Master's thesis in Computer Science, 173315, 15 cu

Morphological reconstruction of semantic layers in map images

Student: Alexey Podlasov

153329

apodla@cs.joensuu.fi

Supervisor: Dr. Eugene Ageenko

ageenko@cs.joensuu.fi

Department of Computer Science

University of Joensuu

December 2003

TABLE OF CONTENTS

Abstract

Digital spatial library is an electronic archive of geographic imagery data such as multi-layer map images. The real-time or mobile imaging application provides user with view of geographic map in the real time. The map image is retrieved from a remote server via network or served from the data base located on the user's mobile device. The main problem in digital spatial libraries is the huge storage size of the images, which affects equally storage and transmission performance. It has been shown that the best compression results for map images can be achieved if the images are decomposed into binary semantic layers, which are consequently compressed by the algorithm designed to handle binary data (e.g. JBIG). There is no problem if semantic layers forming the map image are originally available. However, when the map exists only as a color raster image, the semantic layers can be obtained only through the separation process. The separation leads to appearance of severe artifacts in places when one layer overlaps another. These artifacts affect statistical properties and consistency of the layers and result in degraded visual quality and compression performance.

In the current work, we design the technique to restore semantic layers of the map images by removing or suppressing the artifacts caused by layer separation process. We restrict our technique in such a way that when the restored layers will be again combined into a color map image, it will be identical to the original. Because of possible applications in the mobile devices with restricted computational and memory resource, we must also restrict ourselves with the complexity of the algorithms. Therefore we chose *Mathematical Morphology* as the base tool for the design of restoration technique.

The designed restoration technique provides better performance for the compression of reconstructed layers instead of corrupted ones with popular lossless compression algorithms (ITU Group 4, PNG, and JBIG). Besides that, reconstructed layers have good visual appearance, which is especially important in applications where some layers must be removed (or in opposite, extracted) from the map image.

1 Introduction

1.1 Digital Spatial Libraries

Real-time cartography imaging applications provide user with the view of geographic map for the area surrounding the user's location. The user's location can be obtained using GPS (*Global Positioning Service*) or MPS (*Mobile Positioning Service*) services. The geographical map image is obtained from *Digital Spatial Library* and transmitted via network to user's mobile device such as pocket computer (*PDA*) or a mobile phone. *Digital Spatial Library* is an electronic archive of geographic imagery data [F+95, S89].

The images forming the archive are the color (or grayscale) raster images or multi-layer map images. The multi-layer map images consist of the set of semantic layers, each containing the data with distinct semantic content, e.g. roads, elevation lines, state boundaries, water areas, etc. The layers are combined and displayed to the user as a generated color image, in which the data of each type is usually depicted using its own color. For example, let us consider the topographic images from the NLS topographic database, in particular basic map series 1:20,000 [NLS]. 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 poly-lines representing water areas and ways), *Fields* (solid polygonal regions), see Figure 1.

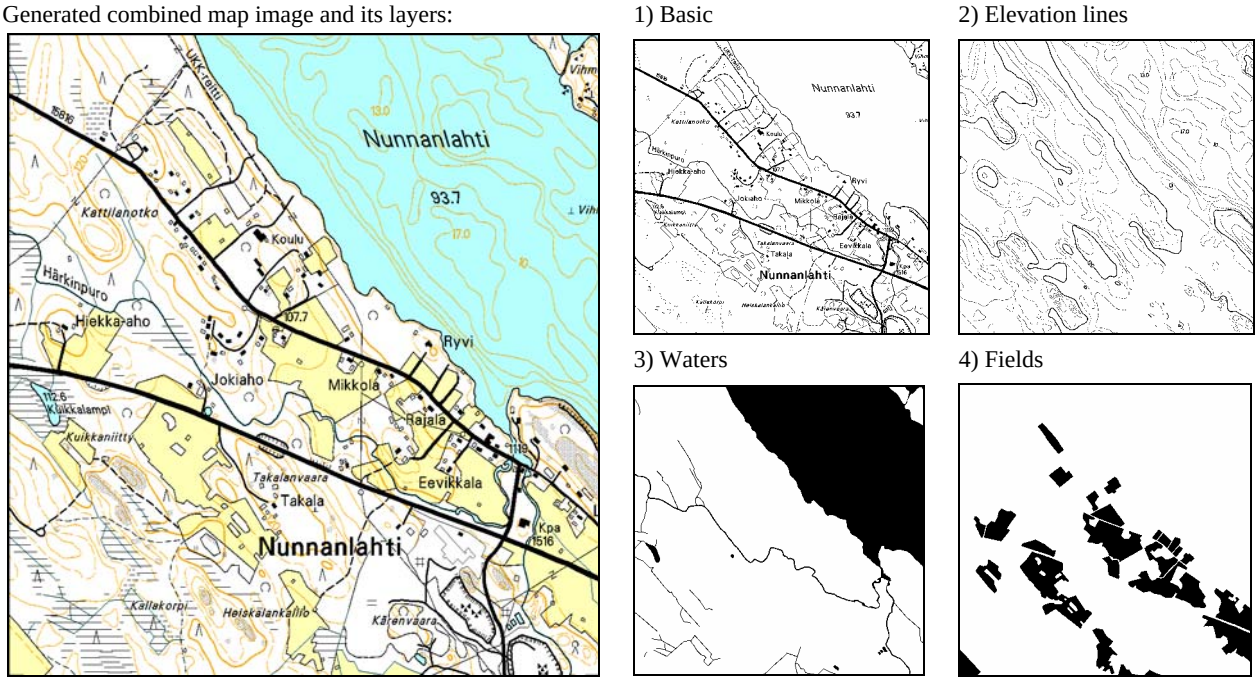


Figure 1: Illustration of a multi-layer map image from NLS Topographic database. The shown fragment has dimensions of 1000 × 1000 pixels.

Semantic layers can be generated from the vector database spawning large territories up to the entire country. Generated raster images are often easier to transmit and handle on a client side since vector database require significant computing resources. Raster images are also often used for digital publishing on CD ROM, or in the web. Even though the DSL may operate on vector database, the raster images are often served to the user.

The main problem of DSL is the huge storage size of the images. Especially it is apparent in applications requiring the use of mobile hardware such as mobile phones or pocket computers. For example, a single map sheet of 10×10 km² is represented by a single map image of 5000×5000 pixels. The image requires about 12 Mb of storage space. In comparison, the typical portable devices have usually only about 32 Mb of the storage space, which can be expanded by about 96 Mb. Even though during the last decade the technology demonstrated significant progress in the development of hardware, mobile devices are still hardly restricted in memory and computational resources. The necessity of compression for saving storage space is therefore obvious. It has been shown that the best compression results for the map images can be achieved if the images are decomposed into binary semantic layers, which are consequently compressed by the algorithm designed to handle binary data (e.g. JBIG) [AF00, FKA02].

Another problem resultant of large image sizes is the image transmission. Larger image size takes longer to transmit and/or require faster (expensive) transmission channels as well as appropriate hardware equipment. For example, 10 seconds transmission via GPRS on 45kb/sec transfers up to 54kB of image data, which is about 500x500 pixels map image in case of 4-layer maps and 1:20 compression rate (which is very optimistic). This data is sufficient to represent a map fragment that covers only 1 km² of territory.

1.2 Semantic layers and their corruption

In the case of multi-layer map images, when semantic data is available, the map images are stored as a set of binary layers each containing different semantic layer. The layers are represented as binary images, which are separately compressed and transmitted to the user, see Figure 2. The user application can reproduce map by plotting each layer in its own color overlapping each other in a given order. Layer separation allows utilizing more efficient context-base compression techniques which benefits in high compression performance. Separation also gives the ability to select specific layers at the time of viewing [FAKGB02, FKV02].

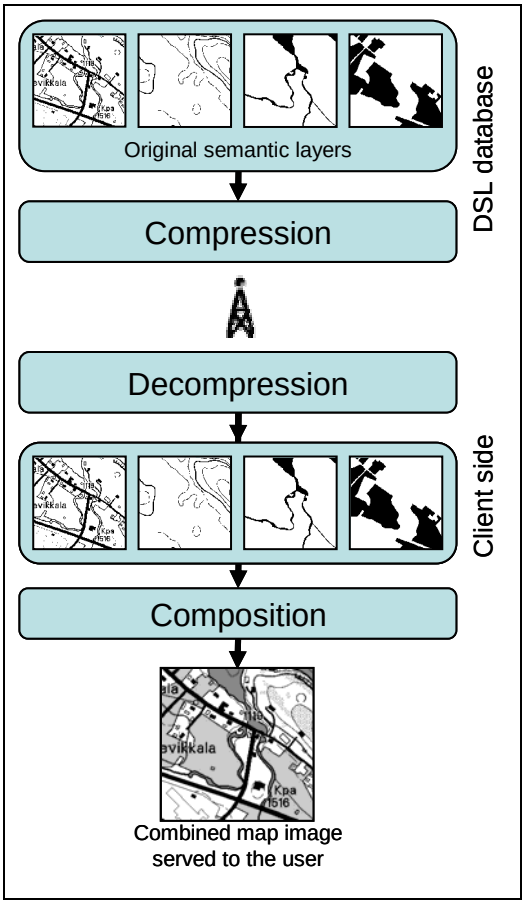


Figure 2: Operation on multi-layer map images.

Often though, the original vector or semantic data are not available. This is possible if the map image is digitized directly from the paper or received from the third party source. In such (still prevailing) situations we have only raster color image as the original map. The semantic layers must be obtained from the color image through the *color separation* process. The map image is divided into binary layers so that each layer represents one color in the original image [FKA02]. Figure 3 illustrates operation on digital map images using color separation. The part after the transmission is essentially the same as in Figure 2 and is therefore skipped for the sake of the space.

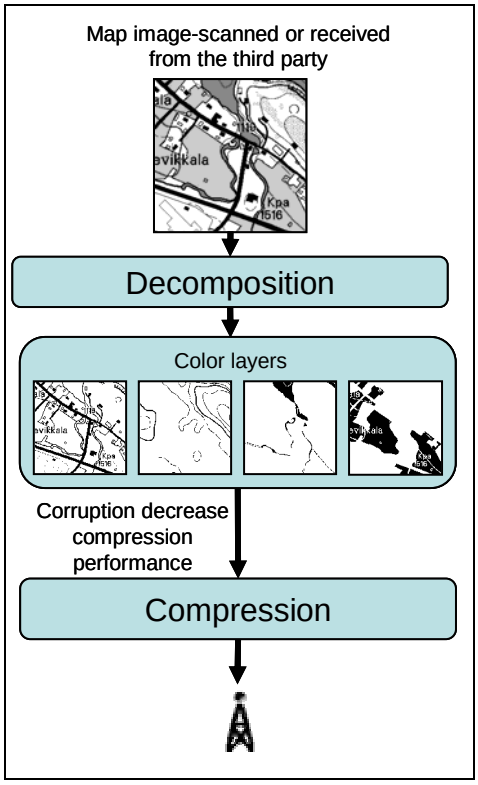


Figure 3: Operation on map images using color separation.

This color separation process however leads to appearance of severe artifacts in places when information in one layer overlaps another. These artifacts affect statistical properties and consistency of the layers and resulting in degraded visual quality and compression performance. A proper image restoration technique shall be therefore applied.

To illustrate the problem, let's look at a small fragment of a map, shown in Figure 4.

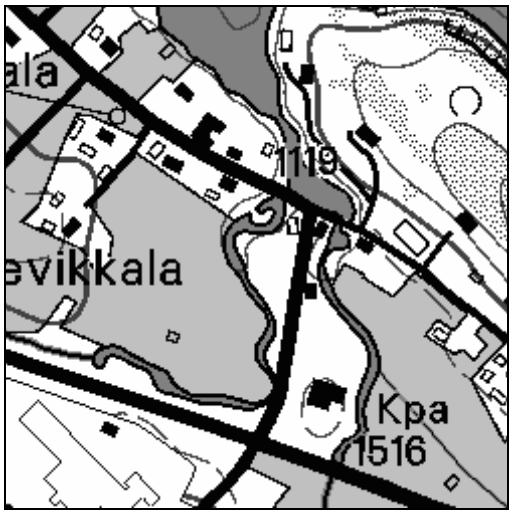


Figure 4: The fragment of multi-layer map image.

The map image could be easily separated into a stack of binary layers: roads, heights, waters, fields and background.

The Figure 5 shows that during separation overlapping layers produce artifacts on underlying layers and cause their corruption. The remains of letters over the fields or breaches on elevation lines are examples of such corruption.

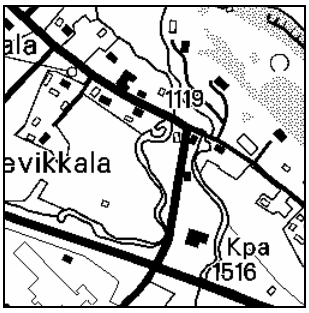
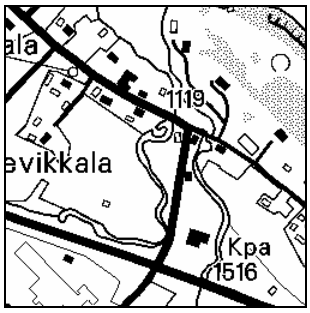
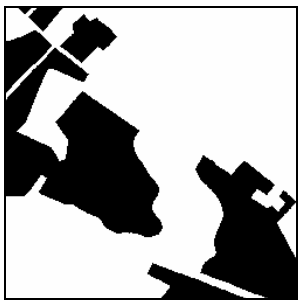
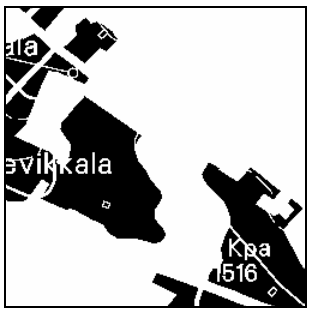
Original semantic layer	**Decomposed layer**
 Roads layer original	 Roads layer decomposed
 Elevation lines layer original	 Elevation lines layer decomposed
 Waters layer original	 Waters layer decomposed
 Fields layer original	 Fields layer decomposed

Figure 5: Original and corrupted layers.

Corruption of semantic layers increases the amount of noise in the layer and introduces irregularities into its statistical properties. This increases the entropy ratio for the image, and consequently reduces performance of compression algorithms.

Secondary problem arises in situations when some of the layers must be extracted or, opposite, removed from the image. Lower layers in hierarchy will suffer more from degradation as well as top-level layers will cause more degradation on under-laying imagery data if being removed.

Therefore, a kind of restoration or filtering technique must be developed for improving the compression and visual performances of decomposed layers.

1.3 Image filtering

In the early development of signal and image processing, linear filters were the primary tools. However, linear filters have poor performance in the presence of noise as well as the problems where system nonlinearities are encountered. For such reasons, nonlinear filtering techniques for signal/image processing were considered as early as 1958 [W58].

In the early sixties investigations of Matheron and Serra led to a new quantitative approach in image analysis, nowadays known as mathematical *morphology* [S82, M75]. 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 the shape of the matching patterns, called structuring elements, one can obtain useful information about the shape of the different parts of the image and their interrelations. In general the procedure results in nonlinear image operators which are well-suited for the analysis of the geometrical and topological structure of an image.

Further, mathematical morphology was significantly developed and achieved a status of a powerful tool for image processing, which is applied in various disciplines such as mineralogy, medical diagnostics, machine vision, pattern recognition, granulometry and the lot of others [DE03].

One of the primary applications of morphology is noise removal, which is restoration of the original image content from the noisy environment. Noise removal is established in the classical works on mathematical morphology such as [S82, M75], and continues to develop nowadays [PV90, D92, H94, DA97].

It has been found that morphological processing of the image as well as noise removal decreases the entropy of the image, and therefore increases the compression performance. The idea of using nonlinear filters for improving compression performance of the image is known as image enhancement and was investigated in [TP80, M75, W86, ZD96]

The mathematical morphology has been continued to develop during last decade. So called *alternating sequential filters* (ASF) originally proposed by Sternberg in 1986 [S86] were proved to be optimal in environments with additive noise in the sense of the least mean difference [SG91]. Further Heijmans in his works [H95] extended the class of AFS filters by using so called *over-* and *under-filters*.

Sarca et al. [SDA99] proposed the method of so called *two-stage optimized binary filter design*. Such filters are advantageous since they are fully optimal with respect to certain subsets of the filter window.

Jin et al [J95] proposed a new class of morphological operators for binary images, it is the *domain operators*. The basic idea is taken from ranked-order filters, but generalized with the incorporation of the fuzzy index function in weight representation.

The standard (so called 'crisp') morphology has been also extended to 'soft' morphology, which is more tolerant to noise and has advantages in image filtering [KA94]. Kuosmainen et al. [KK95] considered *Shape Preservation Criteria* and studied the optimality of *Soft Morphological filtering*. Zmuda and Tamburino [ZT96] developed efficient soft morphological algorithms used in filtering.

Several methods have been considered for image processing by analyzing the local pixel neighborhood defined by a filtering template. Techniques have been proposed based on the analysis of the context information [PL00, RS01, ZK01].

Ping *et al.* [PL00] proposed two algorithms for binary images filtering. The first algorithm called Modified Directional Morphological Filter (MDMF) is introduced with dual properties for eliminating document salt-and-pepper noise and for remedying eroded character stroke distortion. For eliminating larger noise, another algorithm called *Image Geometric Structure Filter* (IGSF) is proposed based on the geometric stroke information of characters.

Randolph and Smith [RS01] used a *binary angular filter banks* for directional decomposition to enhance fax documents. The filter banks provide representations that delineate the directional components in the text letters enabling edges and contours to be smoothed appropriately.

In [ZK01] *morphological degradation model* was proposed for binary images. According the model, the probability of a pixel flipping from foreground to background, or vice-versa, is an exponential function of its distance from the nearest boundary point. Based on the model they offered restoration algorithm, which includes two stages: a training stage to define parameters of the model, and a restoration stage. The training stage includes joint analysis of degraded and the correspondent *ideal* image by computing the conditional distribution between the noise pattern pairs.

However, mathematical morphology is not a universal tool because morphological operators manage within the local window and can not take into account global properties of the image. Other methods utilizing global (semantic) and statistical properties were developed. The statistical approach was considered in [AF00b], where two context-based filtering methods, namely *Simple Context Filters* and *Gain-Loss Filters*, were proposed for the enhancement of document images. They used the 10- and 20-pixel causal templates to collect statistics during the analyzing phase. Then, in the filtering phase all rare pixels in low entropy contexts are flipped.

Semantic properties of the image also could be utilized for image enhancement. For example Fränti *et al*. [FAKK02] used *Hough Transform* (HT) for extracting vector features from binary image. A *feature image* is reconstructed from the extracted linear segments and it is utilized in the filtering phase. The filtering is based on noise removal procedure using the original and feature images. The noise-filtering algorithm was used to improve quality of context-based compression algorithm. The drawback of this approach is that the HT-based feature extraction phase dominates the processing time in the compression phase and makes it an order of magnitude slower than JBIG compression procedure

Some of these methods are not directly applicable to the context of map images, others (as in the case of latter techniques) require significant computational or memory resources which are not available for mobile devices.

Also we must take into account that traditional filtering techniques (and some of reviewed above) improve compression performance by degrading the image. The most common effect is smoothing of the image. However, in our task we can not use that kind of enhancement techniques because we agreed that the content of the map image is critically important and can not be degraded. Therefore, we decided to develop restoration technique dedicated for the map images. The main restriction on that technique is that map image must be prevented from any corruption caused by filtering.

1.4 Compression techniques

As long as we restricted ourselves with lossless restoration, the compression techniques which we are going to use in our research also must be lossless. We will use the following compression techniques and correspondingly image file formats in our evaluation: PNG, ITU Group 4 (TIFF), and JBIG.

PNG

Portable Network Graphic (*PNG*) is a file format for the use in computer networks and Internet [PNG]. PNG uses *Deflate* data compression algorithm [D96] based on a dictionary

based LZH compression scheme [LM00]. The main idea of dictionary based data compression is to extract common sequences of symbols, called *phrases*, from the source data and replace them by the *dictionary* indices. To achieve compression, the indices must be built in a manner so that their representation takes less space than the original phrases. *Ziv-Lempel* base algorithm, known as LZ77, builds dictionary adaptively using a part of the previously encoded data (called sliding window) as a dictionary [ZL77]. The sliding window contains two: the sequence of already encoded symbols used as a dictionary, and the *lookahead buffer*. Idea of the algorithm is to look for the longest match between the sequence in the buffer and the dictionary. The matched phrase is encoded using a pair *(i, j)*, where *i* is the *offset* from the beginning of the buffer, and *j* is a match. After the encoding, the window is slid *j+1* symbols forward and the process is repeated. The LZSS modification of the LZ77 algorithm adds a one-bit flag *f* to the encoded data indicating whether this is an index for the phrase or non-encoded symbol from the source [SS82]. This technique allows for combining together the source symbols and indices in the encoded data stream, thus gaining the advantage in situations when the indices would have longer data size than the original sequence. The LZH is a modification of LZSS described in [B87]. LZH uses dynamic Huffman algorithm use the encoding of offsets and match lengths [V87].

ITU Group 4

The *ITU Group 4* (formerly known as *CCITT Group 4*) standard incorporates simple data compression techniques based on run-length coding, prefix coding, and differential coding to utilize line-to-line coherence [HR80]. The standard specifies two coding methods: a one-dimensional scheme that treats each line independently, and a two-dimensional one that exploits coherence between successive lines. The one-dimensional scheme is used at the beginning of the image (or image stripe in ITU Group 3 method). In this, a line is represented by coding the length of each *run* (the sequence of pixels with same color) using a pre-specified non-adaptive prefix code. The code table is optimized for a particular set of test documents. For the rest of image lines, the two-dimensional method known as *relative element address designate* (*READ*) is applied. The READ algorithm identifies the positions along each line at which image changes from black to white and vice versa, and codes them in respect to a nearby change position (of the same color) on the previous *reference* line. If there is no nearby change within three pixels on the reference line, the ordinary one-dimensional code is used. Originally designed for use in facsimile communication as *ITU Group 3* [ITU T.4] the method has been expanded for encoding monochrome image in digital networks, and is incorporated as an option for TIFF image file format.

JBIG

Context-based statistical modeling and arithmetic coding are implemented in the latest international standard for compression of binary images, *JBIG (Joint Bilevel Image Experts Group)* [JBIG]. To distinguish it from the emerging JBIG2 standard, we will refer to it as to JBIG1.

JBIG *(Joint Bilevel Image Experts Group)* is an International Standard for compression of bi-level images in communications [JBIG]. The standard defines two methods for bi-level compression, *progressive* and *sequential*. In sequential coding, the image is coded in raster scan order using a *context-based probability model* [RL81] and *adaptive arithmetic coder* namely the QM-coder [PM93]. The idea of context-based modeling is to obtain the statistical model of the image by conditioning the probability distribution of the pixels on the context. The context is determined by the combination of the pixels in the local neighborhood, which is defined by the template. The probability distribution of the black and white pixels is conditioned on the context, which is defined by the combination of already coded neighboring pixels. A three-line ten-pixel template is used by default. Both encoder and decoder estimate the model dynamically during the compression. The estimation starts from scratch and adapts the model to the input data. The probability estimation in the QM-coder is derived from the arithmetic coder renormalization and is based on the Bayesian estimation concept [PM88, PMLA88].

The emerging standard JBIG2 improves the compression of text images using pattern matching technique for extracting symbols from the image. This enhancement, however, is of limited direct usage in the case of map images, as they do not contain large number of non-overlapping text regions.

JBIG contains also several different options that are not discussed in this thesis. Among these options are: decomposition of the image into stripes which are independently encoded, deterministic prediction mode, typical prediction mode, etc as defined in the JBIG1 standard. In our experiment we use a default option combination of the option as provided by the "nconvert" image conversion software: stripe size is 128 lines, deterministic and typical prediction is on. JBIG has also possibility to encode multi-bit (such as color-palette or grayscale) images directly by decomposing the images into bit-planes using binary- or gray-code values. However, according to [FKV02] higher compression results are achieved by decomposing the images into the semantic layers.

1.5 Problem definition

Now we can formulate the problem:

> *Our task is to design a proper technique to restore semantic layers in the map images resulting from the decomposition of the image using color separation process. Restoration technique has to provide better compression performance for reconstructed layers than for corrupted ones when using popular compression algorithms (ITU Group 4, PNG, JBIG). Reconstructed layers have to perform good visual appearance, which is useful for removal or extraction of layers from the original map image for further processing.*

To solve a given problem a big variety of restoration techniques could be proposed. But when designing algorithms for use in real-time cartography, one should take into account the capabilities of modern mobile devices. The technical level of the mobile terminals at the moment represents the processing power of the computers in the nineties. Therefore the complexity and memory requirements of the algorithms must be as less as possible to make algorithm applicable and we have to restrict our research with that factors. Thus we decided to avoid too complicated techniques of restoration and chose *Mathematical Morphology* as a base tool to construct our restoration technique. In order to find an optimal set of parameters we performed an empirical investigation of different morphological filters, evaluate results and chose the best one.

