Low-complexity hierarchical lapped transform for lossy-to-lossless image coding in JPEG XR / HD Photo

Chengjie Tu, Sridhar Srinivasan, Gary J. Sullivan, Shankar Regunathan, and Henrique S. Malvar
Microsoft Corporation, One Microsoft Way, Redmond, WA, USA 98052

ABSTRACT

JPEG XR is a draft international standard undergoing standardization within the JPEG committee, based on a Microsoft technology known as HD Photo. One of the key innovations in the draft JPEG XR standard is its integer-reversible hierarchical lapped transform. The transform can provide both bit-exact lossless and lossy compression in the same signal flow path. The transform requires only a small memory footprint while providing the compression benefits of a larger block transform. The hierarchical nature of the transform naturally provides three levels of multi-resolution signal representation. Its small dynamic range expansion, use of only integer arithmetic and its amenability to parallelized implementation lead to reduced computational complexity. This paper provides an overview of the key ideas behind the transform design in JPEG XR, and describes how the transform is constructed from simple building blocks.

Keywords: Image coding, JPEG XR, HD Photo, Transform, Lapped Transform, DCT.

1. INTRODUCTION

Presently, the JPEG committee is in the process of standardizing a new image coding specification known as JPEG XR[1], which is based on the Microsoft HD Photo technology[2]. JPEG XR has been designed to optimize image quality and compression efficiency while also enabling low-complexity encoding and decoding implementations.

JPEG XR is a "block-transform" based image compression scheme which uses some of the same high-level building blocks as traditional image codecs: color conversion, transform, quantization, coefficient scanning and entropy coding. The transform converts the spatial domain image data to frequency domain information. JPEG XR uses a hierarchical two-stage lapped biorthogonal transform (LBT)[4-8], with a novel low-complexity structure that makes it exactly invertible in integer arithmetic (also referred to as integer reversible). This paper provides an overview to the transform used in JPEG XR, and also describes how the transform enables many of the distinguishing features of JPEG XR.

The transform is based on two basic operators: the Photo Core Transform (PCT) and the Photo Overlap Transform (POT). The PCT is similar to the widely used discrete cosine transform (DCT)[5], and can exploit spatial correlation within the block. However, the PCT has some of the same shortcomings as the DCT and other block transforms, in that it cannot exploit redundancy across block boundaries and can result in blocking artifacts at low bit rates. The POT is designed to exploit the correlation across block boundaries as well as mitigate blocking artifacts, and thus address the drawbacks of the PCT. Note that the current draft JPEG XR specification[1] uses the terms "core transform" and "overlap filtering" instead of PCT and POT, respectively.

If the POT and PCT are concatenated appropriately, the combined transform is an LBT[6,9]. The LBT offers state-of-the-art coding performance, both objectively and visually, at low computational complexity[4-9]. JPEG XR further improves the performance of the transform by adopting a hierarchical construction[7,8]. The resulting hierarchical two-stage LBT effectively uses long filters for low frequencies and short filters for high frequency detail. Thus, the transform has a better coding gain as well as reduced ringing and blocking artifacts when compared to traditional block transforms.

Each stage of the transform can be viewed as a flexible concatenation of the POT and PCT. The POT is functionally independent of the PCT, and can be switched on or off, as chosen by the encoder. At each transform stage, the PCT is always performed. However, there are three options for controlling the POT: disabled for both stages, enabled for the first stage but disabled for the second stage, or enabled for both stages. The overlap option that is used at the encoder is signaled to the decoder as part of the compressed bitstream. The flexibility to enable or disable the overlap operators controls the effective filter length. Overlap operators at both levels can be turned off to minimize ringing artifacts related to long filters, as well as to enable a very low complexity decoding mode. Overlap operators at both levels can be turned on at very low bit rates to mitigate blocking artifacts.

Each operator of the JPEG XR transform is designed to be reversible. JPEG XR implements reversible transforms using a lifting-based structure, as described below, which minimizes the dynamic range expansion of the input data, and thus reduces implementation complexity and maximizes lossless compression performance. As the lifting operations use only integer arithmetic, the decoder output is bit-exact for any given compressed bitstream. This paper is organized as follows. We begin by giving a brief overview of the transform design in Section 2 and some basic design concepts in Section 3. Section 4 discusses the basic building blocks for constructing the transform operators, via efficient non-separable 2-D decompositions. Sections 5 and 6 describe how to construct the PCT and POT using these basic building blocks, respectively. The computational complexity and multiresolution aspects of the transform are discussed in Section 7.

