A New Hybrid Method for Image Approximation using the Easy Path Wavelet Transform

GERLIND PLONKA[1], STEFANIE TENORTH[1] AND DANIELA ROŞCA[2]

[1] Faculty of Mathematics, University of Duisburg-Essen,
Campus Duisburg, 47048 Duisburg, Germany
gerlind.plonka@uni-due.de, stefanie.tenorth@uni-due.de

[2] Department of Mathematics, Technical University of Cluj-Napoca, 400020
Cluj-Napoca, Romania Daniela.Rosca@math.utcluj.ro

Abstract

The **Easy Path Wavelet Transform** (EPWT) has recently been proposed by one of the authors as a tool for sparse representations of bivariate functions from discrete data, in particular from image data. The EPWT is a locally adaptive wavelet transform. It works along pathways through the array of function values and exploits the local correlations of the given data in a simple appropriate manner. However, the EPWT suffers from its adaptivity costs that arise from the storage of path vectors. In this paper, we propose a new hybrid method for image compression that exploits the advantages of the usual tensor product wavelet transform for the representation of smooth images and uses the EPWT for an efficient representation of edges and texture. Numerical results show the efficiency of this procedure.

Key words. sparse data representation, tensor product wavelet transform, easy path wavelet transform, linear diffusion, smoothing filters, adaptive wavelet bases, N-term approximation

AMS Subject classifications. 41A25, 42C40, 68U10, 94A08

1 Introduction

Over the last years, wavelets have had a growing impact on signal and image processing. In the one-dimensional case, wavelets provide optimal representations of piecewise smooth functions. Unfortunately, in two dimensions, tensor product wavelet bases are suboptimal for representing geometric structures as edges and texture, since their support is not adapted to directional geometric properties. Only in case of globally smooth images, they provide optimally sparse representations.

Many different approaches have been developed to design approximation schemes that aim at a more efficient representation of two-dimensional data. Curvelets [3], shearlets [12], and contourlets [9] are examples of non-adaptive highly redundant function frames with a strong anisotropic directional selectivity. However, while theoretical results show their good performance for sparse representation of piecewise smooth images with discontinuities along smooth curves [3, 11], these frames cannot be applied for image compression. On the one hand, the known curvelet/shearlet algorithms do not get completely rid of the redundancy of the underlying function systems, see e.g. [4]. On the other hand, the almost optimal image representation can only be proven for images with edges along C^2-curves, and this result is not transferable to other image regularities.

Instead of choosing a priori a basis or a frame to approximate an image u, one can rather adapt the approximation scheme to the image geometry. Different approaches have been developed in this direction. Bandelets [15], wedgelets [10] geometric wavelets [7], grouplets [17], tetrolets [14], and frame constructions with adaptive angular selectivity [13] are examples for adaptive image approximation. In [8], an image compression scheme based on piecewise linear functions over an optimized triangulation has been constructed. Further, nonlinear edge adapted multiscale decompositions based on essentially non-oscillatory (ENO) schemes have been extensively investigated [1, 6].

The idea of a nonlinear locally adaptive easy path wavelet transform (EPWT) has been explored in [18] for sparse image representations. The main idea of this transform is as follows. In a first step, one needs to determine a "path vector" through all indices of a given (two-dimensional) index set, in such a way that there is a strong correlation between the image values that correspond to adjacent pixels in the path vector. Then, one level of a (one-dimensional) wavelet transform is applied to the image values along the path vector. In the following levels, one needs to find path vectors through index sets of a low-pass image and applies again the wavelet transform. The EPWT is very efficient for sparse image representations [20] and can also be applied to other grids [19]. But it suffers from high storage costs for the path vectors in each level. The so-called 'relaxed EPWT' is one way to reduce these adaptivity costs [18].

In this paper, we will exploit the advantages of the well-known tensor-product wavelet transform for representation of smooth images and the ability of the adaptive EPWT to represent edges and texture in images. For that purpose, we propose a new hybrid method for image compression that (roughly) consists of the following steps.

For a given digital image $u^0 = (u^0_{i,j})_{i=1,j=1}^{N_1,N_2}$, we first try to find a suitable separation $u^0 = u^{sm} + u^r$, where u^{sm} is globally smooth, and the difference image u^r contains the remaining part of the image (i.e. edges and texture). The separation will be done by a simple smoothing of u^0 based on local smoothing filters that are derived from linear diffusion. Then the usual tensor product wavelet transform is applied to the smooth image u^{sm}. Here we exploit the fact that smooth functions can be optimally represented by an M-term wavelet expansion u^{sm}_M.

In the next step, the EPWT is applied to the (shrunken) difference image $u^0 - u^{sm}_M$. Assuming that the original image u^0 is piecewise smooth, the difference image $u^0 - u^{sm}_M$ contains a high number of components with very small absolute value. Therefore, we consider a shrunken version $\tilde{u}^r = S(u^0 - u^{sm}_M)$ possessing a smaller number of nonzero values. In our numerical experiments, we shrink the difference, such that $\tilde{u}^r$ contains only $N_1N_2/4$ nonzero values. The EPWT is now applied only to the nonzero values of $\tilde{u}^r$, and the adaptivity costs can be strongly reduced compared with the EPWT for a full image.

Finally, we obtain a very good image approximation as a sum of the M-term wavelet expansion u^{sm}_M of the smooth image part and the N-term EPWT wavelet expansion u^r_N of the difference image.

The outline of the paper is as follows. In Section 2, we study all steps of the new hybrid algorithm separately. Compared with the above rough description, the algorithm, summarized in Section 2.4, will be slightly refined. Section 3 is devoted to the EPWT algorithm that we need to adapt for our purposes. Finally, in Section 4 we present some numerical experiments and show the strong efficiency of the proposed hybrid method.

2 Hybrid-model for image compression

As already mentioned in the introduction, the basic idea of the new hybrid model is to find a suitable partition of a given image $u^0 = (u_{i,j})_{i=1,j=1}^{N_1,N_2}$ into a smooth part u^{sm} and a remainder u^r and to apply different wavelet transforms to these two image parts. While the smooth image is known to be optimally representable by a suitable tensor product wavelet transform, we will use the new EPWT for representation of the remainder u^r that contains textures and edges.

2.1 Separation of images

We are interested in a segmentation of our image u into a "smooth" part u^{sm} and a remainder u^r that contains information about edges and textures. Note that this separation issue is different from image separation problems usually considered for image denoising, where one aims to separate an image into a cartoon part, i.e. a piecewise smooth function (smooth part together with edges of finite length), and a texture part, see e.g. [2, 5, 21] and others.

We suppose that the main portion of the considered image is regular outside a set of (piecewise) regular curves and that the image contains only a small amount of texture. For the smoothing procedure, we want to apply local smoothing filters that are derived from linear diffusion.

Let $\Omega \subset \mathbb{R}^2$ be a suitable region, and let $u^0 = u^0(x_1, x_2) \in L^2(\Omega)$ be a given continuous model of the image. We consider the linear diffusion equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} \qquad (x_1, x_2) \in \Omega,\ t \in [0, 1] \tag{2.1}$$

with Neumann boundary conditions $\frac{\partial u}{\partial n} = 0$ on $\partial\Omega$ and with $u(x_1, x_2, 0) = u^0(x_1, x_2)$ for $(x_1, x_2) \in \Omega$, where u^0 is the given image. It is well-known that the linear diffusion process leads to a smooth solution $u(x, y, t)$. For $\Omega = \mathbb{R}^2$ this solution is obtained by a convolution of u^0 with the two-dimensional Gaussian kernel

$$K_\sigma(x_1, x_2) := \frac{1}{2\pi\sigma^2} \exp(-\frac{|x_1| + |x_2|}{2\sigma^2}),$$

i.e.,

$$u(x_1, x_2, t) = \begin{cases} u^0(x_1, x_2) & t = 0, \\ \int\limits_{\mathbb{R}^2} K_{\sqrt{2\pi}}(x_1 - y_1, x_2 - y_2) u(y_1, y_2)\, dy_1 dy_2 & t > 0, \end{cases}$$

