# Optimized Context Templates for Context-based Compression of Multi-component Map Images.

Eugene Ageenko, Pavel Kopylov, Pasi Fränti

Computer Science Department, University of Joensuu,
Joensuu, FINLAND

## Abstract

We present a method for estimating optimal context templates that are used for conditioning the pixel probabilities in context-based image compression. The algorithm optimizes the location of the context pixels within a limited neighborhood area, and produces the ordered template as a result. The ordering can be used to determine the shape of the context template for a given template size. The optimal size of the template depends on the size of the image. We apply the method for the compression of multi-component map images consisting of several semantic layers represented as binary images. We estimate the shape of the context-template for each layer separately, and compress the layers as generic regions using standard JBIG2 compression technique.

***Keywords:*** *context-based compression, statistical modeling, optimized context template, variable-size modeling*

## 1. INTRODUCTION

The aim of statistical compression is to reduce redundancy in data by assigning shorter codes for symbols with higher probability and longer codes for symbols with lower probability. In an image, pixels form geometrical structures with appropriate spatial dependencies. Dependencies can be localized to a limited neighborhood defined by a local template. Statistical context-based image compression utilizes spatial dependencies in the image. The compression consists of two distinct phases: statistical modeling and arithmetic coding [1].

In the modeling phase, we dynamically estimate the probability distribution of the pixel to be compressed. The probabilities are conditioned on the context that is determined by the combination of neighboring pixel values within the context template. The pixel configuration determines the context, and in this way, the model to be used in compression, see Figure 1. The pixel configuration in the context template is transferred to a binary number ($1110010010_2$ as in Figure), which gives the index ($914_{10}$) of the model that is then used for compressing the pixel. In dynamic modeling, the statistical model is constructed adaptively during the encoding/decoding. It starts from scratch and is updated after each pixel has been coded. In this way, both the encoder and decoder have the same information and no side-information is needed for sending the model.

Arithmetic coding assigns optimal code for the pixels in regards to the given statistical model [2]. The code size can be estimated by the information content of the model measured as the entropy [3]. An example of a context-based statistical compression is JBIG, an international standard for compression of binary images [4]. It uses the ten-pixel context template shown in Figure 1 by default.

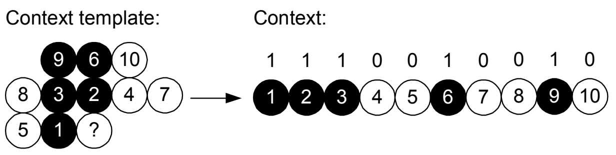

**Figure 1:** Example of a 10-pixel context.
The pixel to be compressed is marked by '?'.

Theoretically, a more accurate probability model can be constructed using a larger context template. In practice, however, the use of larger templates does not always result in compression improvement [5]. The number of contexts grows exponentially with the template size; adding one more pixel to the template doubles the size of the model. This can lead to exessive memory consumption. In addition to this, *context dilution* problem may occur if the statistics are distributed over too many contexts, thus affecting the accuracy of the probability estimates. This is because the model must adapt to the statistics of the image before the model becomes efficient. The coding deficiency in the early stage of compression is known as *learning cost* problem. These two disadvantages can overweigh the improvement of the model if too large context templates are used.

Optimal template size depends on the image size. The location of the template pixels, on the other hand, has no direct effect on the learning cost but they can greatly improve the accuracy of the model if properly designed. It is therefore feasible to optimize the location of the template pixels for the images to be compressed. Usually, the pixels are distributed in the neighborhood using the principle of minimal distance to the current pixel. Standard 1-norm or 2-norm distance functions define two different templates that can be used, see Figure 2 [6]. These templates are well suited for mixed type images. However, they are not necessary the best choices for images of a specific type, and better templates can be obtained.

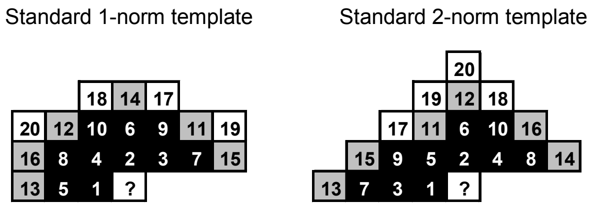

**Figure 2:** Default orderings of the context templates [6]. The pixel location to which the template is applied (seed pixel) is marked with '?'.

