Map Image Compression for Real-Time Applications

Pasi Fränti, Eugene Ageenko, Pavel Kopylov, Sami Gröhn and Florian Berger

Department of Computer Science, University of Joensuu, Box 111, FIN-80101 Joensuu, FINLAND, Email: franti@cs.joensuu.fi

Abstract

Digital maps can be stored and distributed electronically using compressed raster image formats. We introduce a storage system for the map images that supports compact storage size, decompression of partial image, and smooth transitions between various scales. The main objective of the proposed storage system is to provide map images for real-time applications that use portable devices with low memory and computing resources. Compact storage size is achieved by dividing the maps into binary layers, which are compressed using context-based statistical modeling and arithmetic coding. Partial image decompression is supported by tiling the image into blocks and implementing direct access to the compressed blocks. In this paper, we give overview of the system architecture, describe the compression technique, and discuss implementation aspects. Experimental results are given both in terms of compression ratios and image retrieval timings.
Keywords: image compression, map images, real-time applications, personal navigation, spatial access

1 Introduction

Real-time cartography imaging application provides user with the view of geographic map for the area surrounding the user's location (Kraak 1996, 2000). The system may use *global positioning service* (GPS) (Kaplan 1996) or *mobile positioning service* (MPS) (Dye 1999) for obtaining the coordinates of the current location. The location can be updated in real-time (about once or twice in every

second). The system must also support real-time *panning* (spatial movement) and *zooming* (change of resolution) on the map. By panning, we mean scrolling the map; and by zooming, we mean the change of the view on the display in a closer or wider perspective.

Digital map are usually obtained from spatial databases where the maps are stored in vector formats. The visual outlook of maps representing the same region varies depending on the type of the map (topographic or road map), and on the desired scale (local and regional maps). Individual map images are reproduced for each scale separately and stored as separate raster images augmented with the location information of the map. A typical map image needs only a few color tones but high spatial resolution for representing the details such as roads, infrastructure and names of the places.

In on-line map imaging applications, the images are usually stored in an inefficient, uncompressed raster form. The storage size of a map image is huge. For example, electronic library of Finnish road maps of the resolution 1:250 000 takes an entire CD (over 600 Mb) in uncompressed form (GeoData, 1999). In comparison, the portable viewing device, such as pocket computers, have usually about 32 Mb of the storage space, which can be expanded at present by about 96 Mb through using *compact flash* memory cards (Gellersen and Thomas, 2000). The storage requirements of the maps can therefore be a bottleneck, especially in the case of portable devices, in which the maps share the limited memory resources with the operating system, application and other data.

A better approach is to provide the user with the images in compressed form (Arps and Truong 1994, Murray, vanRyper and Russell, 1996). For example, an uncompressed black-and-white topographic image of 5000×5000 pixels takes about 3 megabytes in uncompressed form. The latest compression standards (Haskell et al, 1998), however, can compress typical map images by a factor of about 20:1, which corresponds to the file size of 150 kilobytes. A drawback of the existing compression techniques is that the entire image must be decompressed in memory before the image can be viewed. This can be a problem if the device does not have sufficient computing resources for real time image decompression.

We propose *Map Image Storage System* (further denoted as MISS), in which we present reasonable solutions both to the storage problem and to the real-time requirements of the system. The MISS images are composed of semantic binary layers, which are compressed using a context-based statistical modeling and arithmetic coding. The method is basically the same as in the latest international compression standards, JBIG (*Joint Bi-level Image Group*) and JBIG2 (ISO/IEC 1993, 1999) with a few differences described later.

To meet the real-time requirements, we provide direct access to the compressed image file. Our solution is to divide the image into $b \times b$ non-overlapping rectangular blocks, which are compressed separately. The compressed blocks are stored in the same file, and an index table is stored in the header of the file to locate the starting points of the code blocks. In this way, direct access can be provided with the accuracy of the block size. This kind of file organization is

supported by the JBIG2 standard although there are some limitations as shown by Fränti et al (2002).