see e.g. [22]. We are interested in a simple discretization of this diffusion process. For $\Omega = [0, N_1] \times [0, N_2]$, let $u^0 := (u^0(i,j))_{i=1,j=1}^{N_1,N_2}$ be the given digital image. Using forward and backward differences for approximation of the derivatives, a discretization of (2.1) by the Euler scheme yields in the kth time step an approximation $u^k(i,j)$ of $u(i,j,k\tau)$ with

$$\frac{u^{k+1}(i,j) - u^k(i,j)}{\tau} = u^k(i,j+1) + u^k(i,j-1) + u^k(i-1,j) + u^k(i+1,j) - 4u^k(i,j),$$

where we have used the step size 1 in spatial domain. Hence,

$$u^{k+1}(i,j) = u^k(i,j) + \tau\left(u^k(i+1,j) + u^k(i-1,j) + u^k(i,j-1) + u^k(i,j+1) - 4u^k(i,j)\right)$$

for $i = 1, \ldots, N_1$, $j = 1, \ldots, N_2$. The Neumann boundary conditions are satisfied with the following definitions,

$$\begin{array}{llll} u^k(0,j) := u^k(1,j), & u^k(N_1+1,j) := u^k(N_1,j), & j = 1, \ldots, N_2, \\ u^k(i,0) := u^k(i,1), & u^k(i,N_2+1) := u^k(i,N_2), & i = 1, \ldots, N_1. \end{array} \tag{2.2}$$

The parameter τ controls the amount of smoothing in each step. Using this diffusion process we obtain a smoothed image $u^{sm} = (u^L(i,j))_{i=1,j=1}^{N_1,N_2}$, where L is a fixed number of iterations. In our experiments we have usually taken $L = 5$.

2.2 Tensor-product wavelet transform

Tensor-product wavelet bases are particularly efficient to approximate smooth images.

For given one-dimensional biorthogonal wavelet bases of $L^2(\mathbb{R})$ generated by the dual pairs ϕ, ψ and $\tilde{\phi}$, $\tilde{\psi}$ of scaling functions and wavelets, we consider the two-dimensional wavelets

$$\psi^1(x_1,x_2) = \phi(x_1)\psi(x_2), \quad \psi^2(x_1,x_2) = \psi(x_1)\phi(x_2), \quad \psi^3(x_1,x_2) = \psi(x_1)\psi(x_2),$$

and the dual wavelets

$$\tilde{\psi}^1(x_1,x_2) = \tilde{\phi}(x_1)\tilde{\psi}(x_2), \quad \tilde{\psi}^2(x_1,x_2) = \tilde{\psi}(x_1)\tilde{\phi}(x_2), \quad \tilde{\psi}^3(x_1,x_2) = \tilde{\psi}(x_1)\tilde{\psi}(x_2).$$

Using the notation

$$\psi^k_{j,n}(x_1,x_2) := \frac{1}{2^j}\psi^k\left(\frac{x_1 - 2^j n_1}{2^j}, \frac{x_2 - 2^j n_2}{2^j}\right), \qquad j \in \mathbb{Z},\ n = (n_1,n_2) \in \mathbb{Z}^2,\ k = 1,2,3,$$

and an analogous notation for $\tilde{\psi}^k_{j,n}$, one can verify that $\{\psi^1_{j,n}, \psi^2_{j,n}, \psi^3_{j,n}\}_{(j,n)\in\mathbb{Z}^3}$ and $\{\tilde{\psi}^1_{j,n}, \tilde{\psi}^2_{j,n}, \tilde{\psi}^3_{j,n}\}_{(j,n)\in\mathbb{Z}^3}$ are biorthogonal Riesz bases of $L^2(\mathbb{R}^2)$ (see e.g. [16]).

The fast wavelet transform is based on filter bank algorithms. Let a function f be Hölder-continuous of order α in Ω and let the M-term separable wavelet approximation f_M be obtained by keeping only the M wavelet coefficients with the largest absolute value in a wavelet basis representation of f. Then for a sufficiently smooth wavelet basis we have

$$||f - f_M||_2^2 < CM^{-\alpha},$$

where $||.||_2$ denotes the L^2-norm. The decay exponent is optimal, i.e., tensor product wavelet bases are optimal for sparse representation of smooth images. Therefore we apply the tensor product wavelet transform to the smoothed digital image u^{sm} obtained after linear diffusion (or to a slightly different image $\tilde{u}^{sm}$, see Section 2.4). Using only a fixed number of M wavelet coefficients in the wavelet representation, we obtain an approximation u_M^{sm} of u^{sm} after wavelet reconstruction. In our numerical experiments we use the well-known 9/7 biorthogonal filter bank and $D4$ orthonormal Daubechies wavelets. The image u_M^{sm} is then obtained by using a decomposition algorithm, a shrinkage procedure and a wavelet reconstruction.

2.3 The EPWT for sparse edge representation

Let u be the original digital image and $\tilde{u}_M^{sm}$ the M-term wavelet approximation of the smoothed image $\tilde{u}^{sm}$ obtained by a linear diffusion process (and a slight modification based on shrinkage, see Section 2.4). Now we consider the difference image $u^r = u - \tilde{u}^{sm}$ that mostly contains edges and texture. We want to apply a new locally adaptive wavelet transform to this difference image, the easy path wavelet transform (EPWT). The EPWT is a wavelet transform that works along path vectors through index subsets of the pixel set defining the image u^r. While the EPWT has been shown to be very efficient for sparse image representation [18, 20], we have to keep in mind its adaptivity costs for the storage of path vectors. In order to exploit the ability of the EPWT to sparsely represent edges and texture and, at the same time, to keep adaptivity costs small, we suggest to apply the EPWT not to the complete image u^r, but only to the part with essential image information.

Supposing that the original image u mainly contains piecewise regular segments, which will be hardly changed by the diffusion process, the difference image $u^r = u - \tilde{u}^{sm}$ possesses many very small image values. Therefore, we apply first a shrinkage procedure to u^r and obtain $\tilde{u}^r = (\tilde{u}^r(i,j))_{i=1,j=1}^{N_1,N_2}$, i.e.

$$S_\theta u^r(i,j) = \tilde{u}^r(i,j) := \begin{cases} u^r(i,j) & \text{if } |u^r(i,j)| \geq \theta, \\ 0 & \text{if } |u^r(i,j)| < \theta. \end{cases}$$

The shrinkage parameter θ should be chosen dependently on the image at hand in such a way that $\tilde{u}^r$ contains exactly 2^J nonzero image values, where $2^J < N_1 N_2$. In our numerical experiments, we have taken θ such that $\tilde{u}^r$ has only $\frac{1}{4} N_1 N_2$ nonzero values; these values are situated along the edges/texture of u. Now we apply the EPWT only along the nonzero values of $\tilde{u}^r$ while the vanishing values remain untouched. This procedure can be transferred to the partial image $\tilde{u}^r$, where we only consider the image values corresponding to the index set

$$I_J := \{(i,j) | \, 1 \leq i \leq N_1, \, 1 \leq j \leq N_2, \, |u^r(i,j)| \geq \theta\}$$

of size 2^J.

In Section 3, we will give a short description of the (adapted) EPWT algorithm and a suitable way to store the path vectors in order to minimize our adaptivity costs. The original algorithm of the EPWT is described in detail in [18].

In order to obtain a sparse representation of $\tilde{u}^r$, we apply a shrinkage procedure to the EPWT-wavelet coefficients. In our experiments we use the hard threshold function

$$S_\sigma(x) = \begin{cases} x & |x| \geq \sigma, \\ 0 & |x| < \sigma. \end{cases}$$

By taking only a fixed number of N coefficients, after reconstruction we obtain an EPWT approximation $\tilde{u}^r_N$ of the difference image $\tilde{u}^r$.

2.4 The algorithm

Let us summarize the procedure of the new hybrid algorithm for image compression.

Given: digital image $u^0 = \big(u^0(i,j)\big)_{i=1,j=1}^{N_1,N_2}$.

