# Edge-Based Image Compression with Homogeneous Diffusion

Markus Mainberger and Joachim Weickert

Mathematical Image Analysis Group,
Faculty of Mathematics and Computer Science, Campus E1.1
Saarland University, 66041 Saarbrücken, Germany
{mainberger,weickert}@mia.uni-saarland.de

**Abstract.** It is well-known that edges contain semantically important image information. In this paper we present a lossy compression method for cartoon-like images that exploits information at image edges. These edges are extracted with the Marr–Hildreth operator followed by hysteresis thresholding. Their locations are stored in a lossless way using JBIG. Moreover, we encode the grey or colour values at both sides of each edge by applying quantisation, subsampling and PAQ coding. In the decoding step, information outside these encoded data is recovered by solving the Laplace equation, i.e. we inpaint with the steady state of a homogeneous diffusion process. Our experiments show that the suggested method outperforms the widely-used JPEG standard and can even beat the advanced JPEG2000 standard for cartoon-like images.

**Key words:** image compression, partial differential equations (PDEs), Laplace equation, contour coding

## 1 Introduction

Edges are not only semantically important for humans, they also play a central role in image processing and computer vision. Edge detection can be regarded as an intermediate step from a pixel-based to a semantic image representation. Since it is more compact to describe an image by a few contours than by many pixels, an edge representation is also of potential interest for image compression.

One of the classical edge detectors is based on the Marr-Hildreth operator [1], which extracts edges as zero-crossings of the Laplacian of a Gaussian-smoothed version of the image. Numerous theoretical and experimental papers have been written that investigate if reconstructions from these zero-crossings are possible [2–12]. Unfortunately, it turned out that specifying only the zero-crossing locations is insufficient for typical real-world images. Thus, it has been suggested that one should supplement additional information such as the image gradient, grey values adjacent to the edge, subsampled image data, or scale information. However, none of the before mentioned publications has led to an image compression method that yields results which are competitive to modern compression standards such as JPEG [13] or JPEG2000 [14]: Either the required data cannot

be encoded in a sufficiently compact way, or the results turn out to be of inferior quality.

The goal of the present paper is to address this problem. We provide a proof of concept that information on the edge location together with the adjacent grey or colour values is sufficient to reconstruct cartoon-like images in high quality when the unspecified locations are filled in by the steady state of a homogeneous diffusion process. Moreover, by using state-of-the-art techniques for encoding the edge locations and the adjacent grey/colour values, our method may even outperform the quality of leading compression standards such as JPEG2000.

Our semantic image compression approach can be regarded as a specific implementation of a recent result on optimal point selection for compression with homogeneous diffusion: In [15] it is proven that one should preferently store pixels with large modulus of the Laplacian. For a piecewise constant image this comes down to the pixels left and right of an edge contour. It should be noted that contours can be encoded more efficiently than the same number of individual pixels, and they avoid visually unpleasant singularities of the fundamental solution of the Laplace equation. Moreover, by using homogeneous diffusion, our method is simpler and potentially faster than recent compression methods based on nonlinear anisotropic diffusion processes [16].

The structure of our paper is as follows: In Section 2, we present the encoding algorithm of our lossy compression scheme. Section 3 explains how to decode and reconstruct the image information by means of interpolation with homogeneous diffusion. Experiments and a comparison to JPEG and JPEG2000 are presented in Sect. 4. Finally, the paper is concluded with a summary in Sect. 5.

## 2 Encoding

**Step 1: Detecting Edges.** Our encoding of an image starts with an edge detection. We use the Marr-Hildreth edge detector [1] combined with a hysteresis thresholding as suggested by Canny [17].

