A library of spline based biorthogonal wavelet transforms for image compression

Amir Z. Averbuch* Valery A. Zheludev
{amir,zhel}@math.tau.ac.il
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel

EDICS={2-ADPT,2-FILB,2-FILT,2-POLY}

Abstract

In this paper we describe the design of a library of biorthogonal wavelet transforms and their successful applications to still image compression. The wavelet transforms were constructed from various types of interpolatory and quasi-interpolatory splines. The transforms use FIR and IIR filters and are implemented in a fast lifting mode.

Keywords: Wavelet transform, lifting scheme, spline, image compression.

1 Introduction

By now wavelet transforms grew into becoming one of the basic tools for signal, image processing and numerical computation. In particular, biorthogonal symmetric wavelets manifested remarkable abilities in still image compression, especially the so called 9/7 wavelets [1], which were adopted by JPEG 2000 image compression standard [20]. In this paper we present a new library of biorthogonal wavelet transforms, which proved to be efficient for image compression. Transforms belonging to this library outperform the transform with 9/7 wavelets on most of the benchmark images. Moreover, the technique, which we explored for the design of wavelet transforms, is generic. It enables the construction of transforms with predetermined properties. It provides means for flexible adaptation of the transforms to problems under consideration.

Our approach is based on the combination of the versatile custom design abilities inherent in the lifting schemes [25] with the usage of the well developed theory of interpolatory, quasi-interpolatory, parameterized, continuous and discrete splines [19, 24, 29, 30].

Polynomial splines are a favorite source for wavelet constructions. Until recently, two methods have been used for the construction of wavelet schemes using splines. One is to construct orthogonal (Battle-Lemarié, [7, 14]) and semi–orthogonal wavelets in the spline spaces [9, 28]. There exist compactly supported spline-wavelets. However, their dual wavelets have infinite support. Another option for employing splines in wavelet analysis was introduced by Cohen, Daubechies and Feauveau [10], who constructed symmetric compactly supported spline wavelets whose duals remain compactly supported and symmetric but do not belong to a spline space.

However, since the introduction of the lifting scheme for the design of wavelet transforms [25], a new way was opened to use splines as a tool for devising a full discrete scheme of wavelet transforms.

The basic lifting scheme for wavelet transform of a discrete–time signal consists of three steps:

Split – The signal is split into even and odd subarrays.

Predict – The filtered even array is used to predict the odd array. Then, the odd array is redefined as the difference between the existing array and the predicted one. If the predictor is correctly chosen then this step decorrelates the signal and reveals its high-frequency component.

Update – To eliminate aliasing, which appears while downsampling the original signal, and to obtain the low-frequency component of the signal, the even array is updated using the filtered new odd array.

The newly produced even and odd subarrays are the coefficients of one decomposition step of a wavelet transform. The inverse transform is implemented in reverse order.

The transform generates biorthogonal wavelet bases for the signal space. The specifics of the transform and its generated wavelets are determined by the choice of the predicting and updating filters. In the construction by Donoho [12] an odd sample is predicted from a polynomial interpolation of neighboring even samples. We propose to use various kinds of splines as predicting and updating aggregates in the lifting scheme. In the following we outline how we used splines for this purpose.

Continuous interpolatory splines: The interpolatory spline of odd order (even degree) with equidistant nodes possesses the property of super–convergence at the midpoints of the intervals between grid points [29]. Thus, we built a spline of odd order, which interpolates even samples of the signal and predict odd samples by the values of the spline at the midpoints of the intervals between grid points [5].

Discrete interpolatory splines: Another option is to use the discrete interpolatory splines [3, 4]. In this case explicit formulas for the transforms with any number of vanishing moments are established. Moreover, our investigation revealed an interesting relation between the discrete splines and the Butterworth filters which are used in signal processing. The filter banks, which are used in our scheme, comprise filters which act as bi-directional half-band Butterworth filters. The frequency response of a Butterworth filter is maximally flat and we succeeded in the construction of dual filters with a similar property.

Unlike the construction in [12], the above transforms used causal and anti–causal filters with infinite impulse response (IIR). Fortunately, the transfer functions of the employed filters are rational. Therefore, filtering can be performed in a recursive manner. We established explicit formulas which enable fast cascade or parallel implementation.

The one-directional Butterworth filters were used for devising orthogonal non-symmetric wavelets [13]. The computations there were conducted in time domain using recursive filtering. A scheme that used recursive filters for the construction of biorthogonal symmetric wavelets and their application to image processing was presented in [15, 18].

Quasi- interpolatory splines: There is a way to devise wavelet transforms employing finite impulse response (FIR) filters that have similar properties to those that interpolatory transforms possess. It can be done using the so called quasi-interpolatory splines [29]. These splines also possess the super-convergence property.

The rest of the paper is organized as follows. In Section 2 we outline the lifting scheme of wavelet transforms and discuss its relation to the conventional setting of wavelet transforms. In Section 3 we describe the derivation of the prediction and update filters from interpolatory and quasi-interpolatory splines. We also discuss there the implementation of the derived recursive filters. Note that IIR filters with rational transfer functions, which allow recursive implementation, appear in signal processing algorithms using spline functions. Construction and implementation of such filters were studied in [26, 27]. Our scheme for implementation of these filters is close to that in [27]. More details are provided in Appendix I. By combining the devised filters, we construct in Section 4 a number of biorthogonal wavelet transforms and calculate the computational cost of their implementations. Then, in Section 5 we present the results of compression of four benchmark images using these transforms. These results are compared with the performance of the 9/7 transform. In Appendix II we describe a direct 2D implementation of a transform, which use FIR filters.

2 Biorthogonal wavelet transforms

We call the sequences $\{a(k)\}_{k=-\infty}^{\infty}$, which belong to the space l_1, the discrete-time signals and they are denoted by $\mathcal{S}$. The $z-$transform of a signal $\{a(k)\} \in \mathcal{S}$ is defined to be $a(z) = \sum_{k=-\infty}^{\infty} z^{-k}\, a(k)$. Throughout the paper we assume that $z = e^{i\omega}$. We introduce a family of biorthogonal wavelet-type transforms that operate on the signal $\mathbf{x} = \{x(k)\}_{k=-\infty}^{\infty}$, which we construct through lifting steps.

First we recall a basic lifting scheme of wavelet transform. The lifting scheme [25] can be implemented in either primal or dual mode.

2.1 Lifting scheme of the wavelet transform

2.1.1 Decomposition

Generally, the primal lifting scheme for decomposition of signals consists of four steps: 1. Split. 2. Predict. 3. Update or lifting. 4. Normalization.

Split - The array $\mathbf{x}$ is split into even and odd sub-arrays: $\mathbf{e_1} = \{e_1(k) = x(2k)\}$, $\quad \mathbf{o_1} = \{o_1(k) = x(2k+1)\}$, $k \in \mathbb{Z}$.

Predict - The even array $\mathbf{e_1}$ is used to predict the odd array $\mathbf{o_1}$. Then, the new odd array $\mathbf{o_1^\nu}$ is defined as the difference between the existing array $\mathbf{o_1}$ and the predicted one. To be specific, we apply some *prediction* filter whose transfer function we denote as $zU(z)$ to the array $\mathbf{e_1}$ and predict the function $o_1(z^2)$ which is the z^2-transform of $\mathbf{o_1}$:

$$o_1^\nu(z^2) = o_1(z^2) - zU(z)e_1(z^2). \tag{2.1}$$

From now on the superscript ν designates the new array. Of course, we require $U(-z) = -U(z)$, which implies that $zU(z)$ is a function of z^2. Obviously, $zU(z)e_1(z^2)$ should well approximate $o_1(z^2)$. Then, after this step the signal is decorrelated.

Update (lifting) - The even array is updated using the new odd array, which had been convolved with the *update* filter whose transfer function is $V(z)z^{-1}$ (again we require $V(-z) = -V(z)$):

$$e_1^\nu(z^2) = e_1(z^2) + \frac{1}{2z}V(z)\,o_1^\nu(z^2). \tag{2.2}$$

Generally, the goal of this step is to eliminate aliasing which appears while downsampling the original signal $\mathbf{x}$ into $\mathbf{e_1}$. By doing so we have that $\mathbf{e_1}$ is transformed into downsampled and smoothed (low-pass filtering) replica of $\mathbf{x}$.

Normalization - Finally, the smoothed array $\mathbf{s_1}$ and the array of details $\mathbf{d_1}$ are obtained by the following operation: $\mathbf{s_1} = \sqrt{2}\,\mathbf{e_1^\nu}$, $\quad \mathbf{d_1} = \mathbf{o_1^\nu}/\sqrt{2}$.

The key issue in the lifting scheme is how to properly choose the filters U and V. We address this issue in subsequent sections.

2.1.2 Reconstruction

The reconstruction of the signal $\mathbf{x}$ from the arrays $\mathbf{s_1}$ and $\mathbf{d_1}$ is implemented in reverse order:

Undo Normalization - $\mathbf{e}_1^\nu = \mathbf{s}_1/\sqrt{2} \quad \mathbf{o}_1^\nu = \sqrt{2}\,\mathbf{d}_1$.

Undo Lifting - The even array $e_1(z^2) = e_1^\nu(z^2) - V(z)/(2z)\, o_1^\nu(z^2)$ is restored.

Undo Predict - The odd array $o_1(z^2) = o_1^\nu(z^2) + zU(z)e_1(z^2)$ is restored.

Unsplit - The last step represents the standard restoration of the signal from its even and odd components. In the $z-$ domain it looks as: $x(z) = e_1(z^2) + z^{-1}o_1(z^2)$.

2.1.3 Dual mode

The wavelet transform can be implemented in a dual mode of the lifting scheme. Unlike the primal mode, in the dual mode the Update step foregoes the Predict step:

Update– The even array is averaged with the filtered odd array:
$e_1^\nu(z^2) = \left(e_1(z^2) + V(z)z^{-1}\, o_1(z^2)\right)/2$.

Predict – The odd array is predicted by the filtered new even array:
$o_1^\nu(z^2) = o_1(z^2) - zU(z)e_1^\nu(z^2)$. In the reconstruction the operations are reversed.

2.2 Filter banks

Lifting schemes, that were presented above, yield efficient algorithms for the implementation of the forward and backward transform of $\mathbf{x} \longleftrightarrow \mathbf{s_1} \cup \mathbf{d_1}$. But these operations can be interpreted as a filter bank based transform operated on the signals. These filter bank possesses the perfect reconstruction property ([17]).

Let $\Phi^l(z) \triangleq (1 + U(z))/2, \quad \Phi^h(z) \triangleq (1 - U(z))/2$ then we define the following filter transfer functions

$$\tilde{g}(z) \triangleq \sqrt{2}z^{-1}\Phi^h(z), \quad \tilde{h}(z) \triangleq \sqrt{2}(1 + V(z)\Phi^h(z)) = \sqrt{2} + V(z)\, z\tilde{g}(z),$$
$$h(z) \triangleq \sqrt{2}\Phi^l(z), \quad g(z) \triangleq \sqrt{2}z^{-1}\left(1 - V(z)\Phi^l(z)\right) = z^{-1}\left(1 - V(z)\, h(z)\right).$$

We call $\tilde{h}(z)$ and $\tilde{g}(z)$ the transfer functions of the low-pass and high-pass primal analysis filters, respectively. We call $h(z)$ and $g(z)$ the transfer functions of the low-pass and high-pass primal synthesis filters, respectively. These four filters form a perfect reconstruction filter bank.