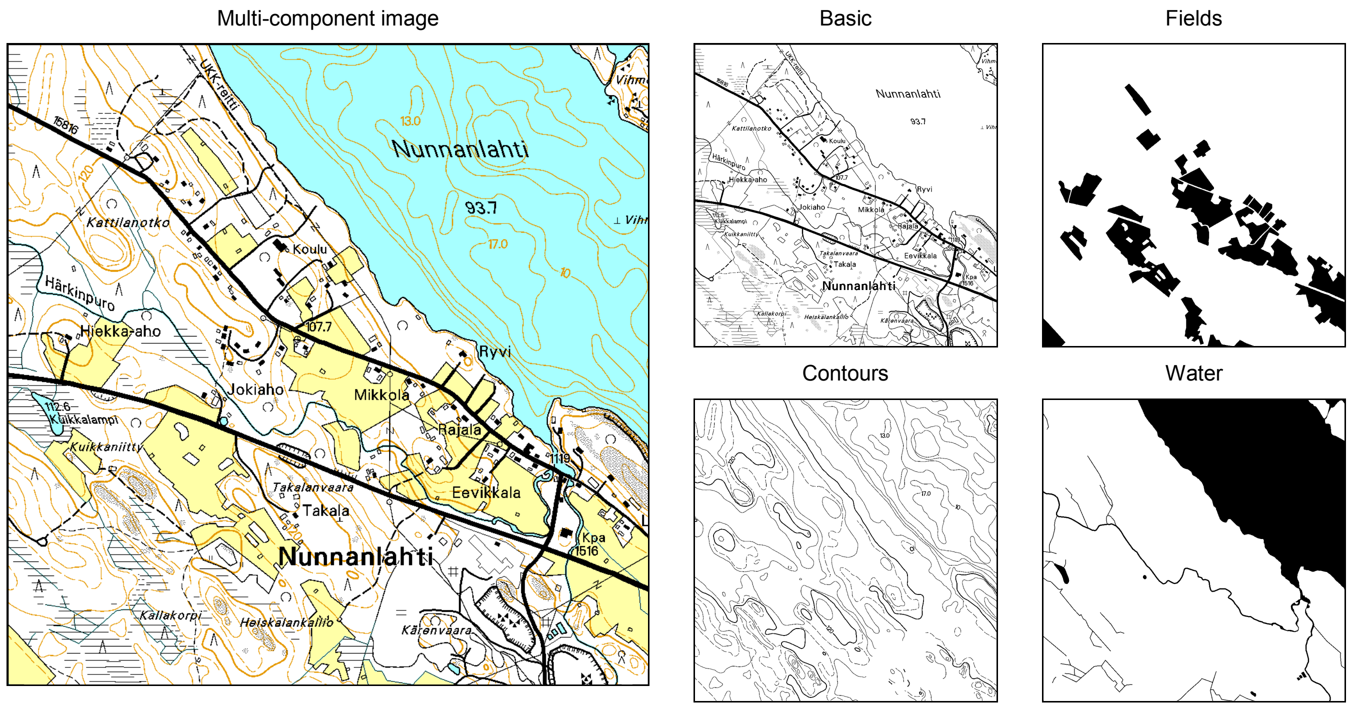

**Figure 3.** Illustration of the multi-component map image. The shown fragment has dimensions of 1000 × 1000 pixels.

We consider multi-component map images. The images consist of several binary layers with different semantic content. Each layer consists of geometrical structures that do not necessarily match to the structures of another layer. In our experiments, we use topographic images from the NLS image database [7]. These images consist of four binary layers corresponding to the topographic data, fields, elevation lines and water area. The layers are combined and displayed to the user as color image, as shown in Figure 3.

In this paper, we propose a method for optimizing the context template for a given image. The method optimizes the location of the template pixels within a limited neighborhood area, and produces the ordered template as the result. The ordering can then be used to derive the context template for any given template size. We apply the method in a static manner for the compression of multi-component map images. The template is optimized for each layer separately using a training image. The optimized context templates are then applied for the compression of a set of NLS images using the JBIG2 standard compression technique [7].

## 2. CONTEXT-BASED STATISTICAL MODELING

The idea of statistical modeling is to describe the pixels of the image according to the probability distribution of the source alphabet (binary alphabet, in our case). The information content of a single pixel can be measured by its *self-entropy*:

$$H_{pixel} = -\log_2 p, \tag{1}$$

where $p$ is the probability of the pixel [3]. The self-entropy of the image can be calculated as the average entropy of all pixels:

$$H_{image} = -\frac{1}{n}\sum_{i=1}^{n}\log_2 p_i, \tag{2}$$

where $p_i$ is the probability of $i$-th pixel and $n$ is the total number of pixels in the image. Self-entropy gives the optimal number of bits required for encoding a single pixel with a given model.