2 Map Image Storage System

Digital maps are usually stored as vector graphics in a database for retrieving the data using the spatial location as the search key. Vector representation is convenient for zooming as the maps can be displayed in any resolution defined by the user. Panning of the map can be performed by retrieving the elements needed for updating the changes in the view. The use of database, however, can be impractical in mobile environment, as the devices may not have enough resources to store the complete map database and the database engine.

The storage problem could be solved by generating spatial views (*map sheets*) from the database and store the maps in a vector format. The storage size can be reduced further by compressing the vector maps (by a factor of about 2:1), or by simplifying the vector representation. This approach, however, does not support real-time panning as separate data structures must be built for this purpose.

The biggest problem of the vector format is that maps are not always available in vector format. Moreover, the maps are stored in various formats and incompatibility between different systems can restrict the use of the maps. To sum up, vector format is a good approach if the user has sufficient hardware and software resources, and if the maps are widely available in a compatible vector format. Otherwise, raster image format is the only choice.

We introduce next a map image storage system (denoted as MISS) based on compressed raster format. The system support the following properties of the maps:

1. Compact storage size
2. Multi-scale representation (*zooming*)
3. Fast scrolling ability (*panning*)

The idea is that the maps are stored in a server-side database. Spatial views are generated for the client-side application in compressed raster image format organized so that it supports the zooming and panning requirements. In this way, raster format is suitable in applications, where the maps are needed for viewing purposes only.

Furthermore, the system does not depend on any database or vector format as digitized raster maps can be easily generated and reproduced from any source format, including paper maps. Another advantage of the system is that it requires only a modest memory and computing resources in order to be operational in real-time environment.

2.1 Multi-Scale Representation

The visual outlook of the maps varies depending on the scale. It is therefore not convenient to use multi-resolution image representation. Instead, several different map images should be reproduced for each desired scale. In addition to this, intermediate scales can also be provided by zooming the raster image. For example, Fig. 1 includes two different scales of a map (1:20 000 and 1:100 000) of the same location. The intermediate scale 1:40 000 has been generated from the detailed map (1:20 000) in order to provide the user with smoother zooming. The image has the same level of details but the size and quality of the features suffer because of the change in resolution.

The organization of the data is illustrated in Fig. 2. In this example, the large rectangles represent the map images of four different scales. The size of the rectangles corresponds to the size of the individual images in pixels. The images have usually the same size when printed on paper or shown on display. The thin grid lines drawn across the rectangles correspond to the spatial territories shown in the maps that have same size in reality but different resolution.

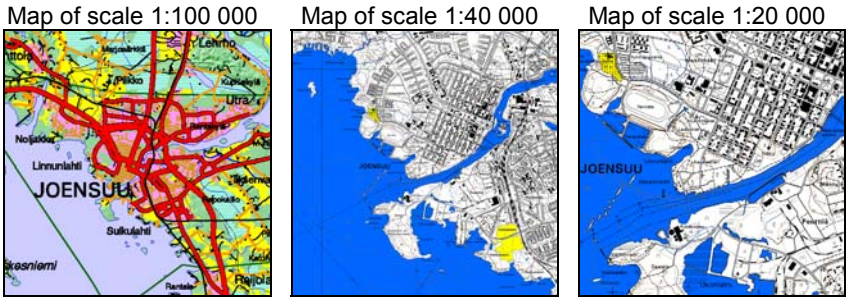

Fig. 1. Example of a map shown in three different scales. The highest and smallest scales have different representation of the content, but the intermediate scale (1:40 000) has been generated from the map of lowest scale (1:20 000)

