15th European Signal Processing Conference (EUSIPCO 2007), Poznan, Poland, September 3-7, 2007, copyright by EURASIP

IMAGE CODING USING A COMPLEX DUAL-TREE WAVELET TRANSFORM

James E. Fowler, Joseph B. Boettcher*, and Béatrice Pesquet-Popescu†*

*Department of Electrical & Computer Engineering
GeoResources Institute
Mississippi State University, MS USA
{fowler,jbb15}@ece.msstate.edu

†École Nationale Supérieure des Télécommunications
Paris, France
pesquet@tsi.enst.fr

ABSTRACT

An embedded wavelet-based coder is deployed to exploit the directional selectivity of a 2D complex dual-tree discrete wavelet transform. Although the dual-tree transform is redundant, a noise-shaping process increases the sparsity of the transform coefficients, resulting in a high degree of spatially coherent regions of insignificant coefficients. The transform coefficients are coded with binary set-partitioning using k-d trees, and experimental results reveal rate-distortion results superior to the state-of-the-art JPEG2000 standard at low bitrates, particularly for images with strong directional features.

1. INTRODUCTION

Although the discrete wavelet transform (DWT) has dominated the field of image compression for well over a decade, DWTs in their traditional critically sampled form are known to be somewhat deficient in several characteristics, lacking such properties as shift invariance and significant directional selectivity [1]. Recently, complex-valued wavelet transforms have been proposed to improve upon these DWT deficiencies, with the dual-tree DWT (DDWT) [2] becoming a preferred approach due to the ease of its implementation. In the DDWT, real-valued wavelet filters produce the real and imaginary parts of the transform in parallel decomposition trees, permitting exploitation of well-established real-valued wavelet implementations and methodologies. A primary advantage of the DDWT lies in that it results in a decomposition with a much higher degree of directionality than that possessed by the traditional DWT. However, since both trees of the DDWT are themselves orthonormal or biorthogonal decompositions, the DDWT taken as a whole is a redundant tight frame [1]. Although present to a significantly less degree than in other overcomplete transforms (e.g., the redundant or oversampled DWT [3]), this redundancy still poses a challenge to certain applications, mostly notably, compression.

In this paper, we adapt an embedded wavelet-based image coder to the task of coding DDWT coefficients. We employ a DDWT with an anisotropic wavelet-packet decomposition to increase directionality beyond that of the more common dyadic decomposition without further increasing the redundancy of the transform. We couple this anisotropic DDWT with a modified version of the embedded wavelet-based coder of [4], binary set splitting with k-d trees (BISK). Experimental results reveal that the resulting DDWT-BISK coder achieves rate-distortion performance superior to that of JPEG2000 at low bitrates, with strongest performance gains resulting for images with strong directional features.

In the following, we first overview the DDWT in Sec. 2 before describing our DDWT-BISK coder in Sec. 3. A detailed presentation of the DDWT-BISK algorithm is made in Sec. 4. Experimental results follow in Sec. 5, and we present some concluding remarks in Sec. 6.

2. THE DDWT

In order to overcome shortcomings of the traditional critically sampled DWT, Kingsbury [2] introduced the DDWT consisting of two trees of real-valued wavelet filters operating on the same data in parallel, with the filters designed such that the two trees produce the real and imaginary parts of the complex-valued coefficients. This approach was extended to higher dimensions to develop 2D and 3D versions of the DDWT for images and video in [5]. While the DWT lacks shift invariance, the DDWT is approximately shift invariant and offers higher directional selectivity. However, the DDWT is 2^m:1 redundant for an m-dimensional signal. It turns out that the degree of redundancy can be reduced without sacrificing perfect reconstruction by simply discarding the complex parts of the coefficients, resulting in 2:1 redundancy for a 2D DDWT. For this real-valued transform, two separable 2D DWTs based on Hilbert pairs of wavelets are applied to the original signal. The resulting two sets of transform data are then combined with linear operations, thereby effectuating the retaining of only the real-valued coefficients. The resulting DDWT subbands are arranged in two separate transform trees with each tree having the same subband organization as would a 2D DWT of the original data, but with each tree containing subbands of different orientation. We employ the popular 9/7 biorthogonal wavelet filters for the first level of decomposition and the Q-shift filters of [6] for the remaining levels. The reader is referred to [1] for a thorough introduction to the DDWT.