The pixels in an image form geometrical structures with appropriate spatial dependencies. The dependencies can be localized to a limited neighborhood, and described by a *context-based statistical model* [1]. In this model, the pixel probability is conditioned on the *context* $C$, which is defined as distinct black-white configuration of neighboring pixels within the local template. For binary images, the pixel probability is calculated by counting the number of black, $n_B(C)$, and white, $n_W(C)$, pixels appeared in that context in the entire image:

$$p(x|C) = \begin{cases} p_W(C) = \dfrac{n_W(C)}{n_W(C)+n_B(C)}, & \text{if } x \text{ is } \textit{white} \\ p_B(C) = 1 - p_W(C), & \text{if } x \text{ is } \textit{black} \end{cases}, \tag{3}$$

Here, $p_B(C)$ and $p_W(C)$ are the corresponding probabilities of the black and white pixels. The entropy $H(C)$ of a context $C$ is defined as the average entropy of all pixels within the context:

$$H(C) = -p_W(C)\cdot\log_2 p_W(C) - p_B(C)\cdot\log_2 p_B(C) \tag{4}$$

The entropy of an $N$-pixel context model is the weighted sum of the entropies of individual contexts:

$$H_N = -\sum_{j=1}^{N} p(C_j)\cdot(H(C_j)) \tag{5}$$

In dynamic modeling, the encoder and decoder then adaptively construct the model during the compression/decompression on the basis of the preceding data. A uniform probability distribution ($p_W^0 = p_B^0 = 0.5$) is assumed in the beginning. Time-dependent counters $n_W^t$ and $n_B^t$ start from zero and are updated after the pixel has been coded (decoded). The probability of a pixel is calculated on the basis of the observed frequencies using a Bayesian sequential estimator:

$$p^t(C) = \begin{cases} p_W^t(C) = \dfrac{n_W^t(C) + \delta}{n_W^t(C) + n_B^t(C) + 2\delta}, & \text{if } t^{\text{th}} \text{ pixel is } \textit{white} \\ p_B^t(C) = 1 - p_W^t(C), & \text{if } t^{\text{th}} \text{ pixel is } \textit{black} \end{cases} \quad (6)$$

where $n_W^t$, $n_B^t$ are the time-dependent counters, $p_W^t$, $p_B^t$ are the probabilities for white and black colors respectively, and $\delta = 0.45$, due to JBIG. The model is inefficient at early stage of compression, since it takes time to adapt to the correct model, but the dynamic modeling is highly applicable for compression large volumes of data, such as map images.

## 3. TEMPLATE CONSTRUCTION

The optimal context template can be solved for a given template size $k$ by compressing the image using all possible templates and selecting the one with best compression performance. However, this is not computationally feasible as there are approximately $32^k$ different template configurations to be tested. Therefore we take a more practical approach and construct the template stepwise by optimizing the location of one pixel at a time. The sketch of the algorithm is shown in Figure 4.

The algorithm starts with an empty template and expands it by one pixel at a time. At each iteration, we add a new pixel to each unoccupied location in the neighborhood area. We use the 40-pixel neighborhood shown in Figure 5. For each candidate pixel location, we make a pass over the input image and construct the statistical model. We evaluate the models by estimating their code length using the equations (2) and (6). We then select the location providing minimum entropy, and add it permanently to the context template. The selected location is marked as occupied, and the process is then repeated until the template size reaches the predefined maximum $k_{\text{MAX}}$. The process of the algorithm is illustrated in Figure 6.

The result of the algorithm is not only the final template of $k_{\text{MAX}}$ pixels but also the ordering of the pixels. From the ordering we can derive all possible templates of the size 1 to $k_{\text{MAX}}$. The size of the context template is a parameter of the compression method and it mainly depends on the size of the image. For example, the map images are very large and therefore relatively large templates can be applied without the risqué been weighed down by the learning cost and context dilution problems.