1. Apply an iterative local smoothing filter for image separation: Fix $\tau > 0$ and $L \in \mathbb{N}$.

   For $k = 1, \ldots, L$ **do**

   $$\begin{aligned} u^k(i,j) := u^{k-1}(i,j) + \tau(u^{k-1}(i+1,j) + u^{k-1}(i-1,j) \\ + u^{k-1}(i,j-1) + u^{k-1}(i,j+1) - 4u^{k-1}(i,j)) \end{aligned}$$

   using Neumann boundary conditions in (2.2).

   Put $u^{sm} := \big(u^L(i,j)\big)_{i=1,j=1}^{N_1,N_2}$.

2. Apply a shrinkage procedure to the difference image $d = u^0 - u^{sm}$ by a hard threshold procedure. Choose θ in such a way that

   $$\tilde{d}(i,j) := S_\theta d(i,j) := \begin{cases} d(i,j) & \text{if } |d(i,j)| \geq \theta \\ 0 & \text{if } |d(i,j)| < \theta \end{cases}$$

   possesses exactly 2^J nonzero image values, where $2^J < N_1 N_2$.

   Now compute a (slightly changed) smooth part of the original image u^0, namely

   $$\tilde{u}^{sm} := u^0 - \tilde{d} = u^0 - S_\theta d.$$

3. Apply a wavelet shrinkage procedure to the smoothed image $\tilde{u}^{sm}$ using an orthogonal or biorthogonal two-dimensional wavelet transform.
   Let $\tilde{u}^{sm}_M$ be the approximation of $\tilde{u}^{sm}$ that is reconstructed using only M wavelet coefficients.

4. Consider the difference image $u^r := u^0 - \tilde{u}^{sm}_M$ that contains edges and texture. Apply again a shrinkage procedure to u^r obtaining $\tilde{u}^r = S_{\tilde{\theta}} u^r$, where $\tilde{u}^r$ possesses exactly 2^J nonzero image values.

5. Apply the EPWT with shrinkage to the detail image $\tilde{u}^r$, where only the nonzero coefficients of $\tilde{u}^r$ are used. Let $\tilde{u}^r_N$ be the approximation of $\tilde{u}^r$ using only N EPWT wavelet coefficients.

Result

Then $\tilde{u}^0 := \tilde{u}^{sm}_M + \tilde{u}^r_N$ is an approximation of u^0 where we have used only $M + N$ wavelet coefficients.

For illustration of the above algorithm, we present an example, where the partial results after each step of the algorithm are displayed.

The original image u^0 in Figure 1(a) shows a 256×256-part of the image "sails". After the first step of our algorithm, we get a smoothed version u^{sm}, see Figure 1(b). Now we apply the second step, i.e., we calculate a difference image, keep the 16384 components with largest absolute values, and add the other values to u^{sm}. In this way we obtain a slightly changed smooth image $\tilde{u}^{sm}$, see Figure 1(c). Compared with u^{sm}, it contains slightly more details; the numbers on the sails are a bit less blurry now.

According to step 3 of the algorithm, we apply a wavelet shrinkage procedure with a hard threshold to $\tilde{u}^{sm}$, and keep only 1200 coefficients; here we use 5 levels of the biorthogonal 9/7-wavelet filter bank. We obtain $\tilde{u}^{sm}_{1200}$, see Figure 1(d). The difference image $u^r = u^0 - \tilde{u}^{sm}_{1200}$ is presented in Figure 1(e). (The image shown here contains the absolute values of the difference and is inverted, i.e., white stands for 0 and black for 255). We apply again a shrinkage to this difference image keeping only $16384 = \frac{256 \times 256}{4}$ nonzero coefficients according to step 4 of the algorithm. Figure 1(f) shows an inverted version of the obtained difference $|\tilde{u}^r|$.

We apply the EPWT, and a hard threshold to keep only 800 EPWT wavelet coefficients of $\tilde{u}^r$. The reconstruction $\tilde{u}^r_{800}$ is shown in Figure 1(g), again we present here the absolute values of its components, where white stands for zero and black for 255. Finally, we add the results of wavelet shrinkage $\tilde{u}^{sm}_{1200}$ in Figure 1(d) and the result $\tilde{u}^r_{800}$ of the EPWT shrinkage in Figure 1(g) and obtain the result in Figure 1(h). For comparison, we show in Figure 1(i) the wavelet approximation of the original image by the 9/7-transform using 2000 nonzero-coefficients.

3 Application of the EPWT

In this section we describe the EPWT algorithm that we need to adapt here to a partial difference image. Let $I_J \subset \{(i,j) \mid i = 1, \ldots, N_1, j = 1, \ldots, N_2\}$ with $\#I_J = 2^J < N_1 N_2$ be the index set of nonzero image values in the thresholded difference image $\tilde{u}^r$.

We say that the nonzero value $\tilde{u}^r(i,j)$ corresponds to the index $(i,j) \in I_J$. For simplicity we use a one-dimensional representation of the index set I_J by taking the bijective mapping

$$\gamma(i,j) := i + (j-1)N_1 \qquad\qquad \text{for } i = 1, \ldots, N_1,\ j = 1, \ldots, N_2,$$

and with $\gamma(I_J)$ we denote the set of one-dimensional indices. Since $\gamma(I_J) \subset \{1, \ldots, N_1 N_2\}$, we can order the indices in $\gamma(I_J)$ by size.

Let $\tilde{u}^J$ be the vector of nonzero image values corresponding to the ordered set of indices in $\gamma(I_J)$, where we say that for $l = \gamma(i,j)$ the value $\tilde{u}^J(l) := \tilde{u}^r(i,j)$ corresponds to l. We define a neighborhood of an index $(i,j) \in I_J$ (respectively $l = \gamma(i,j) \in \gamma(I_J)$) by

$$N(i,j) := \{(i_1, j_1) \in I_J \setminus \{(i,j)\} \mid \quad |i - i_1| \le 1,\ |j - j_1| \le 1\}$$

