the worst by CALIC. The PNG presented an intermediate performance in both cases. NCT, on the other hand, has a slight edge over the other methods, performing best every-

Table 8 Compression results for gray depth reduction. Results are average bit rate over test set for every color depth separately.

	Proposed		Grayscale optimized	Palette optimized	Universal
Color depth, bpp.	MCT-GCPES	NCT-BPS	CALIC	PWC-P	PNG
1 (binary image)	0.42	0.16	0.23	0.18	0.43
2	0.55	0.18	0.39	0.23	0.55
3	2.28	0.50	1.10	0.53	1.11
4	3.46	1.00	2.13	1.07	1.65
5	4.30	1.72	3.14	1.86	2.66
6	4.50	2.59	3.85	2.89	3.37
7	4.46	3.56	4.08	3.97	4.01
8 (original image)	4.42	4.53	4.11	4.84	4.56

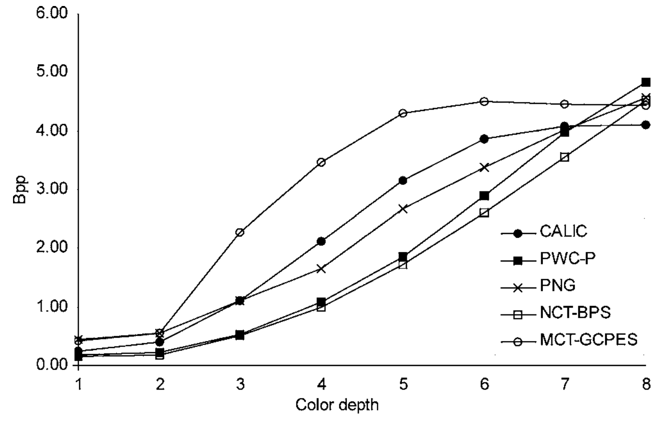


Fig. 11 Compression efficiency (bpp) depending on the color depth.

where else between 2 and 8 bpp. It seems to be the best choice when the images are not clearly of one type: photographic or palette images.

5 Conclusion

We study the efficiency of binary-oriented compression algorithms based on statistical probability estimation and arithmetic coding applied to grayscale and palette images. We consider two modeling schemes. The first scheme (MCT) uses two-layer free-tree modeling with optimized layer ordering. The second scheme (NCT) extended the context modeling to a true multilayer case with fixed ordering. We use four schemes for bit-plane decomposition: bit plane separation (BPS), gray code separation (GCS), prediction error separation (PES), and gray-coded prediction error separation (GCPES).

For prediction-based schemes, we evaluate three predictors: a simple linear scheme, median predictor employed by JPEG-LS, and gradient-adjusted prediction (GAP) used by CALIC. We find that the gray-coded GAP predictor together with MCT (MCT-GCPES) modeling provides the most efficient compression for natural images. The results also show that prediction-based bit-plane separation is inefficient for palette images. For this class of images, NCT with BPS separation (NCT-BPS) is the most efficient, though its advantage over NCT-GCS is minor. We conclude that NCT modeling is less dependent on the chosen bit-plane separation method.

The comparison with the existing compression algorithms on the Natural test set showed that MCT-GCPES outperforms JBIG, which is of similar nature, dictionary-based PNG, and palette-optimized PWC. Its performance is also close to that of lossless JPEG2000. However, other grayscale optimized algorithms—CALIC, JPEG-LS, and grayscale optimized PWC—are not outperformed. For this test set, we conclude that binary-based compression, even if applied with a very high degree of optimization, cannot outperform grayscale-oriented algorithms due to its binary nature.

The same comparison applied to a test set of palette images shows that NCT-BPS outperforms all binary-based techniques (JBIG-BPS and JBIG-GCS) as well as grayscale-optimized algorithms (CALIC, JPEG-LS, PWC-G, and universal PNG). Palette-optimized compressors EIDAC and PWC-P, however, are not outperformed, though the performance of the proposed method is closer to the best algorithm than to the worst.

The results of the palette test set inspired us to perform a detailed investigation of the algorithm's behavior depending on the amount of colors in the image. We designed eight test sets of images where color depth is sequentially decreased from 8 (grayscale case) to 1 bpp (binary case) and examined the performance of best palette (PWC-P and the proposed NCT-BPS), grayscale (CALIC and the proposed MCT-GCPES), and universal PNG. We found out that NCT-BPS performs closely to PWC-P and even outperforms it on some bit depths whereas MCT-GCPES loses to CALIC in all cases. From this observation, we conclude that first, bit-plane separation and binary modeling such as MCT-GCPES cannot be considered to be efficient for natural images, even if strong optimization is involved. Second, context-based compression techniques, such as NCT-BPS, could be considered efficient to be applied for compression of simple (palette) images and the optimization results in compression improvement.

Acknowledgments

The work of the first author was supported by the Finnish Foundation for Technical Development (Tekniikan Edistämissäätiö). The authors also thank Mantao Xu for his advice and support.

References

1. K. Sayood, *Introduction to Data Compression*, 2nd ed., Academic Press, San Diego (2000).
2. ISO/IEC JTC1/SC29/WG1, "JPEG-LS part-2 WD," ISO/IEC JTC1/SC29/WG1 N938, (1998).
3. G. G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Commun.* **29**(6), 858–867, (June 1981).
4. X. Wu, "Lossless compression of continuous-tone images via context selection, quantization and modeling," *IEEE Trans. Image Process.* **6**, 656–664, (1997).
5. X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Commun.*, **45**(4), 437–444 (1997).
6. D. S. Taubman and M. W. Marcellin, *JPEG2000: Compression Fundamentals, Standards, and Practice*, Kluwer, Norwell, MA (2001).
7. P. J. Ausbeck, Jr., "The piecewise-constant image model," *Proc. IEEE*, **88**(11), 1779–1789, (Nov. 2000).
8. International Telegraph and Telephone Consulative Committee (CCITT), Progressive Bilevel Compression, Recommendation T.82, (1993); also appears as: International Organization for Standards/ International Electrotechnical Commission (ISO/IEC), Digital Compression and Coding of Continuous-Tone Still Images, International Standard 10918–1:1993.
9. Y. Yoo, Y. G. Kwon, and A. Ortega, "Embedded image-domain compression using context models," *Proc. IEEE Int. Conf. Image Process 1999 (ICIP 99)*, **1**, 477–481, (1999).
10. B. Martins and S. Forchhammer, "Tree coding of bilevel images," *IEEE Trans. Image Process.* **7**(4), 517–528, (Apr. 1998).
11. E. I. Ageenko, P. Kopylov, and P. Fränti, "On the size and shape of multilevel context templates for compression of map images," *IEEE Int. Conf. Image Process.* **3**, 458–461 (2001).
12. P. Kopylov and P. Fränti, "Context tree compression of multicomponent map images," *Proc. IEEE Data Compress. Conf.*, pp. 212–221 (2002).
13. P. Kopylov and P. Fränti "Compression of map images by multilayer context tree modeling," *IEEE Trans. Image Process.* **14**, 1–11, (Jan. 2005).
14. J. J. Rissanen and G. G. Langdon, "Arithmetic coding," *IBM J. Res. Dev.* **23**, 146–162 (1979).
15. W. Pennebaker and J. Mitchell, "Probability estimation for the Q-coder," *IBM J. Res. Dev.* **32**(6), 737–759, (1988).
16. M. J. Weinberger, J. Rissanen, and R. B. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. Image Process.* pp. 575–586, (Apr. 1996).

17. P. Fränti and E. Ageenko, "On the use of context tree for binary image compression," *Proc. IEEE Int. Conf. Image Process. (ICIP'99)* 3, 752–756, (1999).
18. M. Kunt and O. Johnsen, "Block coding of graphics: A tutorial review," *Proc. IEEE* **6**(11), pp 770–786, (1980).
19. M. Abdat and M. G. Bellanger, "Combining gray coding and JBIG for lossless image compression," *Proc. IEEE Int. Conf. Image Process. 1994 (ICIP 94)*, 3, 851–855, (Nov. 1994).
20. K. Denecker, W. Philips, and I. Lemahieu, "Universal binary context modeling of image residue for lossless image compression," *Proc. ProRRISC/IEEE Circ. Syst. Signal Process.* pp. 119–124 (1998).
21. B. J. Falkowski, "Compact representation of logic functions for lossless compression of gray-scale images," *IEE Proc. Comput. Digit. Tech.* **151**(3), 221–230, (May 2004).

Alexey Podlasov received his MSc degree in applied mathematics from Saint-Petersburg State University, Russia, in 2002, and his MSc degree in computer science from the University of Joensuu, Finland, in 2004. Currently, he is a doctoral student in computer science in the University of Joensuu. His research topics include processing and compression of map images.

Pasi Fränti received his MSc and PhD degrees in computer science in 1991 and 1994, respectively, from the University of Turku, Finland. From 1996 to 1999 he was a postdoctoral researcher of the Academy of Finland. Since 2000, he has been a professor in the University of Joensuu, Finland. His primary research interests are in image compression, clustering and speech technology.

Publication P3

3

Podlasov A., Fränti P., Merge-based color quantization and context tree modeling for compression of color quantized images, *IEEE International Conference on Image Processing (ICIP'06),* Atlanta, Georgia, USA, pp. 2277–2280, October 2006.

MERGE–BASED COLOR QUANTIZATION AND CONTEXT TREE MODELING FOR COMPRESSION OF COLOR QUANTIZED IMAGES

Alexey Podlasov and Pasi Fränti

Speech and Image Processing Unit
Department of Computer Science
University of Joensuu
P.O.Box 111, 80101 Joensuu, Finland
{apodla, franti}@cs.joensuu.

ABSTRACT

A two-stage lossless compression method based on a binary tree representation of colors and on context-based arithmetic coding has been recently proposed. We propose two improvements to this method: merge-based color quantization instead of the original splitting strategy, and context tree modeling optimized for each layer separately. The proposed method achieves better compression performance, and a better reproduction quality in the color progression.

Keywords: progressive image coding.

1. INTRODUCTION

Color-quantized and palette images are widely used in web and on low-cost devices, which are typically restricted by a low number of colors displayed simultaneously. Lossy compression is not always applicable since the degradation of the quality cannot be tolerated in many applications. On the other hand, lossless algorithms optimized for compression of photographic images lack the compression efficiency since the correlation between image indexes in palette images can be lost.

Significant progress has been made in recent years in palette- and color-quantized-oriented lossless compression. Typically, two principal approaches have been considered. The first approach uses color map reordering [1], which revives index correlation, followed by compression with existing techniques such as JPEG-LS [2] or CALIC [3]. The second approach concentrates on the development of specific, well-tuned coding techniques. Successful examples of such algorithms are PWC [4] and EIDAC [5].

Chen *et al*. [6] has proposed an algorithm following the second approach. Besides the efficient compression it also provides lossy-to-lossless progressive encoding allowing to stop the transmission when the desired quality level is reached. The algorithm consists of two main stages: color indexing and context-based arithmetic coding. At the indexing stage, the palette of the image is represented by a binary tree. The root of the tree corresponds to all colors appeared in the image. Two children of the root divide the root's color set into two subsets. Every node is divided in the same manner until every single color gets its own leaf node. The tree is constructed minimizing the distortion caused by the color quantization. At the coding stage, the tree is traversed starting from the root. For each node, the encoder sends the weighted average color of each of its two children and a bitmap indicating the location of the pixels having a color change. The bitmap is encoded by a binary context-based arithmetic coder.

