Towards PDE-Based Image Compression

Irena Galić[1,2], Joachim Weickert[1], Martin Welk[1],
Andrés Bruhn[1], Alexander Belyaev[2], and Hans-Peter Seidel[2]

[1] Mathematical Image Analysis Group, Faculty of Math. and Computer Science,
Saarland University, Building 27, 66041 Saarbrücken, Germany
{galic, weickert, welk, bruhn}@mia.uni-saarland.de
[2] Max-Planck Institute for Informatics, Stuhlsatzenhausweg 85,
66123 Saarbrücken, Germany
{belyaev, hpseidel}@mpi-sb.mpg.de

Abstract. While methods based on partial differential equations (PDEs) and variational techniques are powerful tools for denoising and inpainting digital images, their use for image compression was mainly focussing on pre- or post-processing so far. In our paper we investigate their potential within the decoding step. We start with the observation that edge-enhancing diffusion (EED), an anisotropic nonlinear diffusion filter with a diffusion tensor, is well-suited for scattered data interpolation: Even when the interpolation data are very sparse, good results are obtained that respect discontinuities and satisfy a maximum–minimum principle. This property is exploited in our studies on PDE-based image compression. We use an adaptive triangulation method based on B-tree coding for removing less significant pixels from the image. The remaining points serve as scattered interpolation data for the EED process. They can be coded in a compact and elegant way that reflects the B-tree structure. Our experiments illustrate that for high compression rates and non-textured images, this PDE-based approach gives visually better results than the widely-used JPEG coding.

1 Introduction

In recent years several partial differential equations (PDEs) and variational techniques have shown their usefulness in so-called inpainting methods [19,2,4,13,27]. Here one aims at filling in missing informations in certain corrupted image areas by means of second or higher-order PDES. The basic idea is to regard the given image data as Dirichlet boundary conditions, and interpolate the data in the inpainting regions by solving appropriate boundary value problems. Related variational and PDE methods have also been investigated for more classical interpolation problems such as zooming into an image by increasing its resolution [3,18]. For such interpolation problems with data given on a regular grid, these techniques are in competition with cubic or quintic splines, radial basis functions and sinc-based interpolation techniques; see e.g. [17,20]. If the data are not available on a regular grid, scattered data interpolation techniques have been proposed [11,22], among which radial basis functions such as thin plate splines [9] are popular and well-performing.

Interestingly, not many of the variational and PDE-based interpolation and inpainting techniques have been used for scattered data interpolation. It seems that the sparsity

N. Paragios et al. (Eds.): VLSM 2005, LNCS 3752, pp. 37–48, 2005.

of the scattered data constitutes a real challenge for these techniques: While second-order PDEs may satisfy a maximum–minimum principle, they often create singularities at isolated interpolation points in 2-D. Higher-order PDEs, on the other hand, may give smoother solutions, but the violation of an extremum principle can lead to undesirable over- and undershoots; see e.g [3].

The goal of the present paper is twofold: First we address the before mentioned problem by investigating a partial differential equation that has not been considered for interpolation problems before, namely *edge-enhancing anisotropic diffusion (EED)* [28]. It uses a diffusion tensor that allows smoothing along discontinuities while inhibiting smoothing across them. In our experiments we will see that this technique performs favourably in the context of scattered data interpolation. A second goal of our paper is to investigate if this property can be used for lossy image compression. While contemporary image compression methods are dominated by concepts that involve the discrete cosine transform (such as the widely-used JPEG standard [23]) or the discrete wavelet transform (in JPEG2000 [26]), our goal is to give a proof-of-concept that there are alternatives where PDEs may be beneficial. The basic idea is to reduce the image data to a well-adapted set of significant sparse points that can be coded in an efficient way. Decoding is accomplished by using these scattered data and interpolating them by means of edge-enhancing anisotropic diffusion. As a tool for creating a useful sparse point representation, we consider the *B-tree triangular coding (BTTC)* by Distasi et al. [8], since it is relatively simple and allows an efficient coding of the sparsified image data.