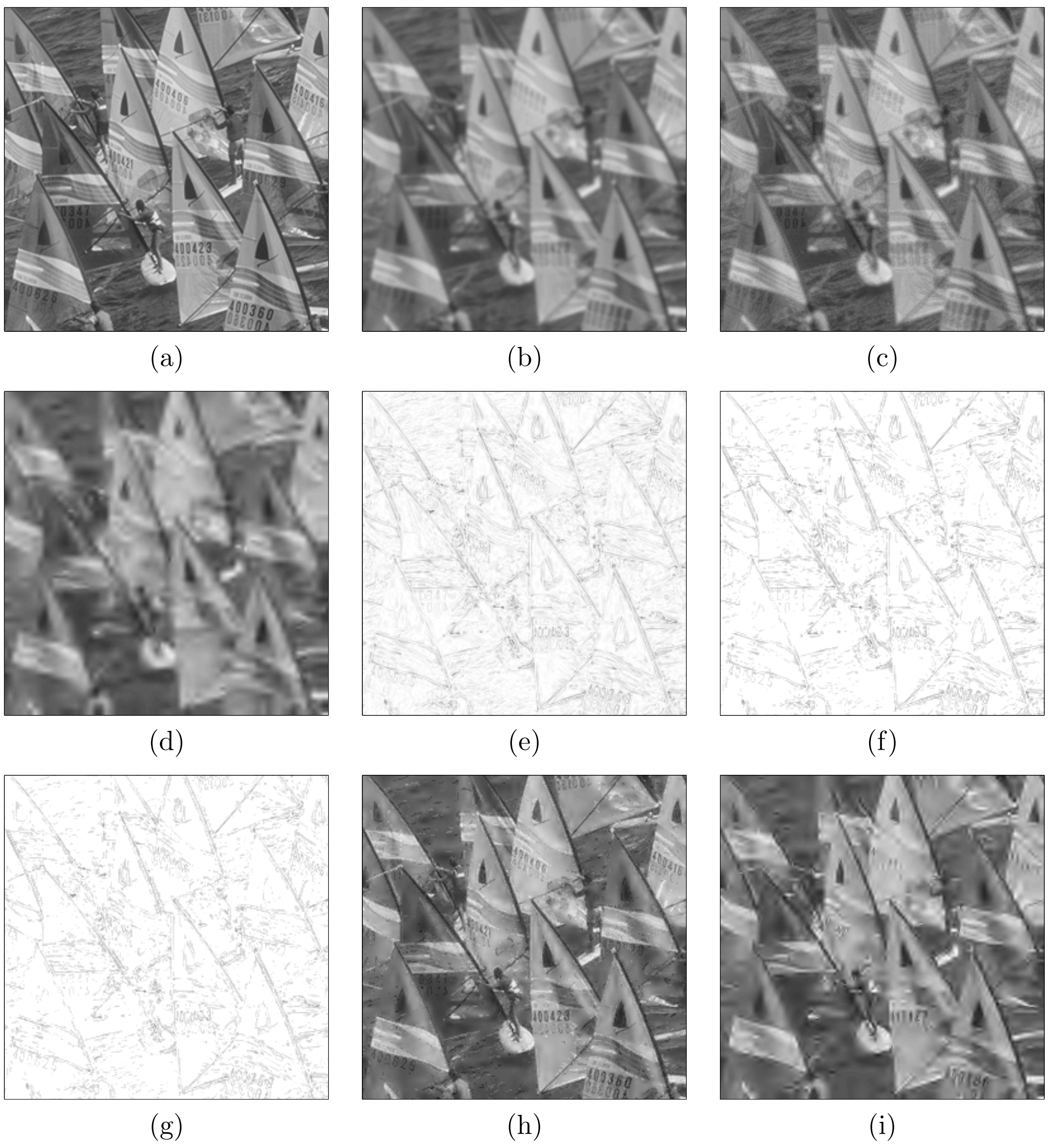

Figure 1: Illustration of the steps in Algorithm 2.4. (a) Original image, (b) smoothed image u^{sm}, (c) smoothed image $\tilde{u}^{sm}$, (d) wavelet approximation $\tilde{u}^{sm}_{1200}$, (e) difference u^{r}, absolute values, inverted, (f) shrunken difference $\tilde{u}^{r}$, inverted, (g) EPWT approximation $\tilde{u}^{r}_{800}$, inverted, (h) Our approximation $\tilde{u}^{sm}_{1200} + \tilde{u}^{r}_{800}$ with 2000 coefficients, (i) Tensor product biorthogonal 9/7 wavelet approximation with 2000 coefficients.

and

$$N(l) := \begin{cases} \{l-1, l+N_1, l-N_1, l+N_1-1, l-N_1-1\} \cap \gamma(I_J) & \text{if} \quad \operatorname{mod}(l, N_1) = 0, \\ \{l+1, l+N_1, l-N_1, l+N_1+1, l-N_1+1\} \cap \gamma(I_J) & \text{if} \quad \operatorname{mod}(l-1, N_1) = 0, \\ \{l+1, l-1, l+N_1, l-N_1, l+N_1+1, l+N_1-1, & \\ \ l-N_1+1, l-N_1-1\} \cap \gamma(I_J) & \text{elsewise.} \end{cases}$$

A vector of indices $(l_k, l_{k+1}, \ldots, l_{k+n})$ with $1 \le k < k+n \le 2^J$ is called *connected* if we have $l_{r+1} \in N(l_r)$ for $r = k, \ldots, k+n-1$. A connected vector of indices is called *pathway.*

For application of the first level of the EPWT, we need to find a complete path vector p^J through the index set $\gamma(I_J)$ that consists of a number of pathways. This first path vector $p^J = \left(p^J(n)\right)_{n=1}^{2^J}$ is a permutation of all integers occurring in $\gamma(I_J)$. It contains the information on the position of the indices in $\gamma(I_J)$, as well as information about the order in which the image values of $\tilde{u}^r$ have to be used in the first level of the EPWT. In order to determine p^J, we want to adapt the idea of the relaxed EPWT which has been introduced in [18] and use the following strategy.

Start with an arbitrary pixel $l_1 \in \gamma(I_J)$, e.g. with

$$l_1 = \min_l \{l \in I_J\},$$

and put $p^J(1) = l_1$. Now, for a given nth component of p^J, $p^J(n) = \tilde{l}$, we choose the path vector's next component $p^J(n+1)$ as follows. We consider the set of all pixels l in the neighborhood of $p^J(n)$ that are in $\gamma(I_J)$ and have not been used yet in the path p^J, i.e.

$$\tilde{N}(p^J(n)) := N(p^J(n)) \cap \gamma(I_J) \setminus \{p^J(1), \ldots, p^J(n)\}.$$

The indices in $\tilde{N}(p^J(n))$ are called "admissible" indices for $p^J(n)$. If $p^J(n-1)$ was in the neighborhood of $p^J(n)$, i.e. $p^J(n) \in \tilde{N}(p^J(n-1))$, then we try to keep the "direction" of the path. More precisely, if

$$p^J(n) + (p^J(n) - p^J(n-1)) = 2p^J(n) - p^J(n-1) \in \tilde{N}(p^J(n))$$

and if this pixel's image value is "suitable", i.e., if

$$\left|\tilde{u}^J(p^J(n)) - \tilde{u}^J\left(2p^J(n) - p^J(n-1)\right)\right| < \vartheta_1, \tag{3.3}$$

where ϑ_1 is a predetermined bound, then we take

$$p^J(n+1) := 2p^J(n) - p^J(n-1),$$

and the direction of the path is preserved. If $2p^J(n) - p^J(n-1)$ is not an admissible index or if the inequality (3.3) is not satisfied, then we apply the following procedure. Sort all admissible indices in $\tilde{N}(p^J(n))$ clockwise starting with the favorite direction index or the first admissible index in clockwise order. Check analogously to (3.3) the bound condition for every element of the obtained vector of indices and take the first index $l \in \tilde{N}_*(p^J(n))$ (here $\tilde{N}_*(p^J(n))$ denotes the ordered index set $\tilde{N}(p^J(n))$) with

$$\left|\tilde{u}^J(p^J(n)) - \tilde{u}^J(l)\right| < \vartheta_1 \tag{3.4}$$

as the next component of p^J . If there is no index $l \in \tilde{N}(p^J(n))$ satisfying the bounding condition (3.4), then we take $p^J(n+1) = l$ with

$$l = \operatorname*{argmin}_{\tilde{l} \in \tilde{N}_*(p^J(n))} \left| \tilde{u}^J(p^J(n)) - \tilde{u}^J(\tilde{l}) \right|.$$

If the set of admissible neighbors $\tilde{N}(p^J(n))$ is empty, we have to start a new pathway. For example, one can take the smallest "free" index in $\gamma(I_J)$ that has not been used yet in the path vector p^J. One might also pick a set N_7 of 7 equally distributed generally admissible (i.e., in $\gamma(I_J)$ and not already chosen) indices, and take that index as a successor of $p^J(n)$, whose corresponding image value has the smallest absolute difference to the image value of $p^J(n)$. That means, we consider, in a way similar to our choice above,

$$l = \operatorname*{argmin}_{\tilde{l} \in N_7} \left| \tilde{u}^J(p^J(n)) - \tilde{u}^J(\tilde{l}) \right|.$$

Indeed, the strategy above ensures that p^J can be cheaply coded since keeping of path direction is preferred, and the bounding condition for the image values of neighbored indices in p_J ensures mostly small differences of function values along the path and hence a small amount of significant wavelet coefficients.

We proceed in this manner and determine a complete path vector $p^J \in \mathbb{Z}^{2^J}$ that contains all indices of $\gamma(I_J)$ as components. In Figure 2(c), we show a model of a thresholded difference image and the corresponding path vector for the first level of EPWT. We consider the difference between the original image in Figure 2(a) and the smoothed image in Figure 2(b), and apply a shrinkage keeping only 64 nonzero coefficients. In the model of the difference image in Figure 2(c), gray corresponds to zero values, the darker pixels correspond to negative values and the brighter entries to positive values of the difference image.