In this paper, we follow the principles proposed by Chen *et al.* and consider two improvements of the original algorithm. First, we propose merge-based tree construction [7]. Then, we consider improved encoding of pixel locations based on highly-optimized context tree modeling [8]. We show that the applied techniques achieve improvement both in compression performance and in quality of the color progression.

2. COMPRESSION SCHEME OUTLINE

In this section, we briefly outline Chen's compression algorithm, and its improved variant [9].

2.1. Binary tree color indexing

For the description of tree construction scheme, we use the formulation as presented in [9]. We denote the color palette of an image as a set of RGB triplets $C = \{c_1, \ldots, c_M\}$, where M is the total number of colors. Let $S_0 = \{1, 2, \ldots, M\}$ be the set of indexes identifying the colors in the palette, where index i corresponds to the color c_i. The number of pixels of color c_i is denoted p_i. Each node j of the binary tree represents a certain subset of the color palette. This node is referred using the corresponding set of indexes s_j. Every color set is associated with a *representative color* q_j, which is given by the weighted average color of the node,

$$q_j = \frac{\sum_{i \in s_j} p_i c_i}{\sum_{i \in s_j} p_i}$$

The tree is constructed in a top-down manner starting from the root. Whenever the color set associated with a node contains more than one color, the node is split into two, in such a way that the two new subsets of colors are separated by a hyperplane that is perpendicular to the principal direction of the data and passes through the average color value q_j. The principal direction of the data is given by the eigenvector of largest eigenvalue of the covariance matrix of the weighted colors in s_j, C_j, where

$$C_j = \sum_{i \in s_j} p_i (c_i c_i^t - q_i q_i^t).$$

When the tree is constructed, every color of the image palette is associated with a unique variable-length binary sequence representing the path in the tree. When decompressing the sequence, we know that the first bits represent the most important part of the color information. In this way, the binary tree representation assigns indexes to the image colors so that neighboring indexes are assigned to neighboring colors, converting the correlation of colors into the correlations of indexes. This property is utilized for obtaining higher compression efficiency for color-quantized images.

2.2. Image encoding

The encoding starts from the first node of the tree for which its representative color value q_0 is transmitted. The following procedure is then applied for every node of the tree:

1. Choose the node m for processing.
2. Transmit the index of the node and the representative colors of its two children q_m^l and q_m^r.
3. Encode the location of the pixels, whose colors belong to the sets s_m^l and s_m^r.

The node to process is chosen according to the largest associated eigenvalue of C_j. The values q_m^l and q_m^r are the representatives of the color sets s_m^l and s_m^r, associated with m's two child nodes (note, that $s_m = s_m^l \cup s_m^r$).

The pixel locations are encoded by a context-based arithmetic coder. Chen *et al.* proposed to use an 8-pixel fixed-order context template (see Figure 1, left context). Since the location of the pixels whose color belongs to s_m is known to the decoder, we only need to encode, for each of those pixels, which now belongs to s_m^l and which to s_m^r. The information to be encoded is binary and for every pixel which color $q \in s_m$ we encode a bit b_0, defined as

$$b_0 = \begin{cases} 0, & q \in s_m^{\ l} \\ 1, & q \in s_m^{\ r} \end{cases}.$$

The context is constructed using a sequence of bits $b_1, \ldots, b_8$, where

$$b_i = \begin{cases} 0, & \left\| q^i - q_m^{\ l} \right\| \le \left\| q^i - q_m^{\ r} \right\| \\ 1, & otherwise \end{cases},$$

and q^i denotes the color of the pixel from the reconstructed image corresponding to the position i of the context template.

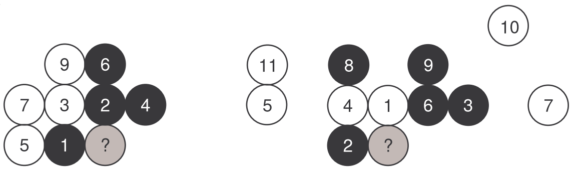


Figure 1. Sample contexts obtained by Chen's algorithm (left) and a sample context constructed by the context-tree approach (right).

In order to achieve better compression performance and avoid context dilution problem, Chen *et al.* considered a variable size context. Instead of using all 8 template locations, only first k are used, where k is defined by the relation

$$k(n) = 9 - \lfloor \log_2 n \rfloor,$$

where n-1 is the number of colors already encoded.

Recently, Pinho *et al.* [9] proposed another context adaptation model for Chen's algorithm. The generalization of context size function $k(n)$ is considered as

$$k(n) = \lceil \alpha(N) - \log_2 n \rceil,$$

where $\alpha(N) = m\log_2 N + b$ is a function of the number of pixels in the image. The values m and b are tuned using test images. This scheme takes into account the size of the image and results in about 4% compression improvement over Chen's algorithm.

3. PROPOSED APPROACH

The original algorithm assumes split-based (top-down) tree construction method. Though this scheme allows combining the tree construction with the encoding into one process, this algorithm lacks quantization quality. Its visual appearance is illustrated on a sample color images (see Figure 2, upper row). We consider another simple and popular quantization heuristics using a bottom-up approach [7]. The algorithm is based on agglomerate (tree-structured) clustering and it provides more precise quantization and better visual appearance (see Figure 2, lower row). Besides that, we apply *free-tree* context modeling [8], instead of a variable-size static-order modeling used by Chen *et al.* The overall scheme is illustrated on Figure 3.

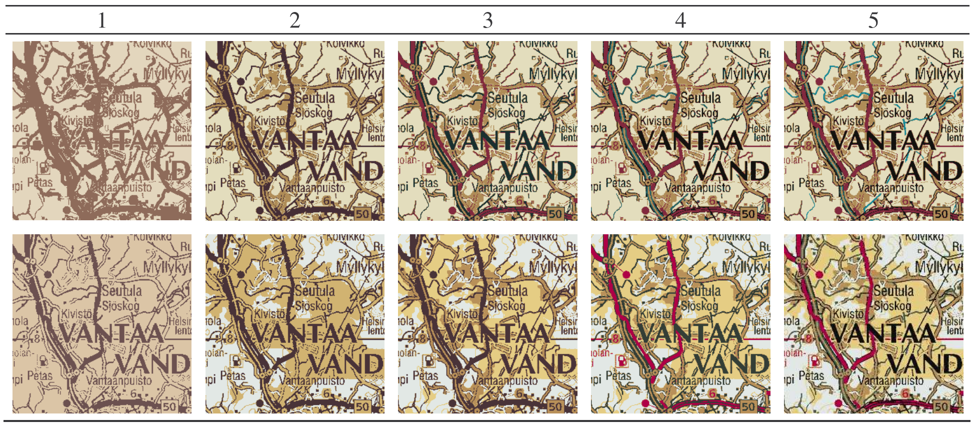


Figure 2. Illustration of the tree-structured quantization. Quantization steps are shown from left to right. Upper sequence corresponds to the split-based tree construction, and the lower sequence represents merge-based tree construction.

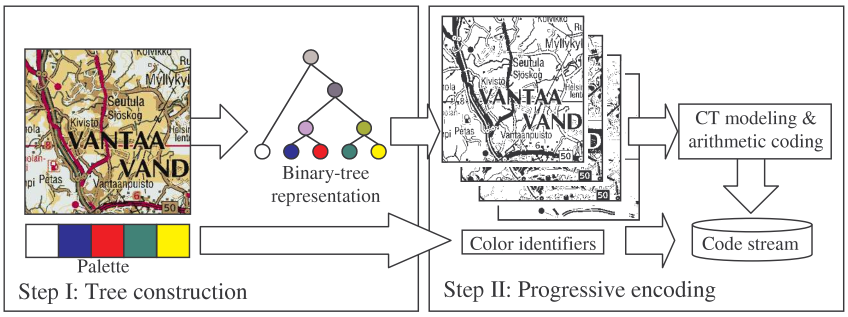

Binary-tree representation
CT modeling & arithmetic coding
Palette
Color identifiers
Code stream
Step I: Tree construction
Step II: Progressive encoding

Figure 3. Two steps of the proposed algorithm.

3.1. Merge-based tree construction

The algorithm constructs the tree bottom–up using a sequence of merge operations. First, every color c_i of the image palette $C = \{c_1, \ldots, c_M\}$ is associated with a leaf node in the tree. At each step, two color nodes are merged. The nodes are selected so that their merging causes minimal increase in distortion. The distortion d_{ij} caused by the merging of two color sets s_i and s_j, is calculated by

$$d_{ij} = \frac{n_i n_j}{n_i + n_j} \left\| q_i - q_j \right\|^2,$$

where q_i and q_j are the color representatives of the corresponding sets, and n_i is the number of pixels that belongs to the set s_i, defined as

$$n_i = \sum_{i \in s_i} p_i$$

The value n_j is defined analogously. The parent node is associated with the merged color set. The process continues until only one node is left in the tree. After the tree is constructed, it is applied for color progression and compression in the same top-down manner as in the original algorithm.

3.2. Context tree modeling

Context tree is a highly optimized form of context modeling technique, which has shown to be an efficient tool in compression. It provides 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. The contexts are represented by a binary tree, in which the context is constructed pixel by pixel and the memory is allocated only for contexts that are really present in the image avoiding extensive memory consumption.

In free-tree variant [8], the location of the template pixels is also optimized. The position of the next context pixel is determined at each step. When a new child is constructed, all possible positions for the next context pixel are analyzed within a predefined search area, and the position resulting in maximal compression gain is chosen. The construction is stopped when increasing the context size does not give any further improvement in the probability estimate, giving the optimal context size. The sample context constructed by the free-tree is illustrated in Figure 1, right context.

Table 1. Compression performance of PNG, Pinho et al. and PNN-Free Tree algorithms on a palette image test set.

Image	PNG	Chen *et al.*	Pinho *et al.*	Proposed
pc	360291	135051	117794	91220
books	20754	8102	8047	7950
music	2800	830	829	814
winaw	33182	13046	12980	11366
party8	10140	3341	3442	2933
netscape	40546	11176	11176	11146
sea dusk	2712	1497	1595	1128
benjerry	6239	2847	2848	2921
gate	47446	15329	15302	15057
descent	39738	17911	17853	17157
sunset	136966	59806	59489	58335
yahoo	11912	5764	5771	6442
Total	712726	274700	257126	226469

Table 2. Compression performance of PNG, Pinho et al. and PNN-Free Tree algorithms on a natural image test set.