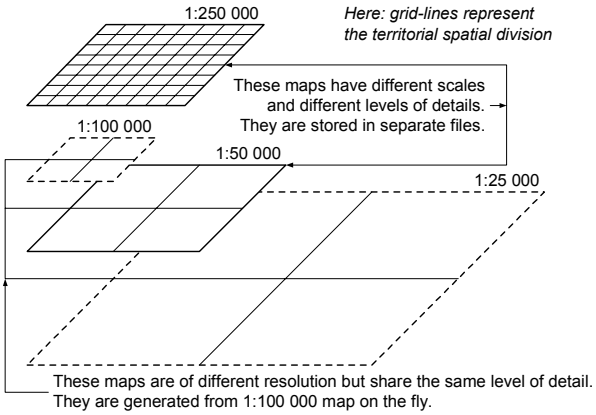

Fig. 2. Representation of the maps as separate map images (1:250 000 and 1:50 000), and as intermediate scales that are obtained by zooming

2.2 Image Compression

A compressed raster image format provides a reasonable solution in the form of compact storage size and compatible map format. Typical map images have high spatial resolution for representing fine details such as text and graphics objects but not so much color tones as photographic images, see Fig. 1. The most suitable compression methods can thus be found among the lossless graphics compression methods such as GIF and PNG (Miano 1999, Roelofs 1999). It is also possible to divide the maps into separate color layers and to apply the lossless binary image compression standards such as *ITU-T Group 4* (Arps and Truong, 1994), or the latest standard *JBIG2* (ISO/IEC, 1999, Howard et al 1998). However, lossy compression methods, such as the JPEG (Pennebaker and Mitchell, 1993), do not apply well for map images.

We take the JBIG1 and JBIG2 as the starting point of our map image storage system. They use context-based statistical modeling and arithmetic coding in the manner as originally proposed by Langdon and Rissanen (1981). The image is processed pixel-by-pixel in raster-scan order. The probability of each pixel is estimated on the basis of previous occurrences in similar context. The *context* is defined as the combination of already processed neighboring pixels defined by a template. Each context is assigned with its own statistical model that is adaptively updated during the compression process. Decompression is synchronous process with compression.

Here we apply the JBIG2 file format but use only the generic mode, which is basically the same as JBIG1. JBIG2 also segments the image into regions of different types, in particular, textual, halftone and generic (other), and utilize the repetitive nature of the textual and halftone images. However, the encoding of the data other than text or halftones remains similar to JBIG with the difference that

a newer version of the arithmetic coder (*MQ-coder*) is used. The pre-ancestor, *Q-coder*, has similar working principles (Pennebaker et al, 1998).

2.3 Decomposition to Binary Layers

In order to utilize the context-based compression, the map must be divided into binary layers. Each layer is then compressed separately, and the compressed layers are stored into the same file. There are three ways to perform the decomposition:

1. Semantic decomposition
2. Color separation
3. Bit-level separation

Semantic decomposition is possible if the maps are obtained from a map database in vector format. The map is output into a set of binary layers each containing different semantic meaning. We consider maps that consist of five layers: *basic* (topographic data and contours), *elevation lines*, *fields*, *water* and *property* (administrative borders), see Fig. 3. The user application can reproduce the map by plotting each layer by its own color overlapping each other in a given order. Color information can be added into the file. The image can be reconstructed as a gray scale image, or using any color set given by the user application. The benefits of the semantic separation are better compression performance, and that the layers to be shown can be selected at the time of viewing.

The second approach, *color separation*, can be used when we have only raster color image as the original map, and it contains only a limited number of colors (Tompkins and Kossentini, 1999). The image is divided into binary layers so that each layer represents one color in the original image. The drawback of the color separation is that information of the original semantic separation cannot be recovered. Furthermore, the color separation create artifacts into the binary layers, see Fig. 3. For example, overlapping text elements break the continuation of the fields and lakes. This does not decrease the quality of the image but it increases the complexity of the image, and thus, the compressed file size.