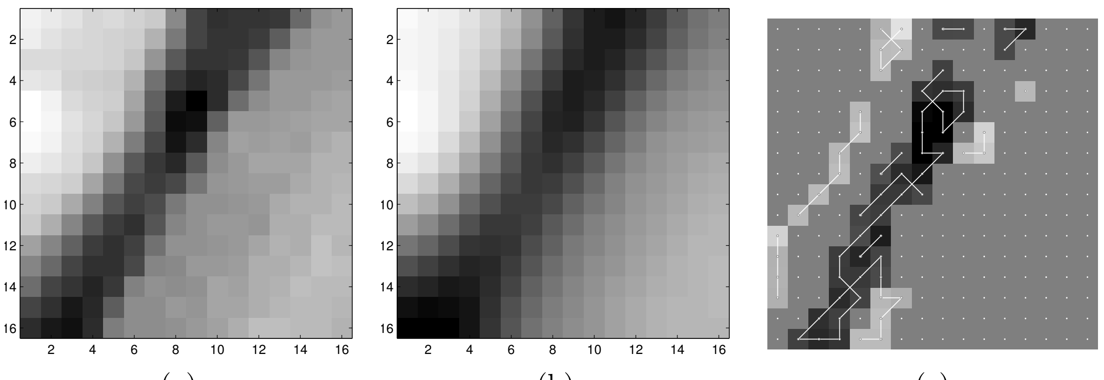

Figure 2: (a) Original image 16 × 16, (b) Smoothed image, (c) Illustration of the shrunken difference with 64 nonzero values and of the first path of the EPWT.

Now we apply one level of a discrete one-dimensional orthogonal or biorthogonal wavelet transform to the vector of function values $\left(\tilde{u}^J(p^J(n))\right)_{n=1}^{2^J}$. We obtain a vector $\tilde{u}^{J-1} \in \mathbb{R}^{2^{J-1}}$ containing the low pass part and a vector of wavelet coefficients $g^{J-1} \in \mathbb{R}^{2^{J-1}}$. Then we proceed further with the low pass vector $\tilde{u}^{J-1}$ at the second level.

The procedure for all further levels is now analogous as described in [18]. We may apply $J - s$ levels of the EPWT, supposing that the length of the used wavelet filters is smaller than or equal to 2^s. In the second level we define the index sets

$$L_n^{J-1} := p^J(2n-1) \cup p^J(2n) \qquad n = 1, \ldots, 2^{J-1},$$

and we say that L_n^{J-1} corresponds to the low-pass value $\tilde{u}^{J-1}(n)$. Observe that, since adjacent indices in the path are usually connected, L_n^{J-1} is (usually) a set of two adjacent pixels. Again we are looking for a path $p^{J-1} \in \mathbb{Z}^{J-1}$ through the index sets L_n^{J-1}, where the components $p^{J-1}(n)$ form now a permutation of the set $\{1, \ldots, 2^{J-1}\}$. As before, this path vector should be cheap to store and the difference of function values of neighbored index sets in the path vector should be small in order to avoid large wavelet coefficients.

Generally, for a given low pass vector $\tilde{u}^{J-j}$, $1 < j < J - s$, we consider the index sets

$$L_n^{J-j} := L^{J-j+1}_{p^{J-j+1}(2n-1)} \cup L^{J-j+1}_{p^{J-j+1}(2n)} \qquad n = 1, \ldots, 2^{J-j},$$

such that the low-pass value $\tilde{u}^{J-j}(n)$ corresponds to L_n^{J-j}.

If in the $(j+1)$th level of the EPWT a path vector $p^{J-j} \in \mathbb{Z}^{2^{J-j}}$ through the index sets L_n^{J-j} has been determined, then the wavelet transform is applied to the vector of function values $\left(\tilde{u}^{J-j}(p^{J-j}(n))\right)_{n=1}^{2^{J-j}}$ along the path vector and we obtain the low pass values $\tilde{u}^{J-j-1} \in \mathbb{R}^{2^{J-j-1}}$ and the wavelet vector $g^{J-j-1} \in \mathbb{R}^{2^{J-j-1}}$.

For determining a path $p^{J-j} \in \mathbb{Z}^{2^{J-j}}$ we fix again a starting index set, e.g. $p^{J-j}(1) = 1$ (corresponding to L_1^{J-j}). Then for given $p^{J-j}(n)$ we choose the next index set as follows.

First we determine all neighbor index sets of $L^{J-j}_{p^{J-j}(n)}$, where we say that L_n^{J-j} and L_m^{J-j} are *neighbors* if there exist indices $k_1 \in L_n^{J-j}$ and $k_2 \in L_m^{J-j}$ with $k_1 \in N(k_2)$. Then these neighbor index sets are ordered by "strength" of neighborhood, i.e. by the number of indices that are directly neighbors of a pixel in $L^{J-j}_{p^{J-j}(n)}$. Finally, we also use a bound, analogously to (3.4) in the first level, in order to define $p^{J-j}(n+1)$. Further strategies for determining path vectors that can be coded efficiently, are described in Section 4.

As output of the EPWT algorithm after L iterations ($L \leq J$) we obtain a vector

$$\left((g^0)^T, (g^1)^T, \ldots, (g^{L-1})^T\right)^T \in \mathbb{R}^{2^J - 2^{J-L}}$$

of EPWT-wavelet coefficients and a vector of scaling coefficients $f^0 \in \mathbb{R}^{2^{J-L}}$ as well as the path vectors determining the paths in each iteration step

$$p = \left((p^1)^T, (p^2)^T, \ldots, (p^L)^T\right)^T \quad \text{with} \quad p^k \in \mathbb{N}_0^{2^{J-L+k}} \qquad k = 1, \ldots, L.$$

For EPWT reconstruction the inverse wavelet transform along the path vectors is used.

4 Numerical Results

In this section we want to give some numerical examples of our proposed hybrid method. We apply the algorithm to different images and especially compare the results with the compression results of the 9/7 tensor product wavelet transform. We consider several images of size 256×256 (i.e. 65536 coefficients), namely `cameraman`, `clock`, `lena`, `pepper` and 256×256-details of the `barbara`, `goldhill` and `sails` image, respectively.

The PSNR (peak-signal-to-noise-ratio) is determined by

$$\mathrm{PSNR} = 20 \log_2 \frac{m}{\left\| f - \tilde{f} \right\|_2},$$

where m is the maximum possible pixel value of the image, $\|.\|_2$ is the L^2-norm and f and $\tilde{f}$ are the original data and the reconstructed sparse image, respectively. The entropy of the path vector p (i.e. a concatenation of path vectors in all levels) for the EPWT is calculated by

$$\text{entropy} := -\sum_{j=0}^{n} \frac{h_j}{N_1 N_2} \log_2(\frac{h_j}{l}), \tag{4.5}$$

where l denotes the length of the path vector, n is the number of different values that appear in the path vector and $h_1,\ldots,h_n$ denote the frequencies of their occurrence.

In a first experiment we compute a sparse representation of the image using only 500 wavelet coefficients. In the first step of Algorithm 2.4, we apply 5 iterations of the smoothing filter with time step $\tau = 0.17$. The threshold parameters θ and $\tilde{\theta}$ in step 2 and in step 4 of Algorithm 2.4 are chosen in such a way that 16384 nonzero coefficients are kept in $\tilde{u}^{sm}$ and $\tilde{u}^r$, respectively. We approximate the smooth part of the image (step 3 of the algorithm) using 5 levels of the 9/7-biorthogonal tensor product filter bank and applying a shrinkage procedure to keep only 300 coefficients. Further, we apply the EPWT with 11 iterations of the one-dimensional 9/7-filter bank to the texture part $\tilde{u}^r$ and keep 200 coefficients in step 5. Table 1 shows the total number of nonzero coefficients (nzc, second column) and the PSNR-values for the hybrid methods "Hybrid", "Center Hybrid", and "Simple Hybrid" that differ only by the choice of the EPWT path vectors in the further levels (levels $2, \ldots, 11$), see further explanations below. Further, the adaptivity costs (according to formula 4.5) for the path vectors of the EPWT for the hybrid methods are given. For comparison, the PSNR obtained using a tensor product wavelet shrinkage with 9/7-filter and with 500 kept nonzero wavelet coefficients is given in the third column of Table 1.