1.6 Structure of the thesis

This thesis is organized as follows. Section 1.1 gives an introduction to basic concepts of mathematical morphology. In Section 3 we use morphological notations for defining the problem under investigation. Section 4 is devoted to reconstruction of solid regions represented by Waters and Fields layers. Restoration algorithm is proposed and its modifications are considered. Section 5 investigates the problem of restoration of Elevation layer. The application of proposed algorithm is discussed. Section 6 tackles the problem of removing a layer from multi-layer map. Algorithm of removing is proposed and illustrations of its performance are presented. Section 7 discusses the implementation of proposed algorithms and concepts. In Section 8 experimental results are evaluated and studied. Section 9 summarizes results and makes final conclusions. Section 10 is devoted to perspectives of future research.

2 Mathematical morphology – the background

Morphological image processing has become a standard part of the imaging scientist's toolbox and today is applied daily to a wide range of industrial applications, including (and certainly not limited to) inspection, biomedical imaging, document processing, pattern recognition, metallurgy, microscopy, and robot vision. Because the morphological operations can serve as a universal language for image processing, their application is only limited by the ability to design effective algorithms and efficient computational implementation.

Mathematical morphology refers to a branch of nonlinear image processing and analysis developed initially by Georges Matheron [M75] and Jean Serra [S82] that concentrates on the geometric structure within an image. That structure may be of a macro nature, where the goal is the analysis of shapes such as a tools or printed characters, or may be of a micro nature, where one might be interested in particle distributions or textures generated by small primitives. The main idea is to analyze the shapes of objects in an image by "probing" the image with a small geometric template (e.g. line segment, disc, square) known as the *structuring element*. The choice of the appropriate structuring element strongly depends on the particular application at hand. This however should not be viewed as a limitation, since it usually leads to additional flexibility in algorithm design.

Today mathematical morphology is an established discipline in the areas of image and signal processing, filtering, segmentation and image and signal coding, among others. It has strong links with more recent theories such as scale-space and level set methods. On the practical level most introductory books in image analysis and most available commercial imaging software include at least few morphological operators.

In this section we consider basic notations and terminology of mathematical morphology.

2.1 Classic morphology

2.1.1 Basic definitions

Consider $\mathbb{E}$ is an Abel group and $E = \mathbb{E}^d$ is the d-dimensional product $\mathbb{E} \times ... \times \mathbb{E}$. In practical cases $\mathbb{E} = \mathbb{Z}$ or $\mathbb{R}$ with the additive group structure, and E is a discrete or *Euclidian* space.

Then we denote by $\mathcal{P}(E)$ the power set of E comprising all subsets of E. In that case $\mathcal{P}(\mathbb{E}^d)$ is a Boolean algebra and have following properties:

- $\mathcal{P}(E)$ – is a complete lattice over $\mathbb{E}^d$.

Complete lattice — is a set $\mathcal{L}$ such as any subset $K \subseteq \mathcal{L}$ has its infimum ∇K and supremum ΔK. Supremum of $\mathcal{L}$ is called maximal element and denoted as $\mathbf{I}$. Infimum is called minimal element and denoted by $\mathbf{O}$.

- $\mathcal{P}(E)$ is distributive:

  $$X \cup (Y \cup Z) = (X \cup Y) \cap (X \cup Z), \quad \forall X, Y, Z \in \mathcal{P}(E)$$

- $\mathcal{P}(E)$ is complementary, i.e. for each set $X \in \mathcal{P}(X)$ exists its *complement* $X^c = E \setminus X$, such that:

  $$X^c \cup X = E, \text{ and } X \cap X_c = \varnothing$$

In case of discrete binary images E is defined as $E = \mathbb{Z}^2$ and binary image X – as a set $X \subseteq E$:

$$X = \{z \mid f(z) = 1, z = (i, j) \in \mathbb{Z}^2\}$$

Note that notations $A \subseteq E$ and $A \in \mathcal{P}(E)$ are equal. The function f is called *characteristic function* of X. It may take only two values: 1 and 0, and it separate the *foreground pixels* (ones that form the image X) from the *background pixels*. Foreground image pixels are usually shown in black. The background pixels belong to X^c.

2.1.2 Principles of morphological transformation

For a set $A \subseteq E$ and element $h \in E$ we define the *translate of A along the vector h* as

$$A_h = \{a + h \mid a \in A\}.$$

The main principle of mathematical morphology is to analyze of geometrical and topological structure of the image X by "probing" that image with another small set A called *structuring element*. By the "probing" we understand applying the structuring element A at every pixel location $h \in E$ and determining the cardinal value $card(X \cap A_h)$ that is the number of elements of X coinciding with elements of A translated along h. In this way structuring element defines the location of analyzed pixels relatively to a given pixel location h. Coinciding of pixels within the structuring element is illustrated on the following Figure 6.

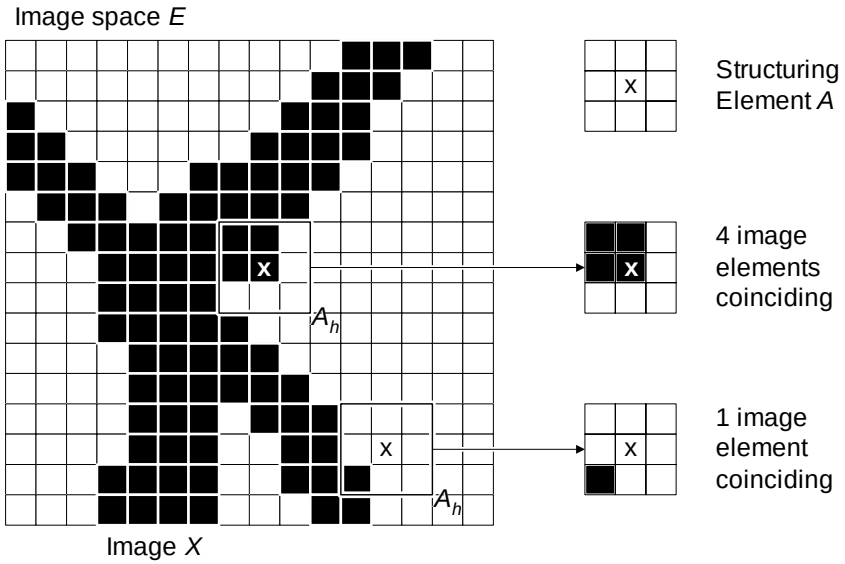


Figure 6. Probing the image with the structuring element.

We define *morphological transformation* $\psi_A(X)$ as some transformation of the image using the information retrieved with "probing" of X with structuring element A.

The performance of a morphological transformation depends on two factors: the structuring element A and the choice of transformation $\psi_A(X)$. There exists a great variety of structuring elements, but they can be decomposed to the simplest ones (as will be shown later). Their choice depends on the particular task. The transformation function $\psi_A(X)$ also can have variety of different properties.

2.1.3 Morphological operators

Let's give some basic definitions:

Operator $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ is called *morphological operator* if it is:

- *Increasing*, if $X \subseteq Y \Rightarrow \psi(X) \subseteq \psi(Y)$, for $\forall X, Y$ (further consider $X, Y \in \mathcal{P}(E)$)

  If increasing operator is applied to the subset X of some set Y, then the result $\psi(X)$ must also be a subset of $\psi(Y)$.

- *Translation invariant*, if $\psi(X_h) = [\psi(X)]_h$, for $\forall X, h$

  The application of translation invariant operator to the translated set X causes equal translation of the result $\psi(X)$. In other words, translation invariant operator is "independent" of the location of set X and the magnitude of translation. One can translate the set X and then apply operator ψ or do it in reverse order — the result remains the same.

The *negative* of an operator ψ is the operator ψ^* defined as

$$\psi^*(X) = \left[\psi(X^c)\right]^c .$$

An operator $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ is called:

- *Self-dual*, if $\psi^* = \psi$

  – this operator affects foreground and background equally.

- *Idempotent*, if $\psi^2 = \psi$

  – the result of applying that kind of operator more than once is equal to the result of single applying.

- *Identity* operator $\psi = id$ if $\psi(X) = X$.

- *Filter* if it is increasing and idempotent.

  – thus *morphological filter* is idempotent morphological operator.

- *Extensive*, if $\psi(X) \supseteq X$.

  – extensive operator "increases" X.

- *Anti-extensive*, if $\psi(X) \subseteq X$.

  – anti-extensive operator "decreases" X.

The following establishes the ordering between two operators. Given two operators ϕ and ψ, the notation '$\phi \le \psi$' means that $\phi(X) \subseteq \psi(X)$ for every $X \in \mathcal{P}(E)$. By $\phi \wedge \psi$ and $\phi \vee \psi$ we denote the infimum and supremum, respectively, of ϕ and ψ. That is $(\phi \wedge \psi)(X) = \phi(X) \cap \psi(X)$ and $(\phi \vee \psi)(X) = \phi(X) \cup \psi(X)$, for every $X \in E$.

2.1.4 Dilation and erosion operators

Given two sets $X, A \subseteq E$, the *Minkowski addition and subtraction* are respectively defined as

$$X \oplus A = \left\{ x + a \,\middle|\, x \in X, \forall a \in A \right\}$$

$$X \ominus A = \left\{ h \,\middle|\, a + h \in X, \forall a \in A \right\}$$

Note that $X \oplus A = A \oplus X$. These operations are the basic ingredients of mathematical morphology.

Given a fixed set $A \subseteq E$ called in morphology the *structuring element*, we define the *dilation of X by A*, denoted by $\delta_A(X)$, as an operator on $\mathcal{P}(E)$ such as

$$\delta_A(X) = X \oplus A .$$

The *erosion by A*, denoted by $\xi_A(X)$, is the operator

$$\varepsilon_A(X) = X \ominus A .$$

Two alternative, but equivalent formulas are:

$$\delta_A(X) = \bigcup_{a \in A} X_a = \{h \in E \mid \widetilde{A}_h \cap X \neq \varnothing\};$$

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

Here $\widetilde{A} = -A = \{-a \mid a \in A\}$ – is the reflectance of A with respect to the origin. For the symmetric structuring element holds

$$A = \widetilde{A}.$$

Further we shall use the following structuring elements (see Figure 7). The $\wedge$ sign points to the location of the origin.

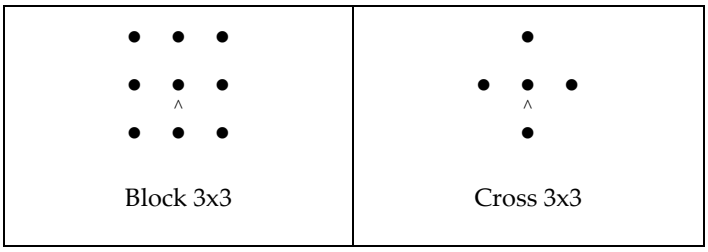


Figure 7: Structuring elements

Dilation and erosion with Block 3x3 are illustrated on the following Figure 8:

Original image | Dilated with Block 3x3 | Eroded with block 3x3

Figure 8: Illustration of dilation and erosion

Compare the results shown above with the same image dilated and eroded with another structuring element Cross 3x3 as illustrated on the Figure 9:

Dilated with Cross 3x3 Eroded with Cross 3x3

Figure 9: Dilation and erosion with Cross 3x3

The following properties of dilation hold for any $X, Y, A, B \subseteq E, h \in E$ and $r \in E$:

- Translation invariance:

$$(X \oplus A)_h = X_h \oplus A$$

- Decomposition of structuring element:

$$(X \oplus A) \oplus B = X \oplus (A \oplus B)$$

- Dilation is an increasing operator:

$$X \subseteq Y \Rightarrow X \oplus A \subseteq Y \oplus A$$

- Dilation is invariant under scaling:

$$rX \oplus rA = r(X \oplus A)$$

For erosion, similar properties hold.

- Translation invariance:

$$(X \ominus A)_h = X_h \ominus A$$

- Decomposition of structuring element:

$$(X \ominus A) \ominus B = X \ominus (A \ominus B)$$

- Dilation is an increasing operator:

$$X \subseteq Y \Rightarrow X \ominus A \subseteq Y \ominus A$$

- Dilation is invariant under scaling:

$$rX \ominus rA = r(X \ominus A)$$

Decomposition of structuring element property is very important for practical implementation. It means that we do not need a very complicated structuring element, i.e. if structuring element could be decomposed, then we replicate original morphological operation with the sequence of morphological operations with the decomposed parts of the

original structuring element. Take a look at the following examples. Figure 10 illustrates the decomposition of structuring element "circle" into three more simple structuring elements.

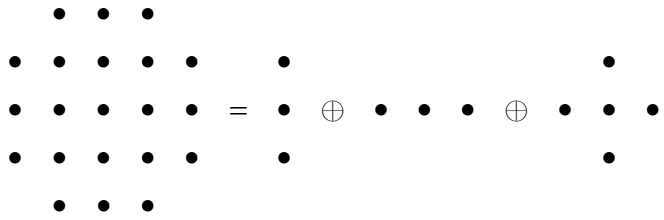


Figure 10: Decomposition of structuring element "circle"

Figure 11 illustrates the decomposition of element "block 5x5" into two "block 3x3"

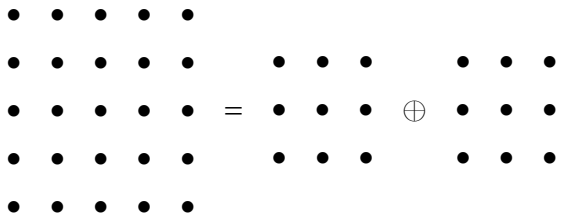


Figure 11: Decomposition of structuring element "block 5x5"

Dilation and erosion are dual morphological operators. It holds that

$$(X \oplus A)^c = X^c \ominus \widetilde{A}$$

$$(X \ominus A)^c = X^c \oplus A$$

for all $X, Y, A \subseteq E$.

Dilation and erosion are dual operators in sense that they satisfy the *adjunction relation:*

$$Y \oplus A \subseteq X \Leftrightarrow Y \subseteq X \ominus A, \text{ for all } X, Y, A \subseteq E$$

As it has been shown in [M75, S82, H94], adjunction relation is one of the basic concepts in mathematical morphology.

2.1.5 Matheron representation theorem

Matheron [M75] has shown that dilations and erosions are the consistent parts of mathematical morphology. To give a formal statement we need the following definition.

The *kernel* $\mathcal{V}(\psi)$ of an operator ψ on $\mathcal{P}(\mathbb{E}^d)$ is defined by

$$\mathcal{V}(\psi) = \{A \subseteq \mathbb{E}^d \mid 0 \in \psi(A)\}.$$

Matheron representation theorem

For any morphological operator ψ holds

$$\psi(X) = \bigcup_{A \in \mathcal{V}(\psi)} X \ominus A = \bigcap_{A \in \mathcal{V}(\psi^*)} X \oplus \widetilde{A}.$$

The theorem states that any morphological operator could be represented as the union of erosions or intersection of dilations.

2.2 Conditional and soft morphology

2.2.1 Rank operator

By using the convention that $[S]=1$ is statement S is true and 0 if it is false consider the *rank function* $r_s(u_1,...,u_n)$

$$r_s(u_1,...,u_n) = \left[\sum_{i=1}^{n} u_i \geq s \right].$$

Consider a set X and its *characteristic function*, i.e., we write $X(h)=1$ if $h \in X$ and $X(h)=0$ if $h \notin X$.

Assume that A is a structuring element containing n elements $a_1,...a_n$, and assume that b is a Boolean function of n variables.

Define the translation invariant operator $\psi_{A,b}$:

$$\psi_{A,b} = \{h | b(X(a_1+h),..., X(a_n+h)) = 1\}.$$

For example:

- If $b(u_1,...,u_n) = u_1 * ... * u_n$, then $\psi_{A,b}(X) = X \ominus A$.
- If $b(u_1,...,u_n) = u_1 + ... + u_n$, then $\psi_{A,b}(X) = X \oplus \widetilde{A}$.

If we take $b(u_1,...,u_n) = r_s(u_1,...,u_n)$ than the resulting operator, denoted by $\rho_{A,s}$, is called *rank operator*.

One finds that $h \in \rho_{A,s}(X)$ if and only if $X \cap A_h$ contains at least n points. That means that operator sets the pixel to the foreground if amount of pixels on the image in a neighborhood defined by the structuring element is greater than s. Otherwise pixel is set to be background. Figure 12 illustrates rank operator performance. Image on the left is the original binary image; image in the middle is the result of applying rank operator with rank parameter equal to 4; right image is the result of rank operator with rank parameter equal to 6.

Original image | Rank operator *s*=4, *A=Block 3x3* | Rank operator *s*=6, *A=Block 3x3*

Figure 12: Illustration of rank operator performance

The operator $\rho_{A,card(A)/2}$ is called *median operator*.

From Matheron's representation theorem follows that rank operator can be treated as the base operator of mathematical morphology. In particular,

$$X \oplus \tilde{A} = \rho_{A,1}(X)$$

and,

$$X \ominus A = \rho_{A,n}(X).$$

Also it holds, that $\rho^*_{A,s} = \rho_{A,card(A)-s+1}$, where $card(A)$ is a *cardinal value* (a number of elements) of a set A.

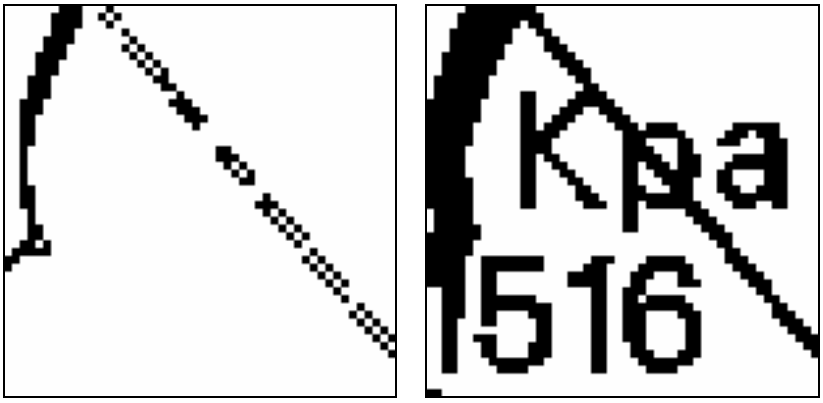
2.2.2 Conditional dilation

If an image is 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 bound. Sometimes it is important to restrict the growth. This can be accomplished by *conditioning* the dilation.

A common form of conditioning restricts the union-forming translations to a superset of the input image: if image A is a subimage of T, and B is a structuring element, then the conditional dilation of A by B relative to T is defined by restricting the translations to T, the result being

$$\delta_B(A|T) = \bigcup_{a \in A} B_a \cap T$$

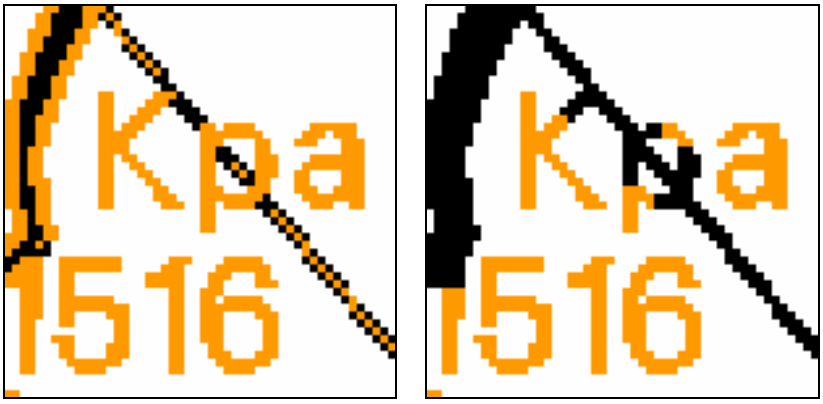
Where notation $\delta_B(A|T)$ instead of $\delta_B(A)$ indicates there is conditioning. Keep in mind that to appreciate the meaning of $\delta_B(A|T)$ in any particular context, one must recognize the type of conditioning being employed. Figure 13 illustrates conditioning principles. Left picture represents an image A, and right picture represents the conditioning mask T.



Original image Conditioning mask

Figure 13: Image and conditioning mask

Going further, the Figure 14 illustrates conditional dilation. On the left picture image and the conditioning mask are represented together: original image with black pixels and conditioning mask with gray. Right image represents original image three times conditionally dilated with *block 3x3*.



Original image and mask Dilated 3 times

Figure 14: Conditional dilation

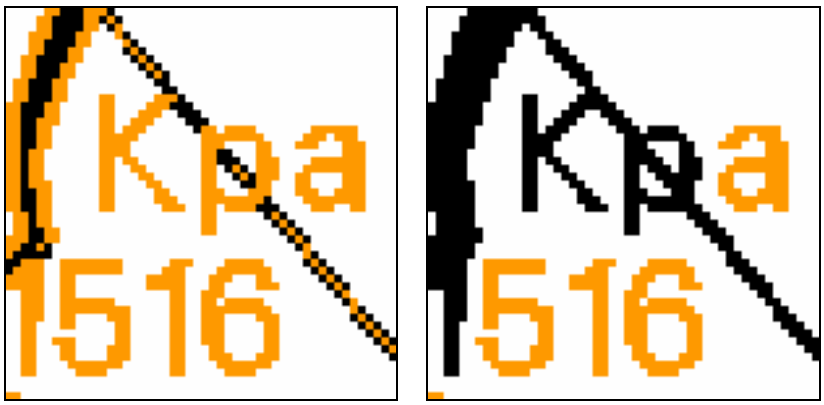
A sequence of n conditional dilations of S relative to T using the structuring element B is called size-n geodesic dilation:

$$\left(\delta_B(S\,|\,T)\right)^n = \delta_B\left(\delta_B \ldots \left(\delta_B(S\,|\,T)\,|\,T\right)\ldots|\,T\right)$$

Viewing S as a marker, reconstruction of T from S is accomplished by repeating the geodesic dilation until stability is reached. In this case, the geodesic dilation is denoted by $T \vartriangle_B S$ and we have

$$T \vartriangle_B S = \left(\delta_B(S\,|\,T)\right)^{\infty}$$

An illustration of reconstruction using geodesic dilation is shown in Figure 15. Here left image represents original and right represents geodesic dilation. Black pixels are binary image pixels and lighter pixels are conditioning mask pixels.



Original image Geodesic dilation

Figure 15: Geodesic dilation

A drawback of the conditional-dilation approach to morphological reconstruction is that it can be time consuming; indeed, it is possible for the number of conditional dilations to equal the number of pixels in the input image. There are, however, very efficient reconstruction algorithms that work recursively and require only a few image scans [DE03].

2.2.3 Soft morphology

Mathematical morphology is based on set theory with transformations exhibiting all-or-nothing precision of pure logic and therefore such transformations could be sensitive to image noise. Sometimes this logic-based paradigm is referred as *standard* or *crisp morphology*. *Soft morphology* is an alternative approach that is more tolerant to noise than standard morphology, yet has many of the desirable characteristics present in the traditional morphological operators [H94, GW02, DE03]. Algorithm designers often use only crisp morphology, leaving noise issues to be handled in other parts of their system. A more effective approach is to process the images using transformations that are tolerant of noise, such as soft morphology.

Standard morphological operators are based on local maximum and minimum operations while soft morphological operations are based on more general weighted order statistics. This makes soft morphological filters to be less sensitive to additive noise and small variations in the shapes of the object to be filtered than standard morphological filters. The definitions of the soft morphological operators are similar to crisp operators but incorporate a factor, r, of how well the structuring element fits within the image.