Proposition 2.1 *The decomposition and reconstruction equations can be represented as follows:*

$$\begin{aligned} s_1(z^2) &= \frac{1}{2}\Big(\overline{\tilde{h}(z)}\,x(z) + \overline{\tilde{h}(-z)}\,x(-z)\Big) \quad d_1(z^2) = \frac{1}{2}\Big(\overline{\tilde{g}(z)}\,x(z) + \overline{\tilde{g}^r(-z)}\,x(-z)\Big) \\ x(z) &= h(z)s_1(z^2) + g(z)\,d_1(z^2). \end{aligned}$$

If U and V satisfy the requirements $U(-z) = -U(z)$ and $V(-z) = -V(z)$ then the functions $\tilde{h}(z)$, $\tilde{g}(z)$, $h(z)$ and $g(z)$ possess the perfect reconstruction property

$$h(z)\,\overline{\tilde{h}(z)} + g(z)\,\overline{\tilde{g}(z)} = 2 \quad h(z)\,\overline{\tilde{h}(-z)} + g(z)\,\overline{\tilde{g}(-z)} = 0.$$

The filter functions are linked to each other as follows: $\tilde{g}(z) = z^{-1}h(-z)$; $\quad g(z) = z^{-1}\,\tilde{h}(-z)$.

2.3 Bases for signal space

The perfect reconstruction filter banks, constructed above, are associated with the biorthogonal pairs of bases in the space $\mathcal{S}$ of discrete-time signals.

In section 2.2 we introduced a family of filters by their transfer functions $h(z)$, $g(z)$, $\tilde{h}(z)$, $\tilde{g}(z)$. We denote by $\varphi^1(k)$, $\psi^1(k)$, $\tilde{\varphi}^1(k)$, $\tilde{\psi}^1(k)$ the impulse response functions of the corresponding filters, respectively. It means, for example, that $h(z) = \sum_{k\in\mathbb{Z}} z^{-k}\varphi^1(k)$ and similar relations hold for the other functions.

Proposition 2.2 *The shifts of the functions $\varphi^1(k)$, $\psi^1(k)$, $\tilde{\varphi}^1(k)$ and $\tilde{\psi}^1(k)$ form a biorthogonal pair of bases for the signal space $\mathcal{S}$. This means that any signal $\mathbf{x} \in \mathcal{S}$ are represented as:*

$$x(l) = \sum_{k\in\mathbb{Z}} s_1(k)\,\varphi^1(l-2k) + \sum_{k\in\mathbb{Z}} d_1\,\psi^1(l-2k).$$

The coordinates $s_1(k)$ and $d_1(k)$ can be represented as inner products:

$$s_1(k) = \langle \mathbf{x}, \tilde{\varphi}^1_k \rangle,\ \textit{where}\ \tilde{\varphi}^1_k(l) = \tilde{\varphi}^1(l-2k), \quad d_1(k) = \langle \mathbf{x}, \tilde{\psi}^1_k \rangle,\ \textit{where}\ \tilde{\psi}^1_k(l) = \tilde{\psi}^1(l-2k).$$

The following biorthogonal relations hold:
$\langle \tilde{\varphi}^1_k, \varphi^1_l \rangle = \langle \psi^1_k, \tilde{\psi}^1_l \rangle = \delta^l_k, \quad \langle \tilde{\varphi}^1_k, \psi^1_l \rangle = \langle \tilde{\psi}^1_l, \varphi^1_k \rangle = 0, \quad \forall l,\ k.$

Proposition 2.2 justifies the following definition.

Definition 2.1 *The functions φ^1 and ψ^1, which belong to the space $\mathcal{S}$, are called the low- and high-frequency synthesis wavelets of the first scale, respectively. The functions $\tilde{\varphi}^1$ and $\tilde{\psi}^1$ are called the low- and high-frequency analysis wavelets of the first scale, respectively.*

We say that a wavelet ψ has m vanishing moments if the following relations hold:
$\sum_{k\in\mathbb{Z}} k^s\psi(k) = 0, \quad s = 0, 1, \ldots, m-1.$

Proposition 2.3 *Suppose the filter $U(z)$ is used for the* predict *step (2.1). If $1 - U(z)$ comprises the factor $(z - 2 + z^{-1})^r$ then the high-frequency analysis wavelets $\tilde{\psi}^1$ have $2r$ vanishing moments.*

2.3.1 Multiscale wavelet transforms

Expansion of the transform to operats on coarser scales is implemented in a recursive way. In this transform we store the array $\mathbf{d}_1$ and decompose the array $\mathbf{s}_1$. The transformed arrays $\mathbf{s}_2$ and $\mathbf{d}_2$ of the second decomposition scale are derived from the even and odd sub-arrays of the array $\mathbf{s}_1$ with the same lifting steps as those that were described in section 2.1. As a result we get that the signal $\mathbf{x}$ is transformed into three subarrays: $\mathbf{x} \leftrightarrow \mathbf{d}_1 \bigcup \mathbf{d}_2 \bigcup \mathbf{s}_2$. The reconstruction is performed in a reverse order.

Again, the transform leads to expansions of the signal with biorthogonal pair of bases:

$$x(l) = \sum_{k\in\mathbb{Z}} s_2(k)\,\varphi^2(l-4k) + \sum_{k\in\mathbb{Z}} d_2(k)\,\psi^2(l-4k) + \sum_{k\in\mathbb{Z}} d_1(k)\,\psi^1(l-2k) \tag{2.3}$$

where low- and high-frequency synthesis wavelets of the second scale are defined as

$$\varphi^2(l) = \sum_{k\in\mathbb{Z}} \varphi^1(k)\varphi^1(l-2k), \quad \psi^2(l) = \sum_{k\in\mathbb{Z}} \psi^1(k)\varphi^1(l-2k).$$

The coordinates in (2.3) are inner products with 4-sample shifts of the following analysis wavelets of the second scale:

$$\tilde{\varphi}^2(l) = \sum_{k\in\mathbb{Z}} \tilde{\varphi}^1(k)\tilde{\varphi}^1(l-2k), \quad \tilde{\psi}^2(l) = \sum_{k\in\mathbb{Z}} \tilde{\psi}^1(k)\tilde{\varphi}^1(l-2k).$$

Namely, $s_2(k) = \langle \mathbf{x}, \tilde{\varphi}^2_k \rangle$, where $\tilde{\varphi}^2_k(l) = \tilde{\varphi}^2(l-4k)$, $d_2(k) = \langle \mathbf{x}, \tilde{\psi}^2_k \rangle$, where $\tilde{\psi}^2_k(l) = \tilde{\psi}^2(l-4k)$.

3 Design of predict and update filters

In this section we first design the filters U for the Predict operation. Our goal here consists in devising filters such that, being applied on the even sub-array of a signal, they will produce a signal, which approximates the odd sub-array. We will use polynomial and discrete splines as a tool for the construction of the appropriate filters. In general, we construct a spline on the grid $\{2k\}$, which interpolates or quasi-interpolates the even samples $\{e_1(k)\}$. Then, we calculate the values of this spline on grid points $\{2k+1\}$. These values are used for the prediction of the odd samples $\{o_1(k)\}$.

3.1 Discrete splines

In [3, 4] we gave a detailed presentation how to construct prediction filters using discrete splines. Therefore, here we only recall the basic idea and present two filters with rational coefficients.

The discrete B-spline of the first order is defined by the following sequence:

$$B_{1,n}(j) \triangleq \begin{cases} 1 & \text{if } j = 0,\ldots,2n-1,\ n\in\mathbb{N}, \quad j\in\mathbb{Z} \\ 0 & \text{otherwise.} \end{cases}$$

Higher order B-splines are defined by recursive usage of discrete convolutions: $B_{p,n} = B_{1,n} * B_{p-1,n},\ p = 2,\ldots$. Obviously, the z-transform of the B-spline of order p is $B_{p,n}(z) = (1 + z^{-1} + z^{-2} + \ldots + z^{-2n+1})^p \quad p = 1, 2, \ldots$.

In this paper we are interested only in the case when $p = 2r,\ r \in \mathbb{N}$ and $n = 1$. The corresponding spline s are denoted as $B^{2r} = B_{2r,1}$. In this case we have $B^{2r}(z) = (1 + z^{-1})^{2r}$. The B-spline $B^{2r}(j)$ is symmetric about the point $j = r$ where it attains its maximal value. The central B-spline $Q^{2r}(j)$ of order $2r$ is a shift of the B-spline: $Q^{2r}(j) \triangleq B^{2r}(j + r), \quad Q^{2r}(z) = z^r B^{2r}(z) = z^r(1 + z^{-1})^{2r}$. The discrete spline of order $2r$ is defined as a linear combination, with real-valued coefficients, of shifts of the central B-spline of order $2r$:

$$S^{2r}(k) \triangleq \sum_{l=-\infty}^{\infty} c(l) Q^{2r}(k - 2l).$$

Definition 3.1 *Let $\{e(k)\} \in \mathcal{S}$ be a given sequence. The discrete spline S^{2r} is called the interpolatory spline if the following relations hold: $S^{2r}(2k) = e(k),\ k \in \mathbb{Z}$. The points $\{2k\}$ are called the nodes of the spline.*

The following proposition shows how to construct the interpolatory splines of any order and to calculate its values in the midpoints between nodes, which we denote by $\sigma(k) = S^{2r}(2k + 1),\ k \in \mathbb{Z}$.

Proposition 3.1 *[3]The interpolatory spline is represented as*

$$S^{2r}(k) = \sum_{l=-\infty}^{\infty} c(l) Q^{2r}(k - 2l), \quad c(z^2) = \frac{2e(z^2)}{z^r\left(1 + z^{-1}\right)^{2r} + (-z)^r\left(1 - z^{-1}\right)^{2r}}.$$

The $z-$transform of the interpolatory spline in the midpoints are

$$\sigma(z^2) = z U_d^{2r}(z) e(z^2), \quad U_d^{2r}(z) \triangleq \frac{\left(1 + z^{-1}\right)^{2r} - (-1)^r\left(1 - z^{-1}\right)^{2r}}{\left(1 + z^{-1}\right)^{2r} + (-1)^r\left(1 - z^{-1}\right)^{2r}}. \tag{3.4}$$

In addition, $U_d^{2r}(-z) = -U_d^{2r}(z)$.

This proposition provided us with a family of prediction filters $U_d^{2r}(z),\ r = 1, 2, \ldots$.

It is interesting to note that these filters are closely related to the discrete-time Butterworth filters [3, 16], which are commonly used in signal processing. These Butterworth filters have maximally flat frequency response. The magnitude squared transfer functions of the low-pass b_l^r and high-pass b_h^r half-band Butterworth filters of order r are:

$$|b_l^r(z)|^2 = \frac{\left(1 + z^{-1}\right)^{2r}}{\left(1 + z^{-1}\right)^{2r} + (-1)^r\left(1 - z^{-1}\right)^{2r}}, \quad |b_h^r(z)|^2 = \frac{(-1)^r\left(1 - z^{-1}\right)^{2r}}{\left(1 + z^{-1}\right)^{2r} + (-1)^r\left(1 - z^{-1}\right)^{2r}}.$$

It is readily seen that the function U defined in (3.4), is related to these transfer functions:

$$1 + U_d^{2r}(z) = 2|b_l^r(z)|^2, \quad 1 - U_d^{2r}(z) = 2|b_h^r(z)|^2.$$

Remark on implementation: Since the transfer function $zU_d^{2r}(z)$ depends on $\zeta \triangleq z^2$, we denote it as $zU_d^{2r}(z) = F_d^{2r}(\zeta)$. Therefore, in line with Eq. (3.4) we have $\sigma(\zeta) = F_d^{2r}(\zeta)e(\zeta)$. Hence, we see that, in order to get the prediction for the odd sub-array $\mathbf{o_1}$ of a signal $\mathbf{x}$, we should convolve the even subarray $\mathbf{e_1}$ with a filter with the transfer function F_d^{2r}. This transfer function is a rational function of its argument (except for the case $r = 1$) and, consequently, the filter has infinite impulse response (IIR). But in [3] we derived explicit formulas, which enable fast implementation of these filters of any order via a cascade of elementary causal and anti-causal recursive filters. A parallel implementation is also possible.

Two examples of filters

Example 1, $r = 1$**:** This is the simplest case. We have

$$U_d^2(z) = \frac{z^{-1}+z}{2}, \; 1 - U_d^2(z) = -\frac{z^{-1}-2+z}{2}, \; F_d^2(\zeta) = (1+\zeta)/2. \tag{3.5}$$

The filter F_d^2 is FIR. Filtering $\mathbf{t} = F_d^2\mathbf{y}$ is implemented as

$$t(k) = \frac{1}{2}(y(k) + y(k+1)). \tag{3.6}$$

Due to Proposition 2.3 , the high-frequency analysis wavelet $\tilde{\psi}^1$ has two vanishing moments.