2. TRANSFORM OVERVIEW

The regions of support for the PCT and POT consist of 4×4 block regions that are offset relative to each other as shown in Figure 1.

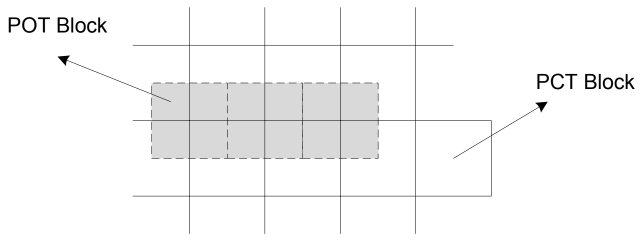


Figure 1 – Regions of support for the basic 4×4 PCT and POT operators; each POT operator is centered on the boundaries of four PCT operators.

The transform operations on the encoder side are performed in the following order to produce *macroblocks* of luma and chroma transform coefficients. One macroblock is produced for each 16×16 region of luma samples and corresponding (16×16, 8×16, or 8×8) region of chroma samples as applicable.

1. First, if it is chosen to be enabled by the encoder, a 4×4 POT is applied. On image boundaries this reduces to 4-point 1-D POTs. The region of support for a 4×4 POT uniformly spans across four blocks as shown in Figure 1.
2. Next, a 4×4 PCT is applied to each block to obtain one DC coefficient and 15 AC coefficients.
3. The DC coefficients of blocks from the first stage are then collected to form a block of such coefficients. For luma and for chroma when chroma is not downsampled relative to luma, this is a 4×4 block. When chroma is downsampled relative to luma, this is a 2×2 or 2×4 block, depending on whether the downsampling is applied both horizontally and vertically or only horizontally.
4. If chosen to be enabled by the encoder, another POT is then performed for the second stage. The only difference between what is done for this POT stage and what is done for the POT in the first stage is that when downsampled chroma channels are used, one or two 2×2 POTs are used instead of a 4×4 POT so as to preserve the macroblock structure for coding granularity (when chroma is not downsampled, an ordinary 4×4 POT is applied). On image boundaries, these reduce to 2-point 1-D POTs and 4-point 1-D POTs, respectively.
5. Then a second stage PCT is applied to the block of first-stage DC coefficients. For luma and when chroma is not downsampled, this is a 4×4 PCT. For processing downsampled chroma, one or two 2×2 PCTs are applied to process the 2×2 or 2×4 block.

Therefore, on the encoder side, transform operations follow the following order: optional first stage POT → first stage PCT → optional second stage POT → second stage PCT. The inverse transform operations on the decoder side follows the reverse order: second stage inverse PCT → optional second stage inverse POT → first stage inverse PCT→ optional first stage inverse POT. As described above, there are six types of operators involved: 4×4 PCT, 2×2 PCT, 4×4 POT, 2×2 POT, 4-point 1-D POT and 2-point 1-D POT.

Of these six operators, the most frequently used ones are the 4×4 PCT and the 4×4 POT, the remaining operators being edge cases. We describe these two operators, along with the 2×2 PCT, in the following sections.

3. BASIC DESIGN CONCEPTS

A separable 2-D transform is typically implemented by performing 1-D transforms on the rows of the data, followed by 1-D transforms on its columns of data (or vice versa). The row-wise and column-wise transforms might be distinct. Let T_c and T_r represent the column and row transform matrices respectively. The 2-D data X is transformed to Y by

$$Y = T_c \; X \; T_r' \tag{1}$$

3.1 Non-separable 2-D transform

It is well known that a separable 2-D transform can be implemented as a 1-D transform operating on the data ordered in 1-D, producing a correspondingly-ordered vector result. The equivalent transform matrix is generated by the Kronecker product of the row and column transforms in the separable case. Let x and y denote reorderings of X and Y to form column vectors, so that (1) becomes

$$y = T\,x \tag{2}$$

where $T = T_c \otimes T_r$ denotes the Kronecker product of T_c and T_r.