Image	PNG	Chen *et al.*	Pinho *et al.*	Proposed
airplane	381946	121424	121121	122337
anemone	575643	164523	164073	163486
arial	715586	238231	235740	228611
baboon	597818	178131	177768	175876
bike3	1074229	287320	278711	274248
boat	520232	150633	148546	151991
clegg	966435	344181	326807	337366
cweel	299722	122990	108225	102530
fractal	245480	241345	238026	239310
frymire	787184	323584	286396	267087
ghouse	699103	214900	208669	201592
girl	398164	134186	132364	135017
Total	7261542	2521448	2426446	2399451

The drawback of the approach is that the context tree should be transmitted to the decoder increasing the size of required side information. For some images this increase overweights the compression improvement. In order to overcome this drawback we implemented a combined scheme. On every encoding step, we compare the performance of Chen's and free-tree context modeling schemes, choose the best and transmit a flag bit indicating which modeling scheme is used.

4. EXPERIMENTS

We tested our algorithm on the set of images used in [9]. The images are separated into two classes: palette images (of synthetic nature) and natural images. The proposed algorithm is compared to the PNG compressor, the original Chen's algorithm [6], and Pinho's modification [9]. For palette images, our algorithm outperforms the Chen's algorithm by 17.5%, in total, and the Pinho's modification by 11.9%, though negative improvement was obtained on some individual images. The tests on natural image test set show that Chen's algorithm is outperformed by 4.8%; and minor improvement of 1.1% is obtained to Pinho's method. All three techniques clearly outperforms PNG compressor for both test sets.

5. CONCLUSION

We proposed bottom-up tree construction based on sequence of merging and tree-based context modeling for color-quantized image coding. The proposed algorithm improves the compression by about 12% for a set of palette images, and about 1% for a set of natural images.

6. REFERENCES

[1] A.J. Pinho and A.J.R. Neves, "A survey on palette reordering methods for improving the compression of color-indexed images", *IEEE Trans. on Image Processing*, vol. 13, no. 11, pp. 1411-1418, Nov. 2004.

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

[3] X. Wu, N. Memon, "Context-based, adaptive, lossless image coding", *IEEE Trans. on communications*, 45(4), pp. 437-444, 1997.

[4] Ausbeck, P.J., Jr. "The piecewise-constant image model" *Proceedings of the IEEE*, Vol. 88, Issue 11, pp. 1779–1789, Nov 2000.

[5] Y. Yoo, Y. G. Kwon, A. Ortega, "Embedded image-domain compression using context models", *Proc. of IEEE Int. Conf. on Image Processing 1999 (ICIP 99)*, Kobe, Japan, vol. 1, pp. 477–481, 1999.

[6] X. Chen, S. Kwong, and J.-F. Feng, "A new compression scheme for color-quantized images", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 904-908, Oct. 2002.

[7] P. Fränti, T. Kaukoranta, D.-F. Shen and K.-S. Chang, "Fast and memory efficient implementation of the exact PNN", *IEEE Trans. Image Process.*, vol. 9, no. 4 pp. 773-777, May 2000.

[8] B. Martins and S. Forchhammer, "Bi-level image compression with tree coding," *IEEE Trans. Image Process.*, vol. 7, no. 4, pp. 517-528, Apr. 1998.

[9] A.J. Pinho and A.J.R. Neves. A context adaptation model for the compression of images with a reduced number of colors. *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2005*, September 2005, Genova, Italy.

Publication P4

Podlasov A., P. Kopylov, Fränti P., Statistical filtering of raster map images using a context tree model, *IEEE International Conference on Signal-Image Technology and Internet-based Systems (IEEE-SITIS'07),* Shanghai, China, December 2007. (to appear)

Statistical filtering of raster map images using a context tree model

Alexey Podlasov*, Pavel Kopylov and Pasi Fränti*
*Speech and Image Processing Unit, Department of Computer Science and Statistics, University of Joensuu, P.O Box-111, 80101, Joensuu, Finland
{apodla,franti}@cs.joensuu.fi

Abstract

We propose a statistical filter using a context tree modeling. The idea of context tree is to perform selective context expansion including only those pixel combinations that really appear in the image. This makes it possible to use much larger spatial neighborhood. The proposed context tree filtering is evaluated for a set of indexed-color raster map images corrupted with generated impulsive and content-dependent noise. The objective evaluation shows improvement of 15% for content-dependent noise and up to 30% for impulsive noise comparing to the closest competitor. Visual comparisons show that the spatial structures are preserved better than by vector median, morphological and peer group averaging filter.

1. Introduction

Geographical map images are typically present in two fundamentally different formats: raster and vector. Vector format is more suitable for large databases providing excellent flexibility and compression even though vector processing can be computationally expensive. Raster images are easier to process and this format is more suitable for final client-side processing for delivery, local archive storage and web-publishing. Typically, vector-to-raster conversion does not affect the quality of the raster image presented to the user. However, cases when the original vector data is not available are common. Raster image can be degraded by noise caused by transformations and lossy compression. Distortion also appears when a printed map is digitized. In these cases, the presence of noise can corrupt the spatial structures in the image.

A great variety of noise removal techniques are known for color image processing [1][2][3]. However, map images require some restrictions to be set. Firstly, the image should not be smoothed and it should remain readable. Secondly, the number of colors is typically small in a map image and

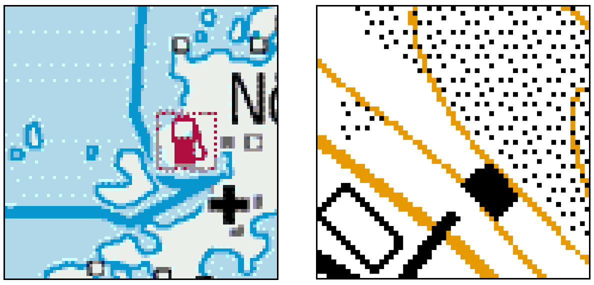


Figure 1. Examples of complicated structures that are treated as noise by most filters.

preferably it should not be increased. Thirdly, the spatial structures in the image should be preserved since they have distinctive meaning. Linear filtering methods cannot be effectively applied to map images because of their smoothing effect, which cannot be tolerated in map images. Among popular non-linear filtering there are methods such as *morphological filters* [4]; *directional vector filters* [5]; a class of *weighted median filters* [6]; its vector extension referred as *vector median filter* (VM) [7] and the adaptive variant referred as *adaptive vector median filter* (AVM) [8]. *Peer group analysis* (PGA) [9] is an edge-preserving smoothing technique based on finding a group of pixels similar to the current one in a local neighborhood. In case there is such group, the pixel is replaced with the average of its peer group.

However, existing filtering methods are mostly designed for continuous-tone images and they do not apply well for map images, web graphics and similar. This kind of images include complicated spatial structures such as one-pixel thin lines, textured areas, dashed and dotted lines, text and symbols. False filtering of this kind of structures is typical for most filters designed for photographic imagery since they tend to consider noise as a local intensity variation without taking into consideration the repeated patterns in the globally in the image. On the other hand, high variance does not necessarily identify the noise. The regions with written text or textured background are far from being uniform but their presence is vital for the usability of the map.

The examples of such structures are illustrated in

Figure 1. The area consisting of isolated black pixels on the map represents sand field in nature. Single pixels and thin lines are considered as noise by most of the existing filters, and thus, they are filtered erasing important geographical information. Morphological filtering would be a natural choice to consider for this kind of imagery. However, the drawback of morphological filtering is the concept of *structural element*, defining the preferred configuration of local patterns where domination of some pixels over the others is emphasized. It is clear that the variety of patterns in a map image is great and one or even a set of structural elements is not able to describe it accurately. Moreover, color morphology is a generalization of gray-scale morphology made by *reduced ordering*, i.e. the 'domination' relationship is defined on color vectors analogously to gray-scale intensity values. However, it seems that in color map images no color can be considered prevailing over the others just by its vector characteristics like energy and intensity.

In this paper, we introduce a statistical filter based on conditional probability estimation allowing the preservation of detailed structures in map images. The proposed filter consists of two stages: *analysis* and *filtering* stage. In the analysis stage, local conditional probabilities are estimated within the image by gathering statistics of how often each particular color appears within the same local neighborhood, called *context*. The size of the context is then optimized by using a *context tree*. The analysis stage does not consider any *a priori* knowledge about the imposed noise characteristics. In the filtering stage, all pixels that have color of low probability in its context, are considered as noise and replaced by the most probable color. In this way, the repetition of local patterns can be discovered within the image. Patterns that appear frequently enough are considered belonging to the image structure and preserved. Pixels that are unlikely to appear in their neighborhood are considered to be noise and filtered out. This property allows the filter to preserve borders and structures independently of their size or variance. Preliminary version of the work has been presented in [10]. Similar filter was considered in [11][12], where context modeling and filtering decision is made in assumption that probabilistic characteristics of noisy channel are known.

The rest of the paper is organized as follows: the proposed filter is described in Section 2; noise models are considered in Section 3; the results of experiments are presented in Section 4; and conclusions drawn in Section 5.

2. Context Tree filter

2.1 Context-based statistical filtering

Consider an image I as a rectangular grid of pixels $I(x,y)$, where (x,y) is a position of a pixel and $I(x,y)$ is its value, or color. Let $I(x,y) \in \{1, \ldots, k\}$, $\forall(x,y)$, where k is the *number of colors* in the image. We assume that k is small enough to allow the storage of the image in palette-indexed format. We define a *context* $c = \{ I(x_1,y_1), \ldots, I(x_n,y_n) \}$ as a set of n pixels, where n is denoted as the a size of the context c. The positions of the pixels in a context $(x_1,y_1), \ldots, (x_n,y_n)$ are defined as a set of offsets to the position of the current pixel, and is referred as a *context template*. In Figure 2, A illustrates a sample 20-pixel context template where the position of the current pixel is marked with '×'. The context B illustrates a sample context for a binary case, *i.e.* $I(x,y) \in \{background, foreground\}$, $\forall(x,y)$, and C illustrates similar example with more colors available.

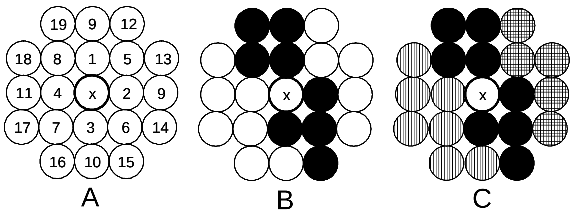


Figure 2. Template used by context tree (A) and sample contexts for the case of binary (B) and color (C) images.

The context defines the configuration of neighboring pixels and the same configuration can repeat in the image on different positions. When the neighborhood of the current pixel $I(x,y)$ equals to a context c we say that pixel $I(x,y)$ appears in a context c, and denote it as $I(x,y) \in c$. Note that the current pixel value is excluded from the context, meaning that different pixel values can appear in the same context. We associate each context c with a vector $p^c = (p^c{}_1, \ldots, p^c{}_k)$ called a *vector of statistics*, where $p^c{}_i$ represents a number of times the pixel of color i appeared in a context c in the image. After the vectors of statistics have been gathered for every context of the image, the conditional probability of every pixel to appear in its context can be estimated as

$$p(I(x,y) = j \mid I(x,y) \in c) = \frac{p^c{}_j}{\sum_{i=1..k} p^c{}_i} \qquad (1)$$

We denote this probability as $p(I(x,y)|c)$.