In a second experiment, we apply the hybrid algorithms (with same parameters) to the example images, but keep 1200 coefficients for the approximation of the smooth image and 800 EPWT coefficients for approximating the difference image, see Table 1.

We present in Figures 3 and 4 the images corresponding to the method "Hybrid", where we keep only 500 nonzero coefficients out of 65536. The original images are presented in the first column, whereas in the second and third column, we show the approximation results using 9/7-tensor product wavelet transform and the proposed hybrid method, respectively.

Let us give some remarks about the entropy of the path vector for the EPWT. Taking only a partial image $\tilde{u}^r$ in step 5 of Algorithm 2.4, we also have to store the positions of the pixels with nonzero image values in the difference image. Using the EPWT, this can easily be done by a suitable coding of the first path vector p^L. In order to minimize the entropy of the path vector, we bound the number n of different values in the stored path. In fact, the path is coded using only the values $0, \ldots, 7$ that correspond to the different possible directions of index neighbors.

		9/7	Hybrid		Center Hybrid		Simple Hybrid	
image	nzc	PSNR	PSNR	entropy	PSNR	entropy	PSNR	entropy
barbara	500	23.33	27.34	1.0497	27.28	1.0070	24.42	0.4010
cameraman	500	22.54	27.61	1.0714	27.49	0.9893	23.79	0.3794
clock	500	24.61	31.06	1.0163	30.87	0.8742	26.69	0.3014
goldhill	500	24.18	28.18	0.9918	28.19	0.8408	25.98	0.3300
lena	500	23.21	28.02	1.0343	27.91	0.9022	24.66	0.3313
pepper	500	23.41	28.07	1.0286	28.03	0.8795	24.89	0.3143
sails	500	21.32	25.52	1.0179	25.42	0.9190	22.95	0.3664
barbara	2000	26.07	30.50	1.1097	30.50	1.0950	28.12	0.4411
cameraman	2000	27.17	31.46	1.1033	31.35	1.0472	28.36	0.4153
clock	2000	29.93	35.48	1.0416	35.55	0.9329	32.49	0.3266
goldhill	2000	27.82	31.41	0.9860	31.37	0.8986	29.90	0.3556
lena	2000	28.16	32.66	1.0715	32.52	0.9699	29.97	0.3790
pepper	2000	28.84	32.97	1.0488	33.01	0.9385	30.62	0.3632
sails	2000	24.57	28.30	1.0195	28.26	0.9666	26.77	0.3916

Table 1: Comparison of 9/7-transform and our hybrid method for several images.

Recall that we determine the path vector for the first EPWT iteration by looking for a certain "well-suited" successor of a certain pixel l_1. Now, we do not store the position $l_2 \in \gamma(I_J)$ of the selected successor, but its position in the ordered neighborhood $\tilde{N}_*(l_1)$. Since we ordered the neighborhood in such a way that the first pixel maintains the direction of the path, we store a zero if we walk into the same direction as before (which happens often when we walk along edges). If the neighborhood is empty, then we consider 7 equally distributed pixels in the set of remaining admissible pixels in $\gamma(I_J)$ as possible successors (starting from the first "admissible" pixel in $\gamma(I_J)$), and again code the choice of the next pixel in the path by a value from $\{0, \ldots, 7\}$. From this easy-to-store-version of p we can later reconstruct the original pixel indices of the path.

In the following levels of the EPWT, one may apply different strategies for efficient coding of the path vectors. In the method "Hybrid", we apply simply the so-called rigorous EPWT in the second level and in all further levels, see [18]. That means, for determining a next component in the path vector, we just look at all neighbored index sets that have not been used in the path so far and choose the index set, where the difference of the corresponding (low-pass) image values is minimal. A more efficient method, called "Center Hybrid", is to compute the centers of index sets and to order neighbor index sets by the Euclidean distance of their centers to the center of L_n. Then, not necessarily the neighbor index set with the most similar image value is taken but the first neighbor index set, for which the difference of corresponding image values is smaller than a certain bound (here $\vartheta_1 = 13$ in the first two experiments). In the path vector, we only store the position of the chosen index set in the ordered set of neighbor index sets.

However, in our application, where the EPWT is only applied along edges and texture, one may expect already good approximation results if all levels of the EPWT use the same path vector p^L, obtained in the first level of the procedure (see columns 8 and 9 of Table 1). This method is called "Simple Hybrid". In this case, only p^J has to be stored.

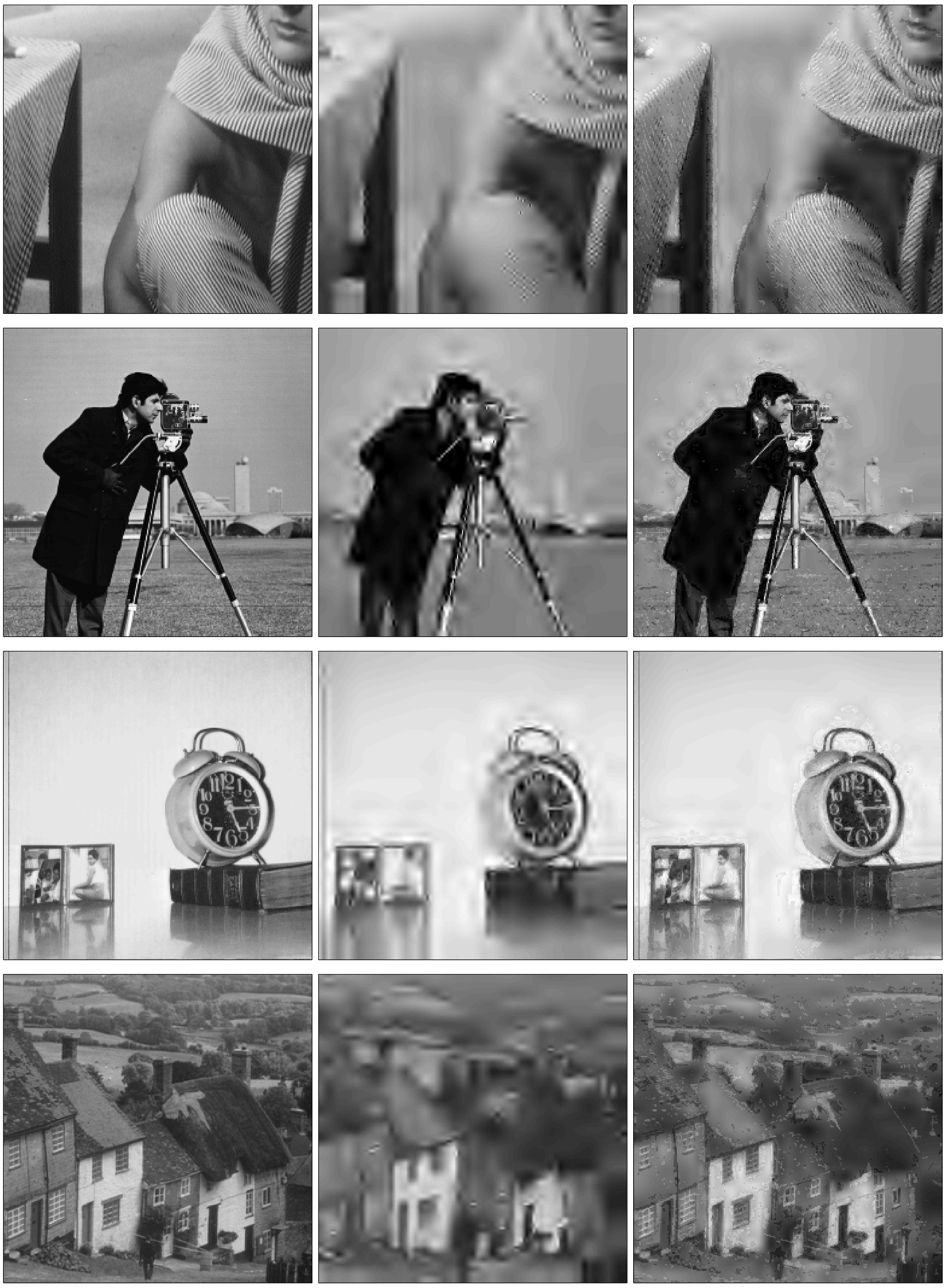

Figure 3: (Left) Original image; (middle) tensor product wavelet transform with 9/7 filter, keeping only 500 nonzero coefficients (nzc); (right) our hybrid method with 9/7 filter, keeping only 500 nzc.

