Compression of line-drawing images using vectorizing and feature-based filtering

Pasi Fränti, Eugene I. Ageenko

Department of Computer Science, University of Joensuu
P.O. Box 111, FIN-80101 Joensuu, FINLAND

Alexander Kolesnikov, Igor O. Chalenko

Institute of Automation and Electrometry, Russian Academy of Sciences
Pr-t Ak. Koptyuga, 1, Novosibirsk-90, 630090, RUSSIA

Abstract:

A three-stage method for compressing bi-level line-drawing images is proposed. In the first stage, the raster image is vectorized using a combination of skeletonizing and line tracing algorithm. A feature image is then reconstructed from the extracted vector elements. In the second stage, the original image is processed by a feature-based filter for removing noise near the borders of the extracted line elements. This improves the image quality and results in more compressible raster image. In the final stage, the filtered raster image is compressed using the baseline JBIG algorithm. For a set of test images, the method achieves a compression ratio of 40:1, in comparison to 32:1 of the baseline JBIG.

Keywords: *image compression, JBIG, raster-to-vector conversion, context modeling, feature based filtering.*

1. INTRODUCTION

Lossless compression of bi-level images has been well studied in the literature and several standards already exist [1]. In the *baseline JBIG* the image is coded pixel by pixel in scan raster order using *context-based probability model* and *arithmetic coding* [2]. The combination of already coded neighboring pixels defines the context. In each context the probability distribution of the black and white pixels are adaptively determined. The current pixel is then coded by *QM-coder* [3], the binary arithmetic coder adopted in JBIG.

The baseline JBIG achieves compression ratios from 10 to 50 for typical A4-size images. The pixelwise dependencies are well utilized and there is no much room for improvement. Remarkable improvement has been achieved only by specializing to some known image types and exploiting global dependencies, or extending the methods to lossy compression. For example, the methods in [4, 5] includes pattern matching technique to extract symbols from text images. The compressed file consists of bitmaps of the library symbols coded by a JBIG-style compressor, location of the extracted marks as offsets, and a pixelwise coding of the matched symbols using two-layer context template.

We study similar approach for *line-drawing images* by utilizing global dependencies in images such as engineering drawings, cartographic maps, architectural and urban plans, schemes, and circuits (radio electrical and topological). These kinds of images contain mainly of straight-line elements. Global information can be gathered by extracting line features from the image and utilizing them in the compression of the raster image.

We propose a three-stage compression method as outlined in Figure 1. In the first stage (*vectorizing*), vector elements are extracted from the image using raster-to-vector conversion. Equal size feature image is created from the extracted line segments to approximate the input image. In the second stage (*filtering*), the original raster image is preprocessed by a feature-based filtering for improving image quality. The feature image is used as a semantic model of the image, consisting of non-local information from the image, which cannot be utilized using local spatial filters. In the third stage (*compression*), the filtered image is compressed by the baseline JBIG. The feature file is used only in the compression phase and therefore it is not needed to store in the compressed file.

The feature extraction and filtering are considered as preprocessing and they are invisible in the decompression phase. The method uses standard image compression component – the baseline JBIG. The resulting output files are therefore standard JBIG files and the decompression is exactly the same as the baseline JBIG. The method can thus be easily integrated into existing compression systems. The vectorizing and filtering parts can be implemented as optional components, and used on-demand only.

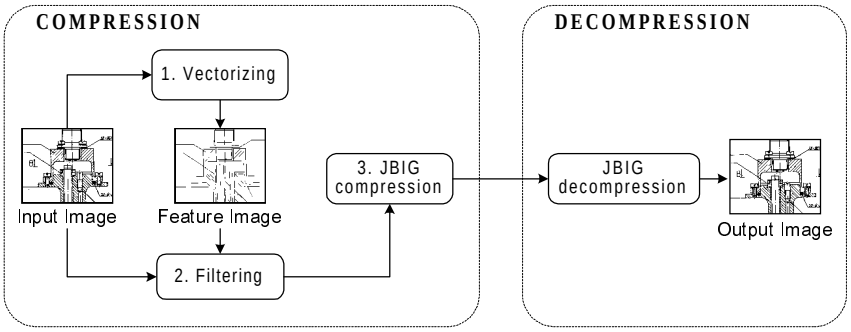


Figure 1: Block diagram of the three-stage compression method.