After the statistics have been gathered, the actual filtering is performed requiring a separate pass over the image. The main idea of the proposed filter is based on the assumption of statistical consistency of the image data. We expect that patterns appear in the image frequently enough, *i.e.* conditional probability $p(I(x,y)|c)$ of a pixel is higher than a predefined threshold for most of the pixels. Otherwise, the pixel is considered as noise and filtered out. As a replacement strategy we consider to replace the noisy pixels with the most probable color in the context. Formally, the algorithm can be outlined as follows:

```
  Analysis stage:
  For each $(x,y)$ do
    C = $\{I(x_1,y_1), \ldots, I(x_n,y_n)\}$;
    $p^c_{I(x,y)} = p^c_{I(x,y)} + 1$;
  For each $C$ do
Calculate $P(I=j|C)$ $\forall j \in [1,k]$ as (1)

  Filtering stage:
  For each $(x,y)$ do
    If $p(I(x,y)|c)$ < Threshold
   $I(x,y) = \arg\max_{j=1..k}(p(I(x,y)=j \mid I(x,y) \in c))$
```

The concept is illustrated in Figure 3 for image consisting of three unique colors. For simplicity we consider context tree filtering within 3×3 neighborhood, and two sample contexts (A and B). In the same context, some pixel values are less probable than the others, *e.g.* black pixel is much less likely to appear than white pixel in Context A, and vice versa, white pixel is much less probable than black pixel in Context B. The probability of these pixels falls below the threshold, and therefore, the pixels are filtered by replacing with the values of the most probable ones. Three examples of contexts and their corresponding probability distributions obtained in experiments with 5-color images are presented in Figure 4. There is a clear domination of the most probable color over the others.

2.2 Context Tree modeling

Gathering pixel occurrence statistics requires one pass over the image and allocating memory for as much as there are different contexts in the image. This number is upper bounded by the number of pixels in the image. In order to optimize the memory allocation we organize the storage of statistics as a tree structure called *context tree* (CT). Similar structures have been used for probability estimation in binary image compression [13] and indexed color image compression [14].

In context tree, a context is sequentially constructed pixel-by-pixel, or to say more precisely, position-by-position according to a predefined ordered context template such as the one in Figure 2.

Each node stores a vector of statistics for its context: f_W for the number of white pixels and f_B for

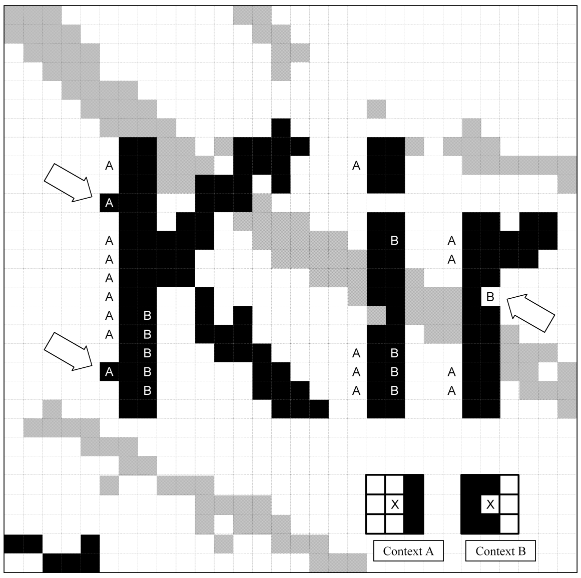


Figure 3. Example of statistical filtering. Two sample contexts are marked by A and B. The filtered less probable pixels are pointed by arrows.

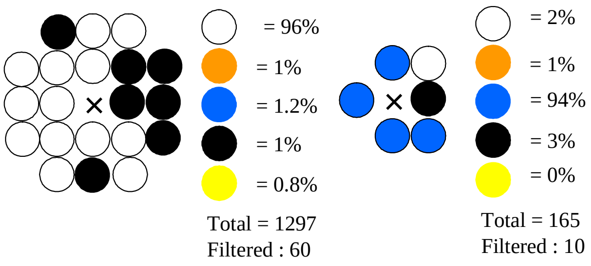


Figure 4. Sample contexts and the statistical distribution of the colors in a 5-color map image.

the number of black pixels in Figure 5. Statistics are gathered only for those contexts that really appear in the image. The principle is illustrated in Figure 5 and Figure 6 for the case of binary and a 4-color images, respectively. Every node of the tree represents a particular combination of the template pixels.

The deeper the tree grows the larger context model is used. Usually the image is processed pixel-by-pixel. For every pixel, the tree is traversed down to the desired depth, and by updating all pixel counters for the corresponding nodes along the path from the root to a leaf. When a context appears first time in the image and the corresponding node tree does not exist in the context tree, it must then be created dynamically at this moment.

Potentially, the final level of the tree can contain k^n nodes, where n is the size of the template. However, since not all possible contexts are present in the image, some nodes will never be constructed and, therefore, memory will be allocated only for existing pixel combinations. For the case of color image (see Figure 6), the construction of the tree proceeds in the same manner as in the case of binary image, expect that there can potentially be as many

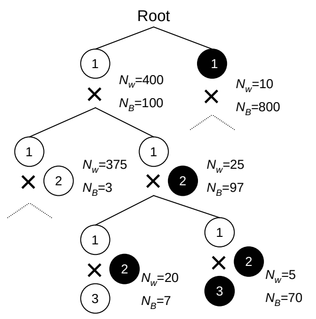


Figure 5. Construction of context tree for a binary image.

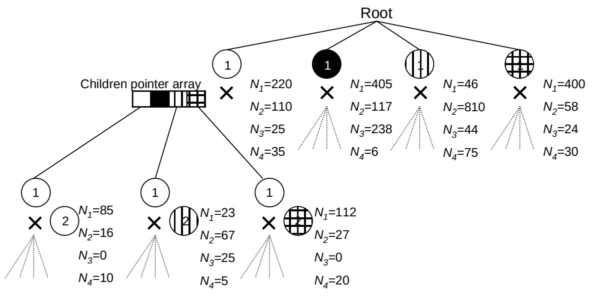


Figure 6. Construction of context tree for a color image.

child pointers and frequency counters as there are colors in the image. The frequency counters (components of the statistics vector) are denoted here as f_1, f_2, … f_k. With a large context size and large number of colors, however, it is unlikely that all colors will appear in a particular node. Our experiments show that for a 25-color image and 15-pixel template, the proportion of non-appearing children pointers and frequency counters can be up to 90% of all memory allocation if linear arrays were used. It is therefore essential to store the children pointers and the frequency vectors as linked lists to optimize memory consumption.

2.3 Pruning the Context Tree

Larger context size allows analyzing of larger structures of the images. However, larger patterns repeat less than smaller patterns and if the size is increased too much, most of the contexts will eventually appear only once or twice. Larger context size tends to make the distribution of the colors in a context more flat. Without enough statistics and clear statistical dominance of one color, the filter is unable to make reliable guess about whether given pixel is noisy, and by which color it could be replaced.

We overcome this drawback by using a pruning technique. Consider a node with the corresponding context c_P, and its children nodes c_1, …, c_k. Denote the number of times the context c_P appears in the image as $N(c_P)$. By definition of CT $N(c_P) = N(c_1) + \ldots + N(c_k)$. When a particular context does not appear frequently enough, it should not be used in filtering. We realize this by applying a simple pruning criterion: if the frequency of a given context falls below a predefined pruning threshold ($\exists\, i : N(c_i) \leq \text{Treshold}$), the corresponding node is pruned out from the context tree.

By definition of CT, all pixels that appear in a child context c_i appear also in their parent context c_p: $\forall \;\; I(x,y) \in c_i$ holds $I(x,y) \in c_P$. When the child context c_i is pruned, traversal in the tree will stop on its parent node, which by definition appears more frequently (or equally frequent in case of only one child) as its child context. The use of pruning criterion guarantees that every context appears in the image frequently enough to be a valid criterion of filtering.

Empirical results support the usefulness of the pruning. Popularity of contexts of size 20 in a sample test image is illustrated in Figure 7. The histogram shows that without pruning most of the contexts (118941) appear only once or twice in the image, and majority of the remaining contexts (21253 + 8971) less frequently than 8 times. Only 6 % of the contexts (about 10000 out of 150000) appear more than 10 times. This means that most of the contexts have too sparse distribution in order to be used for reliable filtering.

Figure 8 illustrates how many pixels are actually filtered in these contexts (filtering with probability threshold 20 %). The less populated contexts (appearing less frequently than 8 times) do not make significant contribution to the filtering. The effect of the pruning is demonstrated in Figure 9 and Figure 10. From Figure 9 one can see that no contexts appearing less than 8 times remain in the tree and Figure 10 shows that the contexts of smaller sizes significantly increase their contribution to the filtering.

3. Noise Models

3.1 Displacement noise

Typically, the map image obtained from a digital scanner is corrupted with specific type of noise. In order to reduce the influence of acquisition device as well as to decrease overall redundancy, that image usually goes through color quantization process. Though pixels of uniform areas are quantized well and are mapped to the same color values, border pixels can be easily mapped to a closer but different color value corrupting the contours of the objects. We refer this kind of noise as *displacement noise*.

We model this type of noise by considering a probability of misplacing the current pixel in a local 3×3 neighborhood. Consider the source image *Source*; the noisy image *Dest* is modeled as follows:

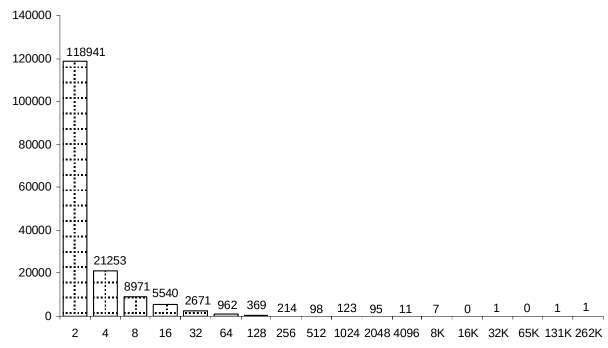


Figure 7. Histogram of contexts popularity for a sample image. The graph represents the number of unique contexts (Y axis) appearing in the image within a given frequency (X axis).

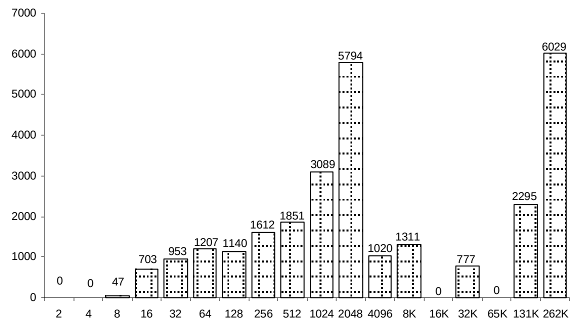


Figure 8. The amount of filtered pixels (Y axis) according to the popularity of the context (X axis) for the statistics of Figure 7.

```
For each Dest(x,y) do
  If rand() < Treshold Then
// do misplacing
    DirX = rand(-1,0,+1)
    DirY = rand(-1,0,+1)
  Dest(x,y)=Source(x+DirX,y+DirY)
  Else
    Dest(x,y) = Source(x,y)
  End If
End For
```

3.2 Impulsive noise

Impulsive noise typically originates from noisy transmitting channels of acquisition devices randomly affecting whole image independently of the region. When the noise level is high, color quantization maps pixels to wrong colors independently of the location of the pixel, and noisy pixels can appear anywhere in the image and can be of any color available in the color palette. We refer this noise as *impulsive noise.* Consider the source image *Source*; the noisy image *Dest* is modeled as follows:

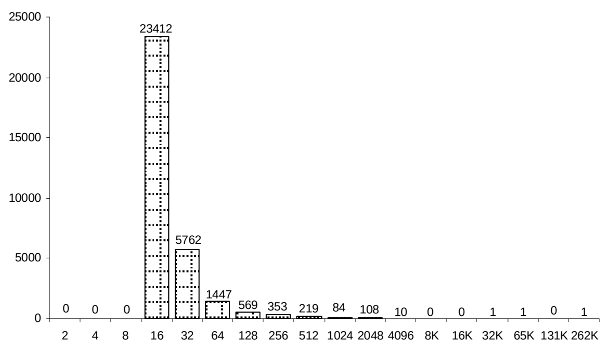


Figure 9. Histogram of the contexts popularity after pruning out contexts appearing less frequently than 8 times.