The third approach, *bit-level separation*, must be applied when we have the original map only as a raster image, and the number of different colors is too high for efficient color separation. For example, the image might have been digitized from a paper copy and stored using lossy compression method, such as JPEG (Pennebaker and Mitchell, 1993). In the bit-level separation, the number of colors are first reduced by quantizing the image into a limited-color representation of a 256 colors or gray-scales. The resulting pixel values are then separated into bit planes using Gray coding (Weinberger, Rissanen and Arps, 1996), and the image is represented as a sequence of binary images.

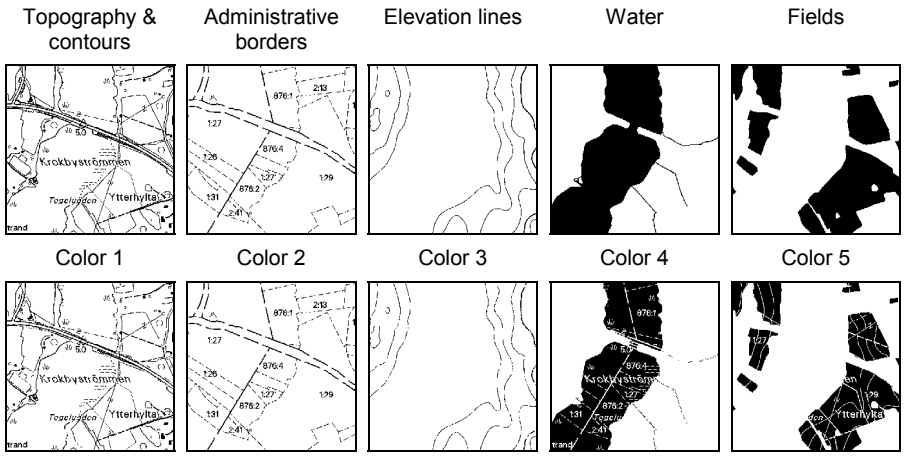

Fig. 3. Example of the image decomposed into binary layers by semantic separation (above), color separation (below)

2.4 Block Decomposition for Direct Access

The binary layers are divided into $b \times b$ non-overlapping rectangular blocks before the compression, and each block is compressed separately from others as proposed by Ageenko and Fränti (1998). The compressed blocks are stored in the same file, and an index table is stored in the header of the file to locate the starting points of the code blocks, see Fig. 4. When the compressed image is accessed, a block index table is constructed. This provides direct access to the compressed image file and enables efficient decompression of smaller fragments of the image.

The block decomposition has the effect that there are fewer pixels to be coded in the same run. It means that the model has less time to adapt to the statistics of the image. Another problem is the compression inefficiency near block boundaries. This is because the pixels located outside the block cannot be used in the context template. Previous studies indicate that the compression inefficiency remains tolerable if the block size is 256×256 pixels or higher (Ageenko and Fränti, 1998).

Somewhat better compression performance can be obtained using the following two modifications. First idea is to apply a forward-adaptive variant of the statistical modeling based on the ideas presented by Ageenko and Fränti (2000a). The forward-adaptive variant uses a pre-calculated initial model, which is constructed using the statistics collected from the entire image layer. This requires an additional pass over the image but it does not affect the speed of the decompression. The second idea is to use variable-size context modeling technique as described by Ageenko and Fränti (2000b). This technique reduces the size of the model, and it allows using larger context templates. These modifications can provide about 20% improvement in the compression performance.