The method is near-lossless because only isolated groups of noise pixels can be inverted. Moreover, undetected objects (such as text characters) are untouched allowing their lossless reconstruction. Uncontrolled loss of image quality cannot therefore appear.

The vectorizing is an essential part of the new method but the method itself is independent on the chosen vectorizing component. The quality of the vectorizing affects only on the amount of compression improvement that can be achieved. The quality of the output image is controlled by the filtering method. The details of the three stages are discussed next in the following sections.

2. FEATURE EXTRACTION USING VECTORIZING

The vectorizing process is outlined in Figure 2. We apply the method described in [6]. The motivation is to find rigid fixed-length straight lines from the image. Each line segment is represented as its two end-points and as the width of the line. A feature image is then reconstructed from the line segments and utilized in the filtering. The vector features are not stored in the compressed files but the vectorizing is considered as an intelligent preprocessing stage. The details of the vectorizing process are described in the following subsections.

2.1 Skeletonization

The black-and-white raster image is processed by a distance transform (defined by 4-connectivity). The resulting width-labeled image is then skeletonized using the algorithm in [7]. It proceeds the pixels layer by layer starting from contour pixels (pixels with distance value 1). The 3×3 neighborhood of each pixel is checked. The pixels satisfying one of the so-called "multiplicity condition" are marked as skeletal pixels. The process is then iterated for the pixels of the next layer (pixels with distance value 2) and so on until all layers are processed. The result of the algorithm is a width-labeled skeletal image. Fast and memory efficient implementation of the algorithm was constructed using the ideas presented in [8].

2.2 Extraction of elementary vectors

Vector primitives are extracted from the skeletal image using a fast and simple line-tracing algorithm. We trace all branches of the skeleton, one pixel at a time, from one delimiter (line end or crossroad) to another. At this stage, no digression from the current direction is allowed during the traversal. The length of the vector primitives is limited to be at most five pixels. The extracted lines are stored as single vector elements using the average width of the skeletal pixel as the line width.

The actual implementation uses LUT and cellular logic. Each pixel is processed by examining its neighborhood in a 3×3 window. An index is constructed from neighboring pixel values and a precalculated look-up table (of size 2^9) is