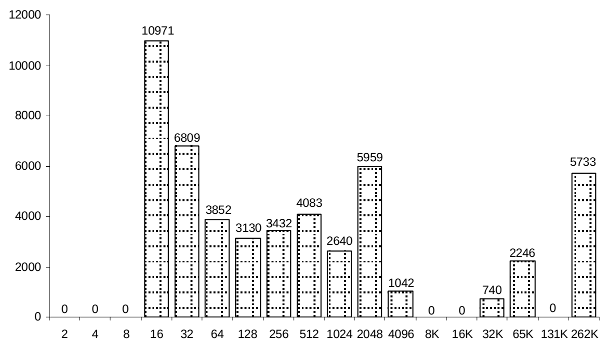


Figure 10. The amount of filtered pixels (Y axis) according to the popularity of the context (X axis) for the statistics of 9.

```
For each Dest(x,y) do
  If rand() < Treshold Then
Dest(x,y)=rnd(1,…,NumberOfColors)
  Else
    Dest(x,y) = Source(x,y)
  End If
End For
```

4. Experiments

We evaluate the proposed Context Tree filter (referred as CT) on a set of six map images chosen from *Finnish National Land Survey* database [15]. Two of them (images #1 and #4) are topographic and the rest are road maps. The images are of different spatial resolution and some of them (images #5 and #6) are affected by quantization noise. In addition to this, we corrupt all images with the noise of two types as described in Section 3.

4.1 Objective evaluation

The proposed filter (CT) is applied with context size 20, probability threshold level 5% and pruning threshold of 128. We compare CT with vector median (VM) [7], adaptive vector median (AVM) [8], morphological (MM) [4] and PGA [9] filters. The efficiency of the filters is evaluated using mean

Table 3. The efficiency of MM, VM, AVM and CT filters measured as Δ*E* distance to the original image for 20% content-dependent (CD) and 5% impulsive noise (I).

	Image 1		Image 2		Image 3		Image 4		Image 5		Image 6	
	CD	I	CD	I	CD	I	CD	I	CD	I	CD	I
MM	23.52	24.37	29.66	30.28	27.75	28.33	14.10	14.48	4.54	8.68	30.45	31.11
VM	3.16	2.51	8.50	7.73	8.58	7.37	3.27	2.46	1.99	1.66	7.81	6.67
AVM	2.51	1.70	4.60	2.46	5.05	3.12	2.18	1.18	1.33	1.15	5.07	3.10
PGA	2.51	1.50	5.48	3.71	5.76	3.79	2.24	1.32	1.75	1.56	5.90	4.02
CT	2.14	0.89	3.95	2.89	3.96	2.44	1.70	0.94	1.19	1.18	3.86	2.94

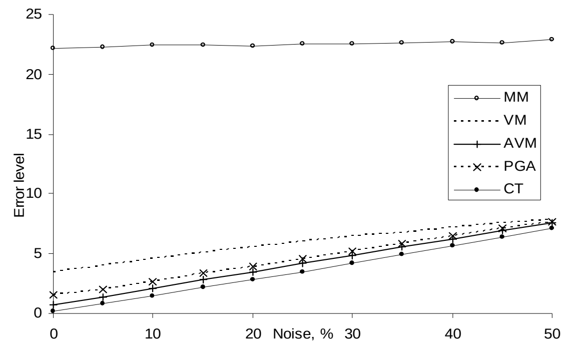


Figure 11. Efficiency of MM, VM, AVM, PGA and CT filters for content-dependent noise.

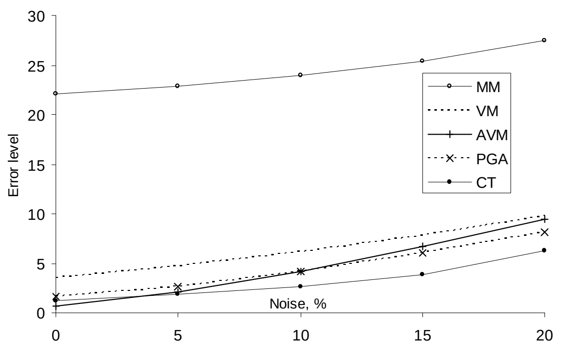


Figure 12. Efficiency of MM, VM, AVM, PGA and CT filters for impulsive noise.

color distance Δ*E* between the original (noiseless) and the filtered images defined by

$$\Delta E = \frac{1}{N}\sum \Delta E^*_{ab}$$

as the normalized sum over all image pixels, where ΔE^*_{ab} is the Euclidean distance between the two color samples in *L*a*b** (CIELAB) uniform color space [16] and is measured as

$$\Delta E^*_{ab} = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2}\ .$$

However, objective distance measure cannot be considered completely relevant for evaluation of the performance because pixelwise measurement does not represent the visual quality. For example when thin and detailed structures are filtered out, this does but it is clearly visible and it corrupts the semantic structures in the map. We therefore present also visual examples of filtered map for subjective evaluation in order to emphasize the ability to preserve repetitive patterns independent of their size.

For content-dependent noise we vary the noise level from 5% to 50% with step of 5%. The results are illustrated in Figure 11. One can see that the proposed filter provides better objective results for all nose levels. On average, the filter outperforms its closest competitor (AVM) by 15%. For impulsive noise we vary the noise level from 5 to 20% with step of 5%; the results are illustrated in Figure 12. The proposed filter outperforms AVM for noise levels higher than 5% noise. On average, CT outperforms AVM up to 30%. Table 3 summarizes the objective measurements for all filters for 20% content-dependent (CD) and for 5% impulsive (I) noise. The measurements are averages over the test set.

4.2 Subjective evaluation

Visual comparisons are presented in Figure 13 for three sample image fragments for 20% content-dependent (CD) and 5% impulsive (I) noise. The VM and AVM filters tend to preserve edges with no blurring. However, thin details of the original data are extensively filtered out since the filters are based on quantitative domination which underlies the median concept. The MM filter is a generalization of gray-scale morphological filter to a color space, and it is based on qualitative dominance. The generalization is considered using *reduced ordering* technique, when an order relation is defined on a vector space by reducing a multivariate object to a single value. For MM filter this order relation is based on a luminance of the color sample [4]. In this way the filter assumes that brighter colors 'dominate' the darker or vice versa. Also, the structuring element defining the operation of the filter is fixed and therefore unable to perform relevant filtering in different areas of the map which have very different structure. All this makes MM filter to perform worst on the selected imagery both by the objective as well as by the subjective comparisons.

The PGA filter performs rather well on impulsive noise. Although some impulses are still visible after one iteration of the filter, they will be removed after

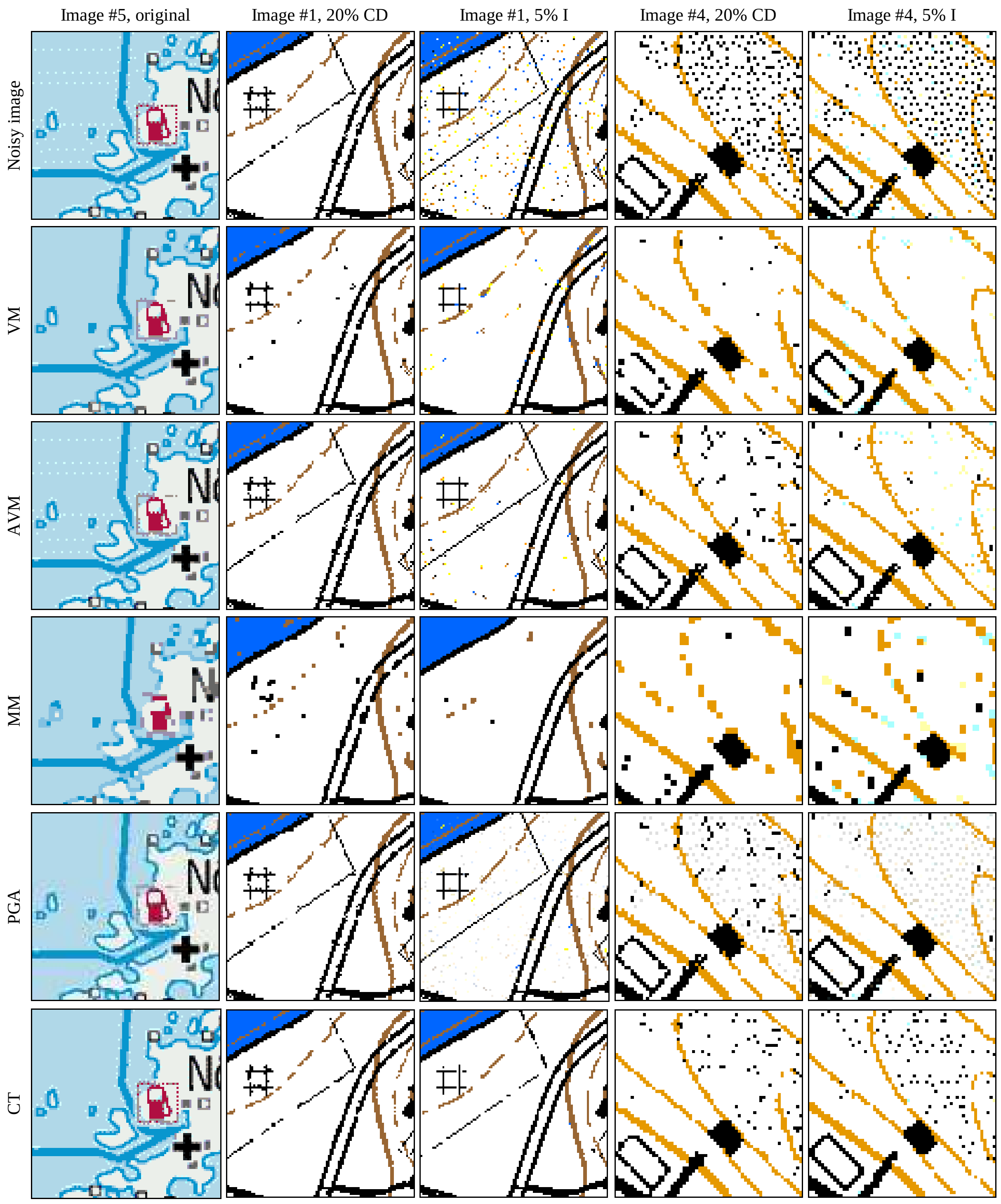


Figure 13. Visual comparison of the competitive filters.

few iterations. However, PGA mostly does not filter the content-dependent noise. This happens because, by its definition, peer group is formed of the neighbor pixels whose color is closest to the processed pixel. In case of content dependent noise, noisy pixels have pixels of the similar (or exactly the same) color in their neighborhood, which makes the peer group averaging ineffective.