Most separable 2-D transforms can be implemented as a cascade simpler 1-D transforms. Assume in Equation (1) that

$$T_c = T_{c1}\; T_{c2} \;\text{ and }\; T_r = T_{r1}\; T_{r2} \tag{3}$$

It is then trivial to show that in Equation (2), the following applies.

$$T = (T_{c1} \otimes T_{r1}) * (T_{c2} \otimes T_{r2}) \tag{4}$$

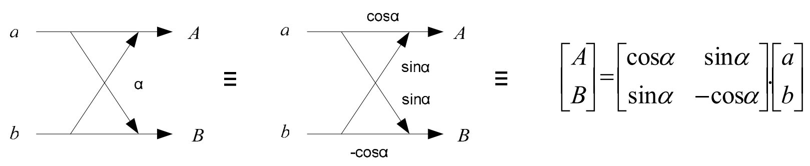
So the corresponding non-separable transform is also a cascade of simpler non-separable transforms.

3.2 Transform factorization

Transforms such as the DCT can be formulated as a cascade of elementary operations, most of which are 2-point rotation operations. This is the key to most fast algorithms. A rotation operator is defined as follows.

$$R(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ \sin(\alpha) & -\cos(\alpha) \end{bmatrix} \tag{5}$$

and its equivalent flowgraph is shown in Figure 2. Note that this actually combines a rotation and mirror flip and is involutory, which means that it is its own inverse.



$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ \sin\alpha & -\cos\alpha \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Figure 2. Flowgraph module of a basic rotation operator.

The 4×4 PCT is loosely based on a 1-D 4-point DCT, as illustrated in Figure 3. It uses a two-stage rotation process: the first stage $R(\pi/4)$ applies to a and d and to b and c; and the second stage $R(\pi/4)$ applies to a and c, and $R(\pi/8)$ applies to b and d. The 4×4 POT is based on a 1-D 4-point overlap operator illustrated in Figure 4. This is a four-stage elementary operation process: the first stage $R(\pi/4)$ applies to a and d and to b and c; the second stage $R(\pi/8)$ applies to c and d; in the third stage a and b are scaled by s and c and d are scaled by $1/s$; and the fourth stage is the same as the first stage. The scaling factor s should have its value set such that the overlapping operator to generate smooth synthesis basis functions, which not only improve visual quality at low bit rates, but also improve coding gain. The scaling matrix $diag(s, s, 1/s, 1/s)$ has determinant equal to one, which is needed for optimal lossless compression performance.

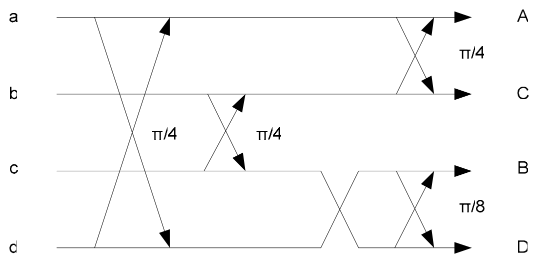


Figure 3. One-dimensional 4-point DCT.

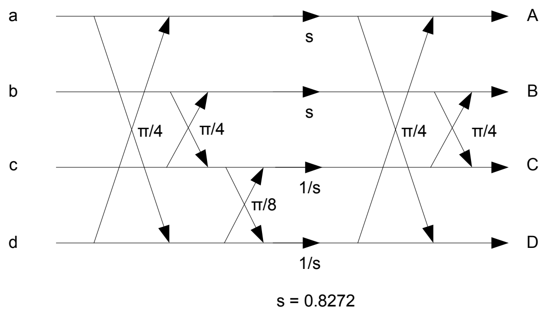


Figure 4. One-dimensional 4-point overlap operator.

3.3 Kronecker product of two rotation operations

The following equation holds

$$R(\alpha) \otimes R(\beta) = B\ diag(1, 1, -1, -1)\ C\ B\ diag(1, -1, -1, 1) \tag{6}$$

where *diag*(*a, b, c, d*) is a diagonal matrix with diagonal entries *a, b, c* and *d*, and *B* and *C* are defined by

$$B = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{bmatrix} \tag{7}$$

$$C = \begin{bmatrix} \cos(\alpha-\beta) & 0 & \sin(\alpha-\beta) & 0 \\ 0 & \cos(\alpha+\beta) & 0 & \sin(\alpha+\beta) \\ \sin(\alpha-\beta) & 0 & -\cos(\alpha-\beta) & 0 \\ 0 & \sin(\alpha+\beta) & 0 & -\cos(\alpha+\beta) \end{bmatrix} \tag{8}$$

B involves two $R(\pi/4)$ operations, and *C* involves a $R(\alpha-\beta)$ operation and a $R(\alpha+\beta)$ operation. If α is equal to β, then $R(\alpha-\beta)$ becomes $R(0) = diag(1,-1)$, which is a simple sign flip.