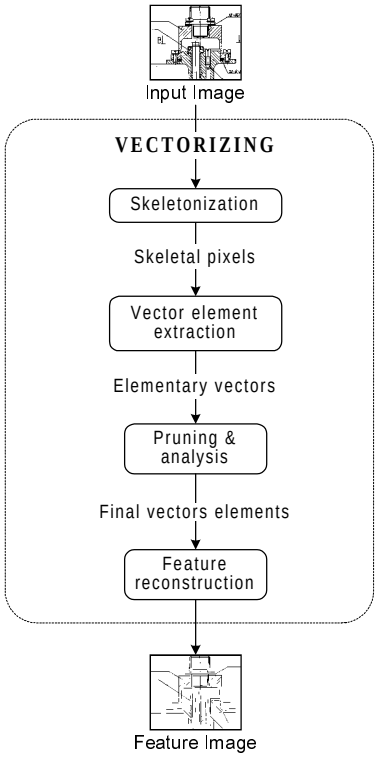


Figure 2: Block diagram of the vectorizing**.**

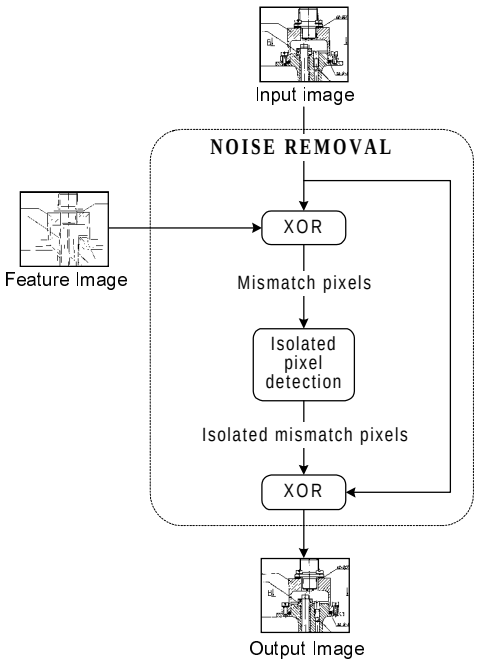


Figure 3: Block diagram of the noise removal procedure.

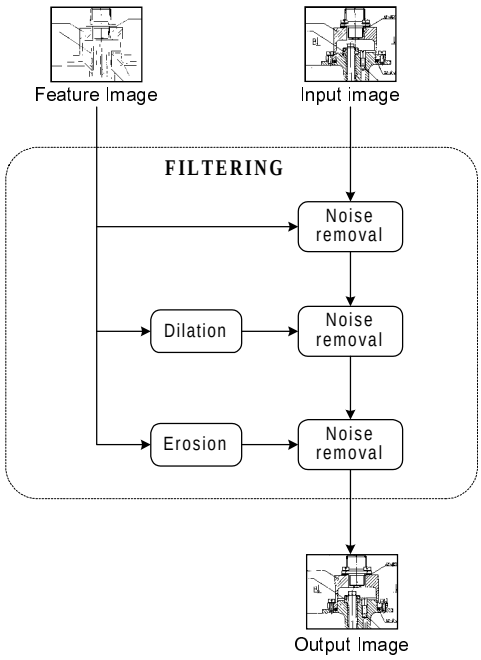


Figure 4: Block diagram of the three-stage filtering procedure.

accessed. The current direction is the second index for the LUT. Actions for all situations are precalculated in advance and stored. The LUT gives a binary answer whether the current pixel is accepted or rejected. The algorithm works extremely fast in practice.

2.3 Pruning and analysis

The vector primitives are further analyzed for constructing larger elements. There are four classes of the combined vector elements, each described by the two end-points and the width of the line:

- Single point: (x_1, y_1, w_1).
- Single vector: (x_1, y_1, w_1), (x_2, y_2, w_2).
- Chain of n vectors: $\{(x_k, y_k, w_k) \mid k = 1,\ldots, n+1\}$.
- Ring of n vectors: $\{(x_k, y_k, w_k) \mid k = 1,\ldots, n+1\}$ where $x_1=x_n$ and $y_1=y_n$.

Vector elements are combined (pruned) from primitives having a common end-point and same orientation (within a certain error marginal). Small gaps between the lines are filled and false branches are removed. The remaining vector chains are then classified either as “good” (linear) or “bad” (noise and non-linear). The good chains are stored by their coordinate differentials using a variable-length code. The bad chains are symbols and spots, and they are stored as raster objects.

3. FEATURE-BASED FILTERING

In the second stage, the original image is processed by a feature-based filter for removing noise near the borders of the extracted line elements. This improves the image quality and results in more compressible raster image. The filtering is based on a simple noise removal procedure, as shown in Figure 3. Mismatch image is constructed from the differences between the original and the feature image. Isolated mismatch pixels (and pixel groups up to two pixels) are detected and the corresponding pixels in the original image are inverted. This removes random noise and smoothes edges along the detected line segments.

The noise removal procedure is successful if the feature image is accurate. The vectorizing method, however, does not always provide exact width of the lines. The noise removal procedure is therefore iterated three times as shown in Figure 4. In the first stage the feature image is applied as such, in the 2nd stage the feature image is *dilated*, and in the 3rd stage it is *eroded* before input into the noise removal procedure. This compensates most of the inaccuracies in the width detection. See [9] for the details of the morphological dilation and erosion.