In contrary with the competitors, the proposed CT filter deals with statistical domination instead of quantitative or qualitative domination, or distance-based grouping. The filter considers a local pattern to be preserved if it is repeated in the image frequently enough. However, irregular areas (the dotted area in the third example) or patterns not repeated frequently enough are filtered out. This property makes the proposed filter sensitive to the original image data. On the other hand, following the statistical assumptions, the filter is able to restore corrupted structures such as smooth distorted lines and borders. The major condition for the filter to be effective is the statistical consistency of the image; it is therefore mostly suitable for indexed-color palette images and images consisting of computer-generated graphics.

4.3 Processing time

We measure the processing time of the proposed filter (CT) for four selected content-dependent noise levels. The measurement is taken as the average over the test set, and the results are summarized in Table 4. The MM and PGA filters are implemented in Matlab and presented the worst performance, which however originates mostly from the chosen implementation environment. The proposed filer is computationally expensive and its performance depends also on how complicated are the structures in the image.

Table 4. Processing time of the filters under evaluation.

Filter	CT, C++	VM, C++	AVM, C++	MM, Matlab	PGA, Matlab
Time, sec.	15.80	1.34	2.29	20.09	74.14

5. Conclusion

We proposed a statistical filter based on a context tree modeling. The proposed filter is based on a local probability estimation followed by a thresholding replacing less probable patterns with the most probable ones. The filter aims at preserving the repetitive structures of the image, which is an essential property for raster map images. The filter is implemented using a memory efficient management of context tree modeling allowing larger local neighborhood and color depth to be utilized. The size of the context template is dynamically optimized by considering a simple and efficient tree pruning technique.

The performance is compared to vector median, adaptive vector median, color morphological and peer group averaging edge-preserving non-linear filters. The experiments show that the proposed filter outperforms these competitors both in objective and subjective comparisons.

The proposed filter, however, has some limitations of its applicability caused by extensive memory consumption of the algorithm. The filter is considered to be practical for color-indexed palette images when the number of colors is less than or equal to 256, but does not generalize well to true-color images as such. Larger irregular patterns are also not captured very well in case of high noise levels. Nevertheless, the main idea of statistical modeling of repeated structures is more general than relying only statistics within a local neighborhood as done in morphological and peer group filtering.

6. References

[1] Gonzalez, R. C., Woods, R. E., *Digital Image Processing*, 2nd ed., Prentice Hall, 2002.

[2] R. Lukac, B. Smolka, K. Martin, K.N. Plataniotis, and A.N. Venetsanopoulos, "Vector filtering for color imaging", *IEEE Signal Processing Magazine*, vol. 22 (1)74-86, January 2005.

[3] R. Lukac, K.N. Plataniotis, "A taxonomy of color image filtering and enhancement solutions", *Advances in Imaging and Electron Physics*, (ed.) P.W. Hawkes, Elsevier/Academic Press, vol. 140, pp.187-264, June 2006.

[4] M.L. Comer, E. J. Delp, Morphological operations for color image processing, *Journal of Electronic Imaging* 8(3), 279-289, July 1999.

[5] K. N. Plataniotis, D. Androutsos, and A. N. Venetsanopoulos, "Color image filters: The vector directional approach," *Optical Engineering*, vol. 36, no. 9, pp. 2375–2383, 1997.

[6] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted median filters: a tutorial," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, issue 3, pp. 157-192, Mar. 1996.

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

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

[9] C. Kenney, Y. Deng, B.S. Manjunath, G. Hewer, "Peer group image enhancement", *IEEE Trans. on Image Processing*, vol. 10 (2), 326-334, 2001.

[10] P. Kopylov, P. Fränti, "Filtering of color map images by context tree modeling", *IEEE Int. Conf. on Image Processing (ICIP'04)*, vol. 1, 267-270, October 2004.

[11] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdru, and M. Weinberger, "Universal discrete denoising: Known channel," *IEEE Trans. Inform. Theory*, 51(1):5-28, Jan 2005.

[12] E. Ordentlich, M. Weinberger, and T. Weissman, "Multi-directional context sets with applications to universal denoising and compression," *Proceedings of IEEE Intern. Sympos. Inform. Theory*, 1270-1274, Sep 2005.

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

[14] A. Akimov, A. Kolesnikov and P. Fränti, "Lossless compression of color map images by context tree modeling", *IEEE Trans. on Image Processing*, vol. 16 (1), pp. 114-120, 2007.

[15] National Land Survey of Finland, (http://www.nls.fi).

[16] CIE, Colorimetry, *CIE Pub.* No. 15.2, Centr. Bureau CIE, Vienna, Austria, 1986.

Publication P5

Podlasov A., A. Kolesnikov, Fränti P., Lossy compression of map images, *17th International Conference on Computer Graphics and Vision (GraphiCon'07)*, Moscow, Russia, pp. 79-83, June 2007.

Lossy Compression of Scanned Map Images

Alexey Podlasov, Alexander Kolesnikov and Pasi Fränti
Speech & Image Processing Unit
Department of Computer Science and Statistics
University of Joensuu, Joensuu, Finland
{apodla, koles, franti}@cs.joensuu.fi

Abstract

An algorithm for lossy compression of scanned map images is proposed. The algorithm is based on color quantization, efficient statistical context tree modeling and arithmetic coding. The rate-distortion performance is evaluated on a set of scanned maps and compared to JPEG2000 lossy compression algorithm, and to ECW, which is a commercially available solution for compression of satellite and aerial images. The proposed algorithm outperforms these competitors in rate-distortion sense for the most part of the operational rate-distortion curve.

Keywords: *Digital map images, lossy image compression, context modeling, color quantization.*

1. INTRODUCTION

Nowadays, digital *Geographical Information Systems* (GIS) became more and more popular among all kind of users. Though at the beginning the price of mobile positioning (e.g. GPS) and processing devices restricted the use of electronic navigation to military or corporate applications, today we are facing the extensive growth of this industry in personal user sector. Recent progress in low-cost mobile hardware and, especially, in low-cost memory made computer-aided navigation possible in personal car on a road trip, as well as in your hand while trekking.

However, raster map image converted from the vector database is not always the case. It is still common that, when needed, geographical information could only be found on the paper printed map. Similar case is the digitization and storage of rare maps, which are too fragile and valuable to be used as such. Though this kind of paper-printed material could be easily digitized and integrated into computerized navigation or archive system, there are still some specific problems. The main problem of raster maps is their storage size. Paper printed material of approximately A4 size scanned with 300dpi in true-color results in about 2500×3500 pixel image requiring 24 bits per pixel, which is 25 megabytes per image. The number of unique colors can vary from hundreds of thousands to several millions depending on the type of the map. For example in our experiments we experienced up to 700 000 unique colors in topographic map images. Standard lossless compression techniques such as PNG, GIF or TIFF are able to provide about 1.2:1 compression ratio, which is not enough for effective transmission of the image to the user's device and processing it there. Lossy compression is therefore needed.

There is a wide variety of standard multi-purpose lossy compressions techniques, as well as techniques developed specifically for compression of scanned material. Among the standard algorithms JPEG and JPEG2000 [13] are the most popular. Wavelet-based *Multiresolution Seamless Image Database* (MrSID) [1] by LizardTech is a patented commercial solution for storing large amounts of satellite and aerial images. It is applied for compression of scanned map imagery as well. Wavelet-based *Enhanced Compression Wavelet* (ECW) [3] format by ER Mapper is also a commercially available solution for GIS-based image compression. Well-known DjVu format [2] by LizardTech and AT&T is specially developed for storage of scanned imagery, especially books.

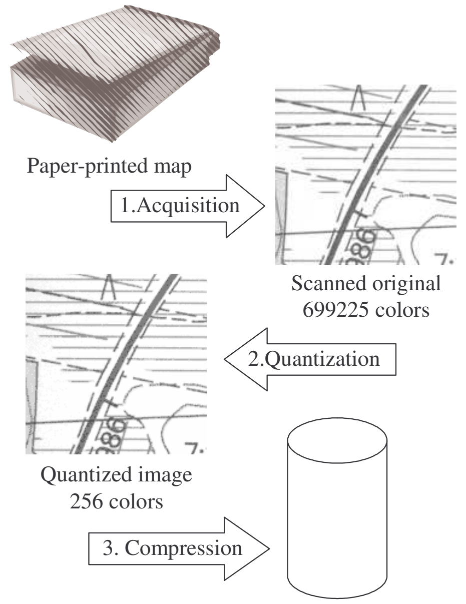


Figure 1: Overall scheme of the proposed compression

However, popular wavelet techniques have some disadvantages when used for compression of scanned maps. Scanned map combines the characteristics of both image classes: discrete-tone and continuous-tone. The image origin is artificial and, therefore, unlike photography, a map image contains of a small number of unique colors and lots of small-size detailed structures such as letters and signs, solid uniform areas such as waters, forests, fields, sharp edges and almost no gradient color gradation. Besides this, typical map image contains a lot of repetitive patterns and textures. This comes as from the map itself, e.g. areas like swamps or sands are usually represented by textures. Besides that when map is printed on the paper, color gradation is usually obtained by dithering the available inks forming uniformly textured areas. This dithering is acquired by the scanner and appears in scanned images as a repetitive pattern of color dots.

Lossy compression based on wavelet transform significantly smoothes the edges of the image and destroys thin well-structured areas, such as textures. When higher level of quality is desired, techniques like JPEG2000 or ECW loose efficiency in compression performance since wavelet transform requires more bits to represent high frequencies of the sharp edges of the image. On the other hand, the compression algorithms optimized for artificial graphics, such as *Piecewise-constant Image Model* (PWC) [4] or *Embedded Image-Domain Adaptive Compression* (EIDAC) [5], are not effective since these algorithms are designed to deal with computer-generated imagery. However, scanned image is affected with noise imposed by the acquisition device – a scanner or a camera. The inconsistency in illumination, sensor's perception and other factors results in blurred edges, and significant increase in the number of colors and intensity gradation. This makes lossless algorithms inefficient in providing necessary compression ratio.

In this work, we propose an alternative lossy compression technique for scanned map images based on color quantization and statistical lossless compression. The overall compression system under consideration is outlined in Figure 1. Firstly, the paper-printed map is digitized with *e.g.* flatbed scanner. The resulting image, referred further as the *original image*, is the input of the proposed compression algorithm. The proposed algorithm consists of two stages: color quantization and lossless compression. In quantization stage, the number of colors of the original image is reduced. This stage is a lossy part of the algorithm and the degradation of the image *i.e.* the information loss occurs here. The resulting image with reduced number of colors is referred further as the *quantized image*. In the second stage, the quantized image is compressed by the lossless image compression algorithm.