With some reordering, a stage of rotation operations of a transform becomes

$$E = \begin{bmatrix} R(\alpha) & 0 \\ 0 & R(\beta) \end{bmatrix} \tag{9}$$

For example, $E = \begin{bmatrix} R(\pi/4) & 0 \\ 0 & R(\pi/8) \end{bmatrix}$ for the second stage of the 1-D 4-point DCT in Figure 3. It is obvious that obtaining $E \otimes E$ with *E* given by (9) involves $R(\alpha)\otimes R(\alpha)$, $R(\alpha)\otimes R(\beta)$, $R(\beta)\otimes R(\alpha)$ and $R(\beta)\otimes R(\beta)$, which apply to the top-left, top-right, bottom-left, and bottom-right quadrants of the data block, respectively.

3.4 Lifting Structure

All transform operations in JPEG XR are implemented using the lifting structure[10-12]. Lifting can be applied as a process for performing a matrix-vector multiplication using successive 'shears'. A shear is defined as a multiplication of the operand vector with a matrix which is an identity matrix plus one non-zero off-diagonal element. The lifting structure is always reversible as the inverse operator can be obtained by reversing the order of the structure and sign of each operator. A lifting implementation showing two shears is illustrated in Figure 5.

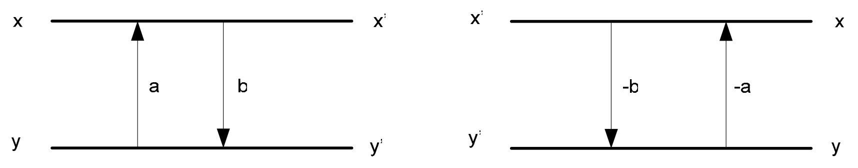


Figure 5. Lifting example (left: forward, right: inverse).

JPEG XR uses only shear operator of the following form: *y* += (*i x* + *r*) >> *b*, where *i* is a small integer and both *r* and *b* are small non-negative integers. The shear is termed a trivial lifting step if $i = 1$, $b \leq 1$ and $r = 0$ since no multiplication is needed, and non-trivial lifting step otherwise. If $i = \pm1$ and $b = r = 0$, it reduces to a simple addition.

3.5 Lifting-based implementation of rotation operations

Figure 6 shows a lifting-based implementation of the rotation operation $R(\alpha)$, which requires three lifting steps plus a sign inversion. If α is small, $R(\alpha)$ can be approximated well as shown in Figure 7, (a) using three lifting steps, and (b) using just two lifting steps.

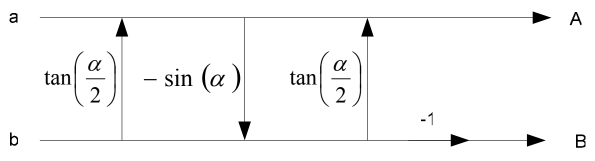

a
A
$\tan\left(\frac{\alpha}{2}\right)$
$-\sin(\alpha)$
$\tan\left(\frac{\alpha}{2}\right)$
-1
b
B

Figure 6. Lifting-based $R(\alpha)$ implementation.

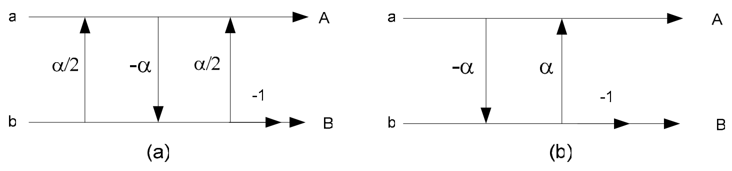


Figure 7. Approximation of $R(\alpha)$ with rational lifting factors for small values of α : (a) using three lifting steps and (b) using only two lifting steps.

4. BASIC BUILDING BLOCKS

The 2-D non-separable implementation of the PCT and POT based on the staged and factorized implementation of the 1-D DCT in Figure 3 and the 1-D overlap operator in Figure 4 involves only the following operators: $R(\pi/4)$, $R(\pi/8)$, $R(\pi/4)\otimes R(\pi/4)$, $R(\pi/8)\otimes R(\pi/8)$, $R(\pi/4)\otimes R(\pi/8)$, and some scaling operations.

4.1 Hadamard transform $R(\pi/4)$

The most import rotation operator in the 1-D DCT (Figure 3) and overlap operator (Figure 4) is $R(\pi/4)$, which is the Hadamard transform:

$$T_H = R(\pi/4) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{10}$$

A lifting-based implementation is shown in Figure 8, which uses an S transform[13] composed only of trivial operations. Note that this is not a normalized implementation; it replaces the sum of the two inputs by their averages.