Our paper is organised as follows. In Section 2 we describe PDE-based interpolation techniques and show that scattered data interpolation with EED performs particularly well. In Section 3 we review the BTTC model for image coding and describe how it can be coupled with PDE-based interpolation. Experiments on EED-based image coding are presented in Section 4, and the paper is concluded with a summary in Section 5.

Related Work. In the context of image compression, PDEs and related variational techniques have mainly been used as a preprocessing step before coding [10,16] or as a post-processing tool for removing coding artifacts [1,12,21,29,30]. Our works differs from these papers by the fact that we use PDEs within the decoding step rather than as pre- or postprocessing tools. Chan and Zhou [5] used total variation regularisation in order to modify the coefficients in a wavelet decomposition such that oscillatory edge artifacts are reduced, while Solé et al. [25] investigate three PDEs for interpolating structures in digital elevation maps and report the most favourable results with the Laplacian operator. An interesting coding scheme that exploits scattered data interpolation has been proposed by Demaret et al. [7]. They construct an adaptive Delaunay triangulation that is used for decoding the image by linear interpolation. Their experiments show that it can be an alternative to JPEG 2000 coding for texture-free images.

2 PDE-Based Interpolation

We start by considering a PDE approach to image interpolation. First we discuss a general model, then we investigate four possibilities for smoothing operators, and finally we present an experiment that illustrates their properties.

2.1 General Model

Let $\Omega \subset \mathbb{R}^n$ denote an n-dimensional image domain. We want to recover some unknown scalar-valued function $v : \Omega \to \mathbb{R}$, from which we know its values on some subset $\Omega_1 \subset \Omega$. Our goal is to find an interpolating function $u : \Omega \to \mathbb{R}$ that is smooth and close to v in $\Omega \setminus \Omega_1$ and identical to v in Ω_1.

We may embed this problem in an evolution setting with some evolution parameter (the "time") $t \geq 0$. It solution $u(x,t)$ gives the desired interpolating function as its steady state ($t \to \infty$). We initialise the evolution with some function $f : \Omega \to \mathbb{R}$ that is identical to v on Ω_1 and that is set to some arbitrary value (e.g. to 0) on $\Omega \setminus \Omega_1$:

$$f(x) := \begin{cases} v(x) & \text{if } x \in \Omega_1 \\ 0 & \text{else.} \end{cases} \tag{1}$$

We consider the evolution

$$\partial_t u \;=\; (1-c(x))\, Lu \;-\; c(x)\,(u-f) \tag{2}$$

with f as initial value,

$$u(x,0) \;=\; f(x), \tag{3}$$

and reflecting (homogeneous Neumann) boundary conditions on the image boundary $\partial\Omega$. The function $c : \Omega \to \mathbb{R}$ is the characteristic function on Ω_1, i.e.

$$c(x) := \begin{cases} 1 & \text{if } x \in \Omega_1 \\ 0 & \text{else,} \end{cases} \tag{4}$$

and L is some elliptic differential operator. The idea is to solve the steady state equation

$$(1-c(x))\, Lu \;-\; c(x)\,(u-f) = 0 \tag{5}$$

with reflecting boundary conditions. In Ω_1 we have $c(x) = 1$ such that the interpolation condition $u(x) = f(x) = v(x)$ is fulfilled. In $\Omega \setminus \Omega_1$ it follows from $c(x) = 0$ that the solution has to satisfy $Lu = 0$. This elliptic PDE can be regarded as the steady state of the evolution equation

$$\partial_t u = Lu \tag{6}$$

with Dirichlet boundary conditions given by the interpolation data on Ω_1.

2.2 Specific Smoothing Operators

Regarding the elliptic differential operator L, many possibilities exist. The simplest one uses the Laplacian $Lu := \Delta u$ leading to *homogeneous diffusion* [15]:

$$\partial_t u = \Delta u. \tag{7}$$

A prototype for a higher order differential operator is the biharmonic operator $Lu := -\Delta^2 u$ giving the *biharmonic smoothing* evolution

$$\partial_t u = -\Delta^2 u. \tag{8}$$

Using it for interpolation comes down to thin plate spline interpolation [9], a rotationally invariant multidimensional generalisations of cubic spline interpolation. Note that only

the second-order differential operators allow a maximum–minimum principle, where the values of u stay within the range of the values of f in Ω_1.