Given a fixed set $A \subseteq E$ called the *structuring element* and a number r called a *factor* (or *rank parameter*), define the *soft dilation by A with factor r*, denoted by $\delta_A(X,r)$, as an operator on $\mathcal{P}(E^d)$ defined by an expression

$$\delta_A(X,r) = \left\{ h \mid card(X \cap \widetilde{A_h}) \geq r \right\}.$$

The *soft erosion by A with factor r*, denoted by $\varepsilon_A(X,r)$, is the operator given by

$$\varepsilon_A(X,r) = \left\{ h \mid card(X \cap A_h) \geq card(A) - r \right\}.$$

The factor r represents the minimum acceptable overlap between X and the displaced structuring element A. Standard (crisp) erosion and dilation are special cases of their soft counterparts:

$$\delta_A(X) = \delta_A(X,1)$$

$$\varepsilon_A(X) = \varepsilon_A(X,1).$$

These operators can also be represented using rank operator as follows:

$$\delta_A(X,r) = \rho_{\tilde{A},r}(X)$$

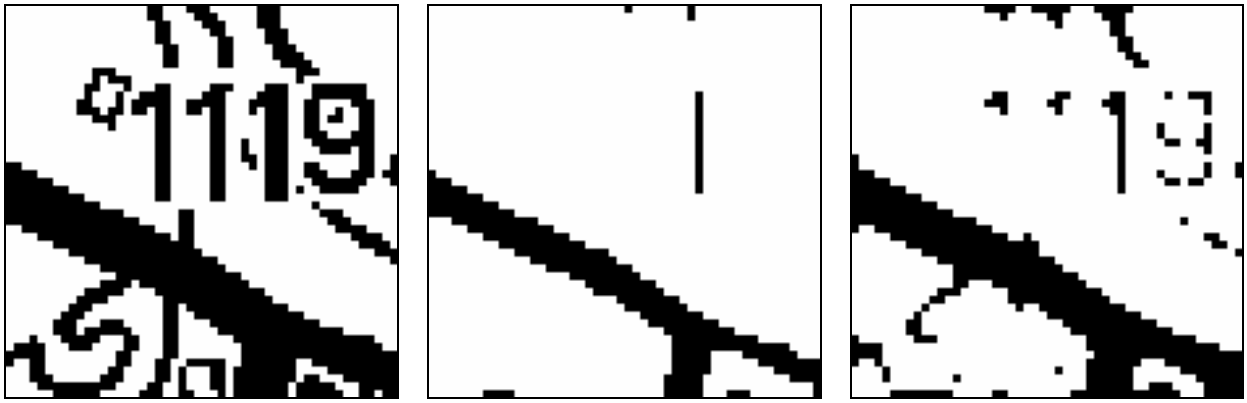
$$\varepsilon_A(X,r) = \rho_{A,card(A)-r}(X)$$

Figure 16 illustrates the performance of soft dilation operator with $r = 2$ (right image) comparing to standard (crisp) dilation (left image) using structuring element *Block 3x3*.

Original image | Standard (crisp) dilation | Soft dilation

Figure 16: Crisp and soft dilation

Figure 17 illustrates the performance of soft erosion with $r = 2$ (right image) comparing to the standard (crisp) erosion (left image) using structuring element *Block 3x3*:



Original image | Standard (crisp) erosion | Soft erosion

Figure 17: Crisp and soft erosion

You can see that parameter r regulates the amount of shrinking (in case of erosion) or expanding (in case of dilation) of objects with the image.

3 Formal problem definition

In the following chapter we define the problem of map image reconstruction using established mathematical formalism.

3.1 Multi-layer map concept

The *binary image space E* is defined as $E = \mathbb{Z}^2$ (the space of all possible image pixel locations), and the *binary image X* – as a set $X \subseteq E$ of foreground pixel locations:

$$X = \left\{ h \,\middle|\, X(h) = \mathit{foreground},\quad h = (i, j) \in \mathbb{Z}^2 \right\}.$$

$X(h)$ is a *characteristic function* $X(h) : E \to \{\mathit{foreground}, \mathit{background}\}$ separating foreground from background. By *N-layer map* we define a set $\mathfrak{M} = \{L_1 ... L_N\}$ of N binary images $L_1, ..., L_N \subseteq E$, where $L_j \subseteq E$ is called a *j-th layer* and *j* is called an *index of a layer*.

We define a gray-scale (or color) image as a pair

$$\mathcal{M} = \begin{pmatrix} C \subseteq E \\ c : \mathbb{Z}^2 \to \mathbb{N} \end{pmatrix},$$

where C is a set of pixel locations (a binary image) and $c(h)$ is intensity (or color) function which yield a gray value or an index into a color palette.

3.2 Combination

Even though the multi-layer map consists of binary layers, the layers must be combined into a single map image before that image can be served to a user. Such operation is called *layer combination* and is denoted by $\mathfrak{M} \xrightarrow{C} \mathcal{M}$. It takes a multi-layer map $\mathfrak{M}$ as an input and generate color image $\mathcal{M}$ as follows

$$\mathcal{M} = \begin{pmatrix} C = \bigcup_{k=1}^{N} L_k \\ c(h) = \min_{\forall k | h \in L_k} k \end{pmatrix}.$$

The resulting image $\mathcal{M}$ is called *combined map image.*

Here C is computed as the union of all layers, and $c(h)$ is a minimal index among layers that contain pixel at location *h*. This means that if there is a pixel location, which belongs to several layers simultaneously, than it will be colored to a color of a layer with smallest index. This way layers overlap other layers if they contain pixels at the same locations. For example if *Basic* layer has index less than *Water* layer, then contours and city names on the *Basic* layer will appear to the user "above" the water.

Observe the difference in the definitions: by the *multi-layer map* we understand a set of layers; and by the *combined map image* – their combination, which is served to the user.

3.3 Decomposition

When semantic layers are not available and all we got is a color raster image, we need to decompose it to layers before the compression using color separation process as described in Section 1.2.

Let us define *decomposition* as the process $\mathcal{M} \xrightarrow{D} \mathfrak{M}$ of splitting N-color map image

$$\mathcal{M} = \begin{pmatrix} C \subseteq E \\ c : \mathbb{Z}^2 \to \mathbb{N} \end{pmatrix}$$

into N-layer map

$$\mathfrak{M} = \{L_1 ... L_N\} \text{ (a set of binary layers),}$$

as follows

$$L_k = \{h \in C \mid c(h) = k\}, k = 1..N .$$

Decomposition separates pixels to different layers by their color value.

3.4 Properties of composition and decomposition.

Proposition 1:

Multi-layer map once composed into a gray-scale image could not be reconstructed back.

$$\forall \mathfrak{M}_1 = \{L_1, ..., L_N\} \xrightarrow{C} \mathcal{M} \xrightarrow{D} \mathfrak{M}_2 = \{L_1^*, ..., L_N^*\} \subseteq \mathfrak{M}_1 .$$

The proposition states, that composition and further decomposition are lossy processes.

Proof:

When we use notation

$$\mathfrak{M}_2 = \{L_1^*, ..., L_N^*\} \subseteq \mathfrak{M}_1 = \{L_1, ..., L_N\},$$

we understand that

$$L_1^* \subseteq L_1, L_2^* \subseteq L_2, ..., L_N^* \subseteq L_N .$$

To prove the proposition, let's recall that in composition process $c(h) = \min_{\forall k \mid h \in L_k} k$, therefore pixels at position *h* on layers with level greater than $c(h)$ (greater than minimum) will be overlapped in composition and lost in decomposition.

For a layer L_k a set of pixels $A_k = \{h \in L_k \mid \exists j < k : h \in L_j\}$ will be overlapped and lost.

Therefore $L_k^* \cup A_k = L_k$ and, obviously, $L_k^* \subseteq L_k$ □.

Further we shall call a set $A_k = \{h \in L_k \mid \exists j < k : h \in L_j\}$ *a set of artifacts* for a layer *k*.

A set of artifacts contains pixels that will be lost in composition-decomposition process. Further we will denote corrupted layers by L_k^* and original layers by L_k.

Proposition 2:

The combined map image once decomposed into a multi-layer map could be safely reconstructed back.

$$\forall \mathcal{M}_1 \xrightarrow{D} \mathfrak{M} \xrightarrow{C} \mathcal{M}_2 = \mathcal{M}_1 .$$

The proposition state, that decomposition and further composition are lossless processes.

Proof:

Decomposition process separates pixels to layers by their color, and therefore layers created this way will not intersect i.e. $L_j^* \cap L_k^* = \varnothing, \forall j,k = 1..N$. As layers not intersect, no information will be lost in composition process □.

3.5 Restoration

The task of restoration of multi-layer map after composition-decomposition process is the problem of reconstruction of artifacts A_k for every layer. But we cannot reconstruct A_k for every layer completely, we can only suppose its structure. Therefore the task of restoration of layer L_k is to construct an operator $\psi(L_k)$ such that

$$\begin{cases} \psi(L_k) : \mathcal{P}(E) \to \mathcal{P}(E) \\ \{L_1^*, ..., L_k^*, ..., L_N^*\} \xrightarrow{C} \mathcal{M}_1 \\ \{L_1^*, ..., \psi(L_k^*), ..., L_N^*\} \xrightarrow{C} \mathcal{M}_2 = \mathcal{M}_1 \end{cases} .$$

In other words a composition of a set of layers where some layer was reconstructed by operator ψ has to be equal to the composition of non-restored layers. It is natural to protect original gray-scale (color) map image from any degradation because of its great semantic importance.

3.6 Masking

Restoration could be maintained by conditioning an operator $\psi(L_k)$ with a mask T_k. As it had been discussed in section 2.2.2, mask is a binary image defining an area where changes of the layer content are allowed.

Keep in mind that composition of binary semantic layers must be preserved unchanged. Therefore restoration operator cannot remove pixels, which are already present on the corrupted layer. It can only add pixels to a layer, which means that $L \subset \psi(L_k \mid T_k)$.

Naturally, pixels could be added to those areas of a layer, which will be overlapped during the composition process. Let's define a set

$$T_k = \{h \mid \exists j \le k : h \in L_j^*\}$$

or, equally,

$$T_k = \bigcup_{j=1}^{k} L_j^* .$$

This set contains all pixel locations, which will be overlapped in composition process by upper laying layers.

Also, note that $A_k \subset T_k$ – a set of artifacts is a subset of mask T_k. Therefore A_k could be reconstructed. The efficiency of reconstruction depends of the chose of operator ψ.

Proposition 3:

Operator $\psi(L_k^* \mid T_k) = \varphi(L_k^*) \cap T_k$ satisfies the condition proposed in 3.5 for any operator $\varphi(L_k^*)$.

The proposition states, any modification of semantic layer performed in area defined by T_k will not affect the combination of layers.

Proof:

Obviously, any modifications that made in L_k^* within T_k will be overlapped by upper layers□.

4 Reconstruction of solid regions

In this chapter we tackle the problem of reconstructing the layers containing solid regions. In our map image examples these regions are represented by *Waters* and *Fields* layers. First we give the description of the basic restoration algorithm. Then, possible algorithm modifications are considered.

4.1 The basic algorithm

In our research we have considered the map image $\mathcal{M}(L_1, L_2, L_3, L_4)$, which is combination of 4-layers (see Figure 4),

- L_1 – *Basics* layer
- L_2 – *Elevation* lines layer
- L_3 – *Waters* layer
- L_4 – *Fields* layer

The task of restoration consists of the following stages.

First we have to decompose combined map image in to a set of corrupted layers

$$\mathcal{M} \xrightarrow{D} \mathfrak{M}(L_1^*, ..., L_4^*)$$

Then we have to form a conditioning mask for conditioning the restoration operator to keep the combination of restored layers untouched. The mask is formed using the following formula:

$$T_i = \bigcup_{j=1}^{i} L_j^*$$

This formula defines a mask with respect to the assumption that layers in multi-layer map are situated one over another in a predefined order. But in our case we can simplify it by taking into account the nature of objects represented on the map. For example, we can expect that *Waters* and *Field* layers could not overlap in reality, and therefore could not overlap on a combined map image. But the concept of multi-layer map enforces that layers must be composed in a predefined order one over another. This enforces *Waters* areas to be included into the mask for Fields. That means that the algorithm will try "to find a field overlapped by the water" which is nonsense semantically. So, when implementing our particular case, we can exclude that kind of layers from the conditioning mask T_i. This adds sense to the concept of conditioning and reduces the size of the mask.

Taking aforesaid into account we can revise formulas defining masks for *Waters* and *Fields*:

$$T_3 = L_3^* \cup L_1^* \cup L_2^*$$

$$T_4 = L_4^* \cup L_1^* \cup L_2^*$$

The Figure 18 illustrates the creation of conditioning mask.

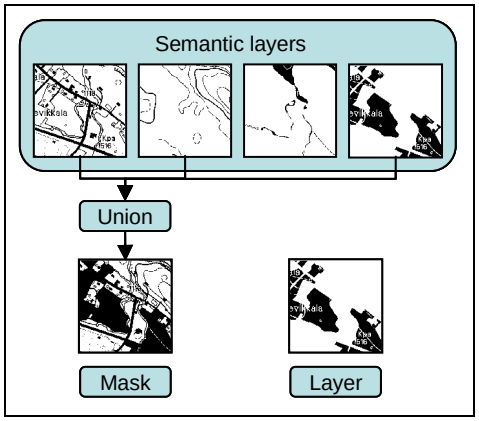
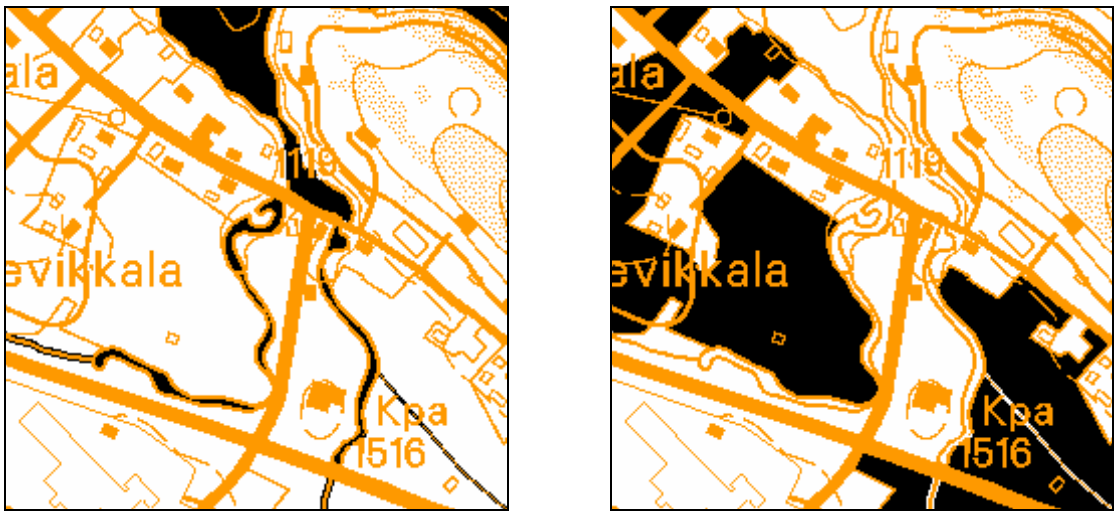


Figure 18: Mask creation

The result of applying that algorithm is illustrated on the Figure 19. Here black pixels represent objects, light pixels are mask and white are background which is restricted for changing. Any restoration of black objects could be done only in a light masking area.



Waters and its mask

Fields and its mask

Figure 19: Waters and Fields with a mask. Objects are black, mask is in light color and background is white.

Following step of the algorithm is the actual reconstruction of the layers. On this step we have to apply some operator ψ conditioned on the mask and to build the reconstruction $\hat{L}_i = \psi(L_i^* \mid T_i)$ of layer L_i^*.

The framework of restoration algorithms could be represented in the following way:

```
Decompose combined map image into a set of Binary layers
For each Binary layer do
  Create Mask
EndFor
For each Binary layer and Mask do
  Apply reconstruction operator
EndFor
```

The problem of restoration is therefore the problem of choosing the operator ψ.

A big variety of different approaches could be proposed to perform such a restoration. For example, the most obvious is to apply conditional dilation operator (see 2.2.2). However, it produces a lot of inappropriate artifacts and cause layer corruption comparable to corruption caused by combination. Therefore, more sophisticated techniques should be designed.

In our work, we have studied variety of different approaches to restoration and developed a restoration operator called *dilation with mask erosion* (see 4.3). This operator is based on the conditional dilation but it avoids from the most its disadvantages (see 4.2). We choose that algorithm to be the basic algorithm for our experiments. In following subchapters we will discuss and illustrate disadvantages of conditional dilation operator and explain the reasons why dilation with mask erosion operator was considered. Also other modifications of the basic algorithm will be proposed.

4.2 Conditional dilation

Waters and *Fields* are of the similar morphological structure. Mostly, the restoration is needed inside objects where inner artifacts have to be filled. Therefore it was natural to try to apply conditional dilation operator

$$\psi(L_3) = \delta_A(L_3 \mid T_3)$$

to reconstruct waters, and

$$\psi(L_4) = \delta_A(L_4 \mid T_4)$$

to reconstruct fields.

The algorithm of applying conditional dilation operator is illustrated on Figure 20.

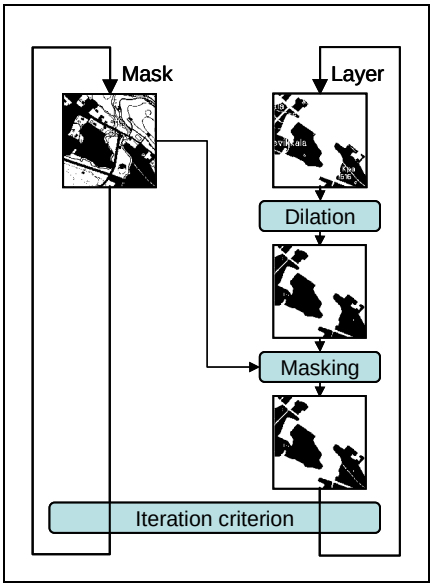
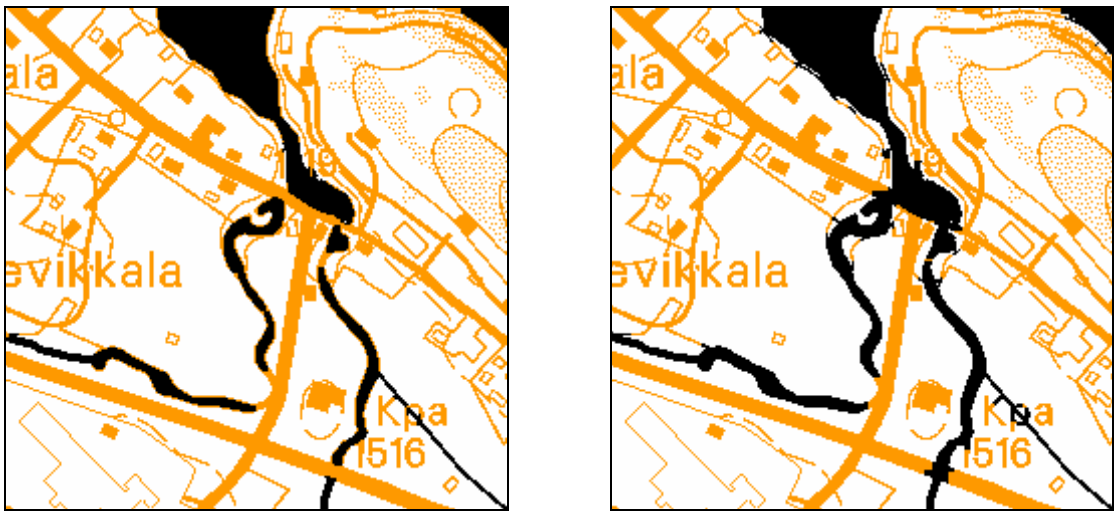


Figure 20. Conditional dilation.

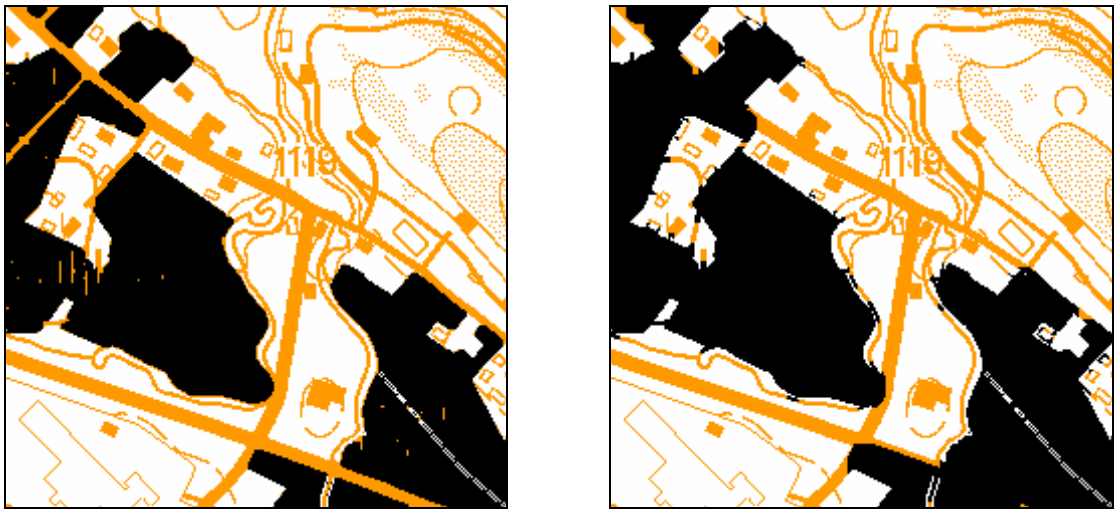
Here we see that mask is not changing during the process of dilation. That causes the appearance of artifacts on the borders of objects and significantly reduces visual performance. Figure 21 and Figure 22 illustrate operator $\delta_A(X \mid T)$ applied once (left image) and 4 times (right image) for *Waters* and *Fields* respectively with *A= block 3x3* (see Figure 7).



Once dilated with block 3x3 Waters

Four times dilated Waters

Figure 21. Once and four times dilated with block 3x3 Waters.



Once dilated with block 3x3 Fields

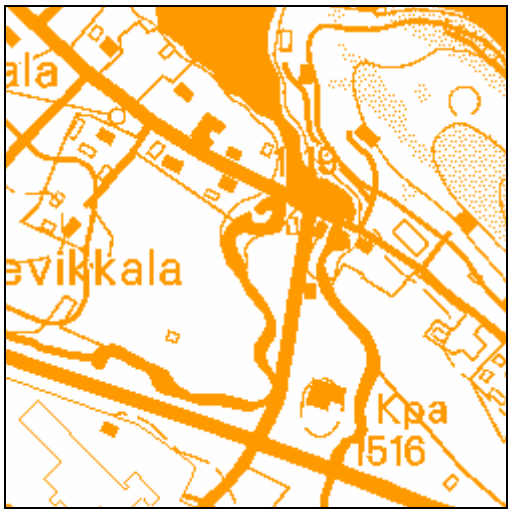
Four times dilated Fields

Figure 22. Once and four times dilated with block 3x3 Fields.

You can see that operator $\delta_A(X \mid T)$ successfully fills artifacts inside objects (right images). The problem is that operator also unacceptably corrupts objects on its borders. The restriction over amount of iterations does not solve the problem, because in that case we cannot guarantee confident filling of inner artifacts. Left images show that borders of objects are smooth but inner artifacts on the *Fields* layer were not filled and *Waters* layer was not reconstructed well because single iteration of dilation is not enough for confident reconstruction. We conclude that conditional dilation is not applicable for layer reconstruction and we have to develop more sophisticated restoration technique.

4.3 Conditional dilation with mask erosion

When observing map images from a test set, we have noticed that border artifacts are typically small areas or thin lines within a mask, which object fills while expanding during dilation. For designing a proper restoration technique, we have to find a way to prevent the expansion of objects to those areas. A simple way to exclude thin areas from the mask is to apply erosion operator. It will erase a lot of thin details, and if those areas will be removed from the mask, as a result they will be removed from the process of conditional dilation. Of course, we have to guarantee that mask will remain to be a superimage of a layer after the erosion. This could be reached by concatenating the result of erosion with the layer itself. The eliminating of thin details from the mask is illustrated on the following Figure 23.



Original mask for Waters

Eroded mask for Waters

Figure 23: Erosion of the mask

You can see that erosion immediately removes small areas. We can establish an iterative process for sequential removing that kind of areas from the mask starting from small ones to bigger ones. Simultaneously, we can dilate objects to perform the restoration. Formally this forms an iterative process:

```
Repeat
  Layer = Dilate( Layer, Mask )
  Mask = Erode( Mask )
  Mask = Union( Mask, Layer )
Until Restoration is complete
```

Or mathematically,

$$X_i = \delta_A(X_{i-1} \mid T^i)$$
$$T^i = \varepsilon_B(T^{i-1}) \cup X_i$$
$$n = 1...number\_of\_iterations$$

Where X_i is a layer image and T^i is a mask on the *i*-th step of iteration process, *A* and *B* are structuring elements of dilation and erosion.

At the first step $X_0 = L_3^*, T^1 = T_3$ for waters and $X_0 = L_4^*, T^1 = T_4$ for fields.