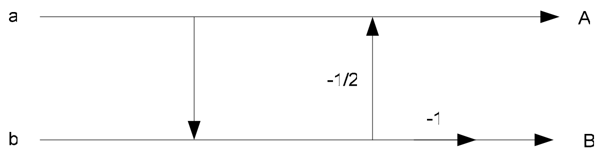


Figure 8. S-transform implementation of T_H .

4.2 2-D Hadamard transform $R(\pi/4) \otimes R(\pi/4)$

The Kronecker product of 2-point Hadamard transforms T_H is the 2-D Hadamard transform:

$$T_{HH} = T_H \otimes T_H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{11}$$

Interestingly it is possible to implement this 2-D Hadamard transform using only trivial lifting steps as shown in Figure 9.

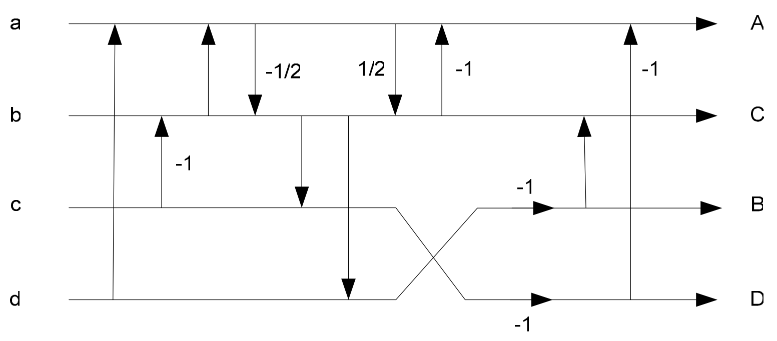


Figure 9. Implementation of T_{HH}.

4.3 2-Point rotation T_R

The rotation operator T_R (which is equal to $R(\pi/8)$) can be implemented as shown in Figure 10, which is an approximation of the module in Figure 7 with two lifting operators with coefficients –½ and ½.

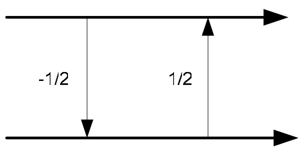


Figure 10. Implementation of T_R.

4.4 2-D rotation T_{RR}

The 2-D rotation T_{RR} (which is equal to $R(\pi/8) \otimes R(\pi/8)$) can be implemented as shown in Figure 11. Referring to Equations (5), (6), and (7), T_{RR} involves B which is a stack of two T_H operators implemented in Figure 8 (with some sign manipulations). Note how two stages of B operations combine to get a normalized implementation. T_{RR} also involves $R(\pi/4)$ which is implemented as an approximation in the form of Figure 7.

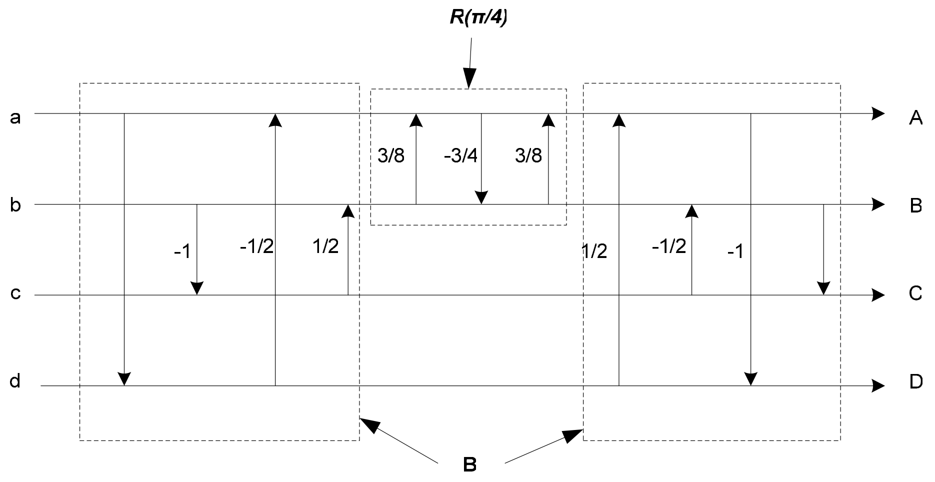


Figure 11. Implementation of T_{RR} .