The proposed method can be applied in two alternative manners: *static* and *semi-adaptive*. In the static approach, as taken here, we optimize the template using a priori knowledge of the image type. This is possible, as we know the type of the images to be compressed. The advantage of this approach is that the optimization can be done off-line. In the semi-adaptive approach, the template is optimized for the image to be compressed and the optimized template are stored in the compressed file. This would be a better solution when the image type is not known beforehand. The compression phase, however, would be very slow and therefore this approach is not suitable for applications, in which real-time compression is required.

```
ConstructTemplate ($k_{MAX}$, SearchTemplate[])
    variables:
        ContextTemplate[]: array;
        k, i, j: int;

    $k \leftarrow 0$;
    repeat
        $k \leftarrow k + 1$;
        for each i that SearchTtemplate[i] ≠ OCCUPIED
            CollectStatistics(i);
            l(i)=CalculateCodeLength(i);
        $j \leftarrow \min_i l(i)$;
        ContextTemplate[k] ← j;
        SearchTemplate[j] ← OCCUPIED;
    until ($k = k_{MAX}$);
    return (ContextTemplate);
```

**Figure 4.** Algorithm for estimating the optimal context template.

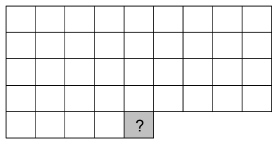

**Figure 5.** The neighborhood area used for optimizing the location of the template pixels.

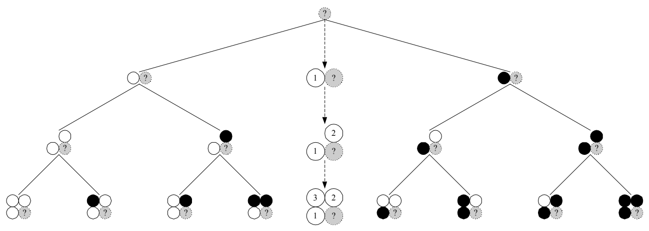

**Figure 6.** Illustration of the context template construction.

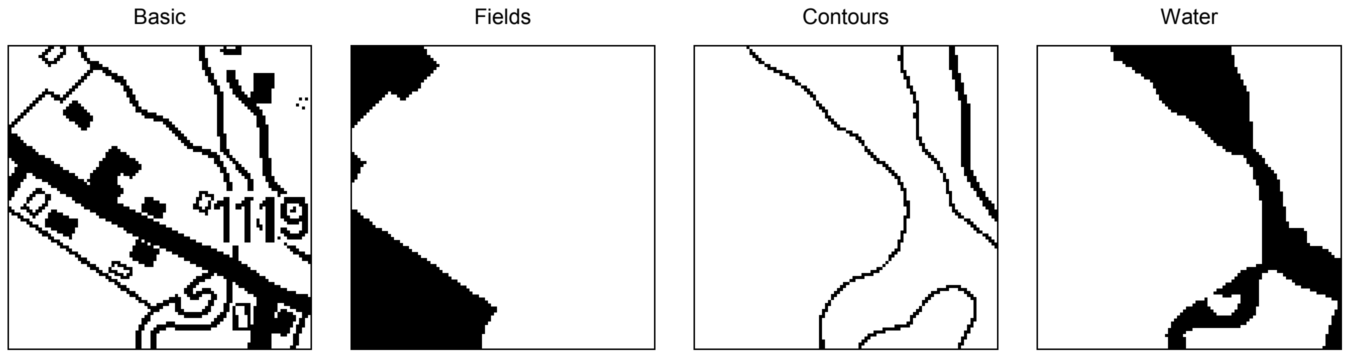

**Figure 7.** Sample 100 x 100 pixels fragments of the layer images.

## 4. EXPERIMENTS

We evaluate the proposed method by compressing a set of map images from the NLS topographic database (Basic map series 1 : 20,000). Each image is of the size 5000×5000 pixels, and represents a 10×10 km$^2$ area. The images consist of four binary layers with different semantic meaning:

- *basic* – topographic image, supplemented with communications networks, buildings, protected sites, benchmarks and administrative boundaries;
- *fields* – solid polygonal regions;
- *contours* – thin lines representing the elevations levels;
- *water* – solid regions, and various width lines representing lakes, rivers, swamps, water streams.

In our experiments, we use five randomly chosen images from the database. The images corresponding to the map sheets *No/No* 431306, 124101, 201401, 263112, and 431204. The image 431306 contains most common geometrical structures, see Figure 3 and Figure 7, and it is therefore used as the training image for optimizing the templates. The rest of the images are used for the actual compression. We use JBIG2 compression technique in its generic mode [8]. Objectives of the evaluation are to determine the compression performance using the constructed context templates in comparison to the standard 1-norm and 2-norm templates. The layers are compressed separately so that user would be able to decompress only the requested layers.