Example 2, $r = 3$**:** We have

$$U_d^6(z) = \frac{(1+z^{-1})^6 + (1-z^{-1})^6}{(1+z^{-1})^6 - (1-z^{-1})^6} \quad 1 - U_d^6(z) = -\frac{(z^{-1}-2+z)^3}{6z^2+20+6z^{-2}}$$

$$F_d^6(\zeta) = zU_d^3(z) = \frac{1}{18}\frac{(\zeta+14+\zeta^{-1})(1+\zeta)}{(1+\zeta^{-1}/3)(1+\zeta/3)}. \tag{3.7}$$

The high-frequency analysis wavelet $\tilde{\psi}^1$ has 6 vanishing moments. Filtering $\mathbf{t} = F_d^6\mathbf{y}$ can be implemented via the following cascade of operations:

$$y_1(k) = \frac{1}{18}(y(k) + y(k+1)), \;\; y_2(k) = y_1(k-1) + 14y_1(k) + y_1(k+1), \tag{3.8}$$

$$y_3(k) = y_2(k) - \frac{1}{3}y_3(k-1)), \;\; t(k) = y_3(k) - \frac{1}{3}t(k+1). \tag{3.9}$$

From (3.9) we can see that the cascade comprises the causal and anti-causal elementary recursive blocks with rational coefficients. Another option is to have a parallel implementation. This option stems from the following decomposition of the function $F_d^6(z)$:

$$F_d^6(\zeta) = \frac{1}{54}\left[-\frac{72}{1+\zeta/3} - \frac{8\zeta^{-1}}{1+\zeta^{-1}/3} + 9(\zeta+105)\right].$$

Then, filtering $\mathbf{t} = F_d^6\mathbf{y}$ is implemented as

$$t_1(k) = 9(105y(k) + y(k+1)), \;\; t_2(k) = y(k-1) - \frac{1}{3}t_2(k-1) \tag{3.10}$$

$$t_3(k) = y(k) - \frac{1}{3}t_3(k+1)), \;\; t(k) = \frac{1}{54}\left[t_1(k) - 8t_2(k) - 72t_3(k)\right].$$

Remark. The application of recursive filters requires initialization of the filter. Namely, if the input samples $\{y(k)\}$ are available only for $k = 1, \ldots, K$ then in order to process the signal with the causal filter $b/(1+az^{-1})$ as $t(k) = by(k) - at(k-1)$, we have to know the initial value $t(0)$. To implement the anti-causal filter $b/(1+az)$ as $t(k) = by(k) - at(k+1)$, the value $t(K+1)$ is required. This initialization problem is discussed in Appendix I.

3.2 Polynomial splines

In this section we deal with polynomial splines. We discuss some properties of polynomial splines and derive the corresponding filters U.

3.2.1 Interpolatory splines

The central $B-$spline of first order on the grid $\{kh\}$ is defined as follows:

$$M_1^h(x) \triangleq \begin{cases} 1/h & \text{if } x \in [-h/2, h/2] \\ 0 & \text{elsewhere.} \end{cases}$$

The central $B-$spline of order p is the convolution $M_h^p(x) = M_h^{p-1}(x) * M_h^1(x) \quad p \geq 2$. Note that the B-spline of order p is supported on the interval $(-ph/2, ph/2)$. It is positive within its support and symmetric around zero. The nodes of B-splines of even orders are located at points $\{kh\}$ and of odd orders at points $\{h(k+1/2)\}$, $k \in \mathbb{Z}$. It is readily verified that $hM_h^p(hx) = M^p(x), \quad$ where $M^p(x) \triangleq M_1^p(x)$. Denote

$$\mathbf{u}^p \triangleq \{hM_h^p(hk) = M^p(k)\}, \quad \mathbf{w}^p \triangleq \left\{hM_h^p\left(h(k+\frac{1}{2})\right) = M^p\left(k+\frac{1}{2}\right)\right\}, \; k \in \mathbb{Z}. \tag{3.11}$$

Due to the compact support of B-splines, these sequences are finite. We will use for our constructions only splines of odd orders $p = 2r-1$. In Table 1 we present the sequences $\mathbf{u}^p$ and $\mathbf{w}^p$ for some values of r.

	-4	-3	-2	-1	0	1	2	3	4
$\mathbf{u}^3 \times 8$	0	0	0	1	6	1	0	0	0
$\mathbf{u}^5 \times 384$	0	0	1	76	230	76	1	0	0
$\mathbf{u}^7 \times 46080$	0	1	722	10543	23548	10543	722	1	0
$\mathbf{w}^3 \times 2$	0	0	0	1	1	0	0	0	0
$\mathbf{w}^5 \times 24$	0	0	1	11	11	1	0	0	0
$\mathbf{w}^7 \times 720$	0	1	57	302	302	57	1	0	0

Table 1: Values of the sequences $\mathbf{u}^p$ and $\mathbf{w}^p$.

We need the z^2–transforms of the sequences $\mathbf{u}^p$ and $\mathbf{w}^p$:

$$u^p(z^2) \triangleq \sum_{k=-\infty}^{\infty} z^{-2k} u^p(k), \quad w^p(z^2) \triangleq \sum_{k=-\infty}^{\infty} z^{-2k} w^p(k).$$

These functions are Laurent polynomials. They are called the Euler-Frobenius polynomials [22]. Any spline S_h^p of order p has the following representation:

$$S_h^p(x) = h \sum_l q(l)\, M_h^p(x - lh). \tag{3.12}$$

Similarly to the previous section 2.1, we use for prediction of the odd samples the values in the midpoints of the splines, which interpolate the even samples. Such a construction is described in [5, 6]. In this case we get for the spline of order p the following prediction filter:

$$U_i^p(z) \triangleq z^{-1} \frac{w^p(z^2)}{u^p(z^2)}. \tag{3.13}$$

Proposition 3.2 *[5] The rational functions $U_i^p(z)$ are real-valued and $U_i^p(-z) = -U_i^p(z)$. If $p = 2r-1$ is odd then*

$$1 - U_i^p(z) = \frac{(z - 2 + z^{-1})^r \xi_r(\alpha)}{u^p(z^2)}. \tag{3.14}$$

where $\alpha \triangleq z + z^{-1}$ and $\xi_r(\alpha)$ is a polynomial of degree $r - 1$.

The property (3.14) results in $2r$ vanishing moments for the high-frequency analysis wavelet $\tilde{\psi}^1$.

Since polynomial splines of order p are piecewise polynomials of degree $p - 1$, their maximal order of approximation is p. It means that the splines can restore polynomials of a degree not exceeding $p - 1$. This maximal order of approximation is achieved while the interpolatory splines are employed. But the interpolatory splines of odd orders $2r + 1$ possess a property of super-convergence in the midpoints between the nodes ([29]). It means that the spline of order $2r + 1$ (degree $2r$), which interpolates samples of a polynomial of degree $2r + 1$ on the grid $2k$, coincides with this polynomial in the points $2k + 1$. This property is valuable for the use of these splines as a source for the construction of prediction filters. Therefore, we use only splines of odd orders $p = 2r + 1$.

Examples

Example 1, $r = 1, p = 3$ **(quadratic interpolatory spline):**

$$U_i^3(z) = \frac{4(z + z^{-1})}{z^2 + 6 + z^{-2}},\ 1 - U_i^3(z) = \frac{(z - 2 + z^{-1})^2}{z^{-2} + 6 + z^2}, \tag{3.15}$$
$$F_i^3(\zeta) = 4\gamma_3 \frac{1 + \zeta}{(1 + \gamma_3 \zeta)(1 + \gamma_3 \zeta^{-1})}$$

where $\gamma_3 = 3 - 2\sqrt{2} \approx 0.172$. Filtering $\mathbf{t} = F_i^3 \mathbf{y}$ can be implemented via the following cascade of operations:

$$y_0(k) = 4\gamma_3(y(k) + y(k+1)),\ y_1(k) = y_0(k) - \gamma_3 y_1(k-1),\ t(k) = y_1(k) - \gamma_3 t(k+1). \tag{3.16}$$

For the parallel implementation we use the following decomposition of the transfer function:

$$F_i^3(\zeta) = \frac{4\gamma_3}{1+\gamma_3}\left(\frac{1}{1+\gamma_3\zeta^{-1}} + \frac{\zeta}{1+\gamma_3\zeta}\right).$$

Then, filtering $\mathbf{t} = F_i^3\mathbf{y}$ is implemented in a parallel mode:

$$t_1(k) = y(k) - \gamma_3 t_1(k-1), \quad t_2(k) = y(k+1) - \gamma_3 t_2(k+1), \quad t = \frac{4\gamma_3}{1+\gamma_3}(t_1+t_2). \tag{3.17}$$

Remark: This filter F_i^3 coincides with the filter F_d^4, which was derived from a cubic discrete spline.

Example 2, $r = 2, p = 5$ (splines of fourth degree):

$$U_i^5(z) = \frac{16(z^3+11z+11z^{-1}+z^{-3})}{z^4+76z^2+230+76z^{-2}+z^{-4}}, \; 1-U_i^2(z) = \frac{(z-2+z^{-1})^3\,(z+z^{-1}-10)}{z^4+76z^2+230+76z^{-2}+z^{-4}},$$

$$F_i^5(\zeta) = \frac{16\gamma_5^1\gamma_5^2(\zeta+10+\zeta^{-1})(1+\zeta)}{(1+\gamma_5^1\zeta)(1+\gamma_5^1\zeta^{-1})(1+\gamma_5^2\zeta)(1+\gamma_5^2\zeta^{-1})}, \tag{3.18}$$

where $\gamma_5^1 = 19-4\sqrt{19}-2\sqrt{166-38\sqrt{19}} \approx 0.361$, $\gamma_5^2 = 19+4\sqrt{19}-2\sqrt{166+38\sqrt{19}} \approx 0.0137$.

3.2.2 Local quasi-interpolatory splines

The valuable property of super-convergence in the midpoints is inherent not only to interpolatory splines but also to the so called *local quasi-interpolatory splines of odd orders*, which can be constructed using finite impulse response (FIR) filtering.

Definition 3.2 *Let a function f be sampled on the grid $\{kh)\}$ and $\mathbf{f} \triangleq \{f_k = f(hk)\}$, $k \in \mathbb{Z}$. A spline S_h^p of order p given by (3.12) is said to be local if the array $\mathbf{q}$ of its coefficients is derived by FIR filtering the array $\mathbf{f}$*

$$\mathbf{q}(z) = \Gamma(z)\mathbf{f}(z) \tag{3.19}$$

where $\Gamma(z)$ is a Laurent polynomial. The local spline of order p is called quasi-interpolatory if it restores polynomials of degree $p-1$. It means that, provided f is a polynomial of degree $p-1$, the spline $S_h^p(x) \equiv f(x)$.

If $\mathbf{w}^p$ is the sequence defined in (3.11) then the values of the spline S_h^p in the midpoints between grid points $\mathbf{m}^p$ are produced by the following FIR filtering the array of samples $\mathbf{f}$

$$\mathbf{m}^{\mathbf{p}}(z^2) = zU_q^p(z)\mathbf{f}(z^2), \quad U_q^p(z) \triangleq z^{-1}\Gamma(z^2)w^p(z^2). \tag{3.20}$$

In [29] explicit formulas for construction of quasi-interpolatory splines were established as well as the estimations of the differences between the splines and approximated functions. In this work we are interested in splines of odd orders $p = 2r+1$. There are many FIR filters which generate quasi-interpolatory splines but there is only one filter of minimal length $2r+1$ for each order $p = 2r+1$. Denote $\lambda(z) \stackrel{\Delta}{=} z^{-1} - 2 + z$.