Objects first dilate over the original mask, and then mask erodes. On the second step objects dilate over the eroded mask and so on. Concatenation of the mask with the current layer is necessary because mask must remain to be the superimage of a layer. Figure 24 illustrates the algorithm.

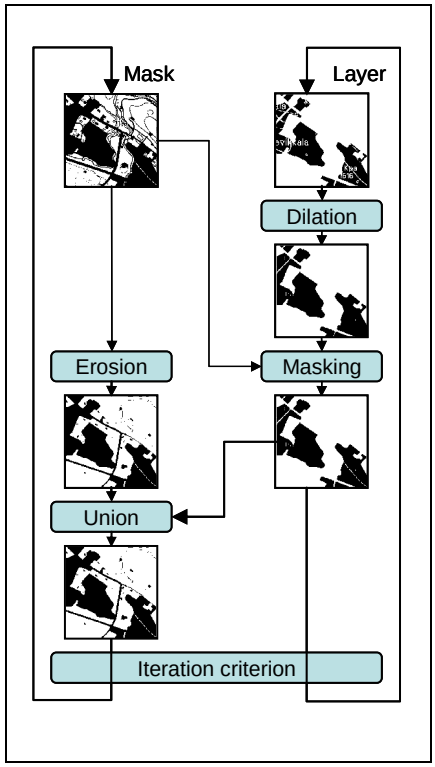
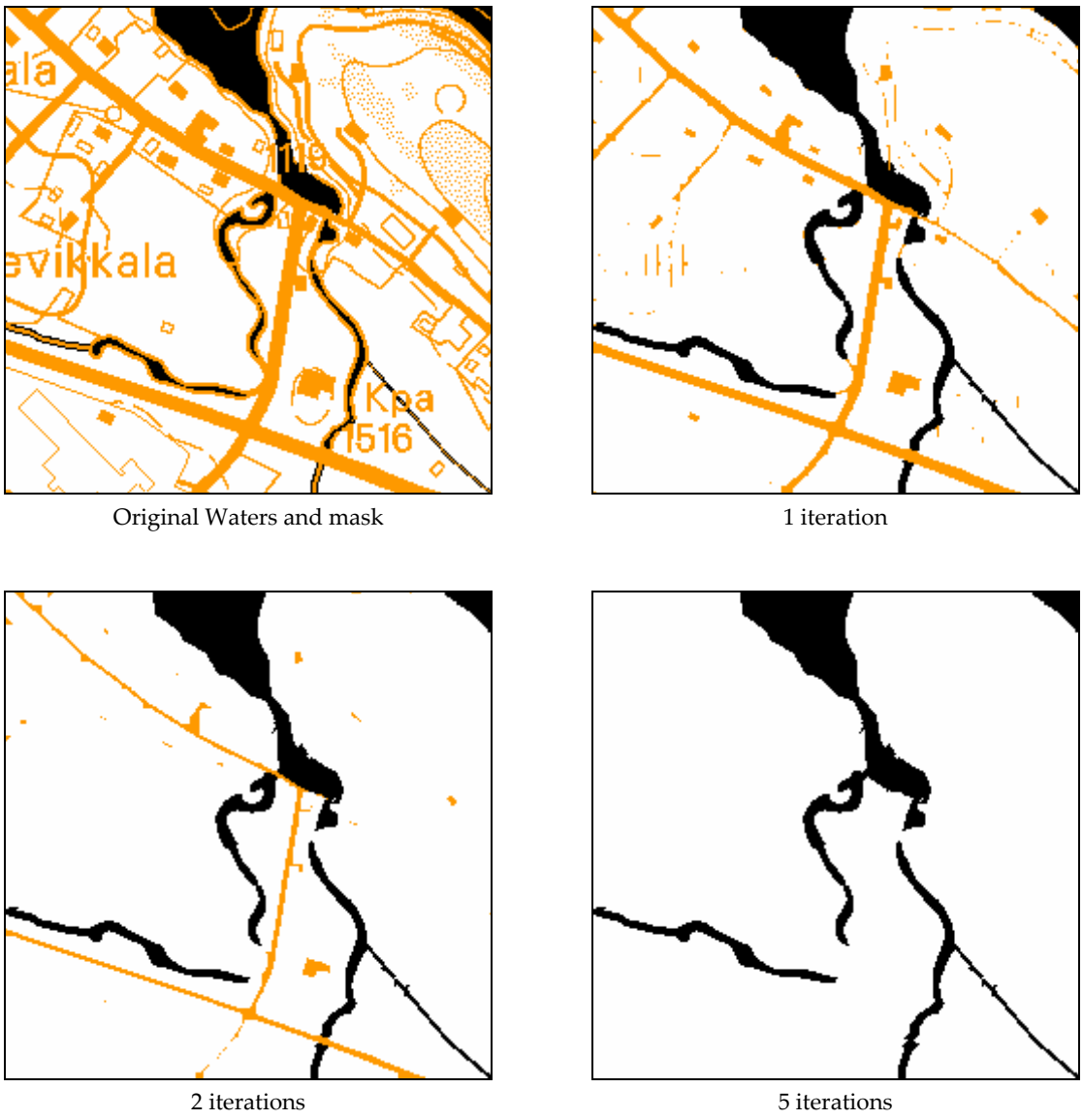


Figure 24: Dilation with mask erosion algorithm

Results of iterative application of conditional dilation with mask erosion operator are illustrated in Figure 25. There are: original image, the result of one iteration, and results of two and five iterations. Objects are shown in black and mask in light color.



Original Waters and mask

1 iteration

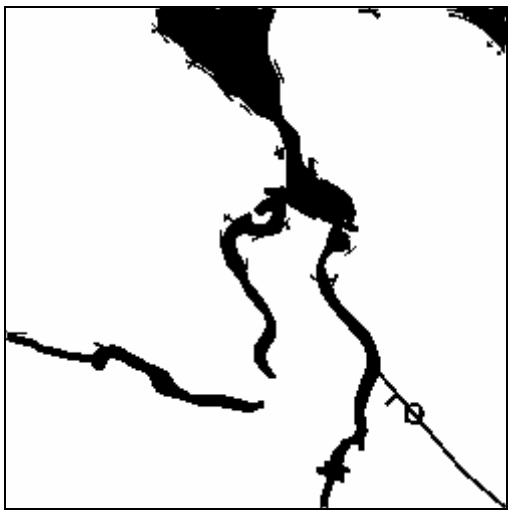
2 iterations

5 iterations

Figure 25: Dilation with mask erosion

Visually the final result is much better that the result of ordinary conditional dilation. It is free from artifacts on object borders and inner artifacts are totally filled.

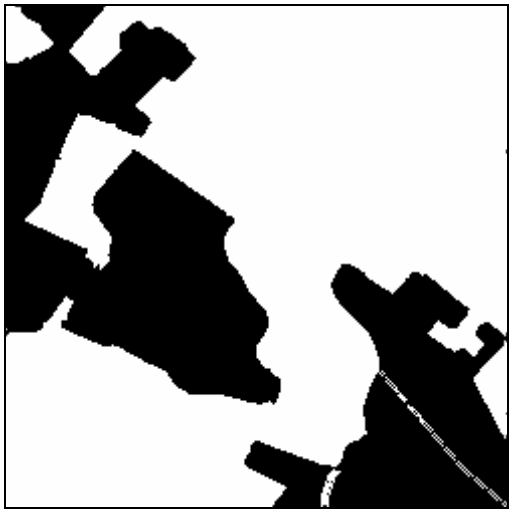
Compare 5-iterational results of ordinary conditioned dilation and of conditioned dilation with mask erosion for *Waters* and *Fields* on a Figure 26. There first column represents layers restored by the conditioned dilation operator and the second column represents same layers reconstructed by conditioned dilation with mask erosion.



Waters. 5 dilated result

Waters. 5 dilated with mask erosion

Fields. 5 dilated result



Fields. 5 dilated with mask erosion

Figure 26: Comparison of conditioned dilation and conditioned dilation with mask erosion

Although the result of conditioned dilation with mask erosion is not absolutely free from artifacts, it performs significantly better visual appearance and object consistency.

4.4 Object Smoothing

In order to improve quality of restoration, we again can investigate the nature of objects under restoration. We know that we are trying to restore waters and fields. These objects typically do not have sharp edges or thin, one or two-pixel details. Therefore, we can try to design some morphological operator to smooth borders of objects.

Small and sharp artifacts could be removed using such operators as structural opening or closing, or using morphological rank operator. For our experiments we chose rank operator

because of its simplicity, which is important property for devices with low computing power.

We expect that restoration algorithm will restore original objects completely and then produce its own artifacts. It means that rank operator have to perform a kind of cleaning over the objects. Therefore, we propose so called "*Subtractive ranking*" operator, which can only remove pixels from a layer, but not those which originate from the original corrupted layer. Formally,

$$\psi(X) = (\rho_{C,r}(X) \cap X) \cup L$$

Here X is a layer restored using dilation with mask erosion technique (see Figure 24), L is an original corrupted layer, C is a structuring element of rank operator and k is a rank parameter. We use that operator on every step of iterative process of reconstruction to smooth borders of objects. The algorithm is modified the following way:

```
Repeat
  Layer = Dilate( Layer, Mask )
  Mask = Erode( Mask )
  Mask = Union( Mask, Layer )
  Layer = Intersection( Layer, Rank(Layer) )
  Layer = Union( Layer, CorruptedLayer )
Until Restoration is complete
```

Steps 5 and 6 perform smoothing of objects. Figure 27 illustrates restoration with object smoothing algorithm.

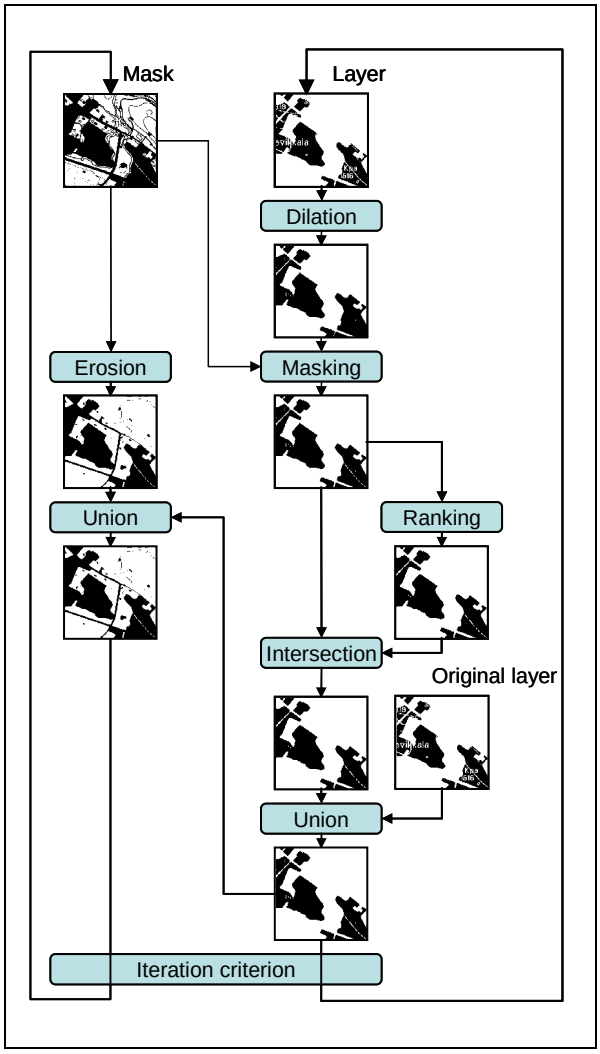
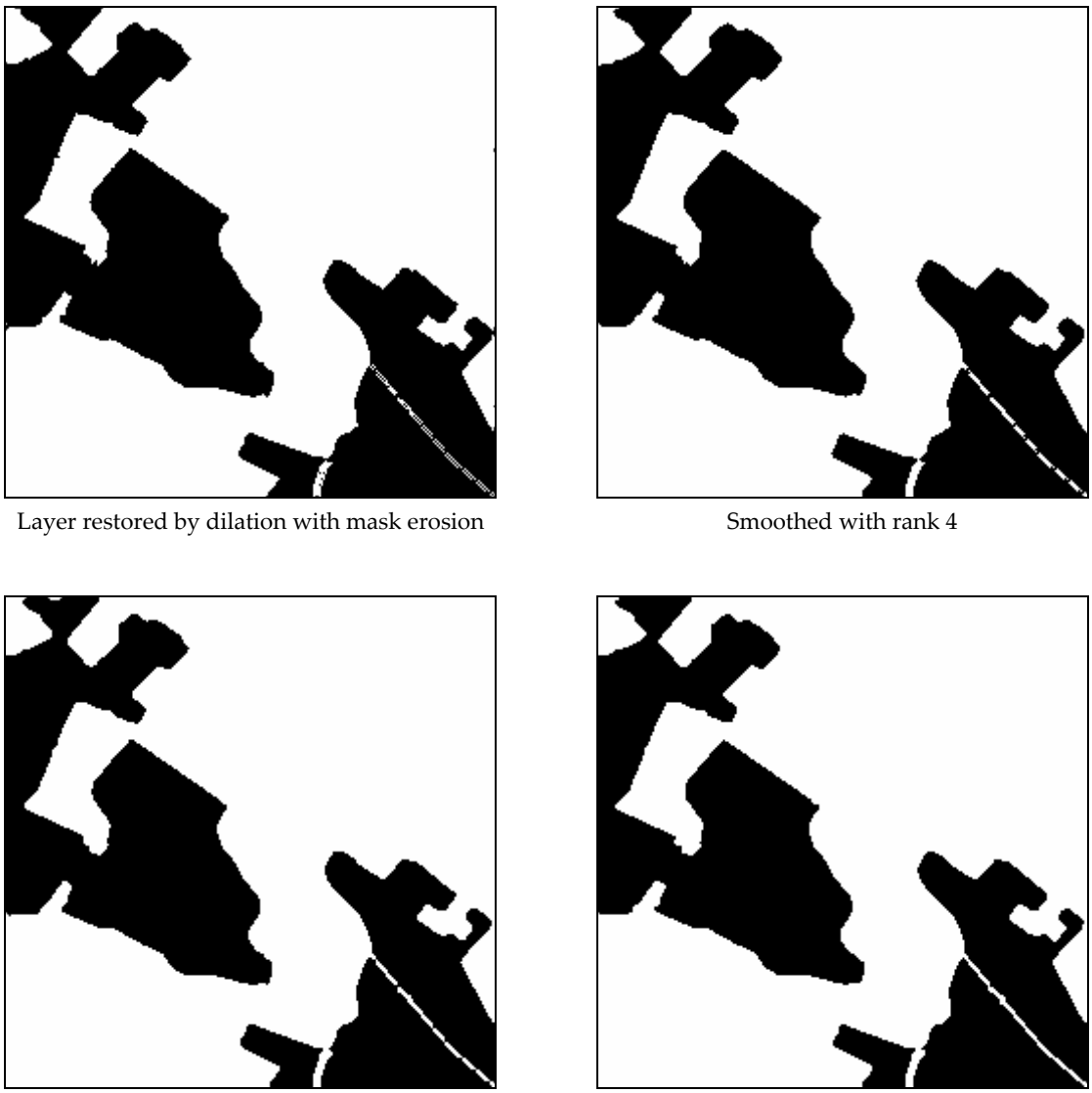


Figure 27: Base algorithm with object smoothing

In Figure 28 you can compare layer restored by dilation with mask erosion operator with the results of smoothing with different rank parameter (4, 5 and 6 respectively) applied to this restored layer.



Layer restored by dilation with mask erosion

Smoothed with rank 4

Smoothed with rank 5

Smoothed with rank 6

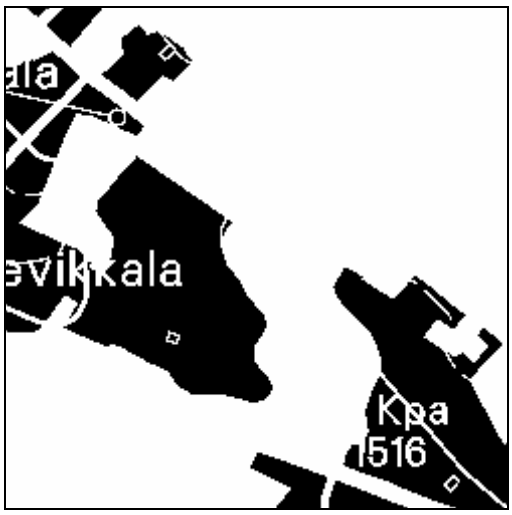
Figure 28: Object smoothing by ranking.

You can see that the value of ranking parameter regulates how efficiently objects will be smoothed.

4.5 Summary

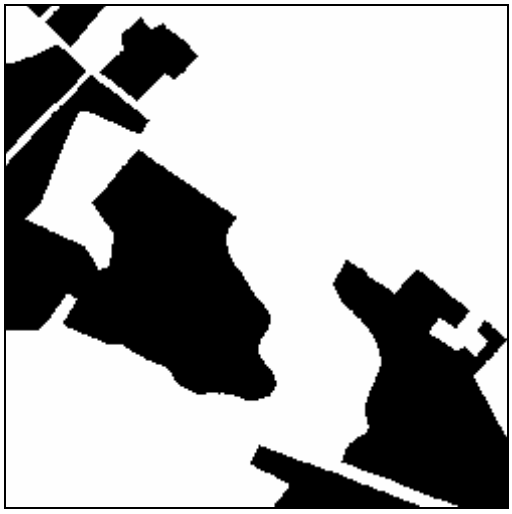
Let's illustrate principles of restoration discussed above. In Figure 29 you can compare restored layers (last row) with the original (non-corrupted, middle row) and examine the efficiency of restoration comparing to corrupted layers (first row).

Waters corrupted



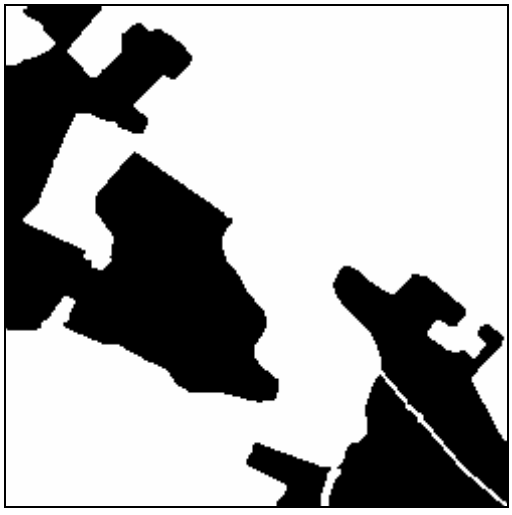
Fields corrupted

Waters original



Fields original

Waters restored



Fields restored

Figure 29: Corrupted, original and restored layer fragments.

You can see that restoration significantly improves the consistency and visual appearance of objects. All inner artifacts (holes made by text and other symbols) were completely filled. Borders of objects smoothed and became more natural. Restored layer is very close visually to the original.

4.6 Algorithm modifications

If we take a look into details of algorithm we can see that there are parameters which could be modified. They are: the criterion controlling the amount of iterations in a process; the structuring element of erosion and dilation operator; dilation and erosion operators could be replaced by other (similar) operators (e.g. soft dilation and erosion).

Criterion of iterations

Criterion of iterations determines how long iterative process should be performed. There are two approaches: *Iterate until stability* and *Iterate fixed amount of times*.

First approach assumes that iterative process will continue until layer (and a mask) stops changing. Erosion of the mask sequentially decreases the mask: $T^i \supseteq \varepsilon_B(T^i) \cup X_i = T^{i+1}$. But each erosion is concatenated with the current restored image X_i. Therefore we can be sure that iterative process has its limit: $\exists n > 0 : T^i = T^n, \forall i \geq n$. Moreover, $T^n = X_n$. Therefore we can iterate the process until layer (and a mask) stops changing.

Second approach assumes that we can restrict the amount of iterations with a fixed number. We can determine average size of artifacts to be restored. Therefore, if we know that, for example, average artifact size is 4 pixels, we can be sure that 2 or 3 dilations with block 3x3 are enough for restoration.

Using alternative structuring elements

There are two structuring elements in our algorithm: in objects dilation and in mask erosion. By changing the first element we can control how fast objects expand over the mask. Changing of the second allows controlling how fast mask shrinks.

Recall the formula of iterative process:

$$X_i = \delta_A(X_{i-1} \mid T^i)$$
$$T^i = \varepsilon_B(T^{i-1}) \cup X_i$$
$$n = 1 \ldots number\_of\_iterations$$

Essential thing is the relation between speeds of dilation and erosion. For our analysis we can select following cases

- Objects dilating faster that mask eroding.

A=block 3x3, *B=cross 3x3*.

- Objects dilating slower than mask eroding.

  A=cross 3x3, *B=block 3x3*.

- Equal speed case

  A=block 3x3, *B=block 3x3* or *A=cross 3x3*, *B=cross 3x3*.

Further, basic algorithm modification with *A=cross 3x3*, *B=cross 3x3* will be referred as *Basic* algorithm.

Using soft erosion and dilation

Dilation and erosion operators could be replaced with their soft counterparts (see 2.2.3). They are relaxed versions of their crisp analogues, and therefore can manage with objects and mask more smoothly. Also, by varying the factor parameter of soft dilation and erosion we can control the speed of expanding and shrinking. Further, that modification will be referred as *Soft* algorithm.

Varying rank parameter in smoothing

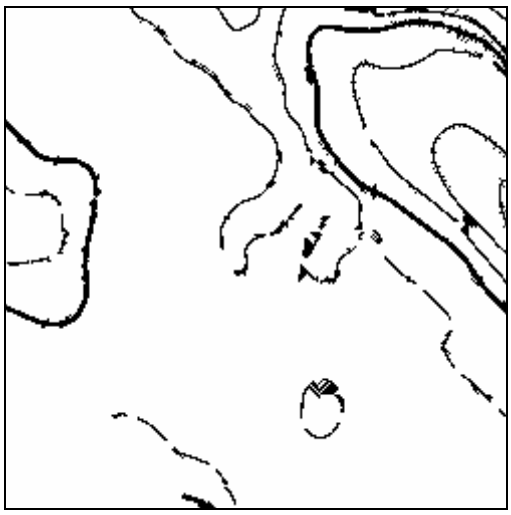
As it had been shown above, varying of ranking parameter allows controlling the accuracy of smoothing (see Section 4.4). Smoothness has an influence to the object consistency and therefore such modifications also have to be considered. Further we will refer to them as *Smooth-1* and *Smooth-2* modifications.

5 Restoration of elevation lines

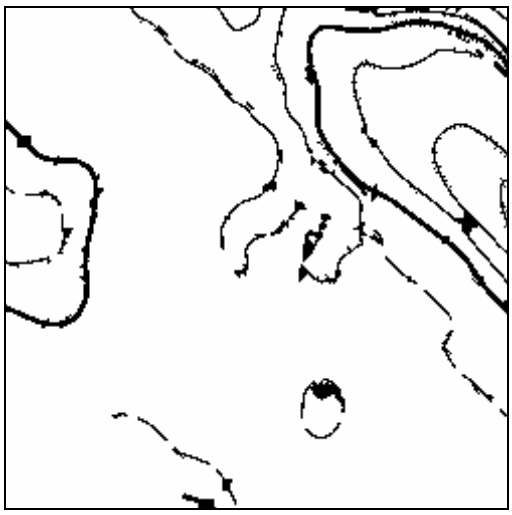
Following chapter is devoted to the solution of the same layer reconstruction problem considered in 1.5 but applying to elevation lines layer. Reconstruction of elevation lines is separated into a single chapter because of a significant difference in morphological structure between elevation lines layer and "solid region" layers such as waters and fields. Therefore morphological operators presenting good results on solid layers are not appropriate when restoring that one. The most natural approach when reconstructing that kind of layer is to design some heuristic algorithm constructing vector approximation of elevation lines, corrupted by overlapping artifacts (e.g. [FAKK02]). But we restricted ourselves with low-complexity algorithms and chose mathematical morphology as a tool. Approximation algorithms and other that kind of techniques are typically much more complex than it is possible when using in mobile devices or other devises restricted in computing power. Therefore we decided to continue using mathematical morphology as a tool and to try to design an algorithm, which of course can not be better than complicated heuristics, but which is based on simple morphological operators and appropriate for using on devices considered above.