Nonlinear isotropic diffusion processes are governed by $Lu := \operatorname{div}(g(|\nabla u|^2)\,\nabla u)$. This gives [24]

$$\partial_t u = \operatorname{div}(g(|\nabla u|^2)\,\nabla u) \tag{9}$$

where the diffusivity function g is decreasing in its argument, since the goal is to reduce smoothing at edges where $|\nabla u|$ is large. One may e.g. choose the *Charbonnier diffusivity* [6]

$$g(s^2) = \frac{1}{\sqrt{1 + s^2/\lambda^2}} \tag{10}$$

with some contrast parameter $\lambda > 0$. Since (9) uses a scalar-valued diffusivity we name this process *isotropic* (in contrast to the nomenclature in [24]).

Real anisotropic behaviour is possible when a diffusion tensor is used. As a prototype for nonlinear anisotropic diffusion filtering we consider *edge-enhancing diffusion (EED)* [28]. The idea is to reduce smoothing across edges while still permitting diffusion along them. The EED diffusion tensor has one eigenvector parallel to ∇u_σ, where u_σ is obtained from convolving u with a Gaussian with standard deviation σ. The corresponding eigenvalue is given by $g(|\nabla u_\sigma|^2)$ with a diffusivity function such as (10). The other eigenvectors are orthogonal to ∇u_σ and have corresponding eigenvalues 1. If we use the convention to extend a scalar-valued function $g(x)$ to a matrix-valued function $g(A)$ by applying g to the eigenvalues on A and leaving the eigenvectors unchanged, then EED can be formally linked to $Lu := \operatorname{div}(g(\nabla u_\sigma \nabla u_\sigma^\top)\,\nabla u)$. Hence, its evolution is governed by

$$\partial_t u = \operatorname{div}(g(\nabla u_\sigma \nabla u_\sigma^\top)\,\nabla u). \tag{11}$$

2.3 Experiments on Interpolation

In order to evaluate the potential of the preceding PDEs for scattered data interpolation, we have discretised them with central finite differences in space. For the diffusion equations, we performed a semi-implicit time discretisation with SOR as solver for the linear systems of equations, while for biharmonic smoothing an explicit scheme was used. Runtimes for a non-optimised C implementation on a 1.5 GHz Centrino laptop range between a few seconds and several minutes for a 256×256 image.

In Figure 1 we present an experiment that illustrates the use of the different smoothing operators for scattered data interpolation. It depicts a zoom into the famous *lena* image, where 2 percent of all pixels have been chosen randomly as scattered interpolation points. We observe that homogeneous diffusion is not very suitable for scattered data interpolation, since it creates singularities at the interpolation points. This can be avoided with interpolation using biharmonic smoothing. It gives fairly good results, but suffers from over- and undershoots near edges due to the violation of an extremum principle (see e.g. the shoulder). Interestingly, going from homogeneous diffusion to nonlinear isotropic diffusion does not give an improvement: Although nonlinear isotropic diffusion may allow discontinuities, its interpolant is too flat and tends to keep many interpolation points as isolated singularities. The fact that EED, on the other hand, gives the best results shows the importance of the anisotropic behaviour. Its ability to smooth

along edges seems to be very beneficial for avoiding singularities at interpolation points. Moreover, this second-order PDE respects a maximum–minimum priciple, such that the solution is within the greyscale bounds of the interpolation points.

Fig. 1. (a) Top left: Zoom into the test image *lena*, 256×256 pixels. **(b) Top middle:** Grey values of the scattered interpolation points (2 percent of all pixels, chosen randomly). **(c) Top right:** Interpolation by linear diffusion. **(d) Bottom left:** Biharmonic smoothing. **(e) Bottom middle:** Nonlinear isotropic diffusion ($\lambda = 0.1$). **(f) Bottom right:** EED ($\lambda = 0.1$, $\sigma = 1$).