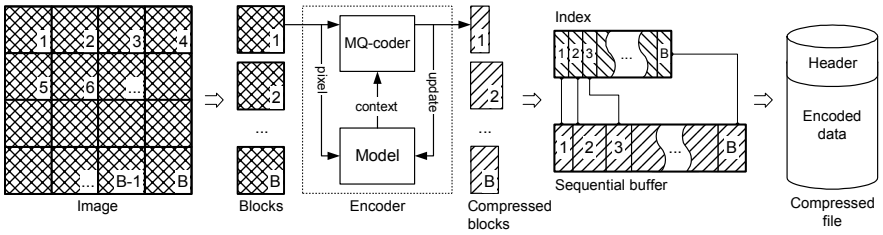

Fig. 4. Diagram of the block decomposition

2.5 Compression phase

The compression of a single map image is performed using the following steps:

1. Layer decomposition
2. Block decomposition
3. Compression

In the first step, the image is decomposed into the binary layers (unless the semantic decomposition already exists). The color space is enumerated and the number of required bit planes are generated. The resulting color palette is stored in the compressed file. If the palette is not stored, the image will be reconstructed as a gray scale.

In the second step, the layers are partitioned into blocks. If the forward-adaptive modeling variant is applied, non-empty blocks of the entire layer are analyzed and the statistical model is built and stored in the compressed file. The emptiness of a block is determined by checking whether all pixel values in the block are of the *default color value*.

In the third step, the series of the bit planes are compressed. Each non-empty block is compressed using context-based modeling and the MQ-coding algorithm. The context is determined by the standard 10-pixel template but we also permit the use of customized multi-level context-templates, e.g. such as described by Kopylov and Fränti (2002). The MQ-coder is reinitialized every time when the compression of a new block starts. The initialization resets the models either to the default model (50-50 probabilities), or to the optimized model (optional).

2.6 Use in the Client Device

A typical use scenario of the map image storage system (MISS) is to show the area surrounding the object whose position is tracked. The scale can be set by the user or it can be automatically determined on the basis of the speed, or other parameters defined in the application. The maps for the particular region are stored in the client's viewing device (flash memory, hard disk, CD), or the images can be

located in a remote server and accessed via communication network (Internet, GSM, GRPS)

The system uses the following steps to show the current view on the map. First, the file possessing the map of the desired location is accessed and its header part is retrieved in memory. From the header, the type and structure of the image are determined, and the block index table is built. The table indicates the size and location of each block in the compressed file. All supplementary data required for image decoding (such as initial model and possible context tree) are also retrieved and kept in memory until this particular map image is no longer used.

Next, depending on the requested location, the system calculates the image fragment needed to be displayed. The blocks covering this fragment are retrieved and decoded. If the map image is accessed remotely, the retrieved blocks are also stored (in compressed form) in the local storage space for further use. When the position of the object changes, the view is updated by decoding new image blocks in the direction of movement, see Fig. 5.

To speed-up the access to the image, the system may use cache for temporary storage of the decompressed image blocks, see Fig. 6. The size of the cache can be fixed, or it can be determined by the amount of free memory. The performance may be further improved by exploiting an idle time for decoding neighboring blocks further in the direction of movement, before the blocks are actually requested. In practice, it is convenient to buffer the data according to the block boundaries.

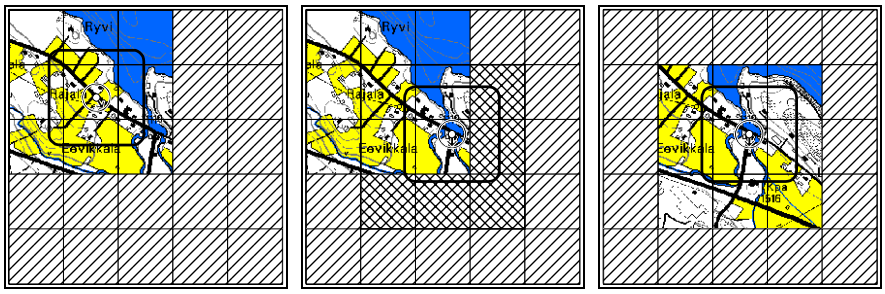

Fig. 5. Image decoding in real-time system. Nine blocks are first decompressed and stored in the cache (left). Change of location is then registered (middle), and new image blocks are then decompressed and the view updated (right)