Theorem 3.1 *[29] A quasi-interpolatory spline of order $p = 2r+1$ can be constructed by filtering (3.19) with filters Γ of length no less than $2r+1$. There exists a unique filter Γ_m^r of length $2r+1$, which produces the minimal quasi-interpolatory spline $\tilde{S}_h^{2r+1}(x)$. Its transfer function is:*

$$\Gamma_m^r(z) = 1 + \sum_{k=1}^{r} \beta_k^r \lambda^k(z).$$

The coefficients β_k^r can be derived from the generating function

$$\left(\frac{2\arcsin t/2}{t}\right)^{2r+1} = \sum_{k=0}^{\infty} (-1)^k \beta_k^r t^{2k}.$$

If the function f has $2r+3$ derivatives then the following asymptotic relations hold for the minimal quasi-interpolatory spline $\tilde{S}_h^{2r+1}(x)$ for $x = h(k+1/2+\tau)$, $\tau \in [0,1]$:

$$\tilde{S}_h^{2r+1}(x) = f(x) - h^{2r+1} f^{(2r+1)}(x) \frac{b_{2r+1}(\tau)}{(2r+1)!} + h^{2r+2} f^{(2r+2)}(x) \left(\frac{(2r+1)b_{2r+2}(\tau)}{(2r+2)!} - \beta_{r+1}^r\right)$$
$$+O(f^{(2r+3)}h^{2r+3}),$$

where $b_s(x)$ is the Bernoulli polynomial of degree s.

We recall that the values $b_s \stackrel{\Delta}{=} b_s(0)$ are called the Bernoulli numbers. If $s > 1$ is odd then $b_s = 0$. Hence, values of the minimal quasi-interpolatory spline in the midpoints between nodes are evaluated as follows:

$$\tilde{S}_h^{2r+1}(h(k+\frac{1}{2})) = f(h(k+\frac{1}{2})) + h^{2r+2} f^{(2r+2)}(h(k+\frac{1}{2}))A^r + O(f^{(2r+3)}h^{2r+3}), \qquad (3.21)$$
$$A^r \stackrel{\Delta}{=} \frac{(2r+1)b_{2r+2}}{(2r+2)!} - \beta_{r+1}^r.$$

Equation (3.21) implies the super-convergence property. This property is similar to the property of interpolatory splines. The minimal quasi-interpolatory spline of order $2r+1$ (degree $2r+1$) restores polynomials of degree $2r+1$ in the midpoints between nodes. The asymptotic representation (3.21) allows to increase the approximation accuracy of the minimal spline in the midpoints. This is achieved via an upgrade the filter Γ_m^r.

Proposition 3.3 *If the coefficients of a spline $\check{S}_h^{2r+1}$ of order $2r+1$ are derived according to Eq. (3.19) using the filter Γ_e^r of length $2r+3$ with the transfer function $\Gamma_e^r(z) = \Gamma_m^r(z) - A^r\lambda^{r+1}(z)$ then the spline restores polynomials of degree $2r+3$ in the midpoints between nodes.*

3.2.3 Examples

Examples for Quadratic splines

Example 1, Minimal spline: The filters are

$$\begin{aligned}
&\Gamma^3_m(z^2) = 1-\frac{1}{8}\lambda(z^2),\ U^3_{qm}(z) = \frac{-z^{-3}+9z^{-1}+9z-z^3}{16}, \\
&1-U^3_{qm}(z) = \frac{(\alpha-2)^2(z^{-1}+4+z)}{16}, \quad F^3_{qm}(\zeta) = \frac{-\zeta^{-1}+9+9\zeta-\zeta^2}{16},
\end{aligned} \tag{3.22}$$

where $\alpha = z+z^{-1}$.

Example 2, Extended spline:

$$\begin{aligned}
&\Gamma^3_e(z) = \Gamma^3_m(z) + \frac{3}{128}\lambda^2(z),\ U^3_{qe}(z) = \frac{3z^{-5}-25z^{-3}+150z^{-1}+150z-25z^3+3z^5}{256}, \\
&1-U^3_{qe}(z) = \frac{(\alpha-2)^3(3z^{-2}+18z^{-1}+38+18z+3z^2)}{256}, \\
&F^3_{qe}(\zeta) = \frac{3\zeta^{-2}-25\zeta^{-1}+150+150\zeta-25\zeta^2+3\zeta^3}{256}.
\end{aligned} \tag{3.23}$$

Remark: Donoho [12] presented a scheme where an odd sample is predicted by the value in the central point of the polynomial of an odd degree which interpolates adjacent even samples. One can observe that our filter U^3_{qm} (3.22) coincides with the filter derived by Donoho's scheme using cubic interpolatory polynomial. The filter U^3_{qe} (3.23) coincides with the filter derived using interpolatory polynomial of fifth degree.

Minimal spline of fifth order (fourth degree)

$$\begin{aligned}
&\Gamma^5_{qm}(z) = 1-\frac{5}{24}\lambda(z)+\frac{47}{1152}\lambda^2(z), \\
&U^5_{qm}(z) = \frac{47(z^{-7}+z^7)+89(z^{-5}+z^5)-2277(z^{-3}+z^3)+15965\alpha}{27648} \\
&1-U^5_{qm}(z) = \frac{-(\alpha-2)^3\left(47(z^{-4}+z^4)+282(z^{-3}+z^3)+1076(z^{-2}+z^2)+3166\alpha+5414\right)}{27648} \\
&F^5_{qm}(\zeta) = \frac{47(\zeta^{-3}+\zeta^4)+89(\zeta^{-2}+\zeta^3)-2277(\zeta^{-1}+\zeta^2)+15965(1+\zeta)}{27648}.
\end{aligned}$$

3.3 Filters for the *update* step

In sections 2.1 and 3.2 we presented a family of filters U for the *predict* step, which was derived from splines of various types. To complete the construction of the transform we need to define the filter V for update part. There is one limitation in the choice of these filters: we have to guarantee $V(-z) = V(z)$. This is needed to have perfect reconstruction of the transform. Hence, the function $\Phi(\zeta) \triangleq V(z)/z,\ \zeta = z^2$ depends on z^2. In order to make synthesis and analysis filters similar in their properties, we choose $V(z) = \breve{U}(z)$, where $\breve{U}$ is one of the filters U presented above. It can be readily verified that in this case $\Phi(\zeta) = \breve{F}(\zeta^{-1})$. In particular, $\breve{U}$ coincided with the filters U which were used for the prediction and $\Phi(\zeta) = F(\zeta^{-1})$.

Proposition 3.4 *Assume that $U(z)$ and $V(z) = \breve{U}(z)$ are filters that were used for* predict *and* update *steps, respectively. If $1 - U(z)$ comprises the factor $(z - 2 + 1/z)^r$ then the high-frequency analysis wavelets $\tilde{\psi}^1$ have $2r$ vanishing moments. If, in addition, $1 - \breve{U}(z)$ comprises the factor $(z - 2 + 1/z)^l$ then the synthesis wavelet ψ^1 has $2q$ vanishing moments, where $q = \min\{l, r\}$.*

4 Transforms for image compression

The above constructed filters were used by the wavelet transforms for compression of multimedia images.

4.1 The employed wavelet transforms

We explored various combinations of filters for *prediction* and *update* steps in the lifting methodology to obtain different transforms for image compression. We describe here several transforms, which we used in the experiments. The four filters were used:

$$
\begin{aligned}
f_1(\zeta) &\triangleq F_i^3(\zeta) = 4\gamma_3 \frac{1+\zeta}{(1+\gamma_3\zeta)(1+\gamma_3\zeta^{-1})} \quad \text{(see (3.15))} \\
f_2(\zeta) &\triangleq F_{qm}^3(\zeta) = \frac{-\zeta^{-1} + 9 + 9\zeta - \zeta^2}{16}, \quad \text{(see (3.22))} \\
f_3(\zeta) &\triangleq F_d^6(\zeta) = \frac{1}{18} \frac{(\zeta + 14 + \zeta^{-1})(1+\zeta)}{(1+\zeta^{-1}/3)(1+\zeta/3)}, \quad \text{(see (3.7))} \\
f_4(\zeta) &\triangleq F_{qe}^3(\zeta) = \frac{3\zeta^{-2} - 25\zeta^{-1} + 150 + 150\zeta - 25\zeta^2 + 3\zeta^3}{256}, \quad \text{(see (3.23))}.
\end{aligned}
$$

Remark: Performance of the transforms using the filter F_i^5 is almost identical to performance of the transforms using the filter F_d^6 but the computational complexity of the former is higher.
By combining the above filters we constructed the following transforms:

T_1^1- The transform where for the *prediction* and *update* steps the IIR filter f_1 was used, namely

$$
o_1^{\nu}(\zeta) = o_1(\zeta) - f_1(\zeta)e_1(\zeta), \quad e_1^{\nu}(\zeta) = e_1(\zeta) + \frac{1}{2} f_1(\zeta^{-1})e_1(\zeta). \tag{4.24}
$$

We display in Figure 1 the synthesis and analysis wavelets related to the transform T_1^1 and their Fourier spectra. We observe that the analysis wavelets are regular and, to some extent, are similar to the synthesis ones. Both the analysis and synthesis high-frequency wavelets have 4 vanishing moments.

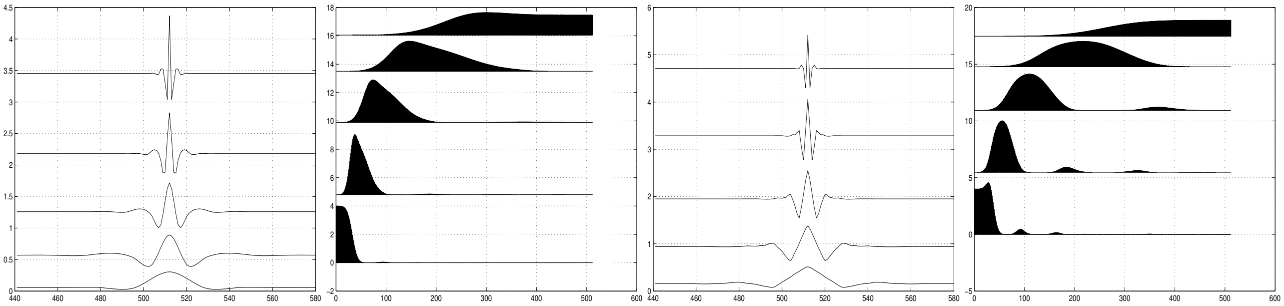

Figure 1: The left pair bottom to top: The synthesis wavelets generated by the transform T_1^1: φ^4 and $\psi^k,\ k=4,3,2,1$ and their Fourier spectra. The right pair bottom to top: The analysis wavelets $\tilde{\varphi}^4$ and $\psi^k,\ k=4,3,2,1$ and their Fourier spectra.

T_2^2- The transform where for the *prediction* and *update* steps the FIR filter f_2 was used. We display in Figure 2 the compactly supported synthesis and analysis wavelets related to the transform T_2^2 and their Fourier spectra. We observe that the synthesis wavelets are more regular than the analysis wavelets and are similar to the synthesis wavelets in Figure 1. Both the analysis and synthesis high-frequency wavelets have 4 vanishing moments.

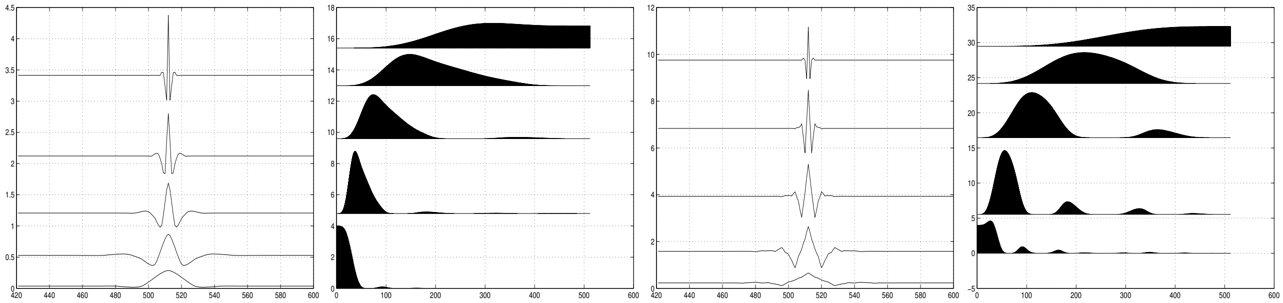

Figure 2: The left pair bottom to top: The synthesis wavelets generated by the transform T_2^2: φ^4 and $\psi^k,\ k=4,3,2,1$ and their Fourier spectra. The right pair bottom to top: The analysis wavelets $\tilde{\varphi}^4$ and $\psi^k,\ k=4,3,2,1$ and their Fourier spectra.