Table 1. Average absolute errors (AAE) for the PDEs used for scattered data interpolation in Figure 1

PDE method	AAE
homogeneous diffusion (7)	16.977
biharmonic smoothing (8)	15.787
Charbonnier diffusion (9)	21.864
edge-enhancing diffusion (11)	14.584

Our visual impression is confirmed by Table 1, in which we have computed the average absolute error (AAE) between the interpolated image and the original image. For two images (u_{ij}) and (v_{ij}) with N pixels, the AAE is defined as

$$\mathrm{AAE}(u, v) = \frac{1}{N} \sum_{i,j} |u_{ij} - v_{ij}|. \tag{12}$$

Nonlinear isotropic diffusion performes worst, followed by homogeneous diffusion and biharmonic smoothing, while EED gives the smallest interpolation error. This shows that for scattered data interpolation, EED is a very promising PDE that has not been investigated in this context before.

3 Image Coding by Binary Trees

Now that we have seen that EED is useful for scattered data interpolation, we want to exploit this technique for image compression. To this end we have to couple it with a method that provides a useful sparse image representation with scattered data.

3.1 Creating Scattered Interpolation Points

Our image compression and decompression scheme relies on an adaptive sparsification of the image data by means of the triangulation from *B-tree triangular coding (BTTC)* [8]. In this coding, an image is decomposed into a number of triangular regions such that within each region it can be recovered in sufficient quality by interpolation from the vertices. In our case, all regions are isosceles rectangular triangles. The decomposition into triangles then is stored in a binary tree structure.

In order to describe the compression procedure, let us assume we have an image $v = (v_{ij})$ of size $(2^m + 1) \times (2^m + 1)$. Smaller images should be filled up to such a size in a suitable way. Initially, the image is split by one of its diagonals into two triangles. The four image corners $(1, 1)$, $(1, 2^m + 1)$, $(2^m + 1, 1)$ and $(2^m + 1, 2^m + 1)$ are vertices of these two triangles.

To refine this initial configuration, an approximation (u_{ij}) of the image (v_{ij}) is calculated by using only the grey values from the vertices and interpolating linearly within each triangle. If the error $e_{ij} := |u_{ij} - f_{ij}|$ satisfies $e_{ij} \leq \varepsilon$ for all pixels (i, j), with a given tolerance parameter $\varepsilon > 0$, the representation by triangles is considered sufficiently fine. Otherwise, for each pixel (i, j) for which $e_{ij} > \varepsilon$ holds, the triangle which contains (i, j) is split into two similar triangles by the height on its hypotenuse. The centre of the hypotenuse thereby becomes an additional vertex of the representation. By recalculating approximation errors within the new smaller triangles, it is determined whether to split these again etc. Since the approximation error is zero at vertices, triangles with legs of length 1 are not split further, which guarantees that the recursive splitting terminates. Moreover, vertices throughout the process have integer coordinates. Which pixels are vertices is indicated in a vertex mask of size equal to the image that is generated during the triangulation.

One point which needs additional consideration is the treatment of pixels located on the sides of triangles during the splitting process. If the error bound is violated in such a pixel, it is sufficient for our compression and decompression method to split one of the two adjacent triangles. This allows to reduce the number of triangles noticeably since in regions with fine details, a large number of small triangles occur, and many pixel positions then happen to be located on sides.

3.2 Coding

To efficiently store the triangulation, we notice that the hierarchical splitting of triangles gives rise to a binary tree structure. Each triangle occurring during the splitting process is represented by a node while leaves correspond to those triangles which are not divided further. When a triangle is split, its two subtriangles become the children of its representing node. To store the structure of the tree, one traverses the tree and stores one bit per node: a 1 for a node that has children, and a 0 for a leave. Preorder or level-order traversal are equally possible. Note that by the tree structure, the vertex mask is fully determined. Further space in storing the tree is saved by measuring globally the minimal and maximal depth of the tree. Only for nodes at intermediate levels, the corresponding bits are stored.