5.1 Problem analysis

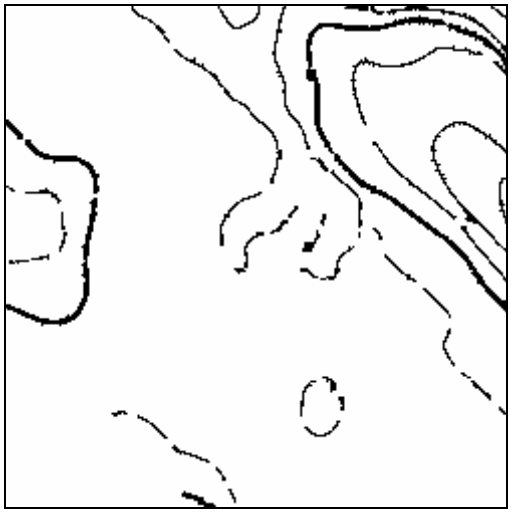
As it was predicted, the most of modifications of basic algorithm designed above (see 4) presented unacceptable results. Following Figure 30 illustrates results of restoration of elevation lines layer with 4 modifications of base algorithm (Figure 24). Here we varied *Cross* and *Block* (see Figure 7) structuring elements of erosion and dilation in different combinations.



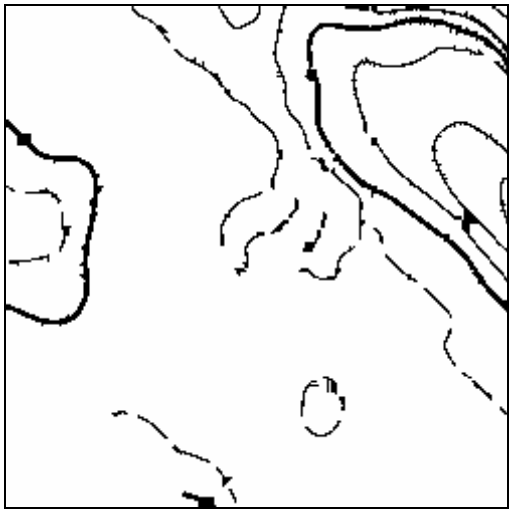
Dilation with Block, erosion with Block restoration



Dilation with Block, erosion with Cross restoration



Dilation with Cross, erosion with Block restoration



Dilation with Cross, erosion with Cross restoration

Figure 30: Restoration of elevation lines with variations of base algorithm

Basic algorithm significantly deteriorates visual performance of the layer. Algorithm version with objects smoothing (see Figure 27) performs better results. Following Figure 31 illustrates results of that algorithm. First row represents elevation lines restored with *Smooth-2* restoration (see 4.4), where smoothing rank parameter equal to 6 (left image) and 5 (right image) respectively. Second row contains results of *Smooth-1* restoration where rank parameter equal to 6 (left image) and 5 (right image) respectively.

Smooth-2 restoration with rank parameter 6

Smooth-2 restoration with rank parameter 5

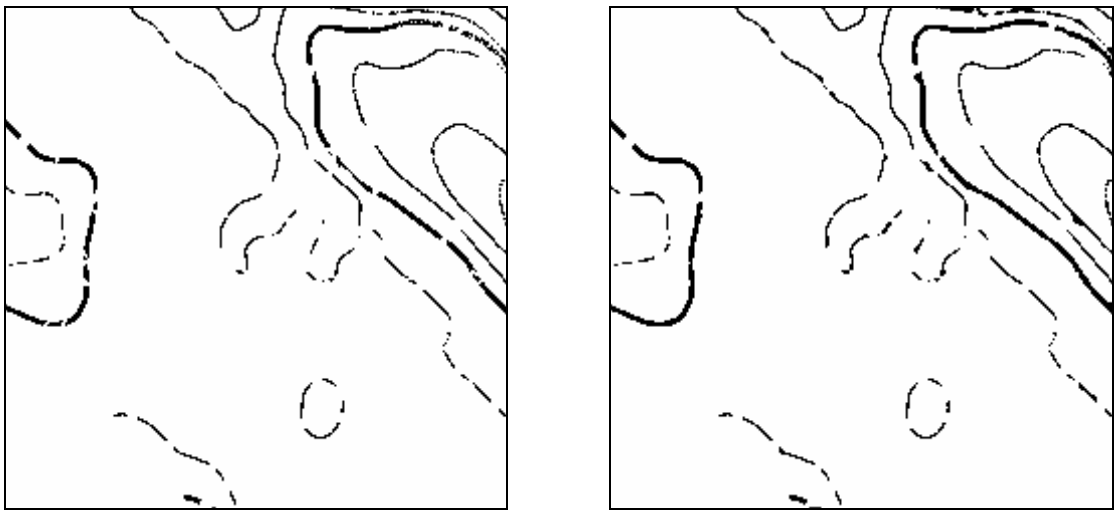
Smooth-1 restoration with rank parameter 6

Smooth-1 restoration with rank parameter 5

Figure 31: Restoration of elevation lines with object smoothing

You can see that though elevation lines were not restored completely, restoration algorithm does not corrupt visual appearance of the layer. Best results were archived by *Smooth-2* algorithm with rank parameter 5 which succeeds in several restoration of the layer. Further we will refer to that algorithm modification as *Contours* algorithm.

Experiments with sample images showed that proposed morphology-based technique does not reconstruct structure of the layer effectively. Restoration algorithms which were effective when applying to solid region layers are not applicable to elevation lines layer (Figure 30) or do not provide effective restoration (Figure 31). In Figure 32 you can compare reconstructed layer with best visual appearance (first row second column of Figure 31) with the corrupted layer.



Corrupted elevation lines | Elevation lines restored

Figure 32: Elevation lines. Corrupted and restored

It is easy to see that restored layer do not differ from the corrupted significantly. And there is no reason to expect significant compression improvement from that restoration technique. Therefore we can conclude that restoration of elevation lines could be skipped because it requires more complicated techniques than mathematical morphology can provide.

6 Removing a single layer from a map

The task of restoration of layers arises also when there is a need for removing a layer (or several layers) from a map. For example some less important layers, like e.g. elevation lines could be unnecessary to the map user driving a car, moreover they can make difficulties for reading a map. We can not just remove that layer because of artifacts it leaves on down-laying layers. Therefore similar restoration technique should be applied.

6.1 Task definition

The task could be formulated in a following way:

> *For a given multi-layer map image and a layer from the map to be removed, design a proper technique that eliminate artifacts (resulting from such removal) from underlying layers. The main quality criterion is the visual appearance of the map without removed layer.*

Consider a N-layer map $\mathfrak{M} = \{L_1, ..., L_r, ..., L_N\}$, where r is the number of layer to be removed. The task is to design an operator $\psi(L)$ such that combined map image $\mathcal{M}$ such that

$$\mathcal{M} : \{L_1, ..., L_{r-1}, \psi(L_{r+1}), ..., \psi(L_N)\} \xrightarrow{C} \mathcal{M}$$

would be as much close to the original multi-layer map as possible.

6.2 Solution

The task of removing of a layer is a subtask of layer restoration. We just have to restore all layers below the removing one using one of techniques proposed in Chapter 4. The removing is performed as follows:

```
Algorithm of removing LayerX
 Decompose combined map image into a set of Binary layers
 For each Binary layer below LayerX do
   Create Mask
 EndFor
 For each Binary layer below LayerX and Mask do
   Apply reconstruction operator
 EndFor
```

The algorithm is illustrated in Figure 33.

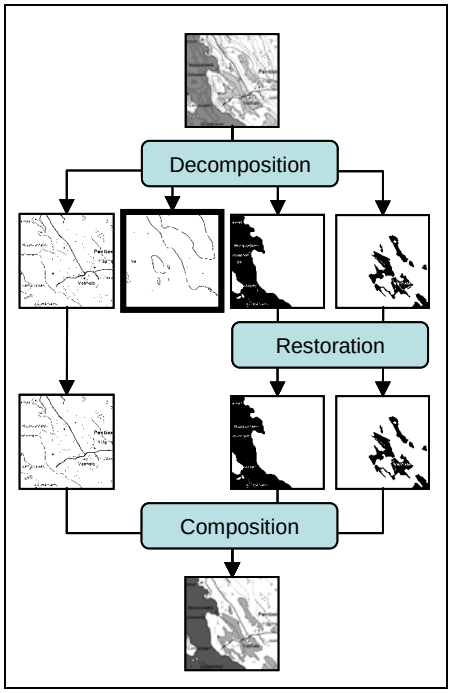
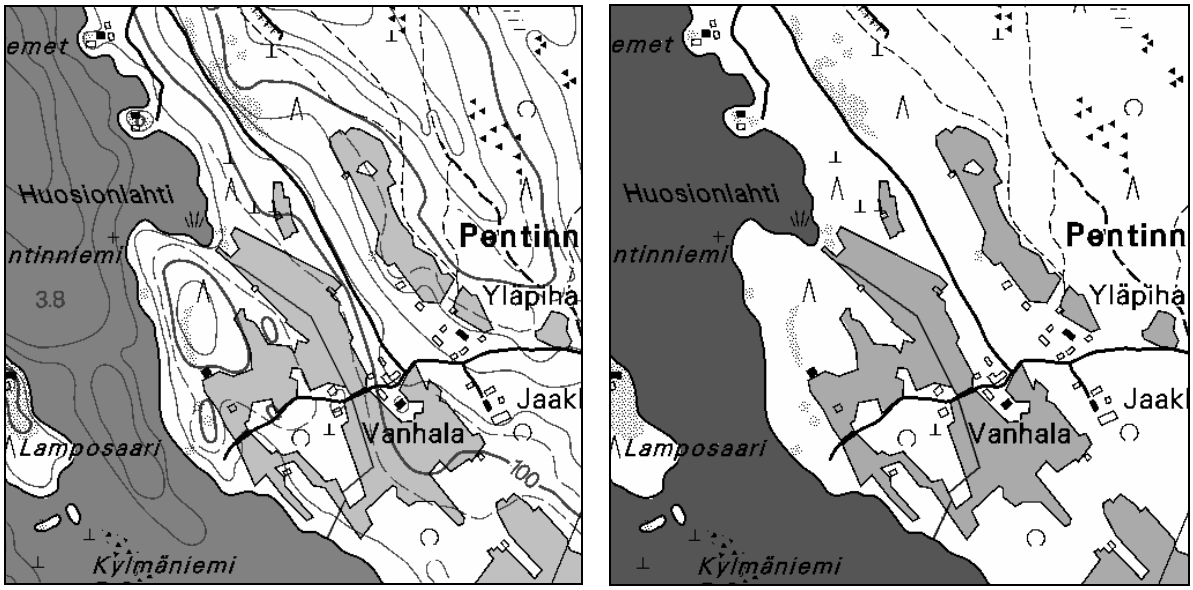


Figure 33: Removing algorithm

Figure 34 illustrates the result of algorithm described above. Here *Basic* algorithm was used as the restoration technique.

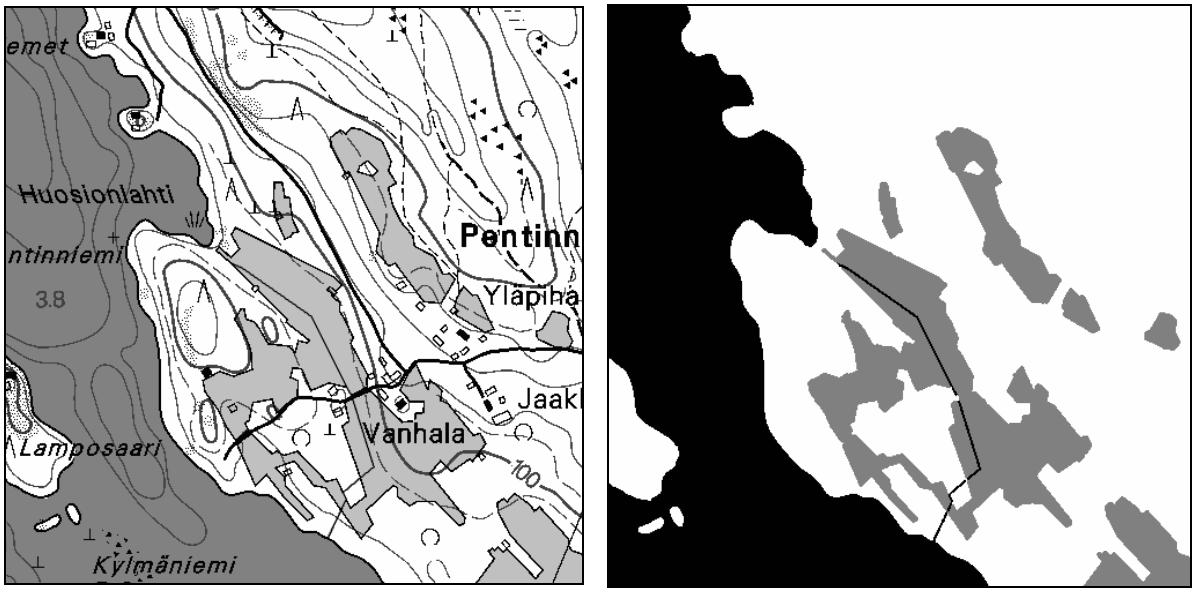


Original map Map with elevation lines removed

Figure 34: Removing elevation lines

We can see that algorithm effectively removed undesired layer from the image (elevation lines in our case) and suppressed artifacts on the down-laying layers, performing good visual appearance. Map readability simplified a lot.

Further, to illustrate our approach we can remove *Basics* layer too. Take a look at the following Figure 35.



Original map

Map with basic and elevation lines removed

Figure 35: Removing basics and elevation lines

Here is another example of removing unnecessary layers. Following map fragment (Figure 36) consists of 5 semantic layers instead of 4. The upper layer is a layer representing territory borders (districts, private areas etc.). Other layers below this one are the same as it had been considered before. Left image on the Figure 36 presents some urban region. Naturally, most of users do not need elevation lines information or information about territory borders when traveling in a city. Therefore it is naturally to remove these layers from the map image.

Original map

Map with borders and elevation lines removed

Figure 36: Removing territory borders and elevation lines

7 Implementation aspects

Following chapter discusses briefly the implementation details of described above algorithms briefly. First we discuss how mathematical concepts proposed in Chapters 1.1 and 3 could be implemented. First subchapter covers the implementation of images, structuring elements, and morphological operators. Second subchapter discusses software implementation of proposed theoretical concepts. Also NIH ImageJ package used in implementation as a base software tool is described briefly.

7.1 Implementing mathematical concepts

7.1.1 Image concept

The *binary image space E* (the space of all possible image pixel locations) is defined as $E = \mathbb{Z}^2$, and the *binary image* X – as a set $X \subseteq E$ of pixel locations:

$$X = \left\{ h = \overrightarrow{(i,j)} \,\middle|\, X(h) = 0, \quad h \in \mathbb{Z}^2 \right\}$$

For simplicity we will use the grayscale image model (in which pixel values range from 0 to 255) to represent binary image so that black pixel will have minimum intensity (0 value) and white pixel will have maximal intensity (255 value). In this representation we assume the following characteristic function $X(h): \mathbb{Z}^2 \to \{0, 255\}$

$$X(h) = \begin{cases} 0, \textit{if } h \in X \\ 255, \textit{otherwise} \end{cases}$$

We will model the binary images: the image X as a rectangular matrix $[X]_{W,H}$ of pixels in which, an image pixel is black and background pixel is white. Then we can write:

$$[X]_{W,H} = \left\{ x_{i,j} \,\middle|\, x_{i,j} = X((i,j)), i \in [1,W], \; j \in [1,H] \right\}$$

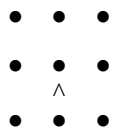
where W and H are the image horizontal and vertical dimensions respectively.

In the following we assume that there is an established relationship between image X (which is a set of pixel location), the model of image (a matrix of pixel values $[X]_{W,H}$), characteristic function $X(h)$ yielding the value of the pixel at a given location $h = \overrightarrow{(i,j)}$, and the pixel value $x_{i,j} = X\left(\overrightarrow{(i,j)}\right)$. We will also use the grayscale image representation as agreed before

7.1.2 Representation of a structuring element

We represent the structural element A as a set of pixel locations relative to the origin (in practice it is implemented as two arrays for each dimension respectively).

For example, for a given structural element:



the set of pixel location is

$$A=\begin{Bmatrix}(-1,-1), & (0,-1), & (1,-1),\\ (-1,0), & (0,0), & (1,0),\\ (-1,1), & (0,1), & (1,1)\end{Bmatrix},$$

and it is implemented as two arrays:

$$A^x=\{-1,0,1,-1,0,1,-1,0,1\},$$

$$A^y=\{1,1,1,0,0,0,-1,-1,-1\}$$

7.1.3 Implementation aspects of morphological operators.

Dilation

Recall definition of dilation operator.

$$\delta_A(X)=\left\{h\in E \mid \tilde{A}_h \cap X \neq \varnothing\right\}$$

Under our image model the dilation $Y=\delta_A(X)$ can be computed as:

$$[Y]_{W,H}=\left\{y_{i,j}=\min_{k=1..\|A\|}\left\{X\left(\overrightarrow{(i,j)}-A[k]\right)\right\} \mid i\in[1,W], \quad j\in[1,H]\right\}$$

Here:

- $A[k]$ denotes the *k*-th element of set A and $\left(\overrightarrow{(i,j)}-A[k]\right)$ is a translation of pixel location $\overrightarrow{(i,j)}$ along vector $-A[k]$. The translation is negative because of the use of a transposed structuring element $\tilde{A}$;
- the $\min\{\cdots\}$ equation is interpreted so that pixel at location (i,j) will be black (has value 0) if any of pixels at locations $\overrightarrow{(i,j)}-A[k]$ is black.

In practice $y_{i,j}$ is computed using $x_{i,j}$ as follows:

$$y_{i,j} = \min_{k=1..\|A\|} \left\{ X\left(\overrightarrow{(i,j)} - A[k]\right)\right\} = \min_{k=1..\|A\|} x_{i-A_x[k],\ j-A_y[k]}$$

Erosion

Similarly for erosion:

For

$$Y = \varepsilon_A(X) = X \ominus A = \left\{h \in E \middle| A_h \subseteq X\right\}$$

it follows that

$$y_{i,j} = \max_{k=1..\|A\|} \left\{ X\left(\overrightarrow{(i,j)} + A[k]\right)\right\} = \max_{k=1..\|A\|} x_{i+A_x[k],\ j+A_y[k]}$$

the $\max\{\cdots\}$ equation is interpreted so that pixel at location (i,j) will be black if and only if all of pixels at locations $\overrightarrow{(i,j)} + A[k]$ are black.

Rank operator

Similarly for rank operator:

$$Y = \rho_{A,s}(X) = \left\{h \middle|\ X \cap A_h \text{ contains at least } s \text{ elements}\right\}.$$

Then

$$y_{i,j} = \begin{cases} 0, \textit{if } \underset{k=1..\|A\|}{\text{count}0} \left\{ X\left(\overrightarrow{(i,j)} + A[k]\right)\right\} \geq s \\ 255, \textit{otherwise} \end{cases}$$

here $\text{count}0\{\cdots\}$ is a function that counts number of black pixels in a given set of locations.

As it has been shown before, every morphological operator (even though erosion and dilation) can be represented using rank operator. It means that when implementing morphological operators we can do it uniformly using rank as a base operator.

7.2 Software implementation

All algorithms considered in chapters 4, 5 and 6 were implemented in Java using NIH ImageJ framework. As our algorithms are based on mathematical morphology, first we implemented a set of morphological operators and structuring elements. Further, basing on morphological operators, we implemented algorithms of the current thesis as ImageJ macro-language scripts. This allowed to concentrate on the realization of actual layer processing and to avoid a lot of specific programming problems and implementation details.

7.2.1 ImageJ in general

ImageJ is a public domain Java image processing program inspired by NIH Image for the Macintosh. It runs, either as an online applet or as a downloadable application, on any computer with a Java 1.1 or later virtual machine.

It can display, edit, analyze, process, save and print 8-bit, 16-bit and 32-bit images in TIFF, GIF, JPEG, BMP, DICOM, FITS or "raw" formats. It supports "stacks", a series of images that share a single window. It is multithreaded, so time-consuming operations such as image file reading can be performed in parallel with other operations.

ImageJ was designed with an open architecture that provides extensibility via Java plugins. Custom acquisition, analysis and processing plugins can be developed using ImageJ's built in editor and Java compiler. User-written plugins make it possible to solve almost any image processing or analysis problem.

The source code is freely available. The author, Wayne Rasband (wayne@codon.nih.gov), is at the Research Services Branch, National Institute of Mental Health, Bethesda, Maryland, USA.

7.2.2 Plugins

Functions provided by ImageJ could be extended by plugins. In fact, the most of ImageJ standard functions are implemented as plugins. Plugins are loadable code modules that extend the capabilities of ImageJ. Using plugins you can implement any needed algorithm concentrating on implementation of how actually image has to be processed. All background work when reading, displaying, magnifying, applying ROI, etc. will be handled by ImageJ. For a developer, plugin is a Java class implementing necessary interface placed into a certain folder. Plugins could be written in ImageJ Java editor or in any existing Java environment, recorded with ImageJ macro recorder or written on ImageJ internal macro language.

All morphological operators on which our algorithms are based were implemented as ImageJ plugins. This allows using them in ImageJ macro language, which is a very flexible tool for combining simple image operators into a complicated algorithm.

7.2.3 Implementing rank operator

As it had been discussed, rank operator was used as a base operator when we were implementing all other operators such as dilation, erosion, etc. For example dilation operator could be implemented as rank operator on the same structuring element with rank parameter equal to 1. Therefore, first we implemented rank operator as ImageJ plugin and then used it in implementation of all other operators. Here is a brief description of plugin

implementing rank operator. It tells us about its source file, input parameters, its output, example of usage in ImageJ and a brief description of how algorithm was implemented.

Source file:

Rank_.java

Input:

binary image

"rank" – integer rank parameter:

"element" – string name of structuring element

Output:

binary image

Usage:

```
IJ.run("Rank ", "rank=2 element='BLOCK_3X3'");
```

The plugin implements rank operator **Rank**(*Element*, *Rank*), where *Element* is a structuring element defining the neighborhood and *Rank* is rank parameter, sets pixel to be black if the local neighborhood of the current pixel location contains at least *Rank* black pixels.

Algorithm:

```
Procedure Rank(A : structuring element, s : integer)
For each y do
   For each x do
         Count pixels in intersection of A
         and source image at current pixel location (x, y)
         If that number is bigger than s then
               Put black pixel to a destination
         Else
               Put white pixel to a destination
   EndFor
EndFor
End
```

Rank plugin is implemented using 2 nested functions: `run()` - interface function which takes input image and parameters and then calls `doFilter()` function, which takes source and destination `ImageProcessor` classes, structuring element `Element` and integer rank parameter and do the actual ranking. If rank parameter is greater than `Element.length` or less than 0 then it is assumed to be equal to `Element.length`. `ImageProcessor` is internal ImageJ class implementing handling of the image data, e.g. accessing pixels values,

changing the type of the image, etc. Besides that it has a lot of predefined frequently used methods like threshold, histogram functions, etc.

Other plugins, implementing other rank based morphological operators, are implemented as a wrapper classes. These classes handle user given parameters (typically it is just a type of structuring element), calculate appropriate rank parameter value and then call rank plugin to perform the operation. This way we can avoid concentrating on the implementation details and use once developed rank operator to implement all others. Such approach also simplifies the bag tracking. If some error had been found once in a source code, then it can be easily localized in rank plugin and the update will be naturally spread to all other operators without any recompilation.

7.2.4 Developed plugins

In order to use mathematical morphology as a tool we have implemented a set of following morphological operators: *Rank* operator (the base morphological operator), *Erosion*, *Dilation*, *Open*, *Close*, *OpenClose*, *CloseOpen*, *Alternating Sequential Filters*, *Rank-max Opening* and *Rank-min Closing*. All operators support 10 different structuring elements. Besides that, the following extensions of classical morphological operators proposed by Heijmans [H94, H95] were implemented: *OverFilter* operator, *UnderFilter* operator, *OverUnderFilter* and *UnderOverFilter* operators.

Restoration algorithms proposed in a current thesis were implemented as macro-language programs. Following the definition every algorithm requires as input two binary images – corrupted layer and conditioning mask. Algorithm applies restoration technique and produces two images – restored layer which actual output of the algorithm and modified mask which can be skipped.

8 Evaluation

8.1 Objectives of evaluation

The restoration techniques have been evaluated on a set of topographic color-palette map images. These map 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 fact gave us possibility to compare restored images with their original undistorted counterparts.

The performance of the proposed restoration techniques has been evaluated quantitatively and qualitatively.