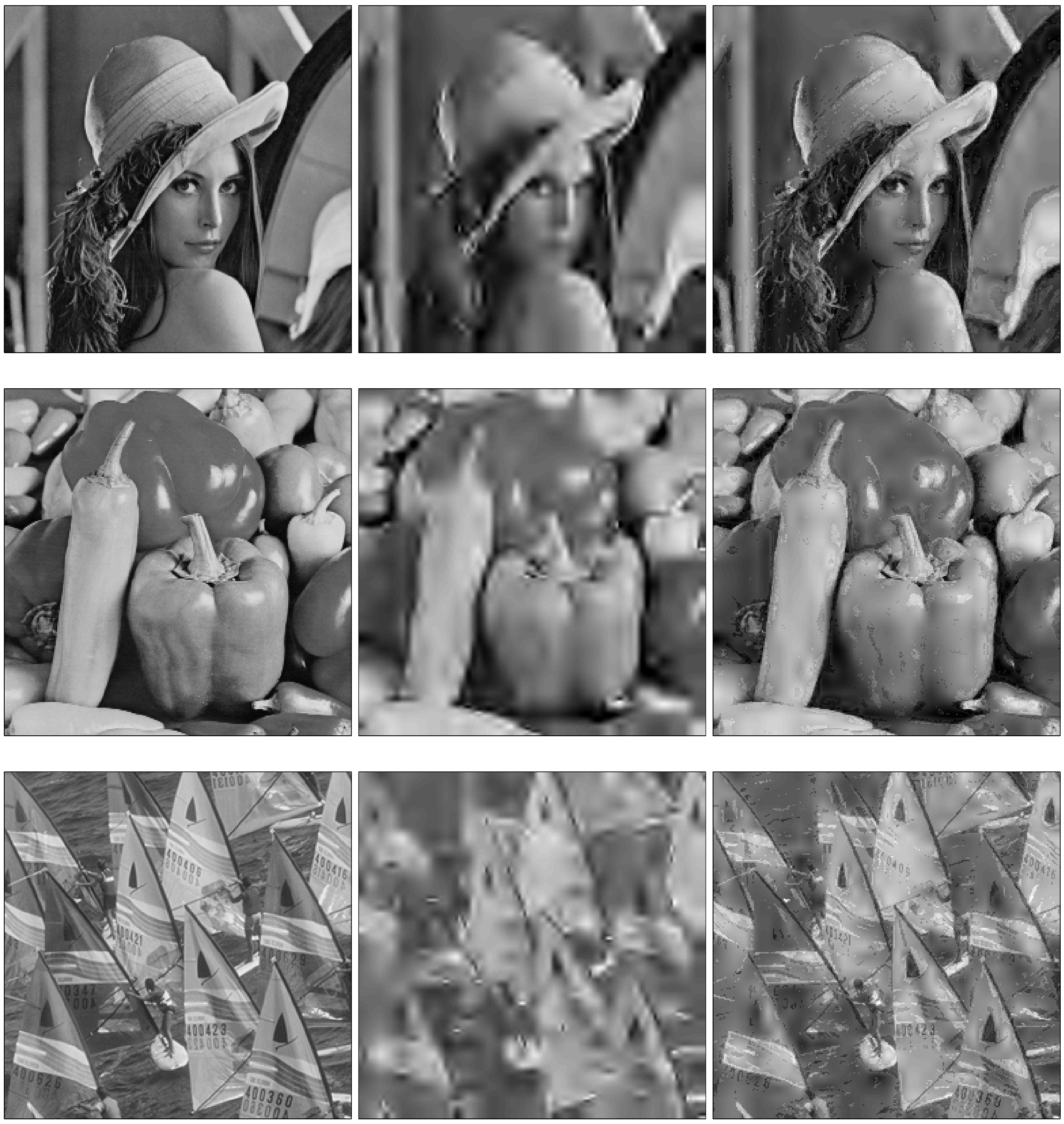

Figure 4: (Left) Original image; (middle) tensor product wavelet transform with 9/7 filter, keeping only 500 nonzero coefficients (nzc); (right) our hybrid method with 9/7 filter, keeping only 500 nzc.

		Hybrid		Center Hybrid		Simple Hybrid	
smooth	differ	PSNR	entropy	PSNR	entropy	PSNR	entropy
D4	D4	27.23	1.0274	27.05	0.8921	24.41	0.3262
D4	7-9	26.86	1.0295	26.92	0.8929	24.13	0.3262
D4	9/7	27.23	1.0248	27.18	0.8909	24.36	0.3262
7-9	D4	26.82	1.0200	26.76	0.8806	24.02	0.3175
7-9	7-9	26.65	1.0132	26.62	0.8820	24.16	0.3175
7-9	9/7	26.90	1.0125	26.86	0.8735	24.30	0.3175
9/7	D4	28.01	1.0273	27.92	0.8772	24.96	0.3143
9/7	7-9	27.67	1.0253	27.65	0.8824	24.68	0.3143
9/7	9/7	28.07	1.0286	28.03	0.8795	24.89	0.3143

Tensor product	PSNR
D4	22.51
7-9	22.15
9/7	23.41

Table 2: Different wavelet transforms used for approximation of the `pepper` image.

In Tables 2 and 3 we present some results that are obtained using different wavelet filter banks for approximation of the smooth image and the difference image. In this experiment we also use $\tau = 0.17$, five iterations of the diffusion process, and we apply 5 iterations of a tensor-product wavelet transform to approximate the smoothed image. As bound for the first level of EPWT we have chosen $\vartheta_1 = 13$. For further levels, simply the neighbored index set with the most similar value is taken, i.e. no bound is applied (see “Hybrid” in columns 3 and 4); the “Center Hybrid” strategy with bound 13 is used (columns 5 and 6); and in columns 7 and 8 the results of the ”Simple Hybrid” strategy are shown. The one-dimensional wavelet transform used for EPWT (11 iterations) is given in the second column. For comparison, we also mention the PSNR-values that are obtained by applying a $D4$-, $7-9$- and 9/7-tensor-product wavelet transform to the original image. In all cases we have used a hard threshold in order to keep only 500 nonzero-coefficients; the original images `pepper` and `lena` are each of size 256×256.

The Tables 4 and 5 illustrate the dependence of the results from the choice of the parameter τ and from the number of iterations in the diffusion process. Here the clock image of size 256×256 is used. In Table 4, five iterations of the diffusion filter are applied but with different time step τ. All other parameters are the same as for the first experiment, where 500 nonzero coefficients are kept. For comparison, the PSNR-value obtained by applying a 9/7 tensor product wavelet transform to the original image is 24.61 dB. We see that the results only slightly depend on τ (if τ stays in a certain range). Table 5 presents the PSNR- and entropy-values that result from the fixed time step $\tau = 0.17$, but different numbers of iterations of the diffusion process (the other parameters are chosen as above).

Our numerical results suggest that we can take five iterations of the smoothing filter with a fixed $\tau = 0.17$ to obtain satisfying results for all considered images.

Finally, in Table 6 we illustrate the influence of the bound ϑ_1 (used for finding the first

		Hybrid		Center Hybrid		Simple Hybrid	
smooth	differ	PSNR	entropy	PSNR	entropy	PSNR	entropy
D4	D4	26.60	1.0097	26.47	0.8775	23.74	0.3216
D4	7-9	26.34	1.0044	26.42	0.8851	23.75	0.3216
D4	9/7	26.70	1.0051	26.57	0.8768	23.97	0.3216
7-9	D4	26.51	1.0016	26.47	0.8771	23.94	0.3191
7-9	7-9	26.33	1.0081	26.31	0.8779	23.88	0.3191
7-9	9/7	26.63	1.0137	26.55	0.8742	23.99	0.3191
9/7	D4	27.90	1.0318	27.75	0.9043	24.66	0.3313
9/7	7-9	27.62	1.0307	27.54	0.9067	24.57	0.3313
9/7	9/7	28.02	1.0343	27.91	0.9022	24.66	0.3313
Tensor product		PSNR					
D4		22.13					
7-9		22.02					
9/7		23.21					

Table 3: Different wavelet transforms used for approximation of the `lena` image.