To this end, we extract the zero-crossings of the Laplacian of a Gaussian-smoothed image. To remove zero-crossings that have no obvious perceptual significance, we apply hysteresis thresholding: Pixels obtained by the Marr-Hildreth edge detector with a gradient magnitude that is larger than a lower threshold are considered as edge candidates. All edge candidates with a gradient magnitude that is larger than a higher threshold become seed points for relevant edges and are automatically edge pixels of the final edge image. In order to keep edge pixels still connected as much as possible, we recursively add all edge candidates that are adjacent to final edge pixels.

For cartoon-like images this algorithm gives well-localised contours that are often closed. However, we are not necessarily bound to the suggested edge detector: For images that contain blurry edges, closed contours may be less important, and the original Canny edge detector [17] can give better results. Choosing alternative edge detectors can also be advantageous regarding noisy images. In our experiments we focus on the edge detector described above.

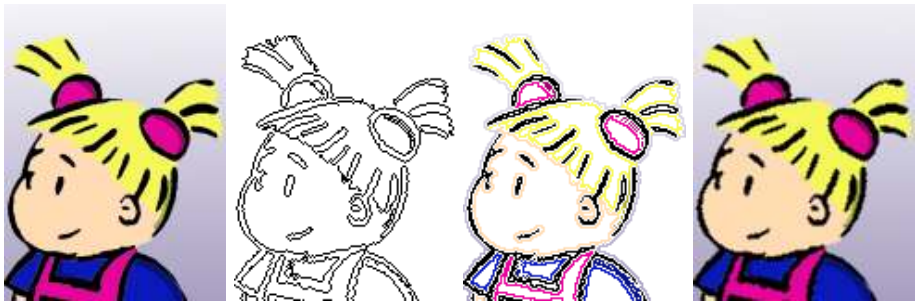


**Fig. 1.** Zoom ($120 \times 160$) into the image *comic* **(a)** Original image. **(b)** Edge image. **(c)** Colours next to the edges and image boundary. **(d)** Interpolated version using homogeneous diffusion.

**Step 2: Encoding the Contour Location.** The result of the edge detection step can be regarded as a bi-level edge image as is illustrated in Fig. 1(b). This bi-level image encodes indirectly the location that is used for interpolation later on. We store this image by using the *JBIG* (*Joint Bi-level Image Experts Group*) standard [18]. It has been developed as a specialised routine for lossless as well as for lossy compression of bi-level images, particularly with regard to textual images for fax transmission.

In this paper we are interested in lossless compression with JBIG. We use the JBIG-KIT [19], which is a free C implementation of the JBIG encoder and decoder. The JBIG standard relies on a context-based arithmetic coding. Two prediction steps are applied beforehand to except pixels from arithmetic coding that can be encoded more efficiently by other methods. In our case only the so-called typical prediction applies since we do not use the progressive mode of JBIG.

**Step 3: Encoding the Contour Pixel Values.** Next, we consider the pixel values we want to store. Since edges usually split areas of different brightness or colour, we do not use the pixel values that lie directly on the edge, but store the values from both sides of the edge instead (see Fig. 1(c) and [7]). Additionally, all pixel values from the border of the image domain are stored such that these values are also available as Dirichlet boundary for the diffusion-based filling-in later on.

Compression methods are often based on the fact that most differences between consecutive values within a signal are small. In our case we know that the pixel values along a contour usually change only gradually. Thus, it is reasonable to store the pixel values by the order of their occurrence along the edge.

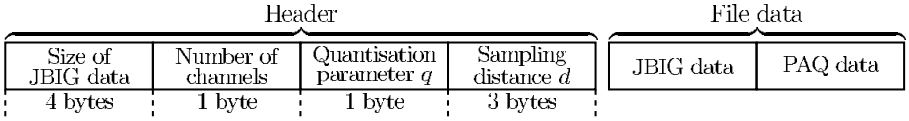
The extracted pixel values can be uniformly quantised to $2^q$ different values, where $q \in \{1, \dots, 8\}$. The parameter $q$ can be chosen by the user depending on compression requirements. For RGB colour images, all channels are quantised equally.