The 2D DDWT in [5] employs a dyadic decomposition structure such that the baseband subband in each tree is recursively decomposed. Here, we employ instead an anisotropic wavelet-packet decomposition structure [7] which, as illustrated in Fig. 1, consists of J-scale 1D DDWTs applied separably in both the horizontal and vertical directions. The anisotropic structure increases the directionality of the decomposition with respect to the dyadic structure by increasing the number of subbands without further increasing the redundancy of the transform [8]. A 3D equivalent of this anisotropic structure exhibited coding advantages in prior work [8, 9]; as a consequence, we use this anisotropic DDWT exclusively here.

Although the 2D DDWT produces twice the data that the 2D DWT does, the DDWT requires fewer critical coefficients to efficiently represent the underlying signal [8]. To wit, Reeves and Kingsbury [10] proposed deliberately reducing the number of DDWT coefficients by discarding small magnitude coefficients and refining the remaining coefficients to compensate. This "noise-shaping" procedure is an iterative projection of signals between the original-signal domain and the DDWT domain. On each iteration, the signal is thresholded in the DDWT domain to remove small coefficients, and the remaining coefficients are compensated by the original-signal-domain error induced by the thresholding.

The noise-shaping process operates as illustrated in Fig. 2. During iteration i, the DDWT-domain coefficients are thresholded with a threshold θ_i. The spatial-domain error, amplified by gain α, is then added back to the DDWT coefficients. The first iteration starts with $\theta_0 = \theta^{\text{start}}$, and the threshold decreases to θ^{stop} in decrements of θ^{step}; i.e.,

$$\theta_{i+1} = \theta_i - \theta^{\text{step}}. \quad (1)$$

Throughout this work, we use $\theta^{\text{step}} = 1$ and $\alpha = 1.8$. We have found that performance of the noise-shaping procedure varies significantly with the values of θ^{start} and θ^{stop}; in the experimental results below, we optimize these values for each image and rate independently.

3. DDWT-BISK

The noise-shaping procedure outlined above ensures that significant coefficients are relatively sparse in the DDWT domain such that insignificant coefficients tend to cluster in large, spatially coherent regions. In order to efficiently code such spatially coherent regions of DDWT coefficients, we propose a modified version of the BISK algorithm [4]. BISK performs bitplane coding in which significant coefficients are located by recursive spatial partitioning. Specifically, k-d trees [11] are used to split sets of coefficients into two subsets of roughly equal size. Once a significant coefficient is located, its sign information is coded, and its magnitude is refined on successive passes. Significance, sign, and magnitude-refinement information are all coded with adaptive arithmetic coding.

In the proposed DDWT-BISK coder, the noise shaping of Fig. 2 is applied to produce sparse DDWT coefficients. The coder maintains multiple lists of insignificant sets (LIS)—sets having roughly the same size are stored in the same LIS list, and sets move between lists as they are split to smaller sizes. The coder is initialized by placing each subband from the two DDWT trees in an appropriate LIS list as a separate set, and then bitplane coding is performed with sorting and refinement passes. Sets containing at least one significant coefficient (significant sets) are split in two along the longest dimension of the set, and the resulting subsets are added back to an LIS list as two new sets to be recursively tested and split if necessary. Eventually, a significant set will be reduced to a single coefficient; at this point, the coefficient is removed from its LIS list and added to the list of significant pixels (LSP). The refinement pass then processes each coefficient in the LSP and outputs the current bitplane value of the coefficient magnitude. Sorting and refinement passes continue until the target bitstream length has been reached. The DDWT-BISK set-partitioning procedure is illustrated in Fig. 3.

4. ALGORITHM

The DDWT-BISK algorithm starts by performing noise shaping on the original image, $\mathcal{X}$, producing two trees of DDWT transform coefficients, $\mathcal{T}_1$ and $\mathcal{T}_2$. These trees are split into individual subbands which are then placed into the appropriate LIS. Afterward, the DDWT-BISK algorithm follows the common bitplane-coding paradigm consisting of sorting and refinement passes.