	Hybrid		Center Hybrid		Simple Hybrid	
τ	PSNR	entropy	PSNR	entropy	PSNR	entropy
0.04	30.95	1.0203	30.81	0.8656	26.65	0.3008
0.08	30.94	1.0209	30.95	0.8738	26.66	0.3047
0.12	30.89	1.0156	30.77	0.8650	26.58	0.2988
0.14	30.90	1.0224	30.67	0.8688	26.68	0.2958
0.18	30.93	1.0186	30.92	0.8744	26.62	0.2978
0.20	30.91	1.0197	31.03	0.8820	26.67	0.3016
0.22	31.09	1.0256	31.14	0.8841	26.74	0.3041
0.24	31.02	1.0235	30.98	0.8798	26.46	0.2982

Table 4: PSNR- and entropy results for different values of τ.

	Hybrid		Center Hybrid		Simple Hybrid	
Iterations	PSNR	entropy	PSNR	entropy	PSNR	entropy
1	30.81	1.0186	30.84	0.8688	26.56	0.3072
3	30.92	1.0221	30.80	0.8691	26.67	0.3033
5	31.06	1.0163	30.87	0.8742	26.69	0.3014
8	31.02	1.0321	30.99	0.8784	26.42	0.3008
12	30.86	1.0297	31.25	0.8795	26.43	0.2939
20	31.03	1.0269	31.09	0.8871	26.15	0.2912
24	31.17	1.0318	31.02	0.8878	26.19	0.2883
28	30.94	1.0298	31.03	0.8891	26.04	0.2929

Table 5: PSNR- and entropy results for different numbers of iterations of the diffusion process.

	Hybrid		Center Hybrid		Simple Hybrid	
bound	PSNR	entropy	PSNR	entropy	PSNR	entropy
0	31.04	1.2176	31.04	1.2138	25.65	0.5342
1	31.05	1.2048	31.19	1.1966	25.59	0.5193
3	30.94	1.1678	30.88	1.1437	25.65	0.4704
6	31.11	1.0936	31.02	1.0390	25.73	0.3817
9	31.10	1.0670	31.14	0.9676	26.17	0.3335
12	31.25	1.0407	31.05	0.9017	25.96	0.2992
13	31.06	1.0163	31.03	0.8891	26.04	0.2929
15	31.17	1.0140	31.05	0.8483	26.12	0.2736
18	31.05	0.9956	30.84	0.7953	26.05	0.2901
21	31.06	0.9944	30.87	0.7678	26.42	0.2376
30	30.97	0.9644	30.72	0.6849	26.13	0.2029
90	29.79	0.8796	28.28	0.3910	24.25	0.0360

Table 6: Different bounds.

path vector for the EPWT) for the clock image. All other parameters are again chosen as in the first experiment. Taking a higher bound ϑ_1, the entropy of the path decreases. Interestingly, the PSNR increases with the bound up to $\vartheta_1 = 12$, and decreases afterwards. It seems that the "best choice" for a particular pixel (i.e., the neighbor pixel with most similar value) might often be only locally the best solution but not the best for the whole image. The bound used for "Center Hybrid" in the further levels is the same bound that is used for the first level.

5 Conclusion

In this paper, we have introduced a first hybrid method that uses the tensor-product wavelet transform for smooth images on the one hand and the EPWT for a sparse representation of the edges and textures of the image on the other hand. Similarly as most known adaptive transforms for image approximation, the EPWT provides very good compression results but produces a non-negligible amount of extra costs due to the adaptivity of the method. Incorporating these "adaptivity costs", adaptive methods only slightly outperform the non-adaptive methods but with essentially higher computational costs. One way to obtain a real improvement for image approximation may be to study hybrid methods as we did in the paper. Also here, the remaining adaptivity costs are not negligible but considerably smaller than for the "pure" EPWT for image approximation. In particular, a further improvement of pathway determination and path coding may lead to a compression algorithm that is truly interesting for practical purposes.

Acknowledgement. The research in this paper is supported by the project 436 RUM 113/31/0-1 of the German Research Foundation (DFG). This is gratefully acknowledged.

References

[1] F. Arandiga, A. Cohen, R. Donat, N. Dyn, and B. Matei, Approximation of piecewise smooth functions and images by edge-adapted (ENO-EA) nonlinear multiresolution techniques, *Appl. Comput. Harmon. Anal.* **24** (2008), 225–250.

[2] X. Bresson and T.F. Chan, Non-local unsupervised variational image segmentation models, preprint, 2008.

[3] E.J. Candès and D.L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise singularities, *Comm. Pure Appl. Math.* **57** (2004), 219–266.

[4] E.J. Candès, L. Demanet, D.L. Donoho and L. Ying, Fast discrete curvelet transforms, *Multiscale Model. Simul.* **5** (2006), 861–899.

[5] T. F. Chan, S. Esedoglu, M. Nikolova, Algorithms for finding global minimizers of denoising and segmentation models, *SIAM J. Appl. Math.* **66** (2006), 1632–1648.

[6] A. Cohen and B. Matei, Compact representation of images by edge adapted multiscale transforms. in Proc. IEEE Int. Conf. on Image Proc. (ICIP), 2001, Thessaloniki, Greece, pp. 8-11.

[7] S. Dekel and D. Leviatan, Adaptive multivariate approximation using binary space partitions and geometric wavelets, *SIAM J. Numer. Anal.* **43** (2006), 707–732.

[8] L. Demaret, N. Dyn, and A. Iske, Image compression by linear splines over adaptive triangulations. *Signal Processing* **86** (2006), 1604–1616.

[9] M.N. Do and M. Vetterli, The contourlet transform: An efficient directional multiresolution image representation, *IEEE Trans. Image Process.* **14** (2005) 2091–2106.

[10] D.L. Donoho, Wedgelets: Nearly minimax estimation of edges, *Ann. Stat.* **27** (1999), 859–897.

[11] K. Guo and D. Labate, Optimally sparse multidimensional representation using shearlets, *SIAM J. Math. Anal.* **39** (2007), 298–318.

[12] K. Guo, W.-Q. Lim, D. Labate, G. Weiss, and E. Wilson, Wavelets with composite dilations, *Electr. res. Announc. of AMS* **10** (2004), 78–87.

[13] L. Jacques and J.-P. Antoine, Multiselective pyramidal decomposition of images: wavelets with adaptive angular selectivity, *Int. J. Wavelets Multiresolut. Inf. Process.* **5** (2007), 785–814.

[14] J. Krommweh, Tetrolet Transform: A New Adaptive Haar Wavelet Algorithm for Sparse Image Representation, preprint, 2009.

[15] E. Le Pennec, and S. Mallat, Bandelet image approximation and compression, *Multiscale Model. Simul.* **4** (2005), 992–1039.

[16] S. Mallat, A wavelet tour of signal processing, Academic Press, San Diego, 1999.

[17] S. Mallat, Geometrical grouplets, *Appl. Comput. Harmon. Anal.* **26** (2009), 161–180.

[18] G. Plonka, The easy path wavelet transform: A new adaptive wavelet transform for sparse representation of two-dimensional data, *Multiscale Model. Simul.* **7** (2009), 1474–1496.

[19] G. Plonka, and D. Roşca, Easy path wavelet transform on triangulations of the sphere, *Math. Geosciences*, to appear.

[20] G. Plonka, S. Tenorth, and A. Iske, Optimally sparse image representation by the easy path wavelet transform, preprint, 2009.

[21] J.-L. Starck, M. Elad, and D.L. Donoho, Image Decomposition via the combination of sparse representations and a variational approach, *IEEE Trans. Image Process.* **14** (2005), 1570–1582.

[22] J. Weickert, *Anisotropic Diffusion in Image Processing*, Teubner, Stuttgart, 1998.