For coding the grey values in all vertices, we first zig zag traverse the sparse image created with the binary tree structure and store it in a sequence of grey values. This sequence is then compressed with Huffman coding [14], a lossless variable-length prefix code that assigns smaller codes to more frequent characters. It also uses a tree structure.

Our entire coded image format then reads as follows:

- image size (4 bytes)
- minimal and maximal depth of the binary tree (together 2 bytes)
- binary string encoding binary tree structure (1 bit for each node between minimal and maximal depth, filled up with zeros to the next byte boundary)
- first grey-value in a sequence of grey values (1 byte)
- minimal and maximal depth of the Huffman coded binary tree (2 bytes)
- binary string for Huffman-coded binary tree (1 bit for each node between minimal and maximal depth, filled up with zeros to the next byte boundary)
- Huffman dictionary (less than 256 bytes)
- sequence of Huffman-coded grey values

We further enhanced this coding by a (lossy) requantisation step that reduced the number of grey values in the initial image from 256 to 64.

3.3 Decoding

Decompression takes place in two steps. In the first step, the vertex mask is recovered from the binary tree representation, and the stored grey values are placed at the appropriate pixel positions to give the sparse image. To recover the vertex mask, the tree is generated in the same order as it was stored. Along with generating nodes, vertex positions are calculated and marked in the vertex mask.

The second step consists in the interpolation of the image, where the vertex mask becomes the interpolation mask. In the BTTC scheme of Distasi et al. [8], linear interpolation within each traingle is used. In the sequel we will denote this technique by BTTC-L.

Since we have already seen that EED performs favourably as a scattered data interpolant, it is natural to renounce the linear interpolation step in the BTTC-L method and apply EED to the interpolation mask that has been created by the BTTC method.

We abbreviate this novel method by BTTC-EED. Note that in contrast to BTTC-L, BTTC-EED does not rely on the triangulation, only on its vertices as interpolation points.

4 Experiments on Compression

Let us now investigate the effects of our EED-based interpolation in the context of image coding. Figures 2 and 3 show two test images and their compressed versions using BTTC-EED, respectively. We have chosen the threshold parameter ϵ such that compressions of 0.8, 0.4 and 0.2 bits per pixel (bpp) are achieved. Since the usual coding uses 1 byte per pixel, this comes down to compression ratios of 1:10, 1:20 and 1:40. In Fig. 3, we display both the coded pixels with their corresponding grey values, and the result after scattered data interpolation with EED. We observe that even at high compression rates, fairly realistic results are possible.

Fig. 2. Test images, 257×257 pixels. **(a) Left:** *trui*. **(b) Right:** *peppers*.

In order to illustrate the differences between BTTC-EED and BTTC-L as well as JPEG, we depict the corresponding results in Fig. 4. We perform this comparison at the high compression rate of 1:40 (or equivalently 0.2 bpp) where the visual differences are well visible. We observe that JPEG coding suffers from severe block artifacts that result from the fact that the discrete cosine transform is computed within blocks of 8×8 pixels. The BTTC-L method, on the other hand, creates artifacts where the underlying triangulation becomes visible. Since BTTC-EED only uses the interpolation points from the BTTC method, but not the corresponding triangulation, it is clear that this method cannot suffer from such a shortcoming. If not enough data are available, its interpolation tends to be on the smoother side. It gives the visually most convincing results among the three methods.

This visual impression is also confirmed by the quantitative measurements in Table 2, where the average absolute error is listed. We see that at the compression rate 1:40, JPEG performs worst, BTTC-L is in the midfield, and BTTC-EED gives the best results.