- The objective of the qualitative evaluation was to compare the images visually and decide on the usability of the restoration techniques in applications where human visual factor plays prevailing role.
- The first objective of quantitative evaluation was to compute and compare the differences between the original and corrupted layers and original and restored layers. The second objective was to calculate the compression performance in applications were map image is compressed as a set of its binary layers. All selected restoration methods were evaluated against three major compression techniques: LZ (PNG), ITU Group 4 (TIFF), JBIG. For each of these compression methods we measured the compressed data size for the original semantic layers, similarly for the corrupted binary layers after decomposition, and similarly for reconstructed binary layer using each of the reconstruction method.

In the following sub-sections we present the algorithms we have evaluated, the test set, the brief description of the compression techniques and image difference measures used in the evaluation process. Finally we give the results of the evaluation.

8.2 Restoration algorithms

We experimented with various modifications of the reconstruction algorithm, and finally chosen most prominent candidates for the evaluation. These four chosen restoration methods are shown in Table 1:

Table 1: Restoration methods

Method:	Basic	Soft	Smooth-1	Smooth-2
Equation (reference)	See 4.3	See 4.6	See 4.4	See 4.4
Structural element for dilation	cross 3x3	block 3x3	cross 3x3	cross 3x3
Structural element for erosion	cross 3x3	block 3x3	cross 3x3	block 3x3
Algorithm modification	none	Soft dilation and erosion	Smoothing added	Smoothing added
Performance	low	average	good	good
Passes over the image	2	2	3	3

The methods are as follows:

Basic

This algorithm is a version of basic algorithm where structuring element for erosion and dilation is *cross 3x3*. This algorithm was chosen due its computational simplicity. Although during prior investigation it has demonstrated relatively low performance.

Soft

The algorithm is a version of basic algorithm where erosion $\varepsilon_A(X)$ and dilation $\delta_A(X)$ is replaced with their "soft" counterparts $\varepsilon_A(X,2)$ and $\delta_A(X,2)$ respectively. The *block 3x3* structuring element is used. Rank parameter for soft operations has been chosen 2. This algorithm modification demonstrated average performance with average computational complexity.

Smooth-1

The algorithm is a version of basic algorithm where structuring element of erosion and dilation is *cross 3x3* and smoothing of objects with rank parameter equal to 6 is implemented. Algorithm demonstrated good performances but computational complexity is relatively high.

Smooth-2

Algorithm is very similar to *Smooth-1* but structuring element of erosion is *cross 3x3* and of dilation is *block 3x3.* Also the smoothing of objects with rank parameter equal to 6 is implemented. Algorithm showed good performance but computational complexity is also the highest.

Contours

Elevation layer are reconstructed using an own reconstruction algorithm. This algorithm is a modification of *Smooth-2* algorithm with smoothing rank parameter equal to 5. All other characteristics of that algorithm are the same as *Smooth-2*. The method was chosen due the best visual performance when applying to *Elevation* layer. Other methods presented very poor performance when restoring elevation lines, therefore that layer was reconstructed this that modification of *Smooth-2* only. Although *Smooth-2* is another modification, for simplicity we will refer to *Elevation* algorithm as to *Smooth-2* when talking about *Elevation* layer.

8.3 Test set

The test set includes five randomly chosen images from the "NLS Basic Map Series 1:20000" corresponding to the map sheets *No/No* 431306, 124101, 201401, 263112, and 431204. Each map image consists of four binary component layers, each has the size of 5000 × 5000 pixels. The images are denoted as *ImageX*, and the layers as *LayerX*, where *Layer* is the layer name, and *X* is the relative domain number from 1 to 4 in a given order. The layer names are following:

- *Basic* – topographic image, supplemented with communications networks, buildings, protected sites, benchmarks and administrative boundaries;
- *Elevation* – elevation lines;
- *Water* – lakes, rivers, swamps, water streams;
- *Fields* – agricultural areas.

8.4 Image difference measures

We will use two following image quality measures to evaluate the quality of restoration.

Measures:

NMAE (*Normalized Mean Absolute Error*) – Hamming Distance:

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

Where $\mathbb{H}$ and $\mathbb{W}$ are image dimensions.

The measure depends on the amount of coinciding pixels contained by images.

NWMAE (*Normalized Weighted Mean Absolute Error*):

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

Where $\mathbb{H}$ and $\mathbb{W}$ are image dimensions and $W_{i,j}$ is a weighting coefficient, given by equation

$$W_{i,j} = \sum_{l=-W_y}^{W_y} \sum_{k=-W_x}^{W_x} w_{k,l} \left| x_{i+k,j+l} - y_{i+k,j+l} \right|$$

Where $w_{k,l}$ are elements of $\left[2W_x + 1\right] \times \left[2W_y + 1\right]$ matrix W called *weighted masking element*. W_x and W_y are half-dimensions of matrix W. The origin of W is assumed to be at the center of the matrix. For example ($W_x = 1$, $W_y = 1$, $\wedge$ sign points the origin),

$$W = \begin{pmatrix} 1 & 2 & 1 \\ 2 & \underset{\wedge}{1} & 2 \\ 1 & 2 & 1 \end{pmatrix}.$$

Idea of this measure is to improve *NMAE* by take into consideration the neighborhood of every pixel. The neighborhood is determined by weighted masking element W. The degree of how neighboring pixel affects the measure depends on its weight in a matrix W.

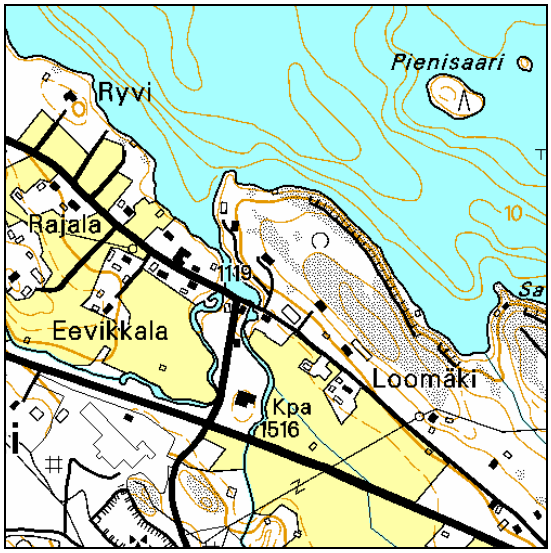
Equations above assume that pixel values are 1 for foreground and 0 for background. In our particular implementation, when pixel values are 0 for foreground and 255 for background, we have to use $255 \cdot \mathbb{H} \cdot \mathbb{W}$ scaling coefficient instead of $\mathbb{H} \cdot \mathbb{W}$ in order to keep measure value in 0 to 1 range.

8.5 Qualitative evaluation

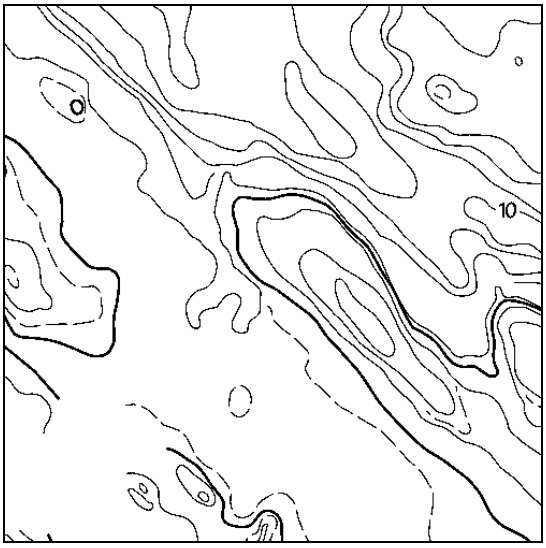
The object of qualitative evaluation is to compare the performance of restoration techniques visually. Figures below represent the performance of restorations algorithms applied to 500x500 map fragment shown on Figure 37. Each figure contains the original (uncorrupted), corrupted and four restored layers for *Elevation* (Figure 37), *Waters* (Figure 38) and *Fields* (Figure 39) layers respectively.

For Waters and Fields layer all restoration techniques presented similar good-looking results. We found that proposed techniques are equally suitable for restoration tasks where the main quality criterion is visual appearance of restored layer. For example, for the task of layer removing, described in Chapter 6. In contrast, visual appearance of restored Elevation layer is not as handsome as appearance of other layers. This caused by the

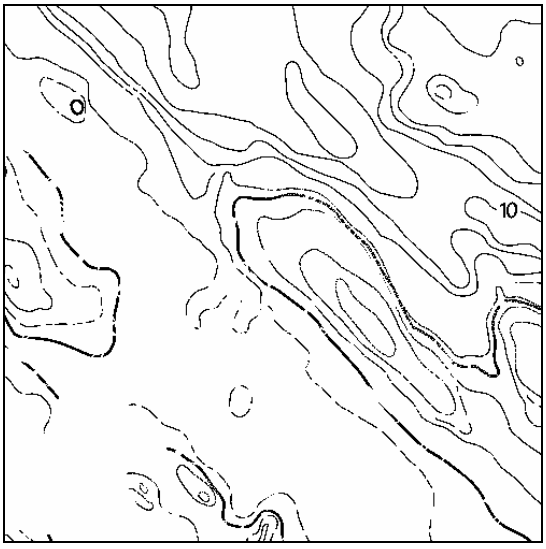
complicated semantic structure of that layer and we conclude that more sophisticated heuristics should be considered for that kind of restoration.



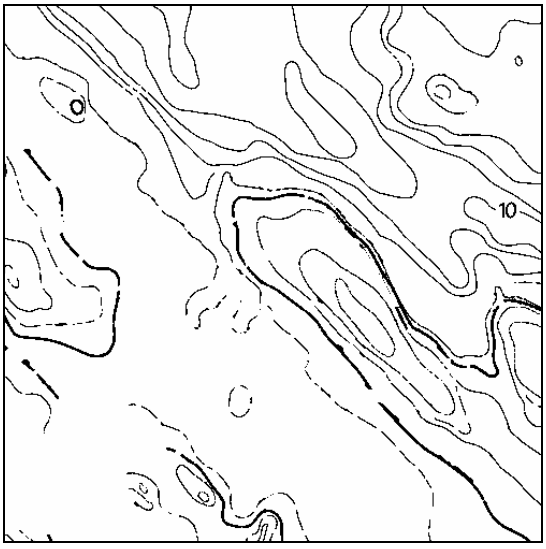
Combined map image



Original (uncorrupted) *Elevation* layer

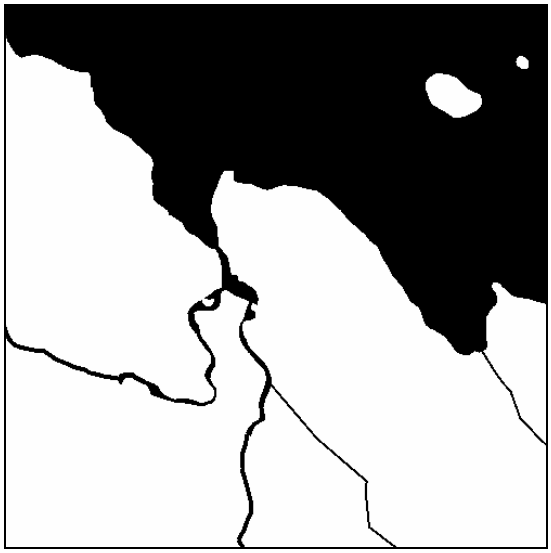


Corrupted *Elevation* layer

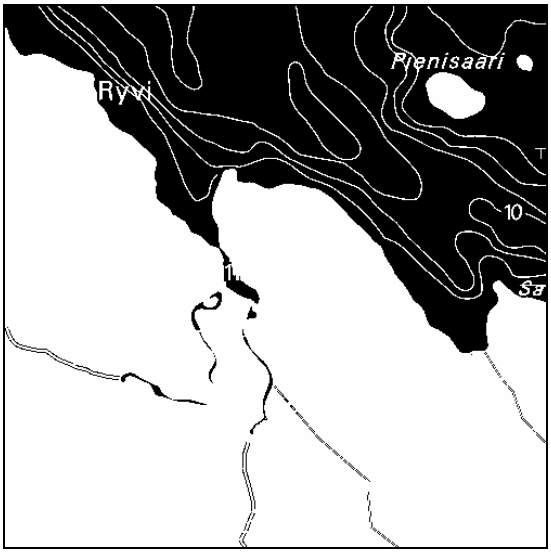


Reconstructed by *Contours* algorithm

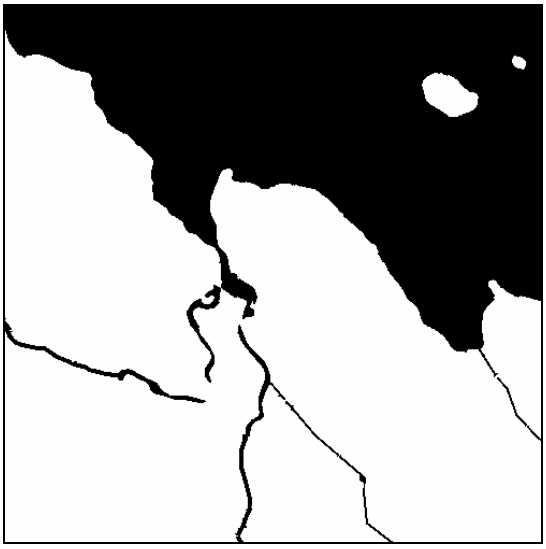
Figure 37. Original combined map image and *Elevation* layer.



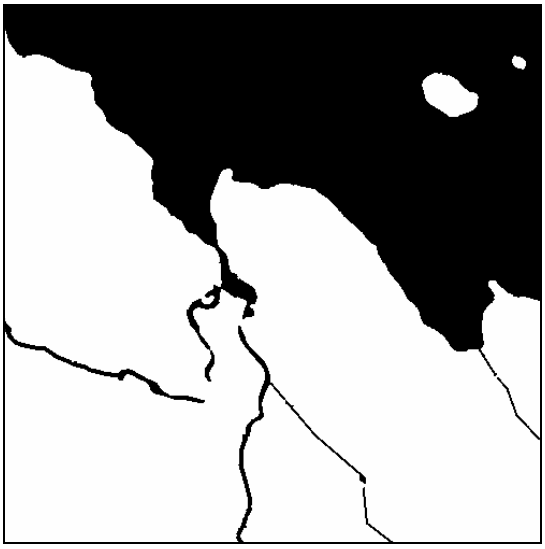
Original



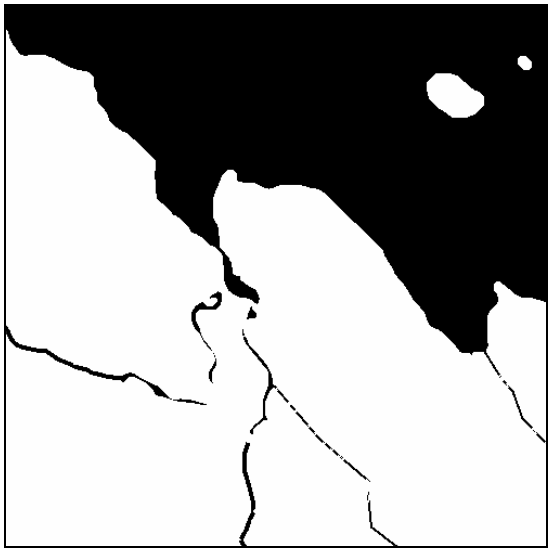
Corrupted



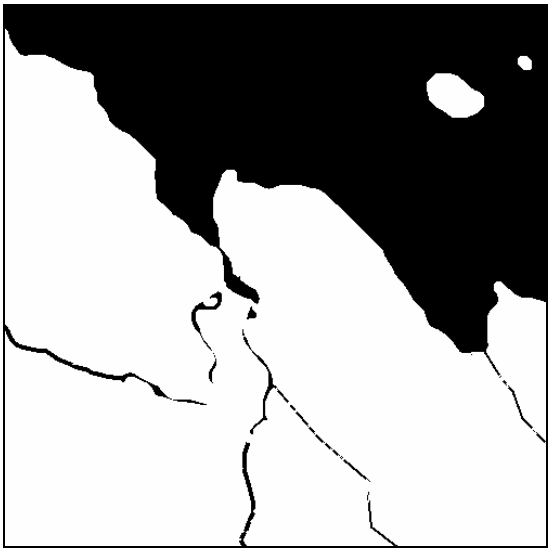
Basic



Soft

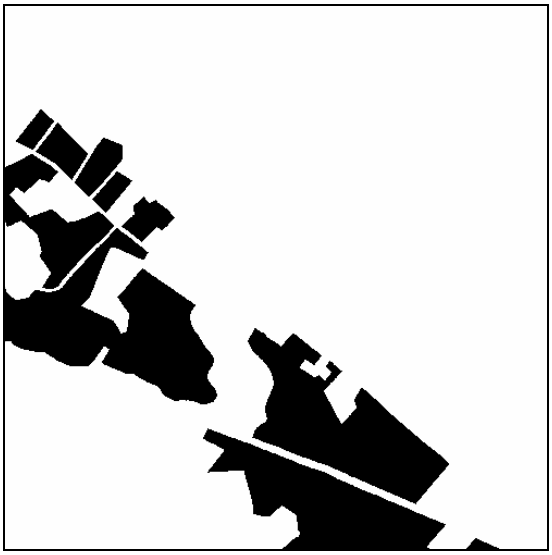


Smooth-1

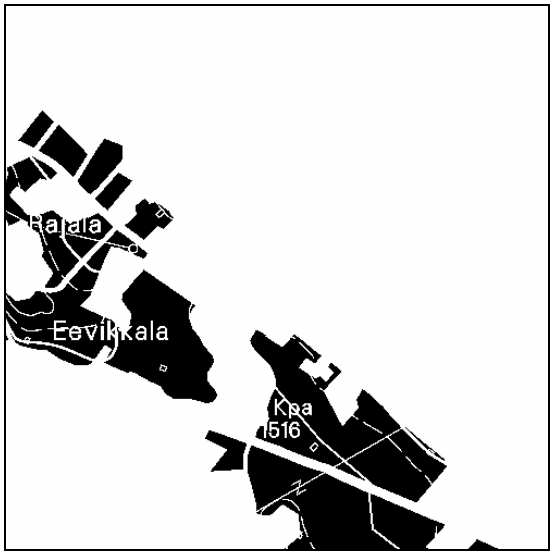


Smooth-2

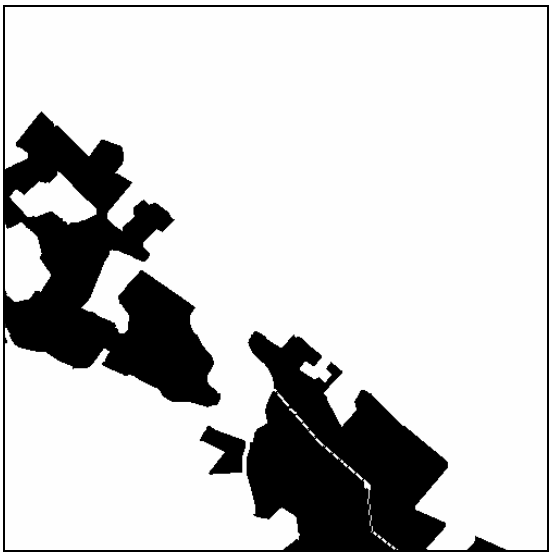
Figure 38. Restoration of *Waters* layer.



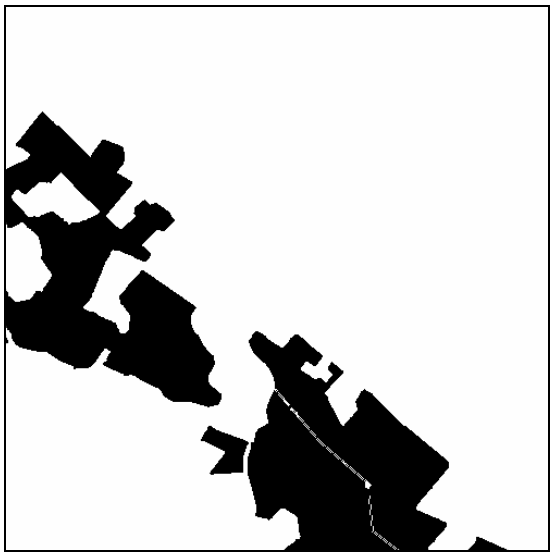
Original



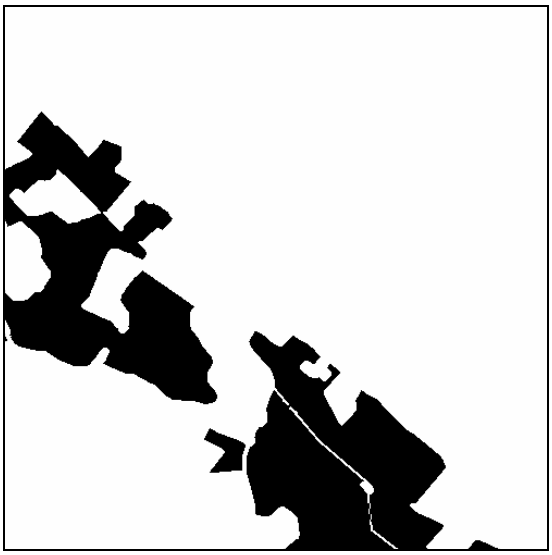
Corrupted



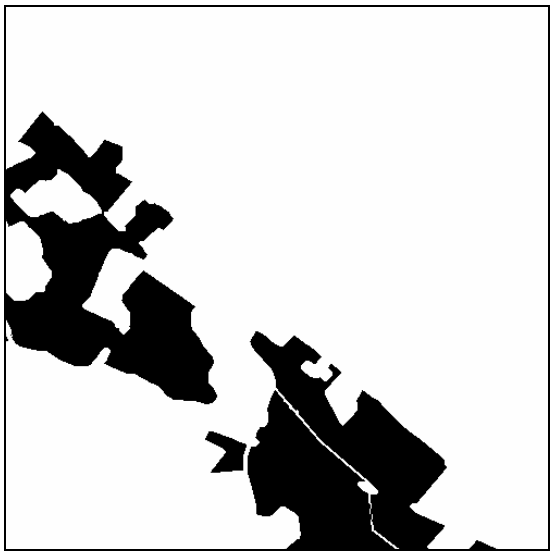
Basic



Soft



Smooth-1



Smooth-2

Figure 39. Restoration of *Fields* layer.

8.6 Compression results

This subsection contains compression results for PNG, TIFF and JBG file formats. Notice that *Basic* layer is not corrupted and therefore do not require any restoration. Also *Elevation* layer was restored only with *Smooth-2* algorithm due unacceptable performance of other techniques. Therefore the results for *Basic* and *Elevation* layers are marked with "—" sign to indicate that given algorithm was not applied to the layer.

On the other hand, when calculating total compressed size (the sum of all compressed layers) for each restoration technique, *Basic* and *Elevation* layers must be taken into an account too, although they were not restored at all (like *Basic* layer) or e.g. *Elevation* layer was restored only with *Smooth-2*. Therefore, when calculating total for *Basic* layer we will use "Semantic layers" number – a number for uncorrupted compressed layer. And for *Elevation* layer *Smooth-2* number (when *Elevation* was restored with *Smooth-2*) will be used in all other restoration techniques.

8.6.1 Results per reconstructed layer

We know that different types of layers have theirs distinct morphological structure. Therefore we expect that different restoration techniques will affect them in a different way. In this subsection results of experiments are classified with respect to the types of layers. This allows evaluating the performance of restoration algorithms depending on the layer which it had been applied to.

Results for Elevation:

As it had been discussed in Chapter 5, the *Elevation* layer was reconstructed with only one algorithm *Smooth-2* because of poor visual performance of alternatives. The experimental results showed, as it also had been predicted in Chapter 5, that reconstruction do not affect the layer significantly and the compression improvement ratio is only about 4% in the best case or even became negative in the worst. This allows to state that mathematical morphology is not a tool for restoration of such kind of layers as *Elevation*, and if another restoration technique is not applicable, then it can be skipped completely.

Following Table 2 represents file sizes in bytes and compression improvement ratios in percents for original, corrupted and restored *Elevation* layer. Figure 40 illustrates data presented in a Table 2 graphically. Figure 41 illustrates compression improvement ratios presented in Table 2.