4.5 2-D rotation T_{HR}

Similar to T_{RR} , T_{HR} involves *B*. T_{HR} also involves $R(\pi/4+\pi/8)=R(3\pi/8)$ and $R(\pi/4-\pi/8)=R(\pi/8)$. Note also that

$$R(3\pi/8)=\begin{bmatrix}0&1\\1&0\end{bmatrix}R(\pi/8)\begin{bmatrix}1&0\\0&-1\end{bmatrix} \tag{12}$$

So $R(3\pi/8)$ can also be implemented based on $R(\pi/8)$. Figure 12 shows the implementation where $R(\pi/8)$ is implemented based on Figure 7.

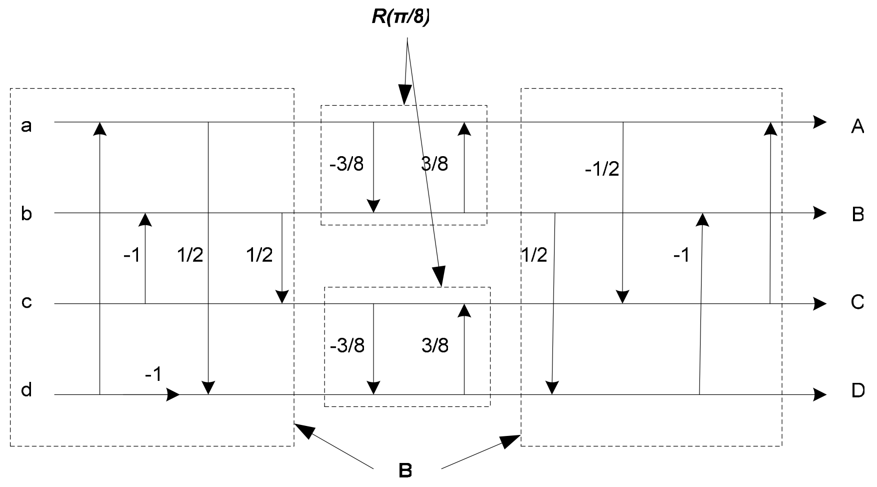


Figure 12. Implementation of T_{HR} .

4.6 2-Point scaling operator T_S

The scaling stage in Figure 4, when applied to 2-D data, produces the resulting scaling matrix

$$\begin{bmatrix} s^2 & s^2 & 1 & 1 \\ s^2 & s^2 & 1 & 1 \\ 1 & 1 & s^{-2} & s^{-2} \\ 1 & 1 & s^{-2} & s^{-2} \end{bmatrix} \tag{13}$$

Half the matrix entries are unity and therefore these points are merely passed through. The remaining entries are paired symmetrically in two-point groups around the center of the matrix so that they can be implemented using lifting-based shear operators. The operator T_S approximately implements a two-point scaling matrix $T_S = \begin{bmatrix} s^2 & 0 \\ 0 & s^{-2} \end{bmatrix}$ with s^2 approximately equal to 0.6842 (choosing a value in the neighborhood of $1/\sqrt{2}$ to provide good compromise between coding gain and smoothness of the LBT basis functions). In the current draft of JPEG XR, this is implemented as shown in Figure 13. In the prior HD Photo design, the T_S operator approximation had somewhat larger off-diagonal magnitudes, which resulted in a less ideal transform for the processing of data with high sample bit-depths. (The magnitude of the off-diagonal terms in the scheme of Figure 13 is 2^{-18}, versus approximately 2^{-7} in the prior HD Photo design.)

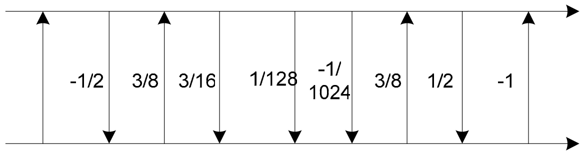


Figure 13. Implementation of T_S in JPEG XR.

5. 4×4 PCT IMPLEMENTATION

Suppose the data block is given by:

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \tag{14}$$

The steps to implement a 4×4 PCT are shown below, where the steps are performed as in-place operations:

1. Hadamard transform stage:
    a. T_{HH} (a, d, m, p)
    b. T_{HH} (b, c, n, o)
    c. T_{HH} (e, h, i, l)
    d. T_{HH} (f, g, j, k)

2. Rotation stage:
    a. T_{HH} (a, b, e, f)
    b. T_{HR} (h, g, d, c)
    c. T_{HR} (n, j, m, i)
    d. $l = -l$; $o = -o$
    e. T_{RR} (k, l, o, p)

Figure 14 contrasts the traditional 4×4 separable transform with the procedure defined above. The PCT interleaves the rotations in the traditional transform, and then combines rotations across the two dimensions to result in the above implementation.