T_3^3- The transform where for the *prediction* and *update* steps the IIR filter f_3 was used. We display in Figure 3 the synthesis and analysis wavelets related to the transform T_3^3 and their Fourier spectra. Both the analysis and synthesis high-frequency wavelets have 6 vanishing moments and they are similar to each other. The wavelets are more regular than former examples. The spectra of the wavelets are well localized in the frequency domain.

T_4^4- The transform where for the *prediction* and *update* steps the FIR filter f_4 was used. We display in Figure 4 the compactly supported synthesis and analysis wavelets related to the transform T_4^4

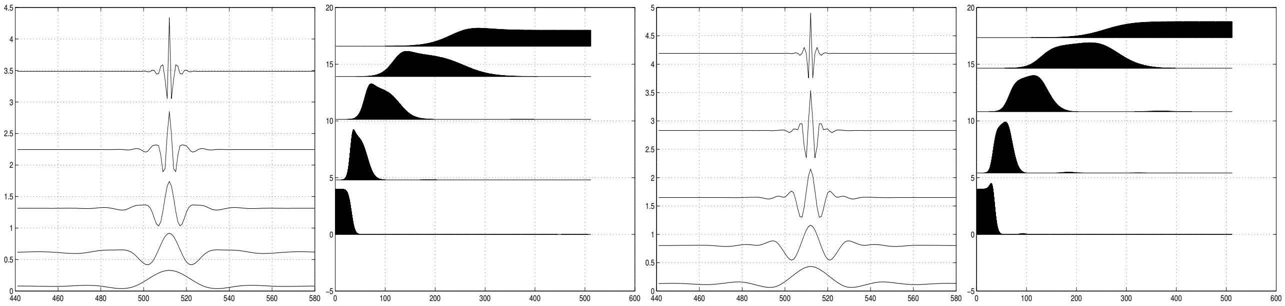

Figure 3: The left pair bottom to top: The synthesis wavelets generated by the transform T_3^3: φ^4 and ψ^k, $k = 4, 3, 2, 1$ and their Fourier spectra. The right pair bottom to top: The analysis wavelets $\tilde{\varphi}^4$ and ψ^k, $k = 4, 3, 2, 1$ and their Fourier spectra.

and their Fourier spectra. One can observe that the synthesis wavelets are more regular than the analysis wavelets and are similar to the synthesis wavelets in Figure 3. Like the transform T_3^3, both the analysis and synthesis high-frequency wavelets have 6 vanishing moments.

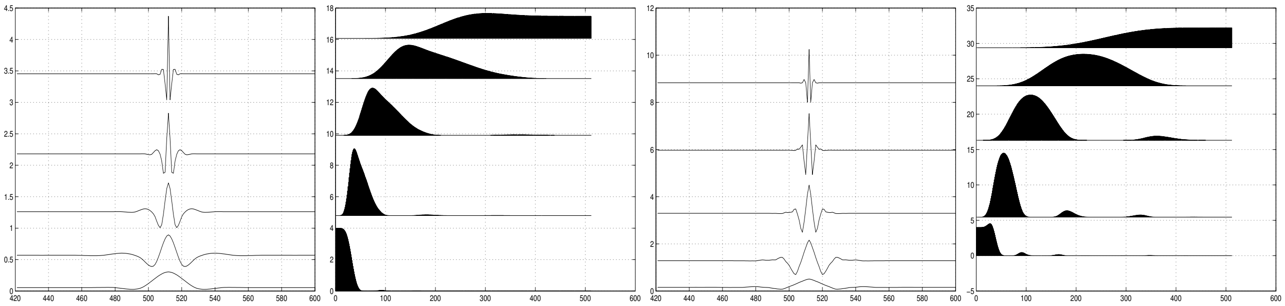

Figure 4: The left pair bottom to top: The synthesis wavelets generated by the transform T_4^4: φ^4 and ψ^k, $k = 4, 3, 2, 1$ and their Fourier spectra. The right pair bottom to top: The analysis wavelets $\tilde{\varphi}^4$ and ψ^k, $k = 4, 3, 2, 1$ and their Fourier spectra.

T_3^1- The transform where for the *prediction* step the filter f_1 was used, whereas the *update* step employed the more complicated filter f_3. We display in Figure 5 the synthesis and analysis wavelets related to the transform T_3^1 and their Fourier spectra. The analysis high-frequency wavelets have 4 vanishing moments, whereas the synthesis wavelets have 6 vanishing moments.

T_4^2- The transform where for the *prediction* step the FIR filter f_2 was used, whereas the *update* step employed the more complicated FIR filter f_4. We display in Figure 6 the synthesis and analysis wavelets related to the transform T_4^2 and their Fourier spectra. The analysis high-frequency wavelets have 4 vanishing moments, whereas the synthesis wavelets have 6 vanishing moments. The transform is, to some extent, similar to the transform T_3^1.

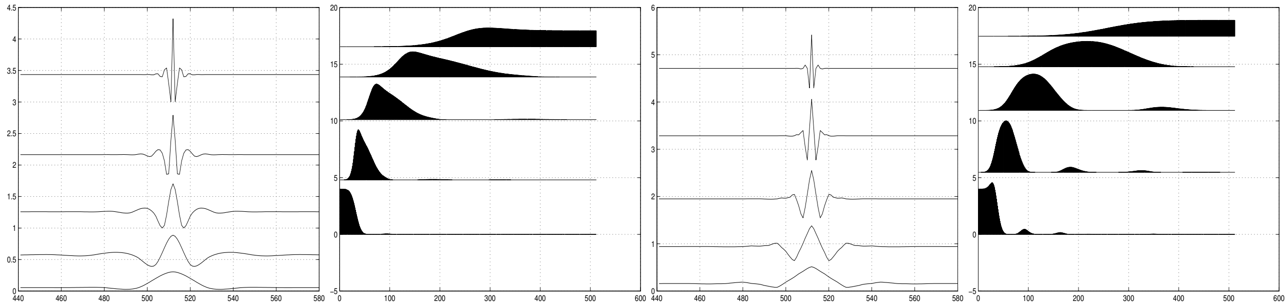

Figure 5: The left pair bottom to top: The synthesis wavelets generated by the transform T_3^1: φ^4 and ψ^k, $k = 4,3,2,1$ and their Fourier spectra. The right pair bottom to top: The analysis wavelets $\tilde{\varphi}^4$ and ψ^k, $k = 4,3,2,1$ and their Fourier spectra.

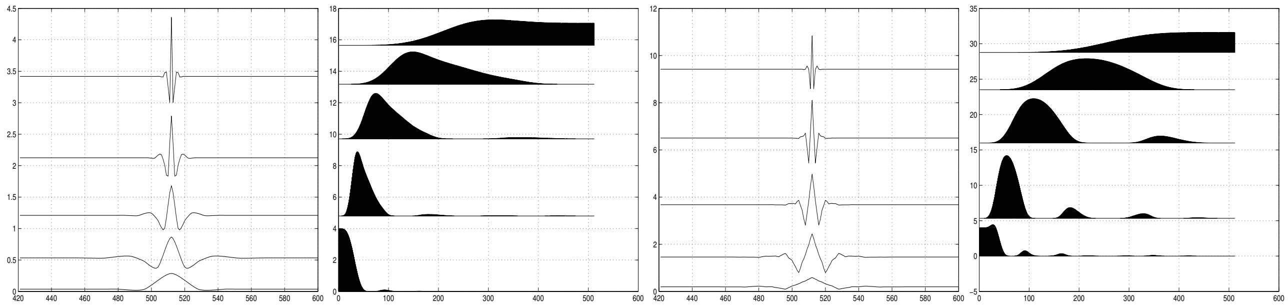

Figure 6: The left pair bottom to top: The synthesis wavelets generated by the transform T_4^2: φ^4 and ψ^k, $k = 4,3,2,1$ and their Fourier spectra. The right pair bottom to top: The analysis wavelets $\tilde{\varphi}^4$ and ψ^k, $k = 4,3,2,1$ and their Fourier spectra.

T_3^4- The transform where for the *prediction* step the FIR filter f_4 was used, whereas the *update* step employed IIR filter f_3. We display in Figure 7 the synthesis and analysis wavelets related to the transform T_3^4 and their Fourier spectra. The analysis and synthesis high-frequency wavelets have 6 vanishing moments. The transform is, to some extent, similar to the transform T_3^3.

We compared the performances of the above transforms and the transform with the B9/7 biorthogonal filter [1], which is very popular in wavelet based image compression systems. Figure 8 displays the synthesis and analysis wavelets related to the transform B9/7 and their Fourier spectra. The analysis and synthesis of high-frequency wavelets have 4 vanishing moment.

4.2 Computational complexity of the transforms

The wavelet transforms, which we presented in section 4.1, were implemented by filtering of even and odd subarrays with the filters f_r, $r = 1,2,3,4$. The filters f_2 anf f_4 are FIR. On the other hand, f_1

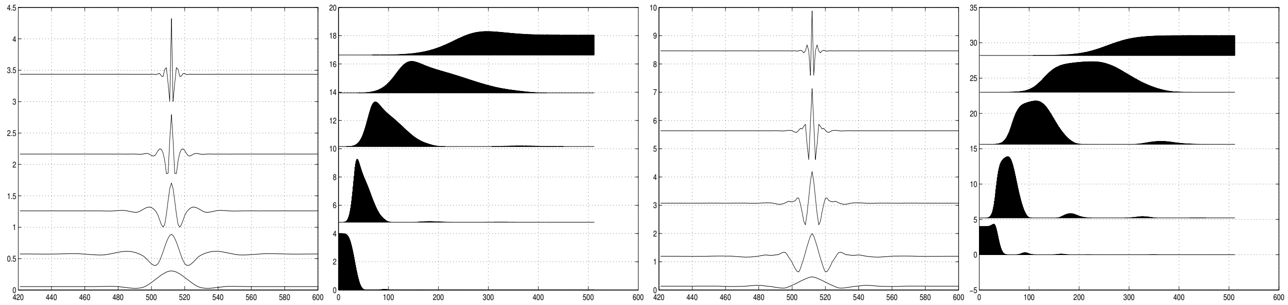

Figure 7: The left pair bottom to top: The synthesis wavelets generated by the transform T_3^4: φ^4 and $\psi^k,\ k = 4,3,2,1$ and their Fourier spectra. The right pair bottom to top: The analysis wavelets $\tilde{\varphi}^4$ and $\psi^k,\ k = 4,3,2,1$ and their Fourier spectra.

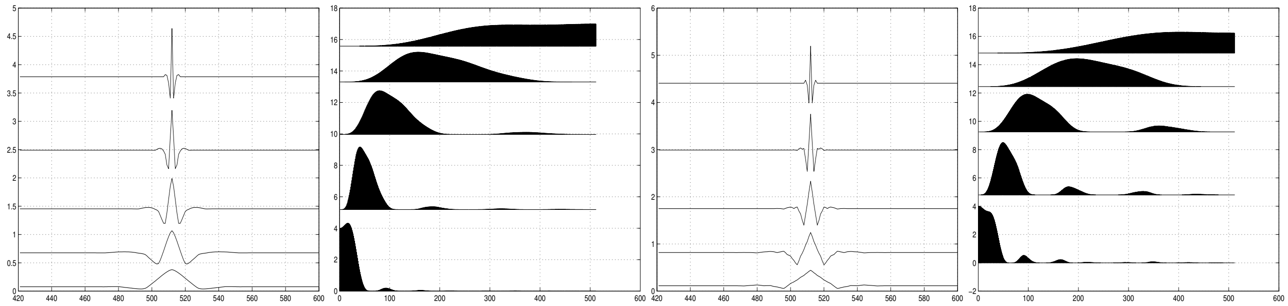

Figure 8: The left pair bottom to top: The synthesis wavelets generated by the transform B9/7: φ^4 and $\psi^k,\ k = 4,3,2,1$ and their Fourier spectra. The right pair bottom to top: The analysis wavelets $\tilde{\varphi}^4$ and $\psi^k,\ k = 4,3,2,1$ and their Fourier spectra.

and f_3 are IIR and have rational transfer functions. These IIR filters can be implemented in either cascade or parallel modes.