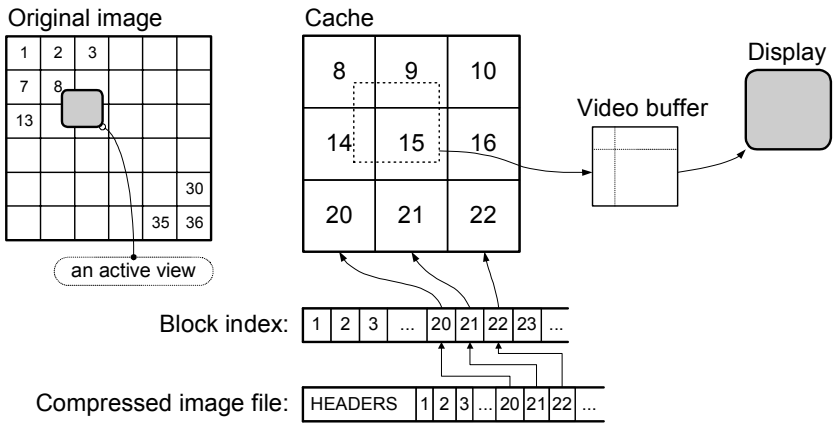

Fig. 6. Illustration of the cache operation

3 Experimental Results

We study next the compression performance and the retrieval times of the proposed storage system. The following methods are considered in the comparisons: MISS (the proposed method), JBIG2 (ISO/IEC, 1999), TIFF G4 (Arps and Truong, 1994), GIF (Miano, 1999), PNG (Roelofs, 1999) and RAW (uncompressed).

We use four images taken from the topographic database by National Land Survey of Finland (1999). The images consists of five layers, each of the size 5000×5000 pixels, and the scale 1:20 000. This corresponds to resolution of two meters per pixel. We use two versions of the images. The Set #1 (semantic decomposition) includes the images when separated into the five semantic layers. The Set #2 (color separation) contains the same set of images but after the following processing. Color image is first constructed from the original semantic layers, and color separation is then performed to estimate the original division, see Fig. 3. This represents the situation, in which only the color image is available as the original.

3.1 Compression Performance

The compression results have been summarized in Table1. The average compression performance of MISS is about 0.20 bits per pixel but the result depends on the complexity of the image. The MISS files take about 5-15% more space than that the JBIG2 files, on average, but 50-65 % less than the comparative methods (TIFF-G4, GIF, PNG). The MISS, on the other hand, is the only method that supports direct access to the compressed file. The results in Fig. 7 shows that most of the bits (about 54 %) originates from the basic information, whereas the

water and fields are rather easy to compress. The compression results between the Set #1 and Set #2 are not significant.

The effect of the block size is illustrated in Fig. 8. The optimal block size is around 350×350 to 500×500 in terms of compression performance. With the chosen 100×100, the files sizes are slightly bigger but this block size allows more dense tiling, and hence, more accurate buffering with less memory resources, and smaller transmission and decompression delays.

Table 1. Compression results (kilobytes) for the Set #1 (semantic decomposition)

	MISS	JBIG2	TIFF G4	GIF	PNG	RAW
Image 1	247	197	372	727	602	15259
Image 2	1109	969	2382	2866	2608	15259
Image 3	395	327	604	1142	1100	15259
Image 4	840	759	1540	2633	2534	15259
Total	2591	2252	4899	7367	6845	61035
Bpp	0.21	0.18	0.40	0.60	0.56	5.00

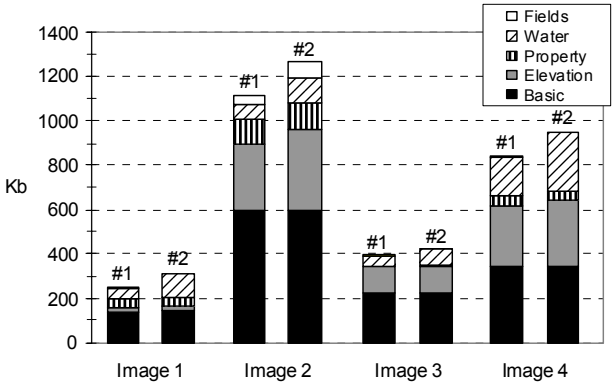

Fig. 7. The proportion of the layers in the compressed MISS files for the Set #1 (left columns), and Set #2 (right columns).