A second compression parameter is the sampling distance $d$: For $d > 1$ a subsampling on the pixel values is performed, i.e. only every $d$-th value along an edge is stored. As already stated, the pixel values change only marginally along an edge. In contrast, the pixel values of distinct edges might differ by a significant amount. Thus, it is indispensable to subsample each edge separately to obtain feasible reconstruction results.

The quantised and subsampled pixel value signal is then compressed by a *PAQ* compression method [20]. PAQ describes a family of lossless data compression archivers, which are licensed under GPL. At the expense of run time and memory requirements, the compression rates of PAQ outperform those of other compression programmes such as the well-established data compression algorithm *bzip2*. PAQ uses a context mixing algorithm, which is related to Prediction by Partial Matching (PPM) [21]. The compression algorithm is divided into a predictor and an arithmetic coder. For the prediction, PAQ is provided with a large number of models conditioned on different contexts often tuned to special file formats. In this paper we use *PowerPAQ* which is a command line and GUI front end using PAQ8o6 for 32 and 64 bit Linux [22].

**Step 4: Storing the Encoded Data.** Now that we have encoded the contour location and pixel values, we want to consider the final image format. Obviously, we need to store the quantisation parameter $q$ and the sampling distance $d$ as header data. Furthermore, we store the size of the JBIG data part in order to be able to split the JBIG data from the PAQ data when decoding. The number of channels (1 or 3) has to be stored explicitly for decoding the PAQ part, whereas the image size is automatically encoded in the JBIG data part.

Our entire coded image format is then given by the following representation.




## 3 Decoding

**Step 1: Decoding the Contour Location and Pixel Values.** As a first step of the decoding phase, we split our encoded file into the JBIG data and PAQ data part. Both parts are then decoded by using the JBIG and the PAQ method, respectively.

We reconstruct the quantised colours obtained by the PAQ part. The pixel values between the sampled points are computed by using linear interpolation along each edge. Higher order interpolations do not give any advantage since the pixel values hardly vary along an edge.

The decoded pixel values have to be placed at their corresponding positions in the final image: The JBIG data provides a bi-level edge image of the original image size. Given the image size and the edge pixel positions, the pixel values are arranged around the edges in the same order in which they have been encoded.

**Step 2: Reconstructing Missing Data.** So far, we have decoded the edge locations and the pixel values surrounding the edges. The idea is now to apply interpolation to reconstruct grey/colour values of pixels that lie between edges. We keep the pixel values at the positions obtained in Step 1 and use homogeneous diffusion for interpolation. This is the simplest and computationally most favourable inpainting approach based on partial differential equations (PDEs) [23]. Missing data is reconstructed by computing the steady state of the diffusion equation [24]

$$\partial_t u = \Delta u \;, \tag{1}$$

where the given pixel values are considered to form Dirichlet boundaries. Thus, the reconstructed data satisfies the Laplace equation $\Delta u = 0$ (see also [7]). Such a PDE can be discretised in a straightforward way by finite differences [25].

Interestingly, diffusion-based inpainting from image edges resembles a classical finding in biological vision: Already in 1935 Werner made the hypothesis that a contour-based filling-in process is responsible for the human perception of surface brightness and colour [26].

## 4 Experiments

Let us now investigate the capabilities of the suggested compression method. Figure 2 shows different test images and their compressed versions using JPEG, JPEG2000, and the suggested PDE-based approach. The compression rates lie between 0.20 and 0.35 bits per pixel (bpp). Since the original colour images use 24 bits per pixel, this comes down to compression ratios of 120:1 to roughly 70:1.