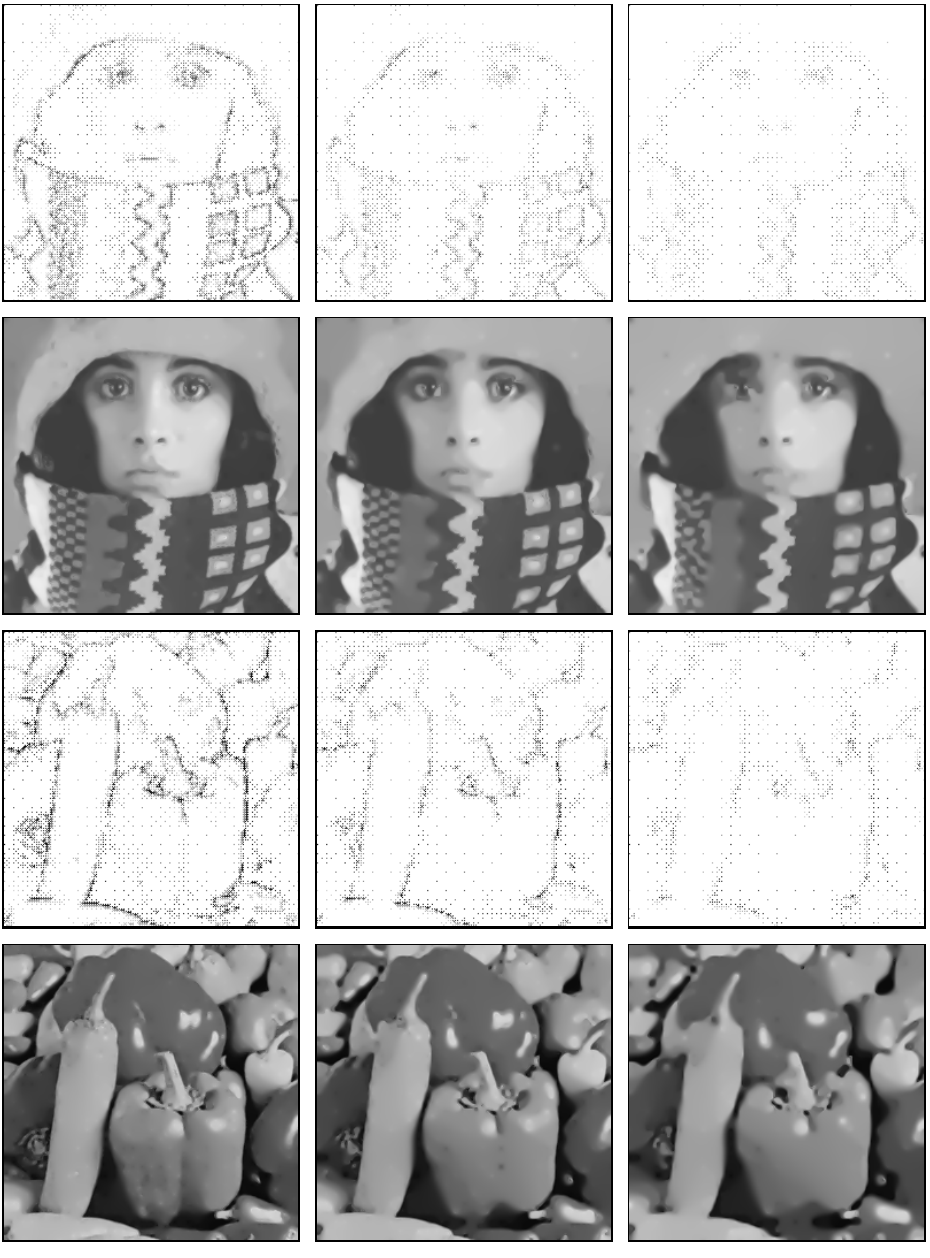

Fig. 3. First row, left to right: Adaptive sparsification of *trui*, using BTTC with compression to 0.8 bpp, 0.4 bpp, 0.2 bpp. **Second row, left to right:** Corresponding EED-based interpolation. **Third row, left to right:** Adaptive sparsification of *peppers*, using BTTC with compression to 0.8 bpp, 0.4 bpp, 0.2 bpp. **Fourth row, left to right:** Corresponding EED-based interpolation.