The stepwise process is demonstrated in Figure 5 for a small image sample. Most of the noise is detected and removed in the first phase. However, in some cases there are too many mismatch pixels grouped together because of a wrong estimation of the line width and therefore no pixels can be filtered. Even if these inaccuracies may have visually

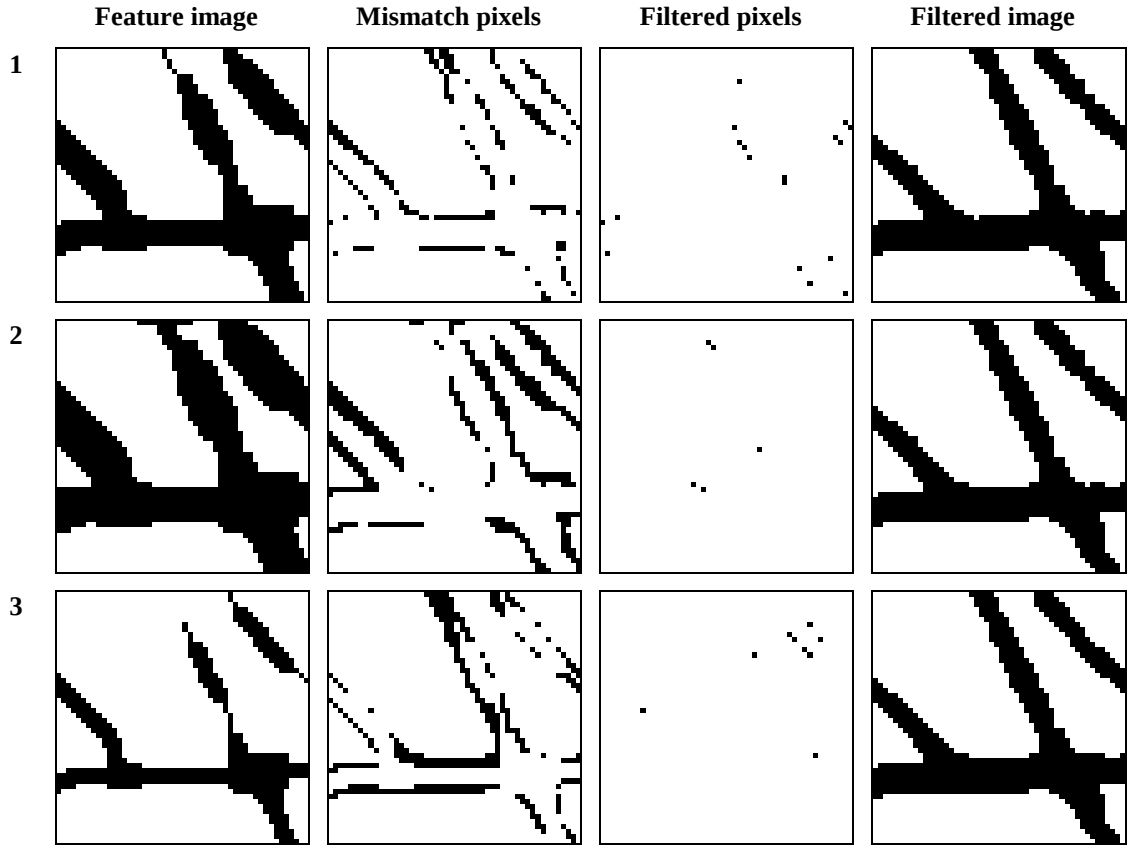


Figure 5: Illustration of the three-stage filtering procedure. The first line corresponds to the first stage, the second line when the feature image is dilated, and the last line when the feature image is eroded.