The templates constructed using the proposed algorithm are shown in Figure 8 for the different semantic layers. The ordering of the pixels is illustrated by the numbering. The first ten pixels are colored by black color, and the next six pixels by gray color. The corresponding compression results are summarized in Figure 9, where the results are given for each layer separately. The results are obtained by varying the template size from 1 to 20.

The resulting templates have different shapes corresponding to the geometrical structures of the images. The *basic* map includes wide variety of different elements: text, solid lines of different width, and single pixel dots, see Figure 7 for details. The optimized template is therefore virtually the same as the standard template of JBIG2, and the corresponding compression results are also close to each other.

The *fields*, on the other hand, have a different template where only the most nearest neighboring pixels are utilized in the ten-pixel template. The most nearest pixels are enough to predict the existence of a field because the images contain merely large solid areas. Additional pixels are chosen far away from the current pixel. The optimized template improves the compression of the fields by about 12 %, on average. The simplicity of the structures also means that relatively small template sizes are sufficient for this kind of images.

*Contours* layer consists of elevation lines, which are one or two pixels wide solid or dashed contours. There are no single pixels or larger structures in these images. *Water* layer contains also contour lines but they are always two or more pixels wide. In addition to that, there are larger black areas representing lakes. The optimized context templates for these two types of images are similar, and they provide moderate improvement in the compression.

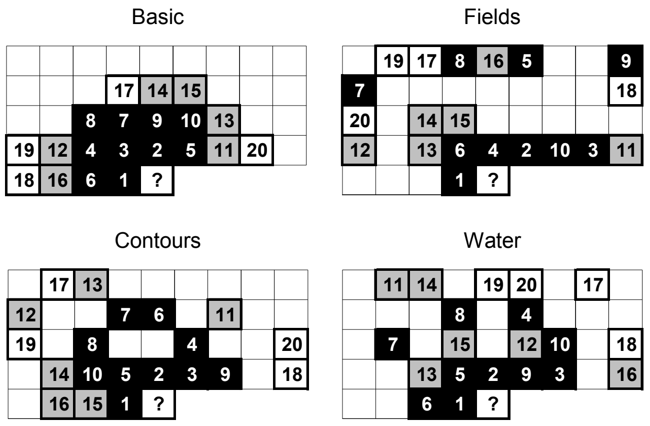

**Figure 8.** Optimized context templates for the semantic layers.

## 5. CONCLUSION

A method for optimizing context templates for a given image was introduced. The algorithm optimizes the location of the context pixels within a limited neighborhood area, and produces the ordered template as a result. It was shown that the optimized templates can be quite different for different types of images. The method can be applied for the compression of multi-component map images, and moderate compression improvement was obtained for a set of map images.