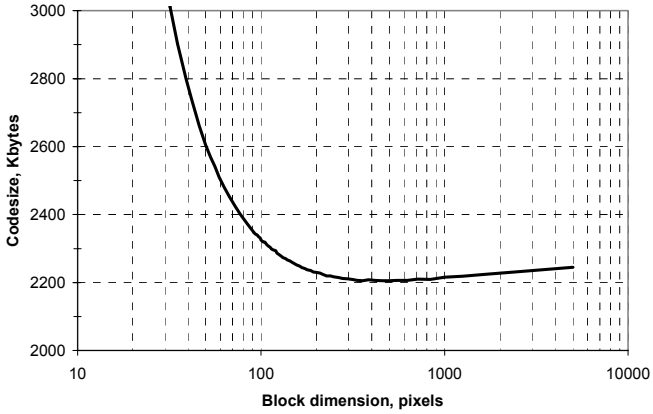

Fig. 8. The effect of the block size on the code size for the Set #1

3.2 Retrieval Timings

We consider next the retrieval performance of the proposed system by measuring time required to transmit and decompress a desired part of the map. We assume that the client device buffers the image data according to the nearest block boundaries.

The decompression times are summarized in Table 2 when decompressing the complete 5000×5000 images. The results show that the better compression performance of the MISS has been obtained at the cost of 10-20 times slower decoding speed in comparison to GIF and PNG. The decoding speed of MISS corresponds to 973,710 pixels per seconds for the images in the Set #1, on average. Using this result, we calculated sample retrieval timings for several screen sizes with varying computing power and transmission speeds.

Table 2. Decompression times (s) for the Set #1 using a processor of 1000 MIPS

	Method:		
Image:	MISS	GIF	PNG
Image 1	25.9	2.1	1.2
Image 2	31.4	2.5	1.4
Image 3	18.4	2.2	1.1
Image 4	27	2.2	1.3
Total	102.7	9.0	5.0

We consider the following three transmission networks: (1) GSM capable of 9600 bits per second; (2) high speed GSM capable of 14400 bits per second; (3) GRPS network, capable of 48 kilobits per second. Typical hand-held devices have

relatively low computing power; we consider three speeds (10, 50, 100 MIPS) that roughly correspond to the computing power of the current low-cost compact devices. The retrieval timings are calculated as:

Transmission time = compressed data size / channel bandwidth

Decompression time = uncompressed image size / decoding speed

The results are summarized in Table 3 for retrieving a full screen. It is shown that the data for a reasonable size screen can be transmitted and decompressed in real-time using existing networks and relatively low-speed devices. For example, screen size of 150×150 equals to 22,500 pixels, and 0.21·22,500 = 4,725 bits.

Table 3. Transmission and decompression times for retrieving full screen

	Transmission times (s):			Decompression times (s):		
Screen size:	9600 bps (GSM)	14400 bps (hs-GSM)	48 kbps (GRPS)	10 MIPS	50 MIPS	100 MIPS
100×100	0.22	0.15	0.04	1.03	0.21	0.10
150×150	0.49	0.33	0.10	2.31	0.46	0.23
200×200	0.88	0.58	0.17	4.11	0.82	0.41
250×250	1.37	0.91	0.27	6.42	1.28	0.64

4 Conclusions

We have proposed a map image storage system (MISS) for real-time applications that use low performance portable devices. The system architecture is designed to minimize storage size, transmission time, and memory requirements. Compact size is achieved by context-based statistical compression. Direct access to the compressed image file is supported and it allows to transmit/decompress partial images. This minimizes the transmission time and memory requirement in the user device. These properties together enable the real-time use of large map images.

Acknowledgments

The work was supported by the National Technology Agency of Finland (TEKES) as the projects Real-time cartography imaging and Dynamic use of maps in mobile environment.

References

Ageenko E and Fränti P (1998), "Enhanced JBIG-based compression for satisfying objectives of engineering document management system", Optical Engineering, 37: 1530-1538.

Ageenko E and Fränti P (2000a) "Forward-adaptive method for compressing large binary images," Software Practice & Experience, 29: 943-952.

Ageenko E and Fränti P (2000b) "Compression of large binary images in digital spatial libraries", Computer & Graphics, 24: 91-98.

Arps RB, Truong TK (1994) "Comparison of international standards for lossless still image compression, Proceedings of the IEEE 82: 889-899.

Dye S, Buckingham S (1999) Mobile Positioning, Mobile Lifestreams.

Gellersen HW, Thomas PJ (eds.) (2000) Proc. 2nd International Symposium on Handheld and Ubiquitous Computing (HUC'2000), Bristol, UK, Springer Verlag, September.

GeoData (1999) CD-tiekartasto Suomi 1:250 000 (CD Roadmap Catalog Finland), WSOY, Helsinki. (in Finnish)

Fränti P, Ageenko E, Kopylov P and Gröhn S (2002), "Compressing multi-component digital maps using JBIG2", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, (ICASSP'02)*, Orlando, Florida.

Haskell BG et al (1998) "Image and video coding – emerging standards and beyond," IEEE Trans. Circuits and Systems for Video Technology, 8: 814-837.

Howard PG, Kossentini F, Martins B, Forchammer S and Rucklidge WJ, (1998) "The emerging JBIG2 standard," IEEE Trans. Circuits and Systems for Video Technology, 8: 838-848.

ISO/IEC (1993) Progressive Bi-level Image Compression, 11544, ITU-T Recommendation T.82.

ISO/IEC (1999) Final Committee Draft for ISO/IEC International Standard 14492. (`http://www.jpeg.org/public/jbigpt2.htm`)

Kaplan ED, (ed.) (1996) Understanding GPS: Principles and Applications, Artech House Telecommunications Library.

Kopylov P and Fränti P (2002), "Context tree compression of multi-component map images", IEEE Data Compression Conference (DCC'02), Snowbird, Utah, April 2002.

Kraak M-J, et al, (1996) Cartography : Visualization of Spatial Data, Addison-Wesley.

Kraak M-J, Brown A (2000) Web Cartography, Taylor & Francis.

Langdon GG, Rissanen J (1981) "Compression of black-white images with arithmetic coding", IEEE Trans. Communications 29: 858-867.

Miano J (1999) Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP, (ACM Press), Addison-Wesley , Boston.

Murray JD, vanRyper W, Russell D (eds.) (1996) Encyclopedia of Graphics File Formats, 2-nd ed, O'Reilly Associates Inc.

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

Pennebaker WB, Mitchell JL (1993) JPEG Still Image Data Compression Standard. Van Nostrand Reinhold, New York.

Pennebaker WB, Mitchell JL, Langdon GG, Arps RB (1988) "An overview of the basic principles of the Q-coder adaptive binary arithmetic coder," IBM Journal of Research and Development, 32: 717-726.

Roelofs G (1999) PNG: The Definitive Guide, O'Reilly & Associates, Cambirdge, MA.

Tompkins D, Kossentini F (1999) "Additional Extension Segments for JBIG2," ISO/IEC JTC 1/SC 29/WG1 (ITU-T SG8), Document No 1318.

Weinberger MJ, Rissanen J, Arps R (1996) "Application of universal context modeling to lossless compression of gray-scale images," IEEE Trans. Image Processing, 5: 575-586.