Specifically, assume a set of coefficients, $\mathcal{S}$, has been already decomposed $l_y(\mathcal{S})$ times vertically and $l_x(\mathcal{S})$ horizontally and resides in LIS$_i$, $i = l_y(\mathcal{S}) + l_x(\mathcal{S})$. Further assume that the number of rows and columns of $\mathcal{S}$ are $y(\mathcal{S})$ and $x(\mathcal{S})$, respectively. When $\mathcal{S}$ is split, the two resulting subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ will reside in LIS$_{i'}$, $i' = l_y(\mathcal{S}) + l_x(\mathcal{S}) + 1$. The split is either horizontal or vertical depending on whether $y(\mathcal{S}) < x(\mathcal{S})$ or not.

```
procedure DDWT-BISK($\mathcal{X}$, $\theta^{\text{start}}$, $\theta^{\text{stop}}$, $\theta^{\text{step}}$, $\alpha$)
  $\{\mathcal{T}_1, \mathcal{T}_2\} \leftarrow$ NoiseShape($\mathcal{X}$, $\theta^{\text{start}}$, $\theta^{\text{stop}}$, $\theta^{\text{step}}$, $\alpha$)
  Initialization($\mathcal{T}_1$, $\mathcal{T}_2$)
  $n \leftarrow$ max bitplane
  while (true)
    SortingPass()
    RefinementPass()
    $n \leftarrow n - 1$

procedure NoiseShape($\mathcal{X}$, $\theta^{\text{start}}$, $\theta^{\text{stop}}$, $\theta^{\text{step}}$, $\alpha$)
  $\theta \leftarrow \theta^{\text{start}}$
  $\{\mathcal{T}_1, \mathcal{T}_2\} \leftarrow$ DDWT($\mathcal{X}$)
  while $\theta \neq \theta^{\text{stop}}$
    $\{\mathcal{T}_1', \mathcal{T}_2'\} \leftarrow$ threshold coefficients in $\{\mathcal{T}_1, \mathcal{T}_2\}$ to $\theta$
    $\mathcal{X}' \leftarrow$ DDWT$^{-1}$($\{\mathcal{T}_1', \mathcal{T}_2'\}$)
    $\{\mathcal{T}_1, \mathcal{T}_2\} \leftarrow \{\mathcal{T}_1, \mathcal{T}_2\} +$ DDWT($\alpha(\mathcal{X} - \mathcal{X}')$)
    $\theta \leftarrow \theta - \theta^{\text{step}}$

procedure Initialization($\mathcal{T}_1$, $\mathcal{T}_2$)
  for $t \in \{1, 2\}$ do
    for each subband $\mathcal{S}$ in $\mathcal{T}_t$
      $l_y(\mathcal{S}) \leftarrow$ vertical transform level of $\mathcal{S}$
      $l_x(\mathcal{S}) \leftarrow$ horizontal transform level of $\mathcal{S}$
      append $\mathcal{S}$ to LIS$_{l_y(\mathcal{S})+l_x(\mathcal{S})}$
  LSP $\leftarrow \emptyset$

procedure SortingPass()
  $l =$ number of LIS lists
  while $l > 0$
    for each $\mathcal{S} \in$ LIS$_l$
      Process($\mathcal{S}$)
    $l \leftarrow l - 1$

procedure RefinementPass()
  for each $\mathcal{S} \in$ LSP
    output $n^{\text{th}}$ bitplane value of coefficient magnitude
```

The sorting pass determines the significance of set $\mathcal{S}$ by comparing the largest coefficient magnitude contained in the set to the current threshold. Sets without a significant coefficient are placed in an LIS, and, during the sorting pass, each set in an LIS is tested for significance against the current threshold. If the set becomes significant, it is split in two as described above. The two new sets are placed into an LIS, recursively tested for significance, and split again if needed. Let $\Gamma_n(\mathcal{S})$ be the significance state of set $\mathcal{S}$, such that $\Gamma_n(\mathcal{S}) = 1$ if $\mathcal{S}$ contains at least one coefficient magnitude greater than the current threshold, 2^n.

```
procedure Process($\mathcal{S}$)
  if $\mathcal{S} = \emptyset$
    remove $\mathcal{S}$ from $\text{LIS}_{l_y(\mathcal{S})+l_x(\mathcal{S})}$
  else
    output $\Gamma_n(\mathcal{S})$
    if $\Gamma_n(\mathcal{S}) = 1$