B9/7 filter: This transform can be implemented in a fast way using factorization of the filters which was suggested in [2, 11]. Briefly, one step of the decomposition (without normalization) comprises the following operations on even $e_1(k)$ and odd $o_1(k)$ subarrays of the signal $x(k)$ of length N:

$$1.\ o_1(k) = o_1(k) + (e_1(k) + e_1(k+1))\alpha_1, \quad 2.\ e_1(k) = e_1(k) + (o_1(k) + o_1(k-1))\alpha_2,$$
$$3.\ o_1^\nu(k) = o_1(k) + (e_1(k) + e_1(k+1))\alpha_3, \quad 4.\ e_1^\nu(k) = e_1(k) + (o_1^\nu(k) + o_1^\nu(k-1))\alpha_4,$$

$$\alpha_1 = -1.586134342,\ \alpha_2 = -0.05298011854,\ a_3 = 0.8829110762,\ a_4 = 1.149604398. \qquad (4.25)$$

Therefore, one step of the 2-D decomposition of an image of size N^2 pixels requires $8A + 4M$ (8 additions and 4 multiplications) operations per pixel.

T_l^r **transforms:** These transforms are implemented by filtering followed by addition:

$$o_1^\nu(k) = o_1(k) - (f_r e_1)(k+1), \; e_1^\nu(k) = e_1(k) + (f_l o_1^\nu)(k-1).$$

The number of operations required by filtering of an array of length $N/2$ by the FIR filter f_2 is $(3A+2M)N/2$. For the FIR filter f_4 the respective number is $(5A+3M)N/2$. Note that in these cases filtering can be implemented with rational numbers whose denominators are powers of two.

If the filter f_1 is implemented in a cascade mode using Eq. (3.16) then the number of operations is $(3A+3M)N/2$. However, with two processors the filtering can be implemented in a parallel using Eq. (3.17). Then, one processor performs only $(2A+2M)N/2$ operations and the other – $(A+M)N/2$.

Implementation of the filter f_3 in a cascade mode using (3.8)-(3.9) requires $(5A+4M)N/2$ operations. With three processors the parallel filtering implmentation using Eq. (3.9) is possible. Then load distribution among the processors is $(3A+3M)N/2$,$(A+2M)N/2$,$(A+M)N/2$ operations. Moreover, this filtering can be conducted with rational numbers.

Keeping in mind the above considerations we summarized in Table 2 the number of operations per pixel for the employed transforms. In the first line we give the number of operations for a cascade implementation. In the second line we give the maximal number of operations in a processor for the transforms using IIR filters in parallel computation.

Tr.	B9/7	T_1^1	T_2^2	T_3^3	T_3^1	T_4^4	T_4^2	T_3^4
Cas.	8A+4M	8A+6M	8A+4M	12A+8M	10A+7M	12A+6M	10A+5M	12A+7M
Par.	–	6A+4M	8A+3M	–	7A+5M	–	–	–

Table 2: The number of operations per pixel for the employed wavelet transforms. First line: cascade implementation. Second line details the number of operations in each processor in the parallel implementation.

Thus, we can see that even by a cascade computation the cost of the implementation of the transforms T_1^1, T_3^3 and T_3^1 is close to the cost of the application of the transform B9/7.

Note that two-dimensional wavelet transforms were applied separately on rows and columns of the 2D array. However, for lifting implementation of 2D transforms, which use FIR filters, it is possible to merge the row and column operations. This merging reduces computational cost of the implementation. Such a scheme was presented in [2]. For the B9/7 and T_2^2 transforms the number of operations per pixel in the 2D implementation is 8A+3M. In Appendix II we provide the formulas for the 2D implementation of the transform T_2^2.

5 Image compression results

The above transforms were applied on multimedia still images to achieve high quality compression. In most experiments they outperform the popular B9/7 biorthogonal transform. In this section we present the results after compression and decompression of four images in Figures 9 and 10. These are 512×512 8 bit per pixel (8bpp) images. The following experiments were done:

Figure 9: Left: "Lena", right: "Barbara".

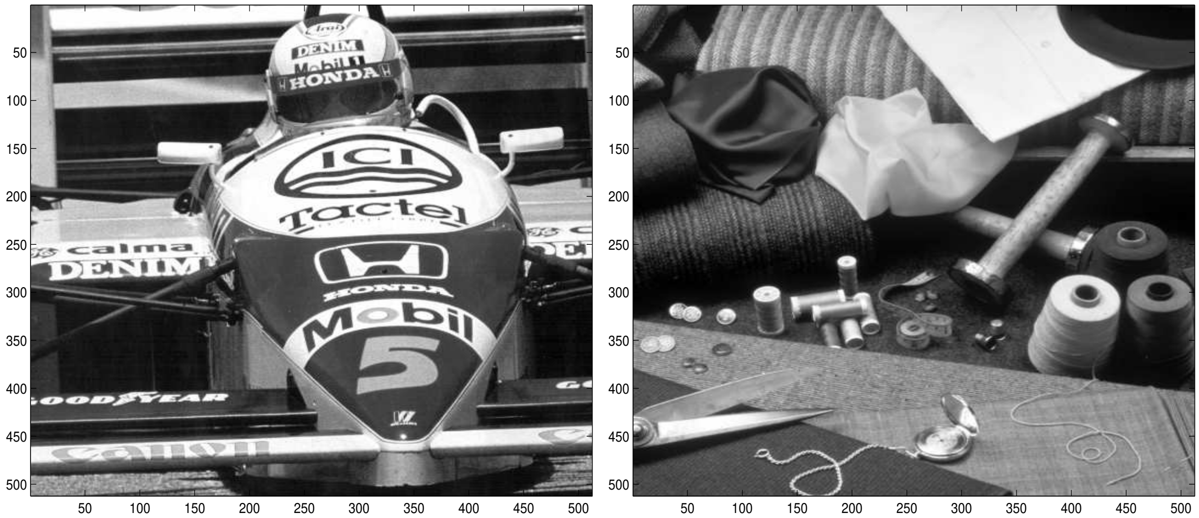

Figure 10: Left: "Car", right: "Fabrics".

1. The image was decomposed up to 6 scales with the wavelet transform using the B9/7 transforms and the transforms listed in Section 4.1.

2. The transform coefficients were coded using the SPIHT algorithm by Said and Pearlman [21]. This algorithm enables to achieve exact prederemined compression rate. We coded the coefficients with different compression ratios (CR) 1:10 (0.8 bpp), 1:20 (0.4 bpp), 1:30 (4/15 bpp), 1:40 (0.2 bpp) and 1:50 (4/25 bpp).

3. The reconstructed image was compared with the original image and the achieved peak signal to noise ratio (PSNR) in decibels was computed:

$$PSNR = 10 \log_{10} \left(\frac{N\, 255^2}{\sum_{k=1}^{N} (x(k) - \tilde{x}(k))^2} \right) dB. \tag{5.26}$$

Lena: The PSNR values of "Lena" are presented in Table 3. All the T_l^k transforms outperform the

CR	B9/7	T_1^1	T_2^2	T_3^3	T_4^4	T_3^1	T_4^2	T_3^4
10	37.70	37.82	37.67	37.84	37.79	37.85	37.71	37.84
20	34.53	34.71	34.70	34.72	34.68	34.75	34.63	34.73
30	32.33	32.62	32.56	32.70	32.59	32.65	32.57	32.67
40	31.42	31.63	31.52	31.66	31.61	31.67	31.54	31.68
50	30.70	30.91	30.82	30.92	30.90	30.95	30.85	30.96

Table 3: PSNR of the "Lena" after the application of SPIHT where the decomposition of the wavelet transform was into 6 scales.

B9/7 filter in any compression rate (with one exception for T_2^2 at CR=10). In Figure 11 we display the closed-up fragments of "Lena" reconstructed from 1:40 compression files of wavelet coefficients of the B9/7 and T_3^4 transforms.

Barbara: The PSNR values of "Barbara" are presented in Table 4. From Table 4 we see that the

CR	B9/7	T_1^1	T_2^2	T_3^3	T_4^4	T_3^1	T_4^2	T_3^4
10	33.01	33.39	32.91	33.72	33.32	33.51	33.06	33.46
20	28.93	29.13	28.90	29.32	29.10	29.24	28.99	29.22
30	26.99	27.03	26.91	27.32	27.17	27.15	27.04	27.15
40	25.78	25.74	25.65	25.94	25.74	25.83	25.69	25.78
50	25.10	25.00	24.86	25.13	24.95	25.05	24.90	25.02

Table 4: PSNR of "Barbara" after the application of SPIHT where the decomposition of the wavelet transform was into 6 scales.

transforms T_3^3 and T_3^1, where the synthesis wavelets have 6 vanishing moments, have better

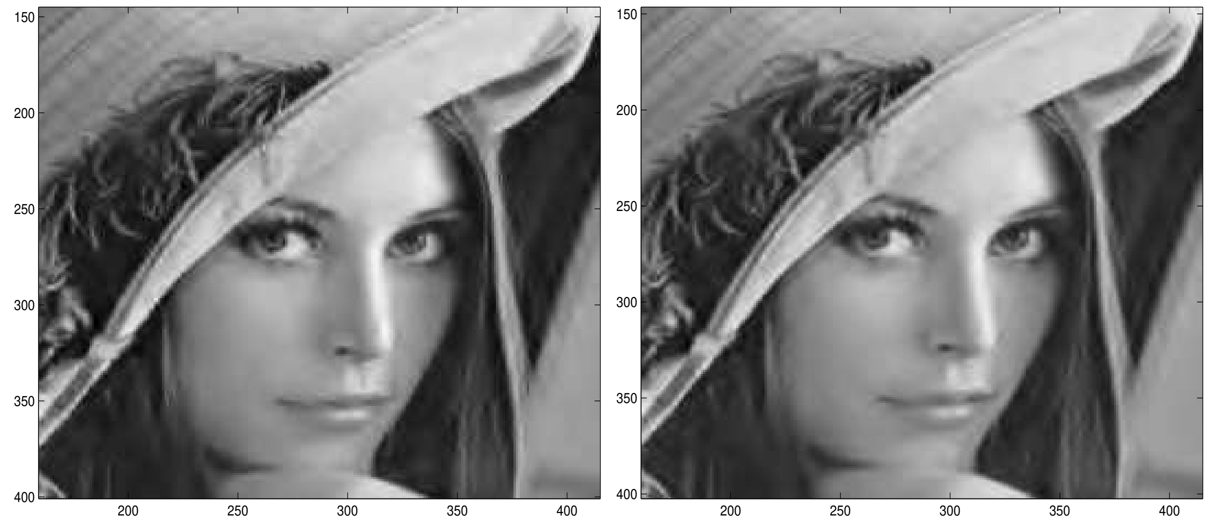

Figure 11: Fragment of "Lena" reconstructed from 40:1 compression. Left: the B7/9 filter was applied: PSNR=31.42. Right: T_3^4 transform: PSNR=31.68.

performance than transforms with 4 vanishing moments. The transform T_3^3 produces the best results amongst the presented examples. Note that even better results were obtained with a similar transform that has 8 vanishing moments. We display in Figure 12 a fragment of the reconstructed "Barbara" from the 1:40 compressed files of wavelet coefficients of the B9/7 and T_3^3 transforms.