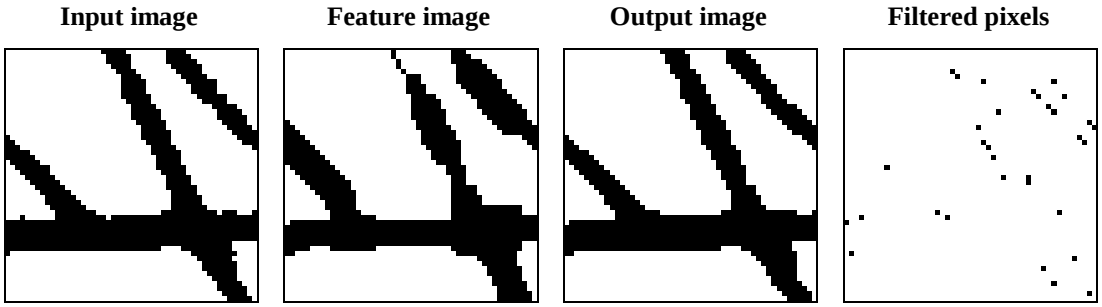
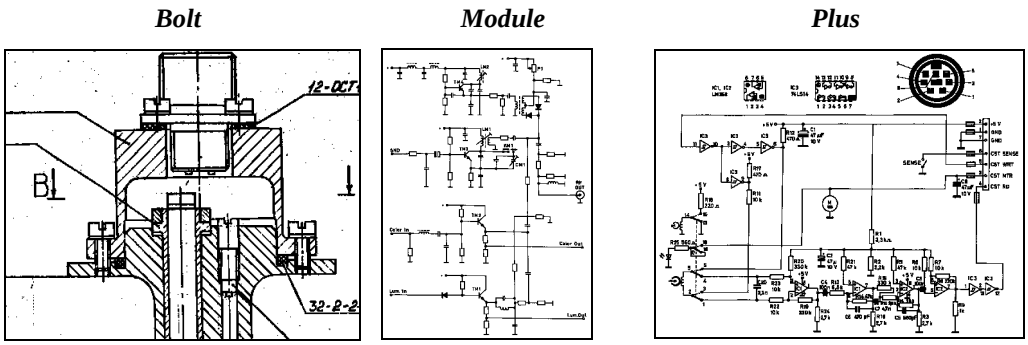


Figure 6: Overall illustration of the feature-dependent filtering process.

unpleasant appearance in the feature image, they do not necessarily prevent effective filtering. For example, the rightmost diagonal line in the feature image is too wide and the pixels are not filtered in the first two stages. The eroded version, however, gives a more accurate version of the line and the noise pixels can be detected and filtered in the 3rd stage. The result of the entire filtering process is illustrated in Figure 6.

4. CONTEXT-BASED COMPRESSION

We apply the baseline JBIG as the compression component, but basically there is no restrictions to use any other compression method. For example, the *progressive mode* of JBIG [2] could also be utilizes. In the baseline JBIG the pixels are coded by arithmetic coding in scan raster order using context-based probability model. The combination of already coded neighboring pixels defines the context. In each context the probability distribution of the black and



Bolt	Module	Plus
1765 × 1437	1480 × 2053	2293 × 1787
317,590 bytes	379,805 bytes	512,199 bytes

Figure 7: The set of test images.

Table 1: Summary of the compression results (in bytes).

	Original raster image	Compressed vector file	ITU Group 3	ITU Group 4	Baseline JBIG	Our method
Bolt	317,590	28,842	26,409	17,203	12,966	10,210
Module	379,805	12,430	18,979	10,801	7,671	5,798
Plus	512,199	38,837	32,269	21,461	17,609	14,581
TOTAL:	**1,209,043**	**80,109**	**77,657**	**49,465**	**38,246**	**30,589**

white pixels are adaptively determined. The current pixel is then coded by QM-coder [3, 10], the binary arithmetic coder adopted in JBIG. Statistics start from scratch and they are updated after the coding of each pixel. The model thus dynamically adapts to the image statistics during the coding process. The probability estimation and update are performed using the state automaton of QM-coder.