Table 2: Results of restoration of “Elevation” layer

Compression algorithm	Semantic layers	Corrupted layers	Reconstructed layers	
			Smooth-2	%
Uncompressed	25 000 000	25 000 000	25 000 000	—
PNG	3 302 040	3 232 502	3 247 832	0.47
TIFF	1 843 244	1 925 352	1 859 736	3.41
JBIG	944 838	1 077 690	1 036 554	3.82

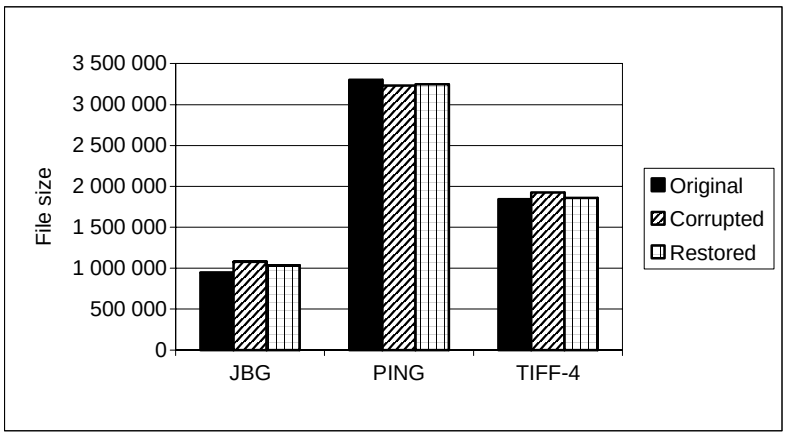


Figure 40: File sizes of “Elevation” layer compressed with JBIG, PNG and TIFF. Bars represent original, corrupted and restored with “Elevation” algorithm layers.

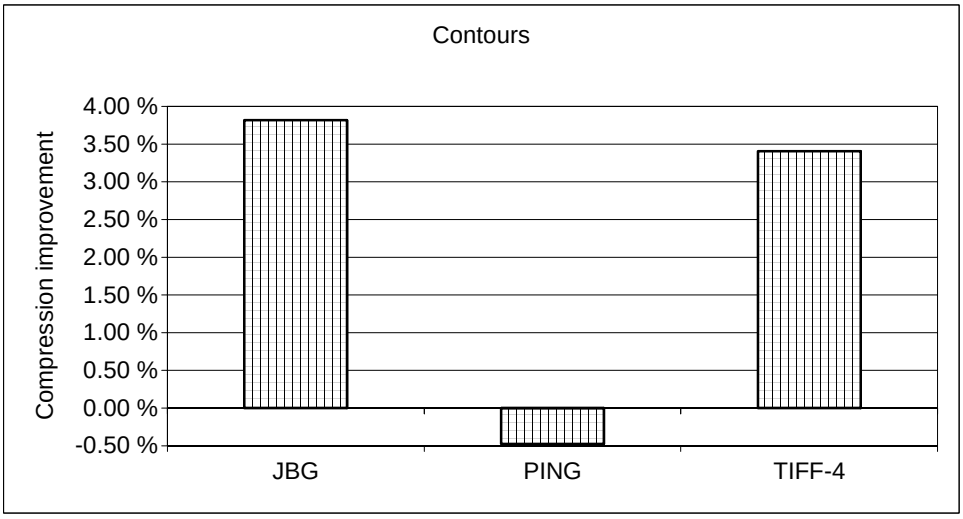


Figure 41: Compression improvement of “Elevation” algorithm for “Elevation” layer

Results for Waters:

Waters layer was reconstructed with four different algorithms. Since that layer has more regular morphological structure, reconstruction demonstrated significantly better

compression improvement up to 50%. Experiments showed that *Soft* restoration algorithm is the best for *Water* layer restoration. Following Table 3 represents file sizes in bytes and compression improvement ratios in percents for original, corrupted and restored *Waters* layer. Figure 42 illustrates data presented in a Table 3 graphically. Figure 43 illustrates compression ratios presented in Table 3.

Table 3: Results of restoration of "Water" layer

Compression algorithm	Semantic layers	Corrupted layers	Reconstructed layers							
			Basic	%	Soft	%	Smooth1	%	Smooth2	%
Uncompressed	25000000	25000000	25000000	—	25000000	—	25000000	—	25000000	—
PNG	1 526 431	1 703 447	1 576 740	7.44	1 539 062	9.65	1 561 013	8.36	1 559 218	8.47
TIFF	669 444	1 428 656	732 704	48.71	674 520	52.79	892 104	37.56	896 906	37.22
JBIG	325 334	549 032	402 749	26.64	372 918	32.08	444 094	19.11	446 740	18.63

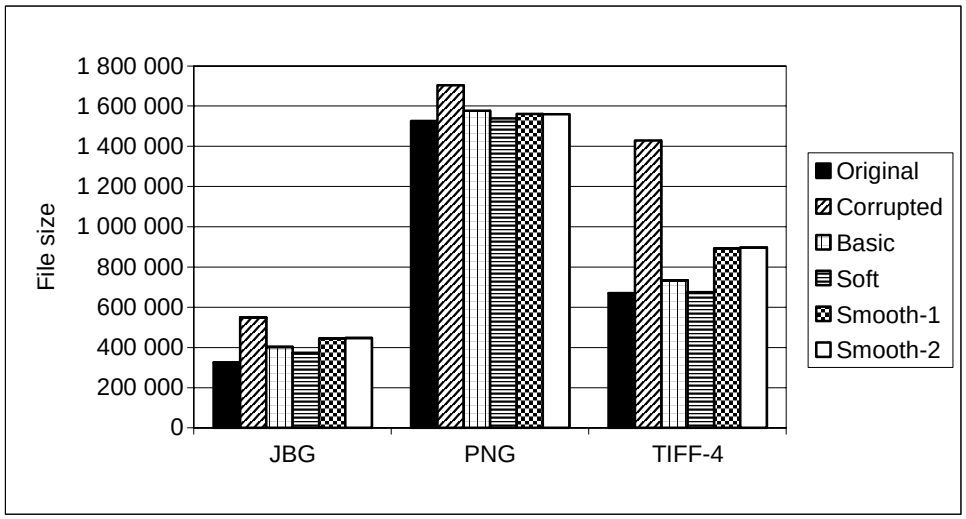


Figure 42: File sizes of "Water" layer compressed with JBIG, PNG and TIFF. Bars represent original, corrupted and restored with four restoration algorithms layers.

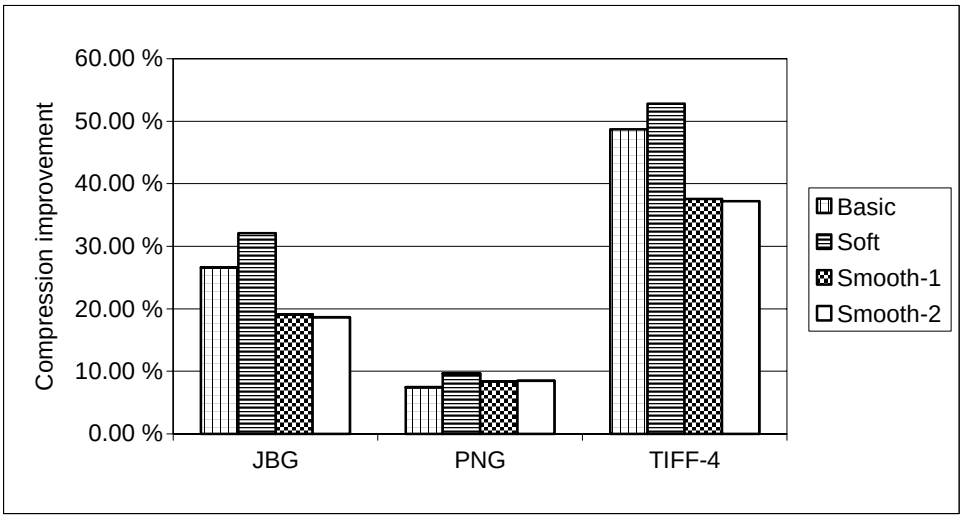


Figure 43: Compression improvement of four different restoration algorithms for "Water" layer when compressing with JBIG, PNG and TIFF respectively.

Results for Fields

Fields layer was also reconstructed with four different algorithms. Reconstruction achieve d significant (up to 55%) compression improvement for all compression techniques. Experiments showed that *Smooth-1* algorithm is the best when reconstructing *Fields* layer.

Following Table 4 represents file sizes in bytes and compression improvement ratios in percents for original, corrupted and restored *Fields* layer. Figure 44 illustrates data presented in a Table 4 graphically. Figure 45 illustrates compression ratios presented in Table 4.

Table 4: Results of restoration of "Fields" layer

Compression algorithm	Semantic layers	Corrupted layers	Reconstructed layers							
			Basic	%	Soft	%	Smooth1	%	Smooth2	%
Uncompressed	25000000	25000000	25000000	—	25000000	—	25000000	—	25000000	—
PNG	309 712	456 710	335 924	26.45	330 811	27.57	320 821	29.75	320 796	29.76
TIFF	99 622	196 456	118 696	39.58	119 484	39.18	105 388	46.36	105 718	46.19
JBIG	49 409	113 977	62 065	45.55	59 879	47.46	50 936	55.31	51 488	54.83

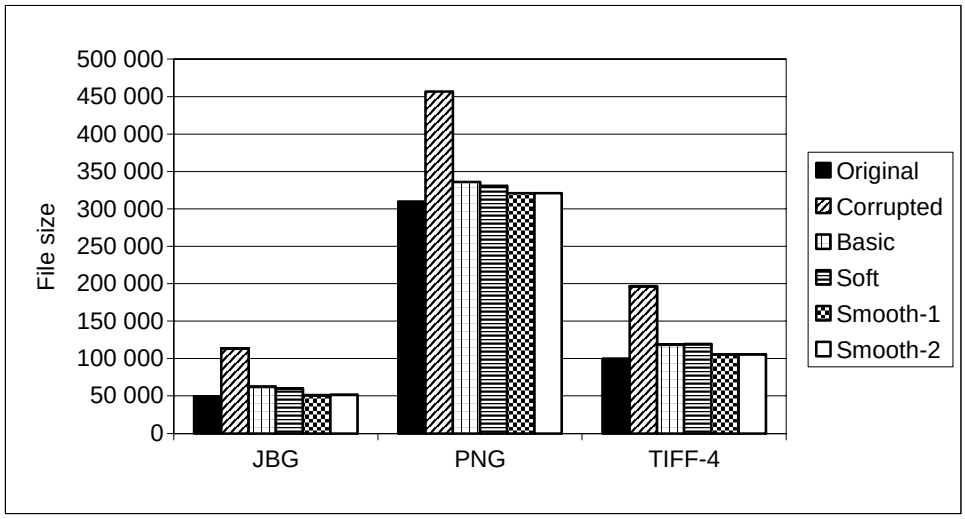


Figure 44: File sizes of "Fields" layer compressed with JBIG, PNG and TIFF. Bars represent original, corrupted and restored with four restoration algorithms layers.

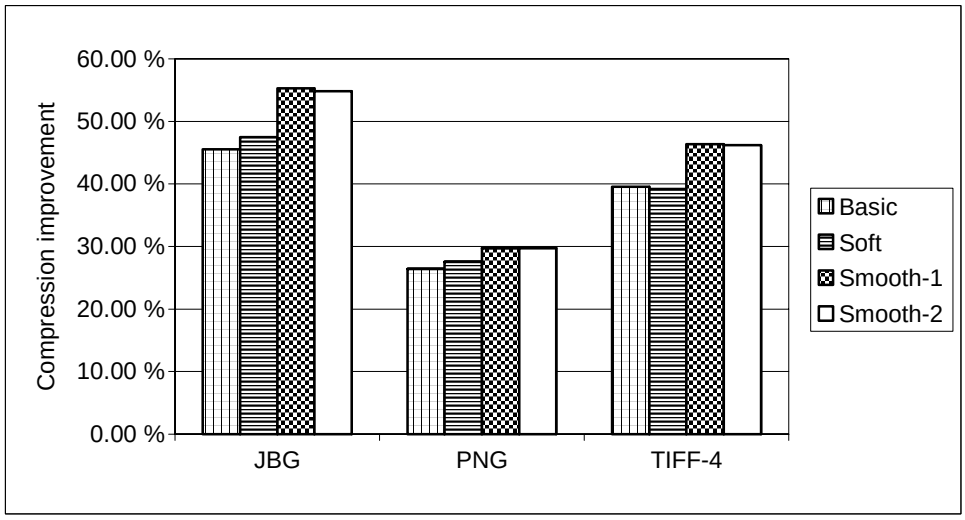


Figure 45: Compression improvement of four different restoration algorithms for "Fields" layer when compressing with JBIG, PNG and TIFF respectively.

8.6.2 Totals per compression methods

The following subsection classifies experimental results with respect to the applied compression technique. In a real Digital Spatial Library we can not vary the compression technique used. Therefore it makes sense evaluating total compression improvement performed by the restoration algorithm for a given compression method.

Compression results for PNG:

Examining experimental data for PNG compressing technique showed that *Soft* restoration algorithm presented the best (up to 3.20%) compression improvement. Table 5 presents the data for PNG compression.

Compression results for ITU G.4 (TIFF):

Examining experimental data for TIFF compressing technique showed that *Soft* restoration algorithm also presented the best (up to 13.1%) compression improvement. Table 6 presents the data for TIFF compression.

Compression results for JBIG:

Examining experimental data for JBIG compressing technique showed that *Soft* restoration algorithm also presented the best (up to 8.58%) compression improvement. Table 7 presents the data for JBIG compression.

Table 5: Experimental results for PNG compression

Test images	Semantic layers	Corrupted layers	Reconstructed layers			
			Basic	Soft	Smooth-1	Smooth-2
Basic 1-4	3 205 301	—	—	—	—	—
Elevation 1-4	3 302 040	3 232 502	—	—	—	3 247 832
Water 1-4	1 526 431	456 710	1 576 740	1 539 062	1 561 013	1 559 218
Fields 1-4	309 712	1 703 447	335 924	330 811	320 821	320 796
TOTAL (16)	8 343 484	8 597 960	8 365 797	8 323 006	8 334 967	8 333 147
Compression improvement	—	—	2.70 %	3.20 %	3.06 %	3.08 %

Table 6: Experimental results for TIFF compression

Test images	Semantic layers	Corrupted layers	Reconstructed layers			
			Basic	Soft	Smooth-1	Smooth-2
Basic 1-4	3 282 984	—	—	—	—	—
Elevation 1-4	1 843 244	1 925 352	—	—	—	1 859 736
Water 1-4	669 444	1 428 656	732 704	674 520	892 104	896 906
Fields 1-4	99 622	196 456	118 696	119 484	105 388	105 718
TOTAL (16)	5 895 294	6 833 448	5 994 120	5 936 724	6 140 212	6 145 344
Compression improvement	—	—	12.28 %	13.12 %	10.14 %	10.07 %

Table 7: Experimental results for JBIG compression

Test images	Semantic layers	Corrupted layers	Reconstructed layers			
			Basic	Soft	Smooth-1	Smooth-2
Basic 1-4	1 420 330	—	—	—	—	—
Elevation 1-4	944 838	1 077 690	—	—	—	1 036 554
Water 1-4	325 334	549 032	402 749	372 918	444 094	446 740
Fields 1-4	49 409	113 977	62 065	59 879	50 936	51 488
TOTAL (16)	2 739 911	3 161 029	2 921 698	2 889 681	2 951 914	2 955 112
Compression improvement	—	—	7.57 %	8.58 %	6.62 %	6.51 %

8.6.3 Total results

The Table 8 represents averaged total sizes and compression improvements of original, corrupted and four restored map sheets. Average total size is the average compressed file size of a map image representing a single map sheet. Figure 46 illustrates the data presented. Figure 46 illustrates compression improvement ratios presented in Table 8.

Table 8: Averaged total results

Compression algorithm	Semantic layers	Corrupted layers	Reconstructed layers							
			Basic	%	Soft	%	Smooth1	%	Smooth2	%
PNG	2 085 871	2 149 490	2 091 449	2.70	2 080 752	3.20	2 083 742	3.06	2 083 287	3.08
TIFF	1 473 824	1 708 362	1 498 530	12.28	1 484 181	13.12	1 535 053	10.14	1 536 336	10.07
JBIG	684 978	790 257	730 425	7.57	722 420	8.58	737 979	6.62	738 778	6.51

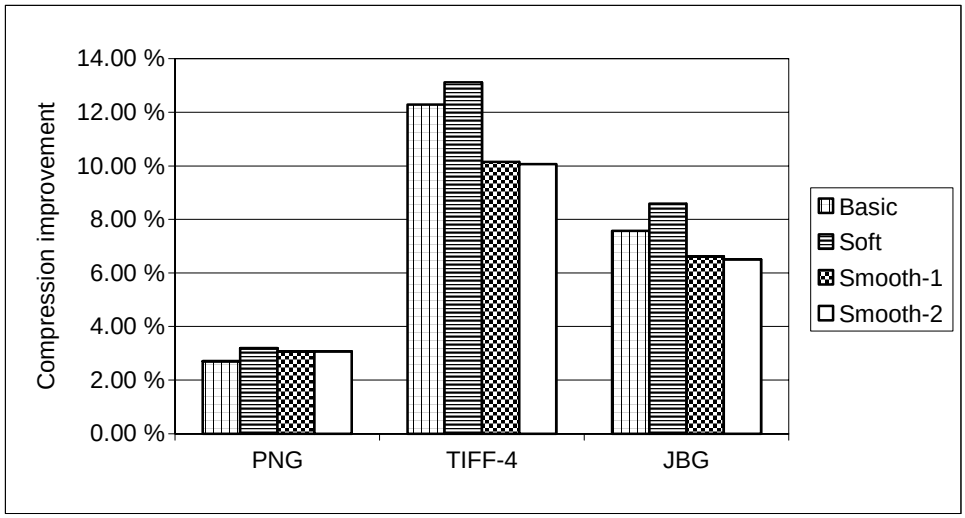


Figure 46: Averaged total compression improvement of restoration algorithms

8.7 Combined algorithm

Since analysis of restoration techniques for every single layer showed that *Soft* algorithm presented best compression improvement for *Waters* layer and *Smooth-1* presented the best improvement for *Fields* layer, we decided to propose and evaluate *Combined* algorithm. *Combined* algorithm is the algorithm, where *Elevation* layer is reconstructed with *Contours* algorithm, *Waters* layer is reconstructed with *Soft* algorithm, and *Fields* layer with *Smoth-1* algorithm. Using of best restoration techniques for every single layer has to improve total compression performance. The Table 9 represents averaged total file sizes and compression improvements for *Combined* algorithm. Figure 47 illustrates averaged total compression improvement of *Combined* algorithm.

Table 9. *Combined* algorithm results.

Compression algorithm	Semantic layers	Corrupted layers	Reconstructed layers	
			Combined approach	%
PNG	2 085 871	2 149 490	2 078 254	3.31 %
TIFF	1 473 824	1 708 362	1 480 657	13.33 %
JBIG	684 978	790 257	720 185	8.87 %

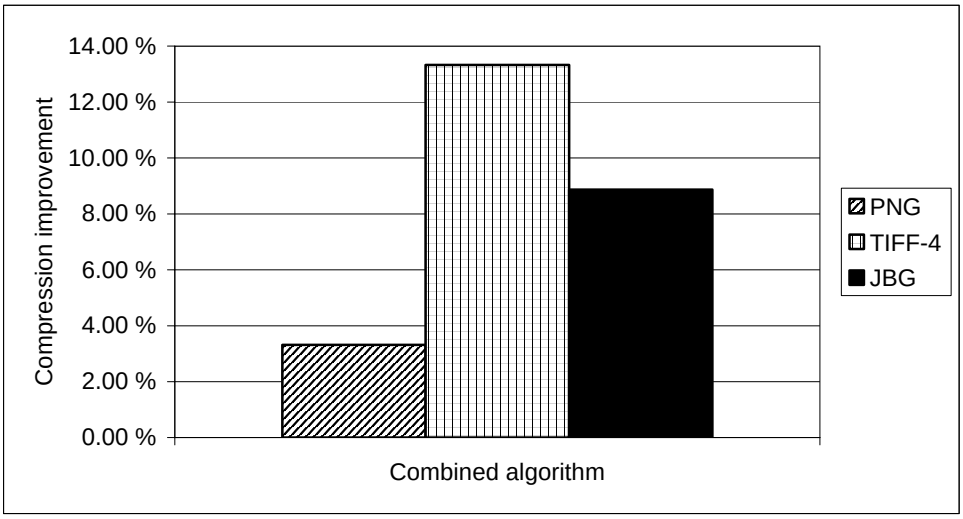


Figure 47. Compression improvements of combined algorithm

Combined algorithm presented several improvements in total compression improvement comparing to averaged total results (Table 8) where all layers were reconstructed with unchanged restoration technique.

8.8 Image difference

We measure the following differences: a difference between the restored layer and the original (uncorrupted) layer; and the difference between the original and corrupted layers. First allows the evaluation of corruption caused by decomposition, and the second shows how close restoration techniques approximate corrupted layer to the original one.

Figure 48 and Figure 49 illustrate the difference measured by NMAE and NMWAE respectively. Here total averaged differences (differences for all layers were calculated for every sample in a test set; then average difference was calculated) are presented.

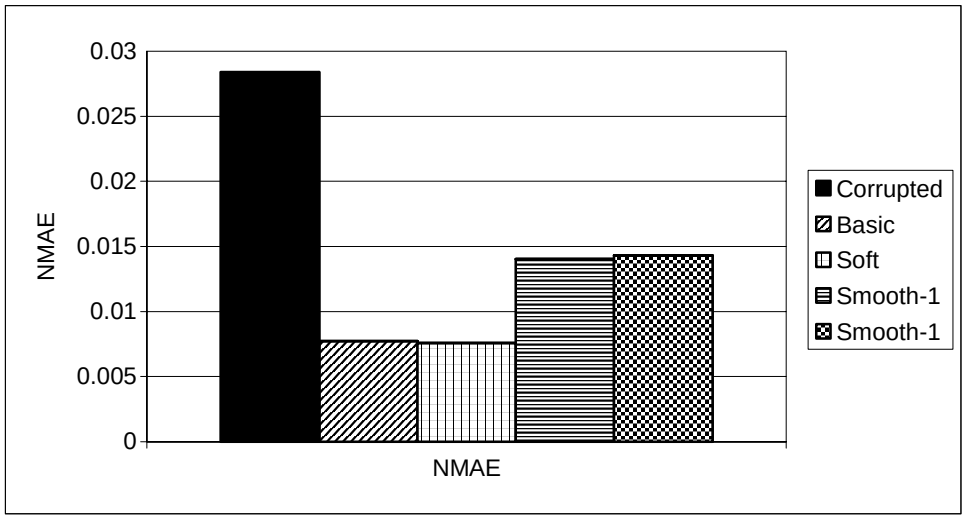


Figure 48. Difference between original (uncorrupted) layer and layer restored by restoration algorithms measured by NMAE.

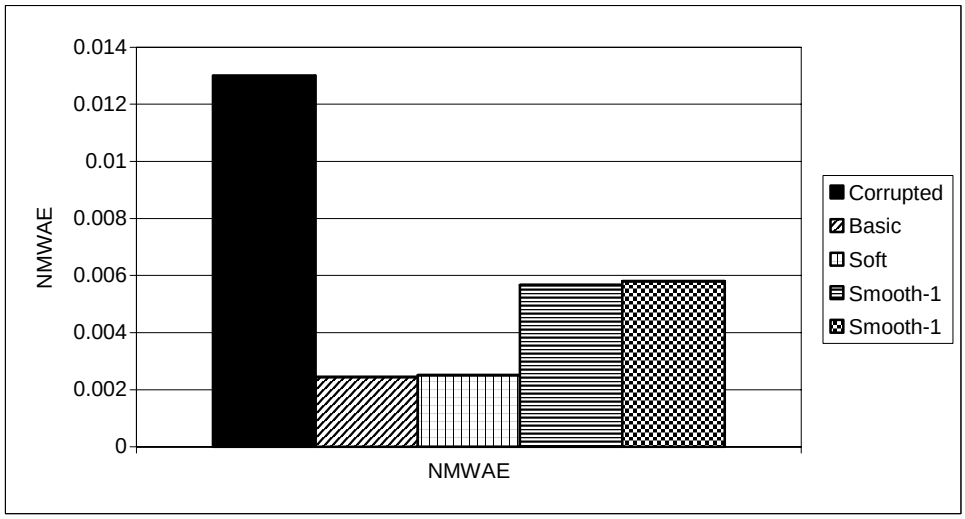


Figure 49. Difference between original (uncorrupted) layer and layer restored by restoration algorithms measured by NMWAE.

In sense of image difference measures, *Basic* and *Soft* algorithms create the best approximation of the corrupted layer to the original.

9 Conclusion

In this work we have developed a technique for restoration of binary semantic layers of the map images resulting from the decomposition of the image using color separation process. The technique is able to reduce the corruption of the layers caused by the decomposition. This also reduces the entropy of the reconstructed images and, therefore, improves the compression performance. The color map image resulting from the combination of the reconstructed layers remain identical to the original because all changes to the layer content are performed only within those areas that will be certainly overlapped during the composition.