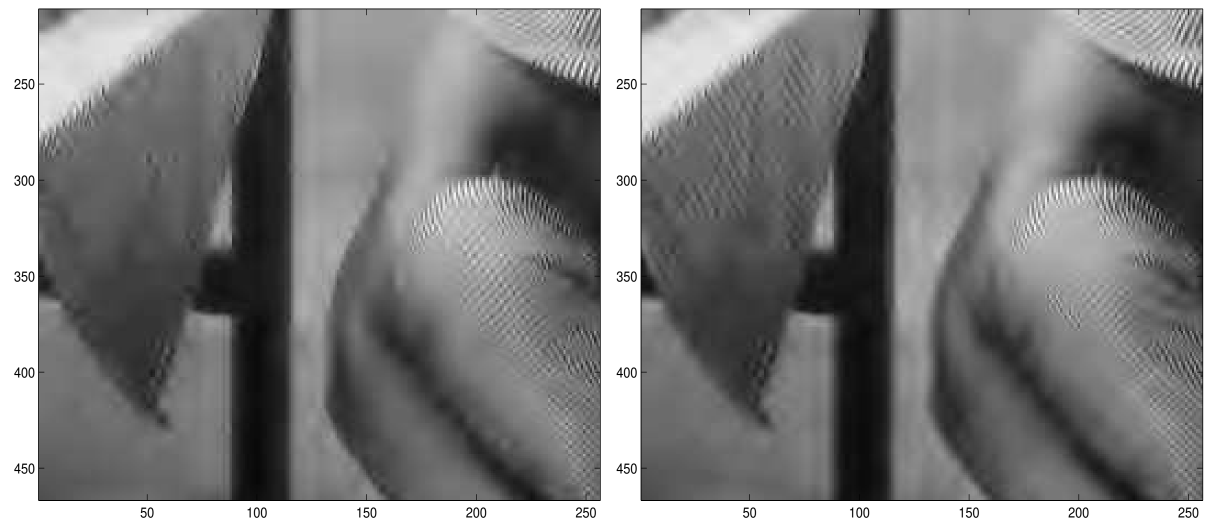

Figure 12: Fragment of "Barbara" reconstructed from 40:1 compression. Left: the B7/9 filter was used: PSNR=25.78. Right: T_3^3 transform: PSNR=25.95.

Car: The PSNR values of "Car" are presented in Table 5. Here, as for the image "Lena" the

CR	B9/7	T_1^1	T_2^2	T_3^3	T_4^4	T_3^1	T_4^2	T_3^4	
10	32.57	32.63	32.72	32.52	32.59	32.66	32.69	32.64	
20	28.38	28.53	28.46	28.46	28.47	28.52	28.49	28.50	
30	26.78	26.99	26.89	26.92	26.94	27.01	26.95	26.98	
40	25.05	25.19	25.11	25.15	25.17	25.21	25.16	25.19	
50	24.40	24.52	24.43	24.46	24.49	24.56	24.48	24.53	

Table 5: PSNR of the "Car" image after the application of SPIHT where the decomposition of the wavelet transform was into 6 scales.

transforms T_l^k outperform the B9/7 filter in any compression rate (with one exception for T_3^3, CR=10). The transforms T_1^1, T_3^1, T_3^4 are working similarly.

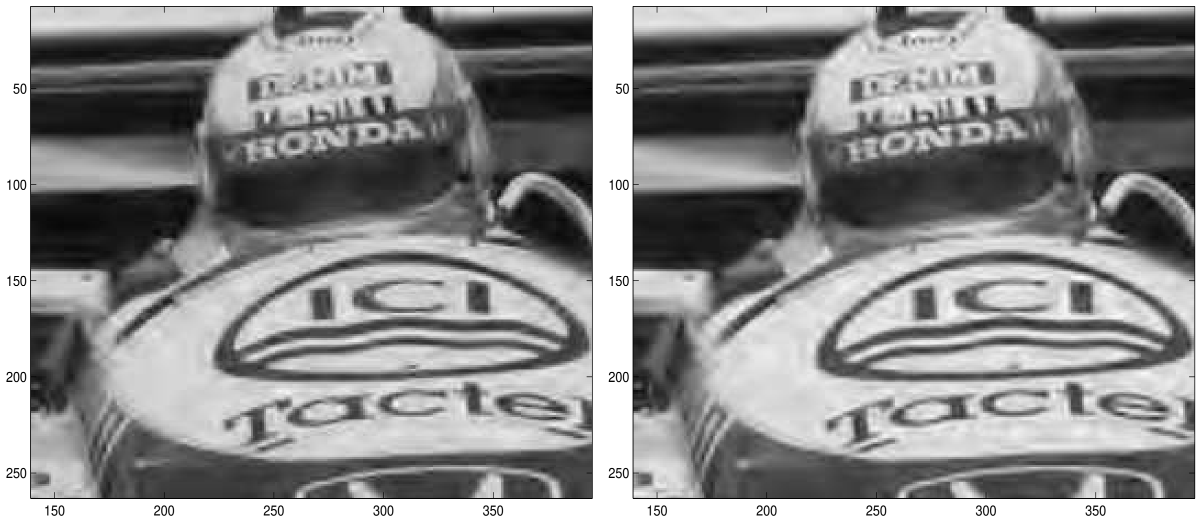

Figure 13: Fragment of "Car" reconstructed from 40:1 compression. Left: the B7/9 filter was applied:PSNR=25.05. Right: T_3^1 transform: PSNR=25.21.

Fabric: The PSNR values of "Fabric"are presented in Table 6. As it was for the above examples, the transform T_2^2 outperforms the B9/7 filter for any compression ratio. The same is true for more complicated transforms such as T_3^3 and T_3^2. The latter transform produces the best PSNR amongst all listed transforms. Again, PSNR values for the simplified transform T_1^2, are comparable to as those for the B9/7 transform. We display in Figure 14 a closed up fragmentof the image "Fabrics" reconstructed from the 1:40 compressed files of wavelet coefficients of the B9/7 and T_3^4 transforms.

CR	B9/7	T_1^1	T_2^2	T_3^3	T_4^4	T_3^1	T_4^2	T_3^4	
10	34.95	34.96	34.86	34.97	34.96	34.99	34.89	35.02	
20	31.53	31.53	31.44	31.53	31.51	31.55	31.47	31.57	
30	29.62	29.79	29.73	29.75	29.77	29.81	29.77	29.83	
40	28.90	29.00	28.93	29.02	28.98	29.02	28.97	29.04	
50	28.41	28.52	28.44	28.52	28.50	28.54	28.46	28.54	

Table 6: PSNR of the "Fabric" image after the application of SPIHT where the decomposition of the wavelet transform was into 6 scales.

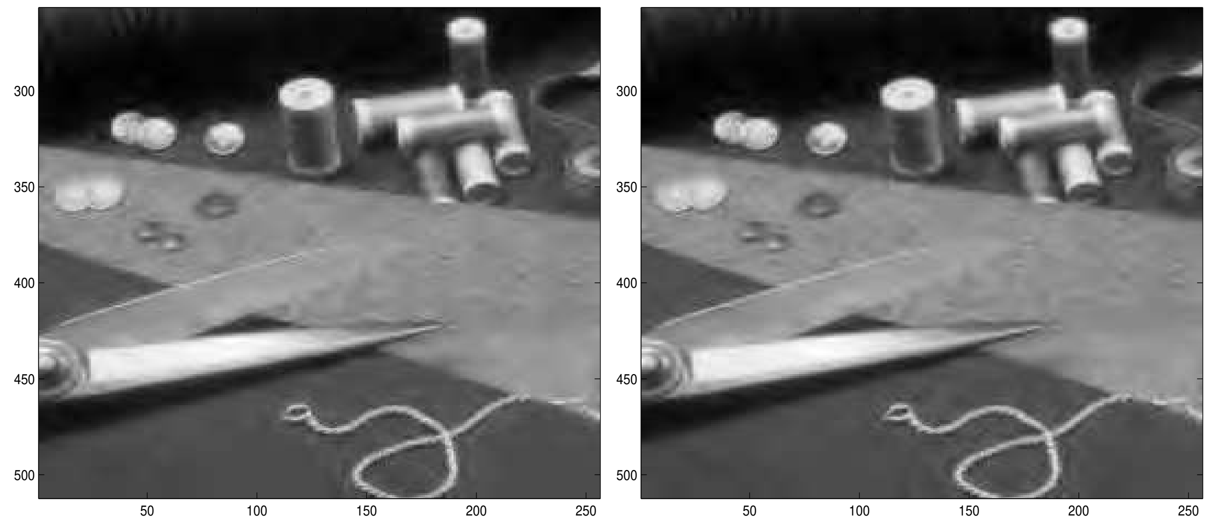

Figure 14: Fragment of "Fabric" reconstructed from 40:1 compression. Left: the B7/9 filter was applied:PSNR=28.90. Right: T_3^4 transform: PSNR=29.04.

6 Conclusions

In this paper we proposed an efficient technique that produces a diversity of new biorthogonal symmetric wavelet transforms. This technique is based on the usage of discrete and polynomial interpolatory and quasi-interpolatory splines for the design of prediction and update filters in lifting schemes of the wavelet transform. The diversity of the constructed transforms enables to have flexible automatic adaptation of the transforms to the image.

The new transforms and the biorthogonal 9/7 transform were incorporated into SPIHT in order to measure and compare their performance with well known codec. The newly designed transforms outperform the 9/7 transform. But the computational cost of their implementation is close to the cost of the implementation of the transform with 9/7 filters.

The technique described in the paper provides tools for extension of the presented library of wavelet transforms.

We plan to design a coding scheme which can utilize specific properties of the newly constructed transforms. We will test the performance of these transforms on video and seismic data.

7 Appendix I: Implementation of recursive filters

Let $\mathbf{x} = \{x(k)\}, k = 1 \ldots N$. To correctly implement recursive filtering of this finite-length signal, we have to extend $\mathbf{x}$ beyond the given interval. Since our filter banks are symmetric, we use the **HH** extension as in the terminology of [8]. It means that the signal is extended through both ends of the interval symmetrically with repetition of boundary samples. Namely, $x(0) \triangleq x(1)$, $x(-1) \triangleq x(2), \ldots x(-k) \triangleq x(k+1)$ and $x(N+1) \triangleq x(N)$, $x(N+2) \triangleq x(N-1), \ldots x(N+k) \triangleq x(N-k+1)$. This results in periodization of the signal with period $2N$. This extended signal is denoted by $\tilde{\mathbf{x}}$.

Each of the recursive filters $f_1(\zeta) \triangleq F_i^3(\zeta)$ (3.15) and $f_3(\zeta) \triangleq F_d^6(\zeta$ (3.7), which we employed in our experiments, comprised a block of type

$$F(\zeta) = \frac{1+\zeta}{(1+\alpha\zeta)(1+\alpha\zeta^{-1})}. \tag{7.27}$$

We describe the application of the filter F, whose transfer function is given by $F(\zeta)$, on a finite-length signal $\mathbf{x}$:

$$y(\zeta) = F(\zeta)x(\zeta). \tag{7.28}$$

Equation (7.27) is equivalent to

$$F(\zeta) = \frac{1}{1+\alpha}\left(\frac{1}{1+\alpha\zeta^{-1}} + \frac{\zeta}{1+\alpha\zeta}\right).$$

Denote

$$y_1(\zeta) = \frac{1}{1+\alpha\zeta^{-1}}x(\zeta) = \sum_{n=0}^{\infty}(-\alpha)^n\zeta^{-n}\tilde{x}(\zeta), \quad y_2(\zeta) = \frac{\zeta}{1+\alpha\zeta}x(\zeta) = \sum_{n=0}^{\infty}(-\alpha)^n\zeta^{n+1}\tilde{x}(\zeta).$$

Then

$$y_1(k) = x(k) - \alpha y_1(k-1), \quad y_2(k) = x(k+1) - \alpha y_2(k+1) \tag{7.29}$$

$$\Longleftrightarrow y_1(k) = \tilde{x}(k) + \sum_{n=1}^{\infty}(-\alpha)^n\tilde{x}(k-n), \quad y_2(k) = \tilde{x}(k+1) + \sum_{n=1}^{\infty}(-\alpha)^n\tilde{x}(k+n+1). \tag{7.30}$$

We can use (7.29) for computation of $y_1(k)$ and $y_2(k)$ provided we know $y_1(1)$ and $y_2(N)$, respectively. To evaluate these samples, we employed (7.30) keeping in mind the extension of the input signal $\mathbf{x}$. We have

$$y_1(1) = x(1) + \sum_{n=1}^{\infty}(-\alpha)^n\tilde{x}(-n+1) \approx x(1) + \sum_{n=1}^{d}(-\alpha)^n x(n), \tag{7.31}$$

$$y_2(N) = \tilde{x}(N+1) + \sum_{n=1}^{\infty}(-\alpha)^n\tilde{x}(N+n+1) \approx x(N) + \sum_{n=1}^{d}(-\alpha)^n x(N-n) \tag{7.32}$$

where d is a prescribed depth of initialization.