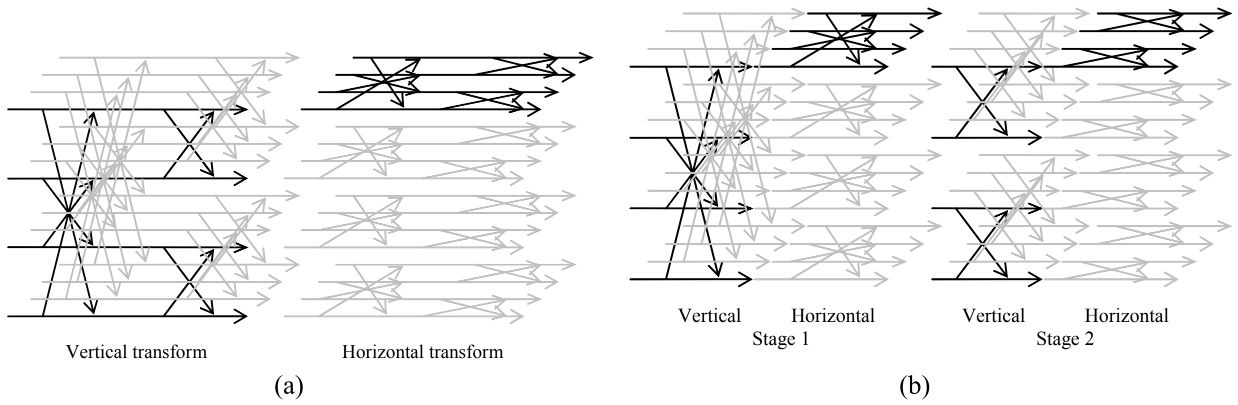


Figure 14. (a) Separable 4×4 transform and (b) PCT compared. Stage 1 and 2 of the PCT above are the Hadamard transform and rotation stages respectively.

6. 4×4 POT IMPLEMENTATION

The steps to implement the 4×4 POT, performed as in-place operations, are:

1. Hadamard transform stage:
    a. T_{HH} (a, d, m, p)
    b. T_{HH} (b, c, n, o)
    c. T_{HH} (e, h, i, l)
    d. T_{HH} (f, g, j, k)
2. Scaling stage:
    a. T_S (a, p)
    b. T_S (b, o)
    c. T_S (e, l)
    d. T_S (f, k)

3. Rotation stage:
    a. $T_R\,(n, m)$
    b. $T_R\,(j, i)$
    c. $T_R\,(h, d)$
    d. $T_R\,(g, c)$
    e. $T_{RR}\,(k, l, o, p)$
4. Hadamard transform stage:
    a. $T_{HH}\,(a, d, m, p)$
    b. $T_{HH}\,(b, c, n, o)$
    c. $T_{HH}\,(e, h, i, l)$
    d. $T_{HH}\,(f, g, j, k)$

It should be apparent that ideas similar to those illustrated in Figure 14 can also be applied for an efficient implementation of the above steps.

7. COMPLEXITY AND MULTIRESOLUTION

The computational complexity of the JPEG XR transform is low, as the basic transform operators are of small size (4×4 at most), and these operators are implemented by a few simple 2-point and 4-point building blocks. Therefore, the transform operators require only a small memory footprint both at the encoder/decoder. Further, the direct implementation of the 2-D transforms instead of implementing two 1-D separable transforms results in further reduction in computational complexity as many lifting steps are combined or canceled. Note that all lifting steps only involve integer additions, and/or right shifts, and/or multiplications. As no division or floating point operations are involved, the transform enables bit-exact decoding. The number of operations required to implement the basic transform operators can be summarized as follows:

1. The 4×4 PCT requires only 91 trivial and 11 non-trivial operations per block, i.e., only 5.69 trivial and 0.69 non-trivial operations per processed sample.
2. The 4×4 POT requires only 156 trivial and 15 non-trivial operations per block, i.e. only 9.75 trivial and 0.94 non-trivial operations per processed sample. The trivial count can be lowered slightly by combining some steps of stages 3 and 4.

Note that each nontrivial operation comprises a multiply by 3, a bit shift and 2 adds. Therefore, only a single unique multiplier needs to be implemented. Further, additional complexity reduction is possible by exploiting the fact that only a single shift and add (or two adds) is necessary to implement the multiply by 3.