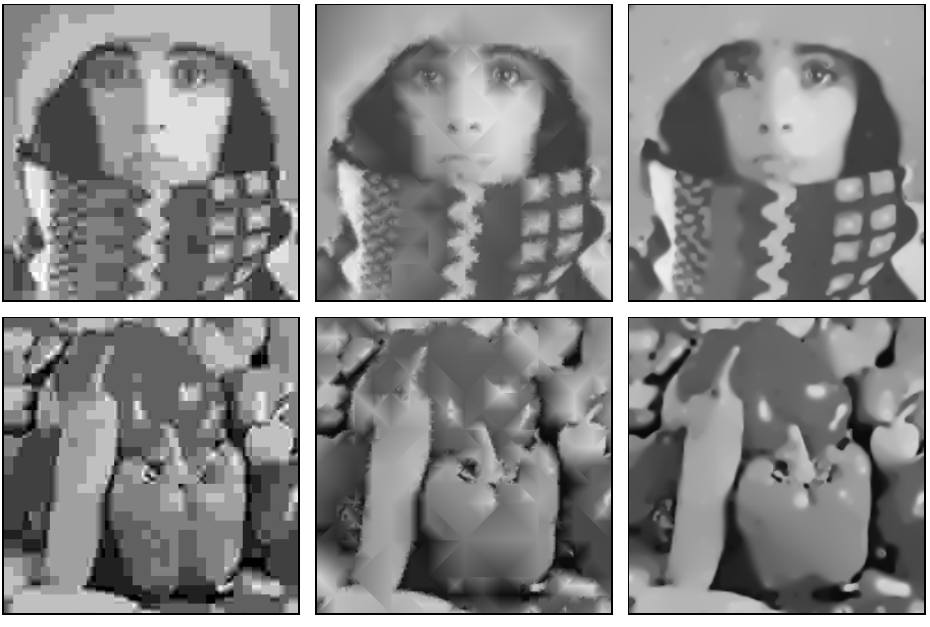

Fig. 4. Comparison at high compression rates (0.2 bpp) for the test images *trui* and *peppers*. **Left column:** JPEG. **Middle column:** BTTC-L. **Right column:** BTTC-EED.

Table 2. Comparison of the average absolute error for the different images and compression methods at 0.2 bpp

compression	trui	peppers
JPEG	11.25	12.99
BTTC-L	8.63	11.22
BTTC-EED	8.45	9.99

5 Conclusions

In this paper we presented a proof-of-concept that edge-enhancing anisotropic diffusion (EED), a PDE originally designed for denoising, can also be useful for scattered data approximation and for image compression. In the latter case we sparsified the image data by B-tree triangular coding, and used EED as a scattered data interpolant for decoding. Our experiments indicate that this strategy is particularly useful when high compression rates become necessary.

Since modern image compression methods have reached a high degree of sophistication, it is evident that our paper can only serve as a first step towards PDE-based image compression. In our ongoing work we are investigating different options for further performance improvements and a more detailed performance evaluation including also upcoming standards such as JPEG 2000.

Acknowledgements. Our research is partly funded by the *International Max-Planck Research School (IMPRS)*. This is gratefully acknowledged. Joachim Weickert also thanks Vicent Caselles for interesting discussions on EED-based interpolation during a stay at the University Pompeu Fabra, Barcelona.