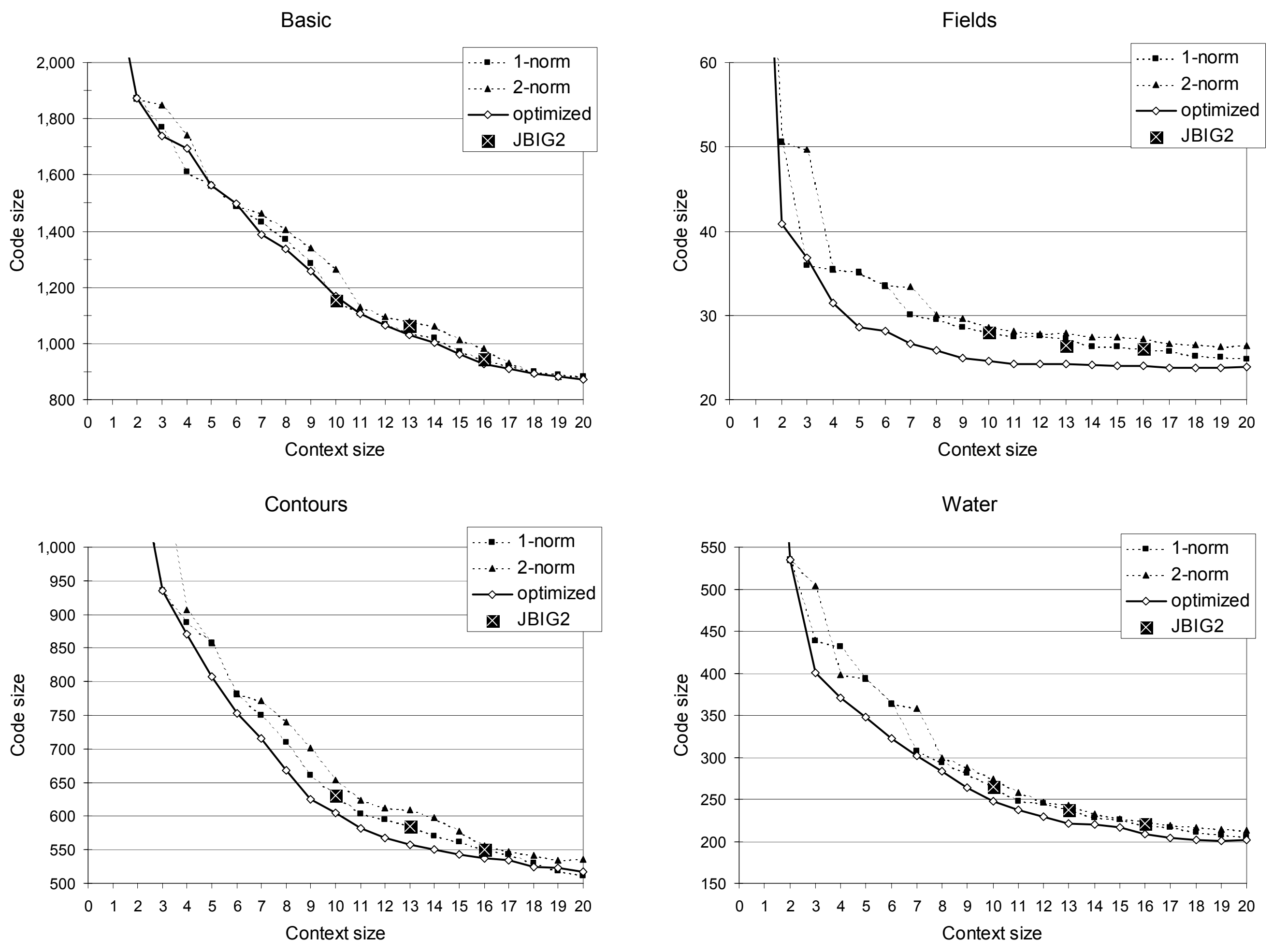

**Figure 9.** Total size of the compressed files in Kilobytes for map layers using JBIG2 compression in generic mode with various context templates: 1-norm, 2-norm, optimized, and three standard templates (with sizes 10, 13, and 16) defined in JBIG2.

The next logical step would be to utilize dependencies between the layers by applying a multi-level context. It is likely that existence of a field is a strong indication of absence of water, and vice versa. The utilization of inter-layer dependencies requires that the images are compressed/decompressed in a predefined order. The optimization of the multi-layer context templates and the proper ordering of the layers is a topic of future research.

## 6. ACKNOWLEDGEMENTS

The project was funded by the grant 954/401/99 of National Technology Agency (TEKES). Pavel Kopylov also acknowledges Centre for International Mobility (CIMO) for financial support of his work.

## 7. REFERENCES

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

[2] **Rissanen J.J., Langdon G.G**. (1979) Arithmetic coding. *IBM Journal of Research, Development 23*: 146-162.

[3] **Shannon C.E**. (1948) A mathematical theory of communication. Bell System Tech Journal 27: 398-403.

[4] **JBIG**: ISO/IEC International Standard 11544 (1993) *ISO/IEC/JTC1/SC29/WG9;* also ITU-T Recommendation T.82. Progressive Bi-level Image Compression.

[5] **Moffat A.** (1991) Two-level context based compression of binary images. *IEEE Proc. Data Compression Conference* (Snowbird, Utah, USA), 382-391.

[6] **Martins B., Forchhammer S.** (1998) Bi-level image compression with tree coding. *IEEE Trans. Image Processing* **7** (4): 517-528.

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

[8] **JBIG2 Working Draft**. http://www.jpeg.org/public/jbigpt2.htm

## About the authors

Dr. Eugene Ageenko is a researcher in the Computer Science Dept., University of Joensuu, Finland; and principal scientist in Arboreal Inc.
E-mail: ageenko@cs.joensuu.fi

Pavel Kopylov is a doctoral student in the Computer Science Dept., University of Joensuu, Finland.
E-mail: justas@cs.joensuu.fi

Dr. Pasi Fränti is a Professor in the Computer Science Dept., University of Joensuu, Finland.
E-mail: franti@cs.joensuu.fi

Contact address: Department of Computer Science, University of Joensuu, PB 111, 80101 Joensuu, FINLAND.