Most of the operations can be implemented in parallel with SIMD instructions and all operations can be parallelized with MIMD instructions. Further, the use of lifting steps ensures that the dynamic range expansion of the transform is limited to 5 bits. Therefore, 16 bit arithmetic is sufficient for performing transform on 8 bit image data, and this allows for further parallelization and consequent reduction in complexity. The limited dynamic range expansion also allows for lossless compression of 24 bit per sample image data using 32 bit arithmetic.

The two-stage hierarchical transform structure naturally offers three layers of multiresolution and spatial scalability. If only the DC coefficients of the second stage transform are used for decoding, a 16:1 downsampled version of the image (that is, a 16:1 "thumbnail" of the image) can be reconstructed. If only the DC coefficients for the first stage transform are used for decoding, a 4:1 thumbnail of the image can be reconstructed. This spatial scalability is important for zooming in and out of large images, and for quickly generating previews of a large number of images. JPEG XR provides support for the coefficients of the three frequency bands to sent in separate bitstream payloads. Therefore, only bitstream payloads of related subbands need to be parsed for decoding the thumbnail . Furthermore, reconstructing the 16:1 thumbnail does not require any inverse transform, and reconstructing the 4:1 thumbnail requires only the second-stage inverse transform.

CONCLUSION

JPEG XR uses a two stage hierarchical lapped biorthogonal transform which offers state-of-the art compression performance. The overlap operator is used to exploit inter-block correlation and reduce artifacts at block boundaries. The two-stage LBT effectively uses long filters for low frequencies and shorter filters for high frequency, and offers higher coding gain as well as lower blocking and ringing artifacts. The hierarchical nature of the transform naturally provides three levels of multiresolution. The overlap operators at each level can be disabled by the encoder to achieve additional control over quality and decoding complexity. The transform is reversible, and supports bit-exact lossy and lossless compression. All the transform operators are implemented as lifting steps using only integer operations. The small dynamic range expansion, amenability of the operators to parallelized implementation, the low computational complexity and small memory footprint reduce the overall encoding/decoding complexity of JPEG XR.

REFERENCES

1. ITU-T SG 16 Q.6 and ISO/IEC JTC 1/SC 29 WG 1, "Study Text for JPEG XR FCD (ISO/IEC 29199-2)", Document N4659, June 2008.
2. S. Srinivasan, C. Tu, S. L. Regunathan, and G. J. Sullivan, "HD Photo: A new image coding technology for digital photography," Proc. SPIE Appl. of Digital Image Processing XXX, San Diego, CA, vol. 6696, Aug. 2007.
3. Microsoft Corp., "HD Photo Device Porting Kit", http://www.microsoft.com/whdc/xps/hdphotodpk.mspx, 2006.
4. H. S. Malvar and D. H. Staelin, "The LOT: transform coding without blocking effects," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 553-559, Apr. 1989.
5. H. S. Malvar, *Signal Processing with Lapped Transforms*. Norwood, MA: Artech House, 1992.
6. H. S. Malvar, "Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts, " *IEEE Trans. Signal Processing*, pp. 1043-1053, Apr. 1998.
7. H. S. Malvar, "Efficient signal coding with hierarchical lapped transforms", in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, pp. 1519-1522, Apr. 1990.
8. H. S. Malvar, "Fast progressive image coding without wavelets," *Proc. IEEE Data Compression Conf.*, Snowbird, UT, pp. 243-252, Mar. 2000.
9. T. D. Tran, J. Liang, C. Tu, "Lapped transform via time-domain pre- and post-filtering", *IEEE Trans. on Signal Processing*, pp. 1557-1571, June 2003.
10. F. A. M. L. Bruekers and W. M. van den Enden, "New networks for perfect inversion and perfect reconstruction", *IEEE J. on Selected Areas in Communications*, vol. 10, pp. 130-137, Jan. 1992.
11. W. Sweldens, "Lifting scheme: a new philosophy in biorthogonal wavelet constructions," *Proc. SPIE Wavelet Applications in Signal and Image Processing III*, A. F. Laine and M. A. Unser, Eds., vol. 2569, pp. 68-79, San Diego, Calif, USA, July 1995.
12. W. Sweldens, "The lifting scheme: A construction of second generation wavelets", *Siam J. Math. Anal.*, vol. 29, No. 2, pp. 511-546, 1997.
13. P. Lux, "A novel set of closed orthogonal functions for picture coding," *Arch. Elek. Ubertragung*, vol. 31, pp. 267-274, 1977.