References

1. F. Alter, S. Durand, and J. Froment. Adapted total variation for artifact free decompression of JPEG images. *Journal of Mathematical Imaging and Vision*, 23(2):199–211, September 2005.
2. M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. SIGGRAPH 2000*, pages 417–424, New Orleans, LI, July 2000.
3. V. Caselles, J.-M. Morel, and C. Sbert. An axiomatic approach to image interpolation. *IEEE Transactions on Image Processing*, 7(3):376–386, March 1998.
4. T. F. Chan and J. Shen. Non-texture inpainting by curvature-driven diffusions (CDD). *Journal of Visual Communication and Image Representation*, 12(4):436–449, 2001.
5. T. F. Chan and H. M. Zhou. Total variation improved wavelet thresholding in image compression. In *Proc. Seventh International Conference on Image Processing*, volume II, pages 391–394, Vancouver, Canada, September 2000.
6. P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2):298–311, 1997.
7. L. Demaret, N. Dyn, and A. Iske. Image compression by linear splines over adaptive triangulations. Technical report, Dept. of Mathematics, University of Leicester, UK, January 2005.
8. R. Distasi, M. Nappi, and S. Vitulano. Image compression by B-tree triangular coding. *IEEE Transactions on Communications*, 45(9):1095–1100, 1997.
9. J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *RAIRO Mathematical Models and Numerical Analysis*, 10:5–12, 1976.
10. G. E. Ford, R. R. Estes, and H. Chen. Scale-space analysis for image sampling and interpolation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 165–168, San Francisco, CA, March 1992.
11. R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38:181–200, 1982.
12. A. Gothandaraman, R. Whitaker, and J. Gregor. Total variation for the removal of blocking effects in DCT based encoding. In *Proc. 2001 IEEE International Conference on Image Processing*, volume 2, pages 455–458, Thessaloniki, Greece, October 2001.
13. H. Grossauer and O. Scherzer. Using the complex Ginzburg–Landau equation for digital impainting in 2D and 3D. In L. D. Griffin and M. Lillholm, editors, *Scale-Space Methods in Computer Vision*, volume 2695 of *Lecture Notes in Computer Science*, pages 225–236, Berlin, 2003. Springer.
14. D. A. Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40:1098–1101, 1952.
15. T. Iijima. Basic theory on normalization of pattern (in case of typical one-dimensional pattern). *Bulletin of the Electrotechnical Laboratory*, 26:368–388, 1962. In Japanese.
16. I. Kopilovic and T. Szirányi. Artifact reduction with diffusion preprocessing for image compression. *Optical Engineering*, 44(2):1–14, February 2005.
17. T. Lehmann, C. Gönner, and K. Spitzer. Survey: Interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075, November 1999.

18. F. Malgouyres and F. Guichard. Edge direction preserving image zooming: A mathematical and numerical analysis. *SIAM Journal on Numerical Analysis*, 39(1):1–37, 2001.
19. S. Masnou and J.-M. Morel. Level lines based disocclusion. In *Proc. 1998 IEEE International Conference on Image Processing*, volume 3, pages 259–263, Chicago, IL, October 1998.
20. E. Meijering. A chronology of interpolation: From ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, 90(3):319–342, March 2002.
21. P. Mrázek. *Nonlinear Diffusion for Image Filtering and Monotonicity Enhancement*. PhD thesis, Czech Technical University, Prague, Czech Republic, June 2001.
22. G. M. Nielson and J. Tvedt. Comparing methods of interpolation for scattered volumetric data. In D. F. Rogers and R. A. Earnshaw, editors, *State of the Art in Computer Graphics: Aspects of Visualization*, pages 67–86. Springer, New York, 1994.
23. W. B. Pennebaker and J. L. Mitchell. *JPEG: Still Image Data Compression Standard*. Springer, New York, 1992.
24. P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.
25. A. Solé, V. Caselles, G. Sapiro, and F. Arandiga. Morse description and geometric encoding of digital elevation maps. *IEEE Transactions on Image Processing*, 13(9):1245–1262, September 2004.
26. D. S. Taubman and M. W. Marcellin, editors. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer, Boston, 2002.
27. D. Tschumperlé and R. Deriche. Vector-valued image regularization with PDEs: A common framework for different applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):506–516, April 2005.
28. J. Weickert. Theoretical foundations of anisotropic diffusion in image processing. *Computing Supplement*, 11:221–236, 1996.
29. S. Yang and Y.-H. Hu. Coding artifact removal using biased anisotropic diffusion. In *Proc. 1997 IEEE International Conference on Image Processing*, volume 2, pages 346–349, Santa Barbara, CA, October 1997.
30. S. Yao, W. Lin, Z. Lu, E. P. Ong, and X. Yang. Adaptive nonlinear diffusion processes for ringing artifacts removal on JPEG 2000 images. In *Proc. 2004 IEEE International Conference on Multimedia and Expo*, pages 691–694, Taipei, Taiwan, June 2004.