The compression ratio for the PDE-based approach can be influenced by the sampling distance $d$ and the quantisation parameter $q$ but also by the underlying edge image. However, for cartoon-like images, there is most often only one reasonable edge set. All parameters were chosen to give visually pleasant results. Experiments have shown that it suffices to set the quantisation parameter $q$ to 4 or 5, i.e. 16 or 32 different pixel values per channel. The sampling distance should be chosen depending on how detailed the input image is. For simple images such as *comic* we could use $d = 30$ without getting obvious visible degradations in the reconstruction. However, common image manipulation and display programmes are not able to create JPEG2000 images with such high compression rates. Thus, for the sake of comparability all given examples use $d = 5$.

In Figure 2 we observe that JPEG as well as JPEG2000 coding suffers from severe ringing artifacts in regions of edges (see *comic*, and zoomed region thereof). This is a result of their quantisation in the frequency domain and the following back transformation. In the JPEG images block artifacts appear because the discrete cosine transform is computed within blocks of $8 \times 8$ pixels. Thus, JPEG cannot even properly describe the smooth gradient in the background of *svalbard* and *comic*. In contrast, our method stores edges explicitly and interpolates in regions between edges. As a result, edges are well-preserved and smooth gradients can still be represented. Visually, this gives the most appealing results.

For a quantitative comparison, we use the *peak-signal-to-noise ratio* (PSNR), a common error measure for the qualitative analysis of images. It is defined as follows: Let $m$ be the maximal possible pixel value, which is 255 in our case. Furthermore, let $N$ be the number of image pixels and $(f_i)_{i=1..N}$ and $(g_i)_{i=1..N}$ the pixel values of the original image and its reconstructed/decompressed version, respectively. The PSNR is then defined via the *mean squared error* (MSE):

$$\text{PSNR} := 10 \cdot \log_{10}\left(\frac{m^2}{\text{MSE}}\right)\text{[dB]} \quad \text{with} \quad \text{MSE} := \frac{1}{N}\sum_{i=1}^{N}(f_i - g_i)^2 \,. \tag{2}$$

Table 1 shows the results for the images presented in Fig. 2 and confirms our visual impression.

## 5 Summary

It is surprising that after almost three decades of intensive research on image reconstruction from the zero-crossings of the Laplacian, substantial progress can be made by a conceptually simple, but carefully engineered approach. By extracting the information on both sides of the zero-crossings, encoding it in an efficient way, and decoding with homogeneous diffusion inpainting, we have presented a codec that can even beat JPEG2000 for cartoon-like images. This was out of reach for all previous methods on edge-based image reconstruction.

Since none of the steps in our codec is highly complex, we are currently performing research on real-time algorithms for sequential and distributed architectures. Moreover, we are investigating extensions that are also optimised for edges with very smooth slope as well as for highly textured images.

**Acknowledgements.** We thank Anna Mainberger for providing the image *comic*.

## References

1. Marr, D., Hildreth, E.: Theory of edge detection. Proceedings of the Royal Society of London, Series B **207** (1980) 187–217
2. Logan, Jr., B.F.: Information in the zero crossings of bandpass signals. Bell System Technical Journal **56** (April 1977) 487–510
3. Yuille, A.L., Poggio, T.A.: Scaling theorems for zero crossings. IEEE Transactions on Pattern Analysis and Machine Intelligence **8**(1) (January 1986) 15–25
4. Curtis, S.R., Oppenheim, A.V., Lim, J.S.: Reconstruction of two-dimensional signals from threshold crossings. In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing. Volume 10., Tampa, FL (March 1985) 1057–1060
5. Zeevi, Y., Rotem, D.: Image reconstruction from zero-crossings. IEEE Transactions on Acoustics, Speech, and Signal Processing **34** (1986) 1269–1277


**Fig. 2.** Comparison of compression methods for different test images. *Rows from top to bottom: coppit* (256 × 256), *svalbard* (380 × 431), *comic* (512 × 512), *comic* (detail) (100 × 100). *Columns from left to right:* original image, JPEG, JPEG2000, and PDE-based compression using contours (sampling distance $d$ *from top to bottom:* 5, 5, 5; quantisation parameter $q$: 4, 5, 5).