In general, the proposed scheme does not require any specific quantizer and compressor to be used. Though a big variety of approaches can be considered for this task, we consider the using of simple, fast *Median Cut* (MC) quantizer [11], which is a classical approach widely used in image processing applications and is able to process map images in reasonable time.

Among the variety of lossless compression algorithms which could be considered to be used to perform the compression stage one should mention that all we deal with color map images when the most of efficient lossless compression techniques are aimed at halftone imagery. Separating the color planes with following halftone-oriented compression typically means sacrifice in compression performance since color components are usually highly correlated. Besides that, linear prediction, which is a standard tool for continuous-tone lossless compression algorithms such as JPEG-LS or CALIC [7][8] fails on map images since the value of the current pixel depends on its neighborhood configuration, not on the local intensity gradation.

This motivates us to choose for compression stage context-based statistical *Generalized Context Tree* (GCT) compression algorithm which has been recently proposed from compression of raster map images [6] and presented compression efficiency surely outperforming its closest competitor PWC. The algorithm, however, is designed to compress raster maps which are directly generated from the vector sources. This means that these images contain low amount of colors (only the colors of the original map) and no blurring or noise. However, the original GCT is inapplicable to the scanned map sources. Together with technical difficulties like memory consumption and great processing time there is a fundamental problem. The great number of colors in the scanned image destroys statistical dependency within the image and GCT approach is not applicable for the same reason as it is

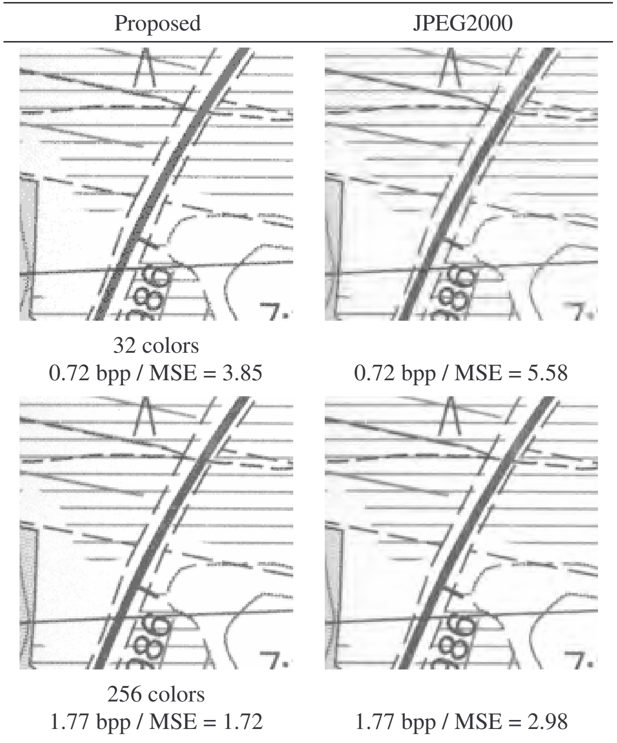


Figure 2: Visual comparison of the proposed and JPEG2000 algorithms**.**

not applicable to photographic imagery. In order to spread the efficiency of GCT to scanned imagery one needs color quantization to be involved to revive the local statistical dependencies featuring map imagery and determining the following use of GCT. Besides that some improvements to the original GCT must be considered since straightforward application would encounter difficulties with processing time and memory consumption. In this work by taking the properties of the imagery into account we successfully apply GCT for up to 256 color images.

The visual comparison of the proposed algorithm and standard JPEG2000 applying to scanned map image is presented in Figure 2. The upper and lower rows represent lower and higher quality levels respectively. The algorithms are applied to compress the test image with the same compression ratio – 0.72 bpp for low quality and 1.77 bpp for higher quality. One can see that for equal bitrate the proposed algorithm provides less degradation according to MSE distance. For lower quality level the proposed algorithm preserves edges and does not employ smoothing as JPEG2000. The performance of the proposed algorithm is evaluated on a set of scanned topographic maps and compared to JPEG2000 – standard lossy compressor and ECW – a commercially available compression system. Also in order to prove the efficiency of GCT compressor we consider the comparison with 'trivial approach' where color quantized image is compressed with PWC – an algorithm for compression of computer generated palette images (referred also as *simple images*). We denote this approach as "MC+PWC" *i.e.* median cut plus PWC.

The rest of the paper is organized as follows: the proposed compression algorithm is described in Section 2; experiments are presented in Section 3, and conclusions are drawn in Section 4.

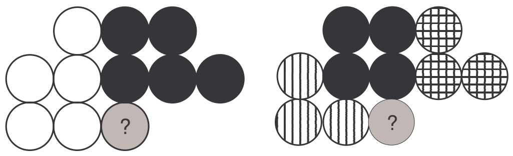


Figure 3: Sample contexts: binary (left) and generalized (right). Pixel which probability is estimated is marked with "?" sign**.**

Future development of the proposed technique is outlined in Section 5.

2. PROPOSED ALGORITHM

We propose two-stage algorithm for lossy compression of scanned map images: firstly, the number of colors in the image is reduced by median cut color quantization; then the resulting image is compressed losslessly by improved GCT lossless compression algorithm.

2.1 Median cut quantization

Median cut algorithm is a very popular method for color quantization widely used in image processing practice originally published in [11]. It is relatively simple both conceptually and computationally still providing good results.

The conceptual idea behind the algorithm is to design a color palette in such a way that each color would represent approximately the same number of pixels of the input image. Firstly, the algorithm computes the color histogram of the image. Typically, the image is pre-quantized with uniform quantizer since 24-bit color histogram would be difficult to handle. Then, from the color histogram one considers a box enclosing the colors of the image. The idea of median cut is to split the box recursively until the desired number of palette colors is reached. At each step of the algorithm, the box containing largest number of pixels is split along the coordinate that spans the largest range. The split is made at the median point so that approximately equal number of pixels falls into sub-boxes.

2.2 GCT compression

Statistical context-based modeling is a well-known tool in image compression and it is widely used in various compression applications. The general idea is to exploit local dependencies among pixels. In typical image, the knowledge about the neighborhood of the unknown pixel significantly improves its probability estimation, e.g. for most of documents, the probability of the current pixel to be white is very high when all its neighbors white. The neighborhood configuration is called a *context* and is defined by the context template. Figure 3, left picture illustrates sample binary contest, where background pixels are drawn as white and foreground as black. The estimated conditional probabilities are usually coded by arithmetic coder [9], as has been done in the very first standard for encoding of bi-level images – JBIG [12].

However, every context-based approach faces two major problems: memory consumption and *context dilution*. The information about estimated probabilities needs to be stored for every context. In case when every possible context is expected to appear in the image this number grows exponentially. For example, for 10-pixel context on a binary alphabet (JBIG) 2^{10} context configurations are possible. In case when K intensity gradations are expected, 10-pixel template results in K^{10} contexts, which is a huge number even for gray-scale images. The problem can be partially solved using the *Context Tree* (CT) modeling originally proposed by Rissanen [10]. This approach organizes the storing of probability estimations in a tree structure. In this way, only the information about the contexts that are really present in the image are stored, which significantly reduces memory consumption.

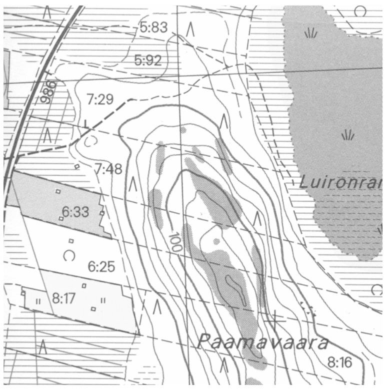
Context dilution problem is of different nature and cannot be solved only with optimized memory allocation. The problem is that larger context template does not always provide the increase in compression performance. With increasing of size, particular contexts do not appear frequently enough in the image for probability to be estimated accurately. Incorrect estimation degrades the efficiency of the entropy coder, and therefore, the compression efficiency. In CT modeling, this problem is solved by applying so called tree pruning technique. The idea is that if the parent node (smaller context) provides better compression than its children (larger context), then the children nodes of the tree are pruned and the parent is used instead for the probability estimation. The efficiency of compression is estimated by the entropy of the model. CT modeling is used mostly in simplified binary case where only two types of pixels are possible.

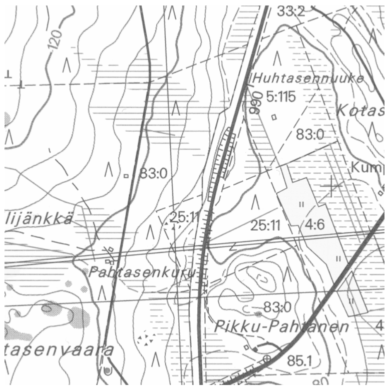
Generalized Context Tree (GCT) generalizes CT model into more color case, sample context is illustrated in Figure 3 (right), where different colors of context pixels are illustrated with texture. Pruning is performed by *steepest descent search* algorithm resulting in sub-optimal tree configuration which, however, is very close to the best one obtained by full search. At the moment, GCT compression presents the best performance for lossless compression of computer-generated raster map images [6].

First, we considered a fast pre-pruning of the tree for GCT. In our experiments we discovered that the most part of the tree is not filled with representative statistics since the most of the contexts do not appear in the image frequently enough but just ones or twice. Though these contexts are pruned out by steepest descent search algorithm, it is computationally expensive and the vast of total processing time is spent on it. Therefore we considered a simple threshold-based pre-pruning. The idea is that the node (and the represented context) is pruned in case that its occurrence number falls below the predefined threshold. The surviving nodes are then processed by standard pruning algorithm.

Then, we optimized the memory allocation for tree nodes. We discovered that in case when storage of pixel counters in tree nodes is implemented as an array, about 90% of array elements are not used. This originates from the fact that in many-color images the actual variety of colors appearing in a particular context is small since typically with increase of colors in the image contexts become less frequent. We consider implementing the storage of pixel counters as a linked list. Basing on the understanding of imagery features, this simple technical improvement dramatically increases the number of colors which GCT compressor is able to process same time making context tree faster to traverse.

The effect of optimization is illustrated in Table 1 for sample 1250×1250 image of 42 colors. Rows of the table represent memory consumption and processing time for original GCT, GCT with optimized memory allocation and for GCT with optimized memory allocation and pre-pruning. For images with more colors the effect is even more significant. In general, the use of these simple and effective optimization techniques made the algorithm applicable for 256-color 3000×3000 pixel images and 20-pixel


Topo1
2461×3345
699925 colors


Topo2
2359×3497
703965 colors

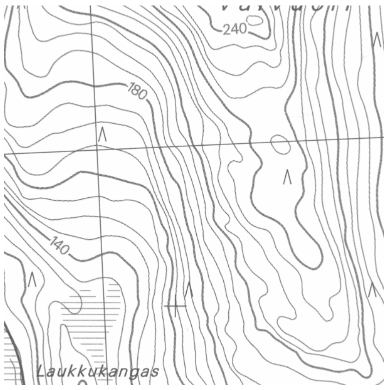

Topo3
2291×3484
669724 colors

Figure 4: Samples of the test set images.