5. TEST RESULTS

We compare the proposed method with the existing compression standards. The following methods are considered:

- Uncompressed raster file
- Compressed vector file
- ITU Group 3
- ITU Group 4
- Baseline JBIG
- Our method

The compressed vector file represents the result of the vectorizing when the chain-coded elements are compressed by ZIP (commonly used universal data compression method). Baseline JBIG is the basic implementation of the JBIG binary image compression standard. ITU Group 3 and Group 4 are the older facsimile standards based on run-length encoding and two-dimensional READ-code [11, 12].

The methods are evaluated by compressing the test images shown in Figure 7. The compression results are summarized in Table 1. The vector representation is not space efficient because it cannot represent small objects efficiently. At the same time, our method and JBIG can compress the raster images using less than half of the size required by the vector file. The corresponding compression ratios (in total) are 15:1 for the vector file, 32:1 for the JBIG, and 40:1 for our method.

To sum up, the proposed compression method outperforms the Group 4 method by 40 % and the baseline JBIG by 20 %. At the same time, the quality of the decompressed images is visually the same as the original since only isolated mismatch pixels were reversed. The quality is sometimes even better because the reversed pixels are mainly random noise or scanning noise near the line segments.

6. CONCLUSIONS

A three-stage compression method for compressing bi-level line-drawing images was proposed. In the first stage, the raster image is vectorized and a feature image is reconstructed from the extracted vector elements. In the second stage, the original image is processed by a feature-based filtering for removing noise from the original image. It improves the image quality and therefore results in better

compression performance. The actual compression is performed by the baseline JBIG. For a set of test images, the method achieves a compression ratio of 40:1, in comparison to 32:1 of the baseline JBIG.

A drawback of the method is that the compression phase is now more complex and the method needs several passes over the image. Fortunately the vectorizing can be performed rather quickly. Moreover, the vector features are not stored in the compressed file and the process is therefore invisible in the decompression phase. The method thus can be considered as a preprocessing step to JBIG and the standard decompression routines can be applied.

The method could be developed further by improving the quality of the vectorizing. The width of the lines was sometimes quite badly predicted resulting in a visually disturbing feature image. This does not effect on the output image quality but it may weaken the compression performance. This was partially compensated by the three-stage filtering process, which makes the noise removal less sensitive to the inaccuracies in the vectorizing part.

7. ACKNOWLEDGEMENTS

The work of Pasi Fränti was supported by a grant from the Academy of Finland.

8. REFERENCES

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

[2] JBIG, Progressive Bi-level Image Compression, ISO/IEC International Standard 11544, ITU Recommendation T.82, 1993.

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

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

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

[6] A.N. Kolesnikov, V.I. Belekhov and I.O. Chalenko, "Vectorization of raster images", *Pattern Recognition and Image Analysis*, **6** (4), 786-794, 1996.

[7] C. Arcelli and G.S. di Baja, "A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **11** (4), 411-414, 1989.

[8] A.N. Kolesnikov and E.V. Trichina, "The parallel algorithm for the binary images thinning", *Optoelectronics, Instrumentation and Data Processing*, No. 6, 7-13, 1995.

[9] J. Serra, Image Analysis and Mathematical morphology. Academic Press, London, 1982.

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

[11] CCITT, Standardization of Group 3 Facsimile Apparatus for Document Transmission, ITU Recommendation T.4, 1980.

[12] CCITT, Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus, ITU Recommendation T.6, 1984.

Authors:

Pasi Fränti, research fellow
Eugene I. Ageenko, PhD student
Dept. of Computer Science, University of Joensuu
P.O. Box 111, FIN-80101 Joensuu, FINLAND
E-mail: franti@cs.joensuu.fi

Alexander Kolesnikov, senior scientist
Igor O. Chalenko, graduate student
Institute of Automation and Electrometry
Pr-t Ak. Koptyuga, 1, Novosibirsk-90, 630090, RUSSIA
E-mail: Kolesnikov@iae.nsk.su