**Table 1.** Comparison of the PSNR for different images (see Fig. 2) and different compression methods.

| image | *coppit* | *svalbard* | *comic* |
|---|---|---|---|
| compression rate | 0.34 bpp | 0.23 bpp | 0.21 bpp |
| JPEG | 26.14 | 26.91 | 25.54 |
| JPEG2000 | 27.56 | 30.06 | 27.44 |
| **our method** | **30.16** | **30.21** | **30.31** |

6. Chen, S.: Image reconstruction from zero-crossings. In: Proc. Eighth International Joint Conference on Artificial Intelligence. Volume 2., Milan, Italy (August 1987) 742–744
7. Carlsson, S.: Sketch based coding of grey level images. Signal Processing **15** (1988) 57–83
8. Hummel, R., Moniot, R.: Reconstructions from zero-crossings in scale space. IEEE Transactions on Acoustics, Speech, and Signal Processing **37** (1989) 2111–2130
9. Grattoni, P., Guiducci, A.: Contour coding for image description. Pattern Recognition Letters **11**(2) (February 1990) 95–105
10. Mallat, S., Zhong, S.: Characterisation of signals from multiscale edges. IEEE Transactions on Pattern Analysis and Machine Intelligence **14** (1992) 720–732
11. Dron, L.: The multiscale veto model: A two-stage analog network for edge detection and image reconstruction. International Journal of Computer Vision **11**(1) (August 1993) 45–61
12. Elder, J.H.: Are edges incomplete? International Journal of Computer Vision **34**(2/3) (1999) 97–122
13. Pennebaker, W.B., Mitchell, J.L.: JPEG: Still Image Data Compression Standard. Springer, New York (1992)
14. Taubman, D.S., Marcellin, M.W., eds.: JPEG 2000: Image Compression Fundamentals, Standards and Practice. Kluwer, Boston (2002)
15. Belhachmi, Z., Bucur, D., Burgeth, B., Weickert, J.: How to choose interpolation data in images. SIAM Journal on Applied Mathematics (2009) In press.
16. Galić, I., Weickert, J., Welk, M., Bruhn, A., Belyaev, A., Seidel, H.P.: Image compression with anisotropic diffusion. Journal of Mathematical Imaging and Vision **31**(2–3) (July 2008) 255–269
17. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **8** (1986) 679–698
18. Joint Bi-level Image Experts Group: Information technology – progressive lossy/lossless coding of bi-level images. ISO/IEC JTC1 11544, ITU-T Rec. T.82 (1993) Final Committee Draft 11544.
19. Kuhn, M.: Effiziente Kompression von bi-level Bilddaten durch kontextsensitive arithmetische Codierung. Studienarbeit, Institut für Mathemathische Maschinen und Datenverarbeitung der Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany (July 1995)
20. Mahoney, M.: Adaptive weighing of context models for lossless data compression. Technical Report CS-2005-16, Florida Institute of Technology, Melbourne, Florida (December 2005)
21. Moffat, A.: Implementing the ppm data compression scheme. IEEE Transactions on Communications **38**(11) (November 1990) 1917–1921
22. Varnavsky, E.: PowerPAQ. http://powerpaq.sourceforge.net/. Last visited March 21, 2009.
23. Masnou, S., Morel, J.M.: Level lines based disocclusion. In: Proc. 1998 IEEE International Conference on Image Processing. Volume 3., Chicago, IL (October 1998) 259–263
24. Iijima, T.: Basic theory on normalization of pattern (in case of typical one-dimensional pattern). Bulletin of the Electrotechnical Laboratory **26** (1962) 368–388 In Japanese.
25. Morton, K.W., Mayers, L.M.: Numerical Solution of Partial Differential Equations. Cambridge University Press, Cambridge, UK (1994)
26. Werner, H.: Studies on contour. The American Journal of Psychology **47**(1) (January 1935) 40–64