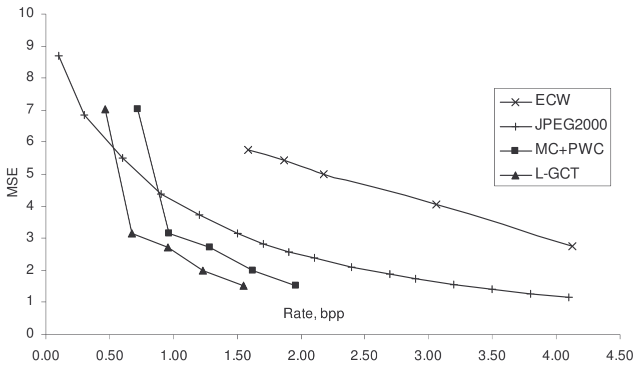


Figure 5: The compression performance of the proposed algorithm (L-GCT) and its competitors.

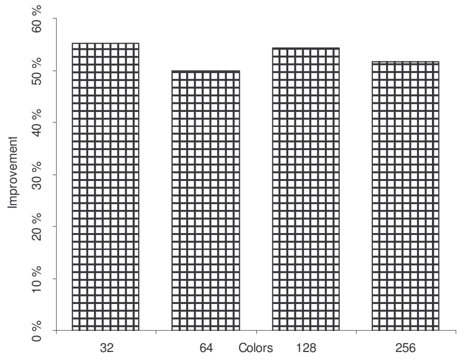


Figure 6: The relative compression improvement provided by L-GCT comparing to JPEG2000.

Table 1: The effect of memory optimization and pre-pruning

	Memory, MB	Time, sec
Original GCT	128	334
Optimized memory	30	326
Opt. memory + pre-pruning	30	72

context on a personal computer with 1G operative memory. Note that no optimization would deal with 256^{20} possible context configurations.

3. EXPERIMENTS

We compare the performance of the proposed algorithm, referred further as Lossy Generalized Context Tree Modeling (L-GCT), with JPEG2000 [13], which is the recent standard for lossy image compression, and with ECW compressor [3] used widely in GIS solutions. For a test set we consider three scanned topographic maps of Finland: topo1, topo2 and topo3. Raster images are acquired by a flatbed scanner at 300 dpi. Samples and image dimensions are illustrated in Figure 4. The experiments are performed on P4-3GHz 1GB memory computer.

We measure the distortion caused by the lossy compression algorithm as MSE distance in $L^*a^*b^*$ color space [14]. The distance is measured from the degraded image to the scanned original. The operational rate-distortion function for JPEG2000 is estimated by considering 16 quality levels varying bit rate approximately from 0.1 to 4 bpp, and respectively, MSE distortion from 8.69 to 1.16. For the proposed compressor we consider 5 quality levels by defining the number of colors in the image as 256, 128, 64, 32 and 16. Images of 256-color are the practical limit of the proposed algorithm. In our experiments for L-GCT, we use 20-pixel context modeling with pre-pruning threshold level set to 32. The compression results – bit rate and MSE distance are measured as the average over the test set.

The compression performance of L-GCT and its competitors is illustrated in Figure 5. The proposed algorithm outperforms its competitors starting from 32-color images. Better performance is presented for the rest of quality levels up to 256-color images. The relative improvement over JPEG2000 with respect to the similar objective quality level is illustrated in Figure 6. The improvement of the proposed algorithm varies around 50% for images of 32 to 256 colors. The comparison with 'trivial approach' MC+PWC proved that GCT provides better lossless compression. ECW in our experiments performs worse than JPEG2000.

The processing time required by the proposed algorithm depending on the quality of the image is represented in Table 2. One can see that the most of the time is spent on the construction of the context tree. Encoding and decoding times are almost equal and are much smaller than the tree construction time.

As a disadvantage of the proposed algorithm one can still consider its compression time and memory consumption. For example for highest quality levels the compression of single image takes about one and a half hour. This restricts the use of the proposed approach in real-time applications, though the offline archiving is practical since decompression does not require significant time or memory.

Table 2: L-GCT processing time (sec) depending on the amount of color in the image.

	16	32	64	128	256
Tree constr.	204	333	591	1816	5021
Encoding	7	12	22	40	62
Decoding	11	16	28	49	71

4. CONCLUSIONS

We proposed a lossy compression algorithm for scanned map images. The algorithm is based on color quantization, which is a lossy part, and context tree modeling, which is a lossless compression technique. The quantization is performed by median cut algorithm. The compression is done by modified Generalized Context Tree lossless compression algorithm, for which pre-pruning and optimized memory management techniques are considered, basing on the features of the target imagery.

The rate-distortion performance of the proposed algorithm is evaluated on a set of scanned topographic maps and compared to JPEG2000 and ECW wavelet-based lossy compressors. JPEG2000 is a recent standard for common lossy image compression and ECW is a commercial proprietary format for aerial and satellite image storage used also for the compression of scanned imagery. Also, in order to prove the efficiency of GCT we compared the proposed algorithm to the ‘trivial approach’ where the compression is performed by standard PWC compressor.

The proposed algorithm surely outperforms the competitors. For JPEG2000 the advantage is about 50% in average by the provided rate for similar MSE distortion level. However, one can consider processing time and memory consumption as the drawbacks of the proposed technique.

5. FUTURE WORK

We believe that the potential of the algorithm needs to be investigated in more details. Such application areas could be considered as lossy compression of simple graphics – architectural schemes, engineering drawings; different types of scanned map images – city plans, navigational and atlas-type maps. The effect of different type of sensor could also be studied; for example, simple graphics obtained with a digital camera. The optimal choice of the quantization scheme is also an open question as well as the question of faster processing time of the algorithm.

6. REFERENCES

[1] LizardTech web site, http://www.lizardtech.com, accessed 18.3.2007.

[2] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, Y. Le Cun, “High Quality Document Image Compression with DjVu”. Journal of Electronic Imaging, vol. 7 (3), pp 410-425, SPIE, 1998.

[3] ER Mapper web site, http://www.ermapper.com/ecw/, accessed 18.3.2007.

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

[5] Y. Yoo, Y. Kwon, A. Ortega, “Embedded image-domain adaptive compression of simple images”, Conference record of the Thirty-Second Asilomar Conference on Signals, Systems and Computers, vol. 2, pp. 1256–1260, 1998.

[6] A. Akimov, A. Kolesnikov and P. Fränti, “Lossless compression of color map images by context tree modeling”, IEEE Trans. on Image Processing, vol. 16 (1), 2007.

[7] M. Weinberger, G. Seroussi, G. Shapiro, “The LOCO-I lossless image compression algorithm: principles and standartization into JPEG-LS”, IEEE Trans. on Image Processing, vol. 9 (8), pp. 1309–1324, August 2000.

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

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

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

[11] P. Heckbert, “Color image quantization for frame buffer display”, Comput. Graph. 16, pp. 297-307, 1982.

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

[13] D. Taubman, M. Marcellin, JPEG2000: Image Compression Fundamentals, Practice and Standards, Kluwer Academic Publishers, 2001.

[14] CIE, Colorimetry, CIE Pub. No. 15.2, Centr. Bureau CIE, Vienna, Austria, 1986.

About the authors

Alexey Podlasov received his MSc degree in applied mathematics from Saint-Petersburg state University, Russia, in 2002, and the MSc degree in computer science from the University of Joensuu, Finland, in 2004. Currently, he is a doctoral student in computer science in the University of Joensuu. His research topics include processing and compression of map images.

Alexander Kolesnikov received the M.Sc. degree in physics in 1976 from the Novosibirsk State University, U.S.S.R., and the Ph.D. degree in computer science in 2003 from the University of Joensuu, Joensuu, Finland. From 1976 to 2003, he was a Senior Research Fellow with the Institute of Automation and Electrometry, Russian Academy of Sciences, Novosibirsk, Russia. In 2003, he joined the Department of Computer Science, University of Joensuu. His main research areas are in signal and image processing, vector map processing, and compression.

Pasi Fränti received his MSc and PhD degrees in computer science in 1991 and 1994, respectively, from the University of Turku, Finland. From 1996 to 1999 he was a postdoctoral researcher of the Academy of Finland. Since 2000, he has been a professor in the University of Joensuu, Finland. His primary research interests are in image compression, clustering and speech technology.

Dissertations at the Department of Computer Science

Rask, Raimo. Automating Estimation of Software Size during the Requirements Specification Phase—Application of Albrecth's Function Point Analysis Within Structured Methods. Joensuun yliopiston luonnontieteellisiä julkaisuja, 28: University of Joensuu. Publications in Sciences, 28. 128 pp. Joensuu, 1992.

Ahonen, Jarmo. Modeling Physical Domains for Knowledge Based Systems. Joensuun yliopiston luonnontieteellisiä julkaisuja, 33: University of Joensuu. Publications in Sciences, 33. 127 pp. Joensuu, 1995.

Kopponen, Marja. CAI in CS. University of Joensuu, Computer Science, Dissertations 1. 97 pp. Joensuu, 1997.

Forsell, Martti. Implementation of Instruction-Level and Thread-Level Parallelism in Computers. University of Joensuu, Computer Science, Dissertations 2. 121 pp. Joensuu, 1997.

Juvaste, Simo. Modeling Parallel Shared Memory Computations. University of Joensuu, Computer Science, Dissertations 3. 190 pp. Joensuu, 1998.

Ageenko, Eugene. Context-based Compression of Binary Images. University of Joensuu, Computer Science, Dissertations 4. 111 pp. Joensuu, 2000.

Tukiainen, Markku. Developing a New Model of Spreadsheet Calculations: A Goals and Plans Approach. University of Joensuu, Computer Science, Dissertations 5. 151 pp. Joensuu, 2001.

Eriksson-Bique, Stephen. An Algebraic Theory of Multidimensional Arrays. University of Joensuu, Computer Science, Dissertations 6. 278 pp. Joensuu, 2002.

Kolesnikov, Alexander. Efficient Algorithms for Vectorization and Polygonal Approximation. University of Joensuu, Computer Science, Dissertations 7. 204 pp. Joensuu, 2003.

Kopylov, Pavel. Processing and Compression of Raster Map Images. University of Joensuu, Computer Science, Dissertations 8. 132 pp. Joensuu, 2004.

Virmajoki, Olli. Pairwise Nearest Neighbor Method Revisited. University of Joensuu, Computer Science, Dissertations 9. 164 pp. Joensuu, 2004.

Julkaisija	Joensuun yliopisto Tietojenkäsittely- ja tilastotieteen laitos
Publisher	University of Joensuu Department of Computer Science and Statistics
Vaihdot	Joensuun yliopiston kirjasto / Vaihdot PL 107, 80101 Joensuu Puh. 013-251 2677, fax 013-251 2691 e-mail: vaihdot@joensuu.fi
Exchanges	Joensuu University Library / Exchanges P.O. Box 107, FI-80101 Joensuu, FINLAND Tel. +358-13-251 2677, fax +358-13-251 2691 e-mail: vaihdot@joensuu.fi