The proposed technique is based on Mathematical Morphology. The base algorithm uses the *conditional dilation* operator. We have modified that operator and proposed the *erosion of the mask* to avoid the most disadvantages of standard conditional dilation. Three possible modifications of the proposed base algorithm were considered. First modification is based on varying the structuring element of dilation and erosion operator. This varying allows controlling the speeds of expanding and shrinking of objects when dilation and erosion operators are applied. Second modification is based on the using of soft counterparts of morphological operators. Soft morphological filters are less sensitive to additive noise and small variations in the shapes of the object. Besides that, soft morphological operators allow controlling the speed of dilation and erosion by varying the value of rank parameter. Third considered modification is a *smoothing* approach. The idea was to use *rank* operator to smooth borders of objects.

The four chosen restoration techniques (*Basic*, *Soft*, *Smooth-1* and *Smooth-2*) have been evaluated on a set of topographic color-palette four-layer (*Basics*, *Evaluation*, *Waters* and *Fields*) map images. Besides that, the technique for layer removal was developed using proposed restoration operator. The restoration algorithms have been applied for reconstruction of 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 fact gave us possibility to compare restored images with their original undistorted counterparts.

The performance of the proposed restoration techniques has been evaluated quantitatively and qualitatively. The objective of the qualitative evaluation was to compare the images visually and decide on the usability of the restoration techniques in applications where human visual factor plays prevailing role. The first objectives of quantitative evaluation were to compute and compare the differences between the original and corrupted layers and between the original and restored layers; and to calculate the compression performance for three major compression techniques: LZ (PNG), ITU Group 4 (TIFF), and

JBIG. Results of experiments were classified in two ways: by semantic layers and by compression techniques.

Calculating and comparing of image differences showed that,

- Results presented by *Basic* and *Soft* algorithms are closest to the original in sense of image difference measures NMAE and NMWAE.

Classification by semantic layers showed that:

- Restoration of *Elevation* layer presented very poor compression improvement for all compression techniques. The best result achieved using JBIG compression was only 3.82% of improvement. We conclude that developed restoration technique is not effective when applied to such kind of layer because of its morphological structure. Compression improvement is too small to be practical.
- Restoration of *Waters* layer showed that developed technique gains up to 52.79% of compression improvement. That result was achieved by *Soft* algorithm using TIFF compression. Using other compressions *Soft* algorithm was also the best one and gained 32.08% with JBIG and 9.65% with PNG. We conclude that *Soft* algorithm is effective in restoration of *Waters* layer and makes sense to be applied.
- Restoration of *Fields* layer showed that developed techniques gains significant compression improvement. The best result 55.31% was shown by *Smooth-1* algorithm using JBIG compression. *Smooth-1* algorithm was also the best in compression with other techniques. It gained 46.36% when compressing with TIFF and 29.75% when compressing with PNG. We conclude that *Smooth-1* algorithm is effective in restoration of *Fields* layer and makes sense to be applied.

Classification by compression techniques showed that:

- Soft algorithm was the best in average compression improvement for all compression techniques. It gained 8.6% for JBG, 13.1% for TIFF and 2.7% for PNG.
- Though restoration can gain significant improvement for a single layer, it is not so effective in sense of total compression improvement. This effect caused by the fact that multi-layer map contains layers (*Basics* and *Elevation*) which can not be compressed more efficiently using restoration. This makes the effect of restoring *Waters* and *Fields* layer less significant.

We conclude that *Basic* and *Soft* algorithms are the best within investigated modifications. The restoration of standalone *Waters* and *Fields* layers gains up to 30–50% compression improvement (depending on the applied compression technique). *Combined* algorithm performs the best results in total compression (sum of all layers) allowing gaining up to 5–10% improvement (depending on the applied compression technique). Low total improvement rates caused by the presence of non- or hardly restorable layers, such as *Basics* and *Elevation*.

10 Future Work

Global modeling and pattern matching.

Remarkable improvements in image compression have been achieved by specializing in some known image types (e.g. text images) and exploiting global dependencies. The emerging standard JBIG2 [H+98] will segment a page into different classes of image data, in particular, textual, halftone and generic (other) [TK99], and utilize the repetitive nature of the textual and halftone images. For textual data, JBIG2 uses either *pattern matching and substitution* or *soft pattern matching* techniques. The first extracts symbols and marks from the image into the dictionary, which is a collection of bitmaps, and substitutes similarly looking patterns in the image with the indices into dictionary [IW94]. The latter technique uses two-layer context template to encode refinement data used to losslesly recreate original symbols in the image [H97]. Some data, such as line art data, may not be identified as either textual or halftone, and is encoded by *cleanup coder*, which is essentially a bitmap coder similar to JBIG [MF99].

It would be worth to develop an algorithm to extract the symbols and marks from the map images into a separate *Symbolic* layer, for which JBIG2 can be applied. The difficulties here are such that, at least in our sample images, the text may be drawn in the same color as topographic data (i.e. located on the same physical layer), which complicates the separation process. In other words, the textual data is integrated into topographic data, and is often overlap each other. Thus, an advanced compression algorithm based on JBIG2 must be developed for the compression of similarly combined layers. The pattern matching technique must be able to extract the symbols even if they overlap with other type of geographic data.

Advanced image compression algorithms.

Lately, there have been proposed advanced map image compression algorithms that gain about 25% in compression performance over JBIG. These algorithms are based on:

- variable-size context-based modeling using context trees, which utilizes higher-order context templates without context dilution problem, see [AF00a];
- multi-level optimal-order context-templates, which utilize inter-layer dependencies, see [AKF01];
- or their optimized combination, see [KF03].

It would be interesting to apply these algorithms in order to evaluate their performance for compression of corrupted and respectively restored semantic layers.

11 List of symbols

Symbol	Meaning	Paragraph reference
$\mathbb{E}$	Abel group	2.1.1
E	Discrete or Euclidian image space	2.1.1
$\mathbb{E}^d$	d-dimensional product	2.1.1
$\mathcal{P}(E)$	Power set of E	2.1.1
X^c	Complement of X	2.1.1
A_h	Translate of A along vector h	2.1.2
$card(A)$	Cardinal value of A	2.1.2
$\psi_A(X)$	Morphological transformation of X using structuring element A	2.1.2
$X \oplus A$	Minkowski addition	2.1.4
$X \ominus A$	Minkowski subtraction	2.1.4
$\delta_A(X)$	Dilation of X by A	2.1.4
$\xi_A(X)$	Erosion of X by A	2.1.4
$\widetilde{A}$	Reflectance of A	2.1.4
$\rho_{A,s}$	Rank operator	2.2.1
$\delta_B(A \mid T)$	Conditional dilation of A by B relative to T	2.2.2
$\delta_A(X,r)$	Soft dilation by A with factor r	2.2.3
$\varepsilon_A(X,r)$	Soft erosion by A with factor r	2.2.3
$\mathfrak{M}$	Multi-layer map	3.1
$\mathcal{M}$	Combined map image	3.2
$\mathfrak{M} \xrightarrow{C} \mathcal{M}$	Layer combination process	3.2
$\mathcal{M} \xrightarrow{D} \mathfrak{M}$	Map decomposition into layers using color separation	3.3

12 References

[AF00a] **Ageenko E., Fränti P.,** "Compression of large binary images in digital spatial libraries", *Computers & Graphics*, **24** (1), 91-98, Elsevier Science, February 2000.

[AF00b] **Ageenko E., Fränti P.**, "Context-based filtering of document images", *Pattern Recognition Letters*, **21** (6-7), 483-491, Elsevier Science, 2000.

[AKF01] **Ageenko E., Kopylov P., Fränti P.,** "On the size and shape of multi-level context templates for compression of map images", *Proc. 2001 IEEE Int. Conf. on Image Processing "ICIP'01"*, Thessaloniki, Greece, vol. 3, pp. 458-461, 2001.

[B87] **Brent R.P.,** "A linear algorithm for data compression", *Australian Computer Journal*, **19** (2), pp. 64-68, 1987.

[D92] **Dougherty E.R**., "Optimal mean-square n-observation digital morphological filters. Part I: Optimal binary filters", *Computer Vision, Graphics, and Image Processing*, 55: 36-54, 1992.

[D96] **Deutsch L.P**.,"DEFLATE Compressed Data Format Specification" v.1.3 (RFC1951), available in ftp://ftp.uu.net/pub/archiving/zip/doc/, or http://www.gzip.org/zlib/zlib_docs.html

[DA97] **Dougherty E.R., Astola J. (eds)** *Nonlinear Filters for Image Processing*, SPIE Optical Engineering Press, 1997.

[DE03] **Doughberty E., Lotufo R.,** *Hands-on morphological image processing*. SPIE Optical Engineering Press, 2003.

[F+95] **Fox E.A., et al.** (Eds.) "Digital Libraries". [*Special issue of*] *Communications of the ACM* **38** (4), 1995.

[FAKGB02] **Fränti P., Ageenko E., Kopylov P., Gröhn S., Berger S**., "Map image compression for real-time applications", *Proc. Spatial Data Handling 2002 Symposium "SDH'2002", (part of 2002 Joint International Symposium on Geospatial Theory, Processing and Applications)*, Ottawa, Canada, 15 pages, July 2002.

[FAKK02] **Fränti P., Ageenko E., Kukkonen S., Kälviäinen H.,** "Using Hough transform for context-based image compression in hybrid raster/vector applications", *Journal of Electronic Imaging*, **11** (2), 236-245, April 2002.

[FKA02] **Fränti P., Kopylov P. & Ageenko E**., "Evaluation of compression methods for digital map images", *IASTED Int. Conf. on Automation, Control and Information Technology (ACIT 2002)*, Novosibirsk, Russia, pp. 418-422, June 2002.

[FKV02] **Fränti P., Kopylov P., Veis V.,** "Dynamic use of map images in mobile environment", IEEE Int. Conf. on Image Processing (ICIP'02), Rochester, New York, USA, vol. 3, 917-920, September 2002.

[GW02] **Gonzalez R.C., Woods R.E.,** *Digital image processing*, 2nd edition, Addison-Wesley, 2002.

[H+98] **Howard P.G., Kossentini F., Martins B., Forchammer S., Rucklidge W. J., Ono F.** "The emerging JBIG2 standard". *IEEE Trans. Circuits, Systems for Video Technology* **8** (7): 838-848, 1998.

[H94] **Heijmans H.J.A.M.,** *Morphological image operators*. Boston: Academic Press, 1994.

[H95] **Heijmans H.J.A.M.**, "A new class of alternating sequential filters" *1995 IEEE Workshop on Nonlinear Signal and Image Processing*, Neos Marmaras, Halkidiki, Greece, June 20-22, 1995.

[H97] **Howard P.G.,** "Text image compression using soft pattern matching". *The Computer Journal*, **40** (2/3): 146-156, 1997.

[HR80] **Hunter R., Robinson A.H.,** "International digital facsimile coding standards", *Proc. IEEE*, **68** (7), pp. 854-867, 1980.

[ITU T.4] **ITU-T (CCITT) Recommendation T.4**. (1980)

[IW94] **Inglis S., Witten I.,** "Compression-based template matching". *In Proc. IEEE Data Compression Conference* (Utah, USA), 106-115, 1994.

[J95] **Jin X.C., Ong S.H., Jajasooriah,** "A domain operator for binary morphological processing", *IEEE Trans. Image Processing*, **4** (7): 1042-1046, 1995.

[JBIG] **"**Progressive Bi-level Image Compression" ISO/IEC International Standard 11544, *ISO/IEC/JTC1/SC29/WG*, 1993; also available as ITU-T Recommendation T.82.

[KA94] **Koskinen L., Astola J.** "Soft morphological filters: A robust morphological filtering method". *Journal of Electronic Imaging* 3: 60-70, 1994.

[KF03] **Kopylov P., Fränti P.,** "Optimal layer ordering in the compression of map images", *IEEE Data Compression Conference (DCC'03)*, Snowbird, Utah, USA, 323-332, April 2003.

[KK95] **Kuosmainen P., Koivisto P., Huttunen H., Astola J.** "Shape Preservation Criteria and Optimal Soft Morphological Filtering". *Journal of Mathematical Imaging and Vision*, 5, 319-335, 1995.

[LM00] **Larsson J., Moffat A.** "Offline dictionary-based compression", *Proceedings IEEE*, **88** (11), pp. 1722-1732, Nov. 2000.

[M75] **Matheron G.** *Random Sets and Integral Geometry*, J. Wiley & Sons, New York, 1975.

[MF99] **Martins B., Forchhammer S.,** "Lossless, near-lossless, and refinment coding of bi-level images. *IEEE Trans. Image Processing* **8** (5): 601-613, 1999.

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

[PV90] **Pitas, I., Venetsanopoulos A.N**., *Nonlinear digital filters: principles and applications*, Boston, Mass.: Kluwer, 1990.

[PL00] **Ping Z., Lihui C., Alex K.C.,** "Text document filters using morphological and geometrical features of characters", *Proc. Int. Conf Signal Processing-ICSP'00*, pp. 472-475, 2000.

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

[PM93] **Pennebaker W.B., Mitchell J.L.** *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.

[PMLA88] **Pennebaker W.B., Mitchell J.L., Langdon G.G., Arps, R.B** "An overview of the basic principles of the Q-coder adaptive binary arithmetic coder". *IBM Journal of Research, Development* **32** (6): 717-726, 1988.

[PNG] **Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification**. *ISO/IEC 15948:2002 (E), W3C Recommendation*, 10 November 2003, http://www.w3.org/TR/PNG/

[RL81] **Rissanen J.J., Langdon G.G.,** "Universal modeling and coding". *IEEE Trans. Inform. Theory* **IT-27**: 12-23, 1981.

[RS01] **Randolph T.R., Smith M.J.T.,** "Enhancement of fax documents using a binary angular representation", *Proc. Int. Symp. on Intelligent Multimedia, Video and Signal Processing*, pp. 125-128, 2001.

[S82] **Serra J.,** *Image Analysis and Mathematical morphology*. London: Academic Press, 1982.

[S86] **Sternberg S.R.**, "Grayscale morphology". *Computer Vision*, *Graphics and Image Processing*, vol. 35, pp. 333-335, 1986.

[SDA99] **Sarca O.V. , Doughberty E.R., Astola J.** "Two-stage binary filters", Journal of Electronic Imaging **8** (3), 219-232, 1999.

[SG91] **Schonfeld D., Goutsias J.,** "Optimal morphological pattern restoration from noisy binary images". *IEEE Trans. on Pattern Analysis, Machine Intelligence,* **13** (1): 14-29, 1991.

[SS82] **Storer J.A., Szymanski T.G.,** "Data compression via textual substitution", *Journal of ACM*, **29** (4), pp. 928-951, Oct.1982.

[TK99] **Tompkins D.A.D., Kossentini F.,** "A fast segmentation algorithm for bi-level image compression using JBIG2". *Proc. IEEE International Conference on Image Processing "ICIP '99"* (Kobe, Japan), 25PS1.4, 1999.

[TP80] **Ting D., Prasada B.,** "Digital Processing Techniques for Encoding of Graphics". Proceedings of IEEE, **68** (7), pp. 757-769, July 1980.

[V87] **Vitter J.,** "Design and Analysis of dynamic Huffman codes". *Journal of Association for Computing Machinery*, 34:825-845, 1987.

[W58] **Wiener N.,** *Nonlinear Problems in Random Theory*, New York, The Technology Press, MIT and John Wiley and Sons, Inc., 1958.

[W86] **Wah, F.M**., "A binary image preprocessor for document quality improvement and data reduction", *Proc. Int. Conf. on Acoustic, Speech, and Signal Processing-ICASSP'86*, 2459-2462, 1986.

[WMB94] **Witten I.H., Moffat A., Bell T.C.,** *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, New York, 1994.

[ZD96] **Zhang Q., Danskin J.M.** "Bitmap reconstruction for document image compression". *SPIE Proc. Multimedia Storage, Archiving Systems*, Boston, MA, USA, Vol. 2916: 188-199, 1996.

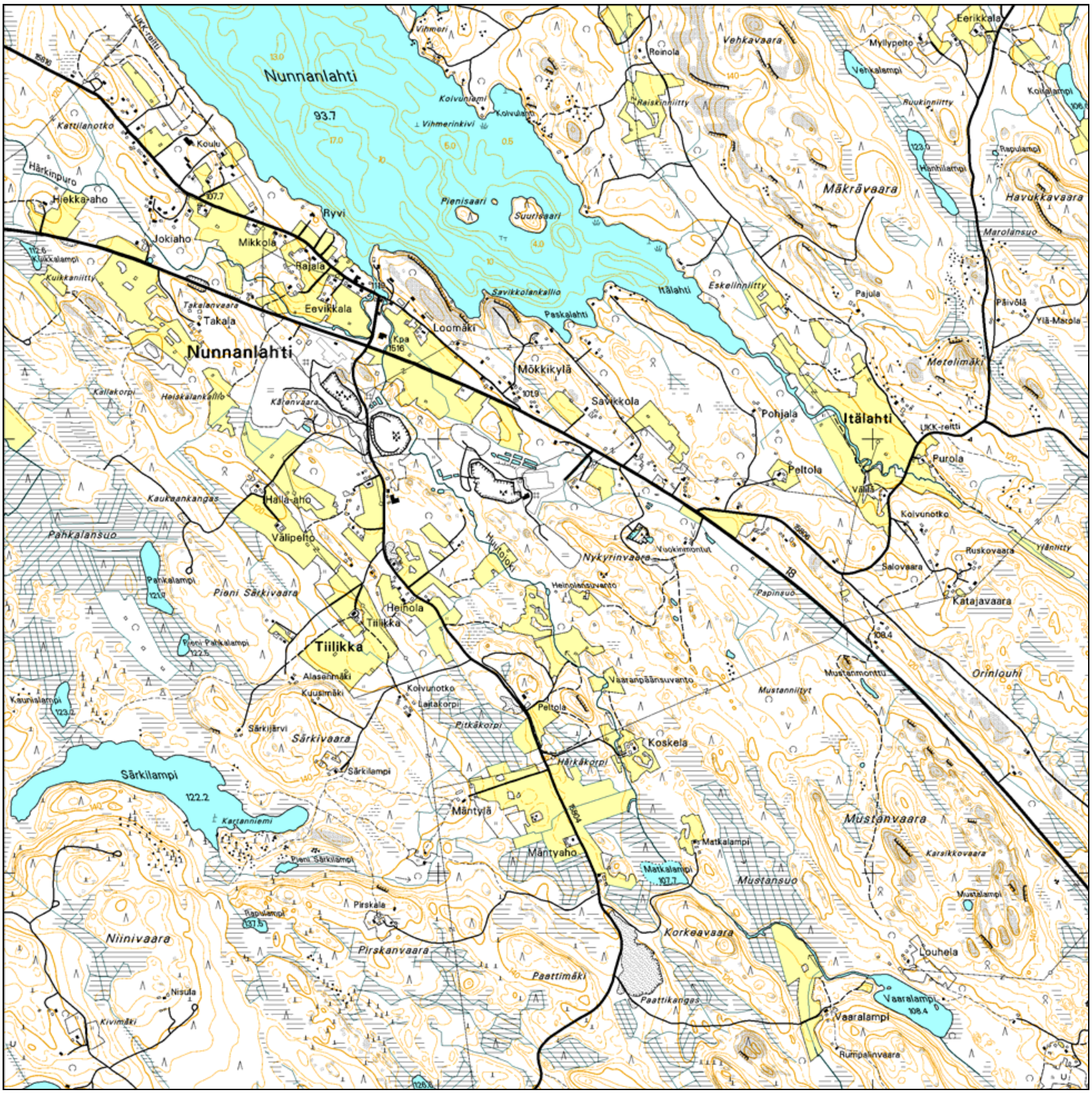
[ZK01] **Zheng Q., Kanungo T**., "Morphological degradation models and their use in document image restoration", University of Maryland, USA, Technical Report, LAMP-TR-065 CAR-TR-962 N660010028910/IIS9987944, 2001.

[ZT96] **Zmuda M.A., Tamburino L.A.,** "Efficient Algorithms for the Soft Morphological Operators". *IEEE Transactions on Pattern Analysis and the Machine Intelligence*, **18** (11), pp. 1142-1147, November 1996.

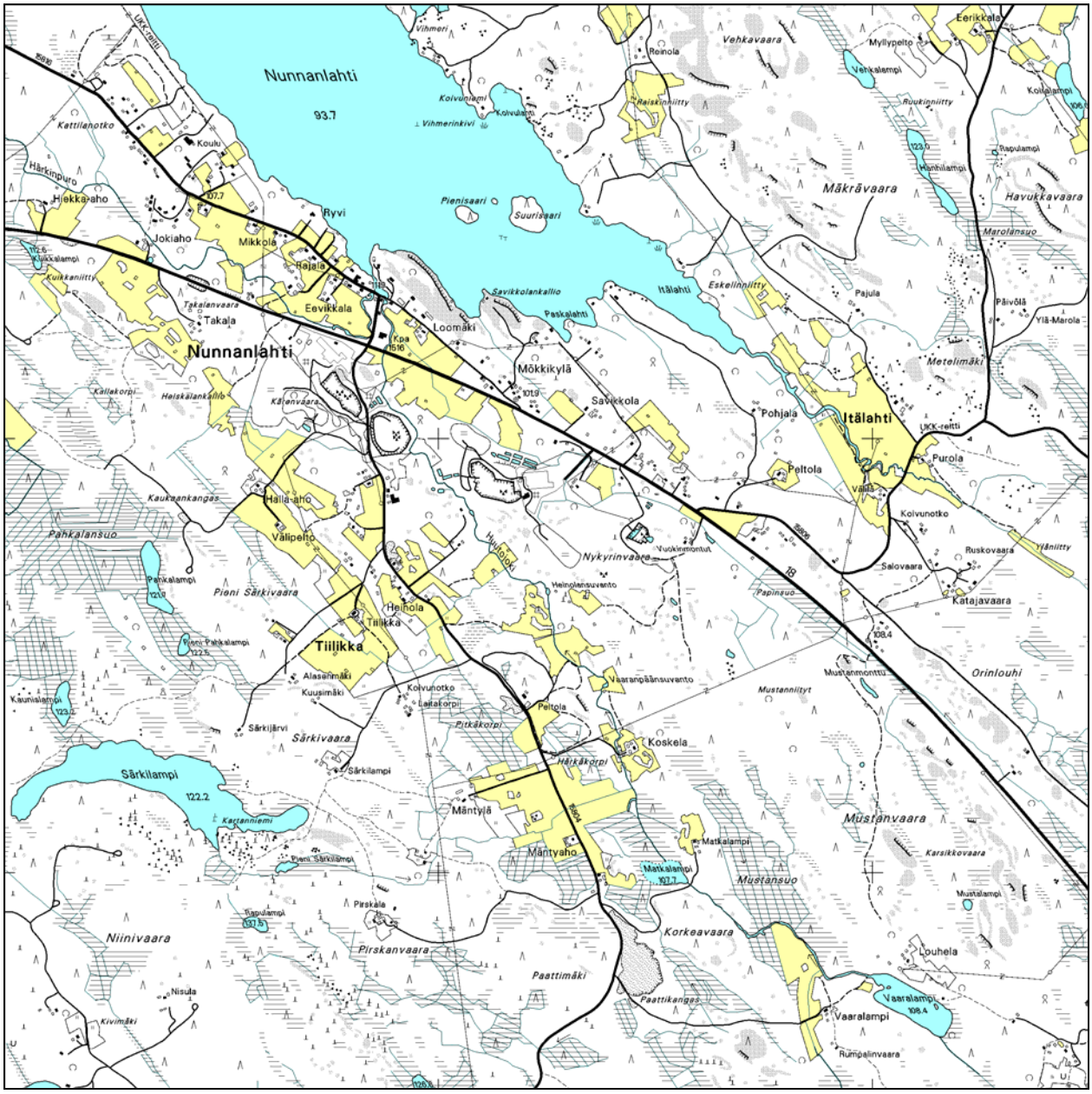
[S89] **Samet H.** (Eds.) *Applications of Spatial data Structures: Computer Graphics, Image Processing*, GIS. MA: Addison-Wesley, Reading, 1989.

[ZL77] **Ziv J., Lempel A**. "A universal algorithm for sequential data compression", *IEEE Transactions on Information Theory*, **23** (3), pp.337-343, May 1977.

Appendix. Illustration of layer removal.



Fragment of the original image containing 4 different semantic layers.



Fragment of the reconstructed map image with elevation lines layer removed.