Filtering (7.28) can be implemented by the following parallel algorithm:

1. Evaluate $y_1(1)$ from (7.31) and $y_2(N)$ from (7.32).
2. Calculate $y_1(k) = x(k) - \alpha y_1(k-1), \quad k = 2, \dots N$ and $y_2(k) = x(k+1) - \alpha y_2(k+1), \quad k = N-1, \dots 1$.
3. $y(k) = (y_1(k) + y_2(k))/(1+\alpha), \quad k = 1 \dots N$

Equations (7.30) and (7.32) imply that $y_2(N) = y_1(N)$. Hence, it follows that

$$y(N) = \frac{y_1(N) + y_2(N)}{1+\alpha} = \frac{2y_1(N)}{1+\alpha}. \tag{7.33}$$

The cascade algorithm has the following form:

1. Evaluate $y_1(1)$ from (7.31). 2. Calculate $y_1(k) = x(k) - \alpha y_1(k-1), \quad k = 2, \dots N$.

3. Evaluate $y(N)$ from (7.33). 4. Calculate $y(k) = y_1(k) + y_1(k+1) - \alpha y(k+1), \quad k = N-1, \dots 1$.

8 Appendix II: 2D implementation of transforms with FIR filters

The 2D array $\mathbf{x} \stackrel{\Delta}{=} \{x(n,m)\}, \quad n,m = 0 \dots N-1$, is to be transformed into four subarrays: $\mathbf{x} \to \mathbf{ss} \bigcup \mathbf{sd} \bigcup \mathbf{ds} \bigcup \mathbf{dd}$, where **ss** is smoothed low-frequency section in horizontal and vertical directions, **sd** – low-frequency in vertical direction, high-frequency in horizontal direction, **ds** – low-frequency in horizontal direction, high-frequency in vertical direction, **dd** – high-frequency section of details in both horizontal and vertical directions. Conventionally the 2D transform of $\mathbf{x}$ is implemented in a separable way (tensor product). In [2] a direct 2D lifting implementation of the transform B9/7 is presented . We describe the 2D lifting implementation of the transform T_2^2.

NOTATION: Let $\mathbf{a} = \{a(n,m)\}, \ n,m = 0 \dots M-1$ be a 2D array and $\tilde{\mathbf{a}}$ be a 2-periodic **HH**-extension of the array in both vertical and horisontal directions. Then $\mathbf{L_r} \cdot \mathbf{a}$ means the left shift of the extended array. Namely, $L_r \cdot a(n,m) = \tilde{a}(n, m+r), \ m = 0 \dots M-1,$. Similarly, the right shift is $\mathbf{R_r} \cdot \mathbf{a} = \{R_r \cdot a(n,m)\} = \{\tilde{a}(n, m-r)\}, \ m = 0 \dots M-1$.

The upper shift is $\mathbf{U_r} \cdot \mathbf{a} = \{U_r \cdot a(n,m)\} = \{a(n+r, m)\}, \ n = 0 \dots M-1$. The down shift is $\mathbf{D_r} \cdot \mathbf{a} = \{D_r \cdot a(n,m)\} = \{\tilde{a}(n-r, m)\}, \ n = 0 \dots M-1$.

8.1 T_2^2 Transform

The parameters are: $\alpha = -9/16, \ \beta = 1/16, \ \gamma = -\alpha/2, \ \delta = -\beta/2$.

8.1.1 Decomposition

Split $\mathbf{x} = \mathbf{ee} \bigcup \mathbf{eo} \bigcup \mathbf{oe} \bigcup \mathbf{oo}$.

Predict 1. $\mathbf{de} = \mathbf{oe} + \alpha(\mathbf{ee} + \mathbf{U_1} \cdot \mathbf{ee}) + \beta(\mathbf{D_1} \cdot \mathbf{ee} + \mathbf{U_2} \cdot \mathbf{ee})$.

2. $\mathbf{dd} = \mathbf{oo} + \alpha(\mathbf{de} + \mathbf{L_1} \cdot \mathbf{de} + \mathbf{eo} + \mathbf{U_1} \cdot \mathbf{eo}) + \beta(\mathbf{R_1} \cdot \mathbf{de} + \mathbf{L_2} \cdot \mathbf{de} + \mathbf{D_1} \cdot \mathbf{eo} + \mathbf{U_2} \cdot \mathbf{eo})$.

Update 1. $\mathbf{sd} = \mathbf{eo} + \alpha(\mathbf{ee} + \mathbf{L_1} \cdot \mathbf{ee}) + \beta(\mathbf{R_1} \cdot \mathbf{ee} + \mathbf{L_2} \cdot \mathbf{ee}) + \gamma(\mathbf{dd} + \mathbf{D_1} \cdot \mathbf{dd}) + \delta(\mathbf{U_1} \cdot \mathbf{dd} + \mathbf{D_2} \cdot \mathbf{dd})$

2. $\mathbf{ss} = \mathbf{ee} + \gamma(\mathbf{de} + \mathbf{D_1} \cdot \mathbf{de} + \mathbf{sd} + \mathbf{R_1} \cdot \mathbf{sd}) + \delta(\mathbf{U_1} \cdot \mathbf{de} + \mathbf{D_2} \cdot \mathbf{de} + \mathbf{L_1} \cdot \mathbf{sd} + \mathbf{R_2} \cdot \mathbf{sd})$.

3. $\mathbf{ds} = \mathbf{de} + \gamma(\mathbf{dd} + \mathbf{R_1} \cdot \mathbf{dd}) + \delta(\mathbf{L_1} \cdot \mathbf{dd} + \mathbf{R_2} \cdot \mathbf{dd})$.

Normalization $\mathbf{ss} = 2\mathbf{ss}$, $\mathbf{dd} = \mathbf{dd}/2$.

Reconstruction of the 2D array $\mathbf{x}$ from four subarrays of coefficients:
$\mathbf{ss} \bigcup \mathbf{sd} \bigcup \mathbf{ds} \bigcup \mathbf{dd}$, $\mathbf{ss} \to \mathbf{x}$ is conducted in reverse order.

References

[1] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, Image Coding using Wavelet Transform, *IEEE Transaction on Image Processing,* 1(2):205-220, 1992.

[2] A. Z. Averbuch, F. Meyer, J-O Stromberg, *Fast Adaptive Wavelet Packet Image Compression*, IEEE Trans. on Image Processing, 9: 792-800, May 2000.

[3] A. Z. Averbuch, A. B. Pevnyi and V. A. Zheludev *Butterworth wavelet transforms derived from discrete interpolatory splines: Recursive implementation*, Signal Processing, **81**, (2001), 2363-2382.

[4] A. Z. Averbuch, A. B. Pevnyi and V. A. Zheludev *Biorthogonal Butterworth wavelets derived from discrete interpolatory splines*, IEEE Trans. Sign. Proc., **49**, No.11, November 2001, 2682-2692.

[5] A. Averbuch, V. Zheludev *Construction of biorthogonal discrete wavelet transforms using interpolatory splines*, to appear in Applied and Comp. Harmonic Analysis.

[6] A. Averbuch, V. Zheludev, *Splines: a new contribution to wavelet analysis* to appear in Proc. Conference "Algorithms for Approximation IV."

[7] G. Battle, *A block spin construction of ondelettes. Part I. Lemarié functions*, Comm. Math. Phys. **110**, (1987), 601–615.

[8] C. M. Brislawn, *Classification of Nonexpansive Symmetric Extension Transforms for Multirate Filter Banks* Applied and Computational Harmonic Analysis, **3**, No. 4, (1996), 337-357.

[9] C. K. Chui and J. Z. Wang, *On compactly supported spline wavelets and a duality principle*, Trans. Amer. Math. Soc., Vol. 330 (1992), pp. 903-915.

[10] A. Cohen, I. Daubechies and J. C. Feauveau, *Biorthogonal bases of compactly supported wavelets*, Comm. Pure Appl. Math., **45**(1992), 485-560.

[11] I. Daubechies, W. Sweldens, *Factoring wavelet transforms into lifting steps*, J. Fourier Anal. Appl., 4: 247–269, 1998.

[12] D. L. Donoho, *Interpolating wavelet transform*, Preprint 408, Department of Statistics, Stanford University, 1992.

[13] C. Herley and M. Vetterli, *Wavelets and recursive filter banks*, IEEE Trans. Signal Proc., **41(12)** (1993), pp. 2536-2556.

[14] P. G. Lemarié, *Ondelettes à localization exponentielle*, J. de Math. Pure et Appl., **67**, (1988) 227–236.

[15] D. Marpe, G. Heising, A. P. Petukhov, and H. L. Cycon, *Video coding using a bilinear image warping motion model and wavelet-based residual coding*, Proc. SPIE Conf. on Wavelet Applications in Signal and Image Processing VII, Denver, CO, July 1999, SPIE, Vol. 3813, pp. 401 – 408.

[16] A. V. Oppenheim, R. W. Shafer, *Discrete-time signal processing*, Englewood Cliffs, New York, Prentice Hall, 1989.

[17] G. Strang, and T. Nguen, *Wavelets and filter banks*, Wellesley-Cambridge Press, 1996.

[18] A. P. Petukhov, *Biorthogonal wavelet bases with rational masks and their applications*, Proc. of St. Petersburg Math. Soc., Vol. 7 (1999), pp. 168 – 193. (Russian)

[19] A. B. Pevnyi and V. A. Zheludev, *Construction of wavelet analysis in the space of discrete splines using Zak transform* , J. Fourier Analysis and Application, **8** No.1, (2002) 55-77.

[20] M. Rabbani, R. Joshi, *An overview of the JPEG 2000 still image compression standard*, Signal Processing, Image communication, **17**, (2002), 3-48.

[21] A. Said and W. W. Pearlman, *A new, fast and efficient image codec based on set partitioning in hierarchical trees*, IEEE Trans. on Circ. and Syst. for Video Tech., 6: 243-250, June 1996.

[22] I. J. Schoenberg, *Cardinal spline interpolation*, CBMS, **12**, SIAM, Philadelphia, 1973.

[23] I. J. Schoenberg, *Contribution to the problem of approximation of equidistant data by analytic functions*, Quart. Appl. Math., **4** (1946), 45-99, 112-141.

[24] L. L. Schumaker, *Spline functions: Basic theory*, Wiley, 1981.

[25] W. Sweldens em The lifting scheme: A custom design construction of biorthogonal wavelets, Appl. Comput. Harm. Anal. Vol. 3(2), (1996), pp. 186-200.

[26] M. Unser, A. Aldroubi and M. Eden, *B-spline Signal Processing: Part I–Theory*, IEEE Trans. Signal Process, Vol. 41, No. 2, Feb. 1993, pp. 821-832.

[27] M. Unser, A. Aldroubi and M. Eden, *B-spline Signal Processing: Part II–Efficient Design and Applications*, IEEE Trans. Signal Process, Vol. 41, No. 2, Feb. 1993, pp. 834-848.

[28] M. Unser, A. Aldroubi and M. Eden, *A family of polynomial spline wavelet transforms*, Signal Processing, Vol. 30, (1993), 141-162.

[29] V. A. Zheludev *Local spline approximation on a uniform grid*, U.S.S.R. Comput. Math. & Math. Phys., **27** No.5, (1987), 8-19.

[30] V.A. Zheludev, *Local smoothing splines with a regularizing parameter,* Comput. Math. & Math Phys., **31**, (1991), 193-211.