

Evaluation of compression methods for digital map images

Pasi Fränti, Pavel Kopylov and Eugene Ageenko
University of Joensuu, Box 111, 80101 Joensuu
FINLAND

ABSTRACT

Digital maps can be stored and distributed electronically using compressed raster image formats. In this paper, we evaluate the most suitable compression methods for storing digital map images. We apply the latest compression standards for binary images (G4, JBIG2), and the standards for computer graphics with limited number of color tones (GIF, PING). Experimental results are given for two sets of clean noiseless map images generated from digital map database, and for one set of noisy images.

KEY WORDS: Image compression, map images, lossless compression, personal navigation.

1. INTRODUCTION

Real-time map imaging applications provide user with the view of geographic map for the area surrounding the user's location [1]. The system may use *global positioning service* (GPS) [2] or *mobile positioning service* (MPS) [3] for obtaining the coordinates of the current location. The location can be updated in real-time about once or twice in every second.

Digital maps are usually stored in vector format in a database for accessing and retrieving the data using spatial location as the search key. 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 biggest problem of the vector format is that maps are not always available for the user in vector format. The maps are also stored in various formats and incompatibility between different systems can restricts the use of the maps. In this case, the raster image format is the only choice. The use of raster map is straightforward: the map is already stored in the form it will be used.

The biggest problem of the raster format is the huge storage size of a raster map. For example, electronic library of Finnish road maps of the resolution 1:250 000 took an entire CD (over 600 Mb) in uncompressed form [4]. 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 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 [5, 6]. 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* [7] or the latest standard *JBIG2* [8].

In this paper, we consider the above methods and study how well they apply to the compression of map images. We consider three different representations of the maps. On the basis of the results, we conclude that JBIG-based approach is the most efficient method for the purpose among the methods considered. Finally, we outline how the JBIG-based compression can be applied to dynamic server-client map imaging application in personal navigation.

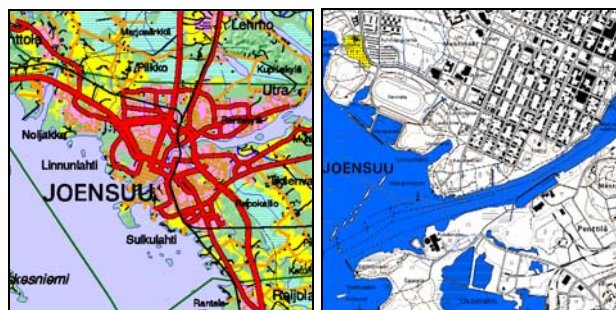


Figure 1: Example of map images with different scales.

2. REPRESENTATION OF DIGITAL MAPS

We study next how the digital maps are stored, and how they should be distributed efficiently in mobile map imaging applications.

2.1 Sources of digital maps

Maps can be stored in map server using common database format such as *ArcShape*. The maps are reproduced for the client in compressed raster format. The advantage of this approach is that the maps can be stored in vector format in the server but the client solution can operate with devices of low memory and computing resources. We consider the NLS map server as an example of such approach [9].

Fig. 2 illustrates two maps from the same NLS topographic database. The images are of the size 5000×5000 pixels, and have the original scale 1:20 000. This corresponds to a resolution of two meters per pixel. These images are composed of five semantic layers representing topography, elevation lines, waterways, fields and administrative boundaries (when available).

Another major source of maps are digitized raster maps. Such maps can contain *digitization noise*, they can be *dithered*, or compressed by lossy methods that are not suitable for map images. For example, sometimes map images can be stored in JPEG format [10], which is a widely used standard for photographic images but it is not suitable for images with small details such as text and graphical objects. Nevertheless, all such images must be handled by the compression method. Two examples are shown in Fig. 3.

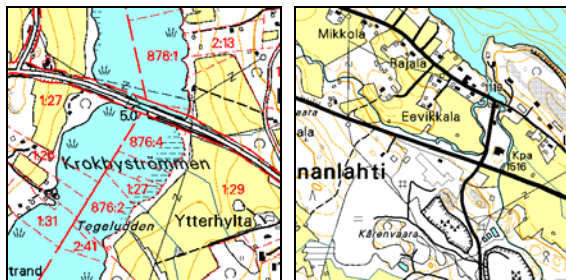


Figure 2: Sample fragments from NLS topographic images of scale 1:20 000 . The sample to the left includes also administrative boundaries (red lines and numbers).



Figure 3: Sample fragments from the road map images *Sea* and *Vantaa* from set #2.

2.2 Image representation

The maps can be represented as raster image in two alternative ways:

- color palette image,
- set of binary layers.

Color palette image is a special case of *true color* images, in which only a limited set of predefined colors are used. The layer separation is an alternative approach, which has the advantage that several efficient compression methods exist for binary images. The layer separation, on the other hand, must be then be solved in some way. There are standard although sometimes little bit naïve solutions for the conversion. Fig. 2 illustrates the overall relationship between different representations of map from our point of view.

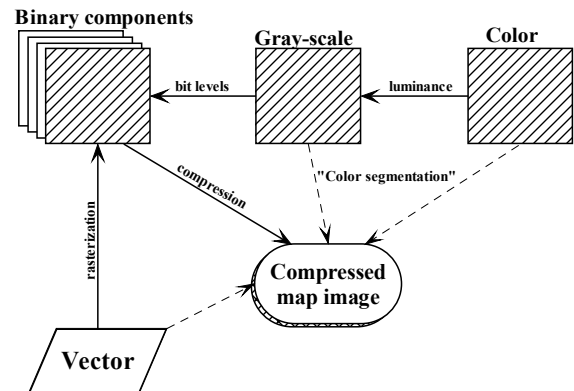


Figure 4: Conversion diagram for using layer separation and binary image compression methods.

2.3 Separation to binary layers

There are three ways to perform the decomposition into binary layers:

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

Semantic layer separation is possible if the maps are obtained from a map database. This is the case of the NLS map images illustrated in Fig. 3. The map is output into a set of binary layers each containing different semantic meaning of the location that the map represents. The user application reproduce the map by plotting each layer by its own color overlapping each other in a given order.

The second approach, *color separation*, can be used when we have raster color image as the original map, and it contains only a limited number of colors. The image is divided into binary layers so that each layer represents one color in the original image. The semantic layer separation and the corresponding color separation are illustrated in Fig. 5.

The third approach, *bit-level separation*, must be applied when the number of different colors is too high for efficient color separation. In the bit-level separation, the number of colors is first reduced by mapping the image into a limited-color representation of a 256 colors, or into a gray-scale image. The resulting pixel values are then separated into eight bit planes, which are represented as a sequence of binary images.

The drawback of the color and bit-level separation is that they lack the semantic information of the map. The separation process can also create artifacts into the binary layers at the color transition points, see Fig. 6. For example, overlapping text elements breaks the continuation of the fields and lakes, which increases the complexity of the binary images and thus, results in higher file size.

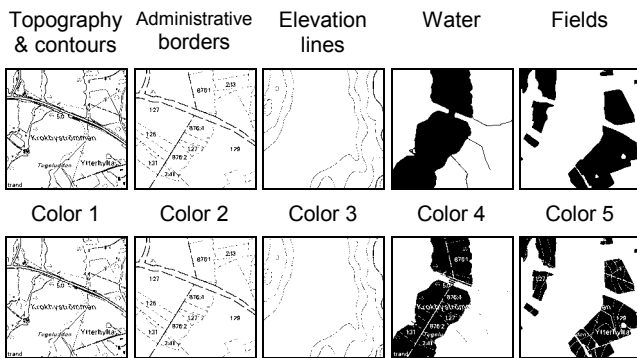


Figure 5: Example of semantic layers separation (above), and color separation (below).



Figure 6: The original water layer (left), and the same information when obtained by color separation (right).

3. COMPRESSION

We consider the following compression methods:

- GIF
- PNG
- TIFF G4
- JBIG

The first two methods (GIF, PNG) use the color representation for the image, and the other two (G4, JBIG) apply layer separation.

3.1 GIF

The *CompuServe Graphics Interchange Format* (GIF) [5] is the most widely used format in Internet for color palette images. It uses the dictionary-based compression method known as LZW [11], which is based on the LZ78 method of the well known Ziv-Lempel methods. The LZW dynamically constructs a dictionary of repeating pixel sequences on the image. Further appearances of the same sequence are then coded by an index to the dictionary. The compression performance depends on how long pixel sequences can be coded by one index.

3.2 PNG

The *Portable Network Graphics* (PNG) [6] is based on the LZ77 dictionary-based compression scheme. In specific, the PNG uses the same *deflate* algorithm that is also used in *gzip*. It parses the pixel sequence and finds the longest sequence that matches to an earlier occurrence of the same sequence in the buffer of previous pixels. The method uses a sliding window of size 32K. The pointer and the length of the sequence are coded by *Huffman coding*.

3.3 TIFF G4

TIFF G4 refers here to the old but still widely used *ITU-T Group 4* fax compression standard [7]. The Group 4 standard is based on *relative element address designate (READ)*, which codes the transition points between black and white runs, in respect to the corresponding transition point in the previous line. This is referred as *vertical mode*. As a special case, the READ uses also *run-length encoding (RLE)*, which codes the lengths of the run by Huffman coding. This is referred in G4 as *horizontal mode*. The G4 is implemented in the *Tagged Image File Format* (TIFF) as the compression method #4. We use this implementation in our experiments.

3.4 JBIG

JBIG1 was originally published in 1993 to as a replacement for the G4 standard for binary images [12]. It was later extended to JBIG2 for better consideration of textual images [13] but the core of both standards is the same. The JBIG-based methods use context-based statistical modeling and arithmetic coding in the manner as originally proposed by Langdon and Rissanen in 1981 [14]. Here we apply the JBIG2 file format but use only the generic mode, which is basically the same as JBIG1. We refer it simply as JBIG.

The JBIG processes the image pixel-by-pixel in raster-scan order. The probability of each pixel is estimated on

the basis of previous occurrences in similar context. This is known as dynamic modeling. 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. Fig. 7 illustrates how the context is determined from the neighboring pixel values. Decompression is synchronous process with compression.

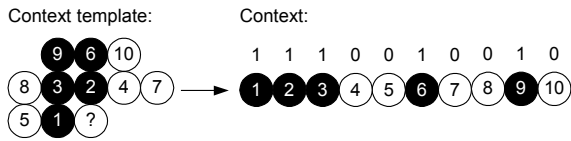


Figure 7: An example of a 10-pixel context.

4. EXPERIMENTS

In the following experiments, we use three image sets: The Set #1 (semantic decomposition) includes four images of size 5000×5000 pixels from the NLS database. These images consist of five semantic layers so we can apply the semantic layer separation as explained in Section 2.3. All compression methods are applied for each layer separately.

The Set #2 (color separation) contains the same set of images after converted to color raster images. This makes the semantic layer separation impossible. Instead, we apply color separation in the case of G4 and JBIG. The GIF and PNG are applied directly to the color images as such.

The Set #3 consists of two color map images including noise and a high number of output colors. The number of different color is 146 (*Sea*) and 194 (*Vantaa*). We first map the images into 256 gray-scales, apply *Gray coding*, and divide the images into 8 bit-planes. The binary image compression methods (G4 and JBIG) are then applied to the bit planes. The GIF and PNG are applied to the gray-scale images as such.

The compression results have been summarized in Tables 1-3 for the three sets. The average compressed file size of JBIG is 50-65 % less than that of the other methods in the case of Set #1 and Set #2. The Set #2 results in about 15% higher file sizes but the relative performance of the different methods is similar.

Fig. 11 illustrates further the source of the bits in the Set #1 and #2. It shows that most of the bits (about 54 %) originates from the basic information, whereas the water and fields are rather easy to compress. So even though the color separation adds into the size of the elevation lines, water and fields, their overall effect is not so significant. It is also noted that the administrative boundaries were absent in the case of image 3.

The images in the Set #3 are much more difficult to compress because of the noise and the level of complexity of the images. The average bit rate for these images is

0.52 for JBIG, and the performance gap between JBIG and the comparative methods is much smaller. In fact, GIF gets slightly lower bit rate than JBIG. The results demonstrate the importance of having the image divided into semantic layers, or at least have a clean original so that the color separation can be performed properly.

Table 1: Compression results (kilobytes) for the set #1 (semantic decomposition). The raw file size would be 15259 Kb per image.

	JBIG	TIFF G4	GIF	PNG
Image 1	197	372	727	602
Image 2	969	2382	2866	2608
Image 3	327	604	1142	1100
Image 4	759	1540	2633	2534
Total	2252	4899	7367	6845
Bpp	0.18	0.40	0.60	0.56

Table 2: Compression results (kilobytes) for the set #2 (color separation). *The GIF and PNG results are obtained by compressing the corresponding color image instead of the binary layers. The raw file size would be 15259 Kb per image.

	JBIG	TIFF G4	GIF*	PNG*
Image 1	258	466	610	654
Image 2	1138	2605	2642	2644
Image 3	360	691	1028	1074
Image 4	875	1902	2295	2384
Total	2631	5664	6575	6757
bpp	0.22	0.46	0.54	0.55

Table 3: Compression results (kilobytes) for the set #3 (bit-level separation). The raw file size would be 1250 Kb per image.

	JBIG	TIFF G4	GIF	PNG
Sea	197	705	165	275
Vantaa	449	878	445	521
Total	646	1583	610	796
Bpp	0.52	1.27	0.49	0.64

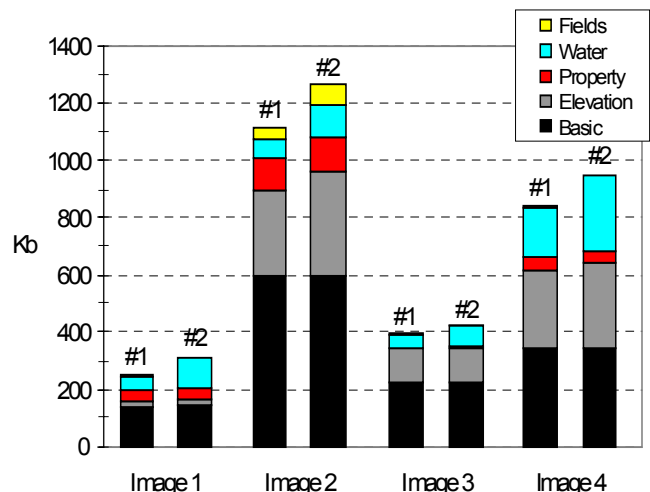


Figure 8: Storage size of the map set #1.

From the test results we can conclude that the JBIG has clearly the best compression performance for storing the map images among the other methods considered. The other functionalities of the map imaging system, such as tiling and direct access, have been recently studied within the JBIG2 standard [15], and on the basis of the results, real-time map imaging system has been proposed in [16].

So far we have restricted our study to existing *standard* methods. The latest research, however, have shown that further compression improvement can still be achieved by the use of better modeling schemes such as *multilevel context tree (MCT)* [17]. Additional comparisons are therefore given in Fig. 9, in which the latest results have also been included.

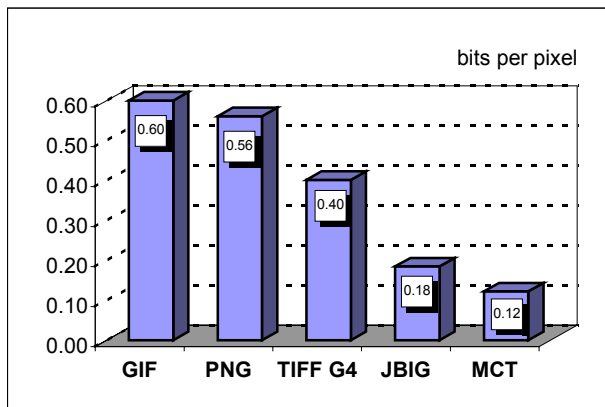


Figure 9: Summary of the compression results for the Set #1 using the standard compression methods plus the MCT method proposed in [17].

5. CONCLUSION

We evaluated methods for compressing digital map images. The results showed that JBIG can provide significant compression by a factor of about 23:1, and the most recent method by a factor of about 47:1. To sum up, raster map images can be compressed efficiently so that map image size would not be the bottleneck.

6. ACKNOWLEDGEMENT

The work was supported by the *National Technology Agency* of Finland (TEKES) under the projects *Real-time cartography imaging*, and *Dynamic use of maps in mobile environment*. (<http://cs.joensuu.fi/pages/franti/dynamap/>)

REFERENCES

[1] M.-J. Kraak and A. Brown, *Web cartography* (Taylor & Francis, 2000).

[2] E.D. Kaplan (ed.), *Understanding GPS: principles and applications* (Artech House Telecommunications Library, March 1996).

[3] S. Dye, S. Buckingham, *Mobile positioning* (Mobile Lifestreams, 1999).

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

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

[6] G. Roelofs, *PNG: The Definitive Guide*, (O'Reilly & Associates, Cambirdge, MA, June 1999).

[7] R.B. Arps and T.K. Truong "Comparison of international standards for lossless still image compression," *Proceedings of the IEEE*, 82 (6), 1994, 889-899.

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

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

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

[11] T. Welch, "A technique for high-performance data compression", *IEEE Computer*, 17 (6), 8-19, June 1984.

[12] ISO/IEC, *Progressive Bi-level Image Compression*, 11544, ITU-T Recommendation T.82, 1993.

[13] ISO/IEC, *Final committee draft for ISO/IEC International Standard 14492*, 1999. (<http://www.jpeg.org/public/jbigpt2.htm>)

[14] G.G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding", *IEEE Trans. Communications*, 29 (6), 858-867, June 1981.

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

[16] P. Fränti, E. Ageenko, P. Kopylov, S. Gröhn and F. Berger, "Compression of map images for real-time applications", Univ. of Joensuu, Dept. of Computer Science, Technical Report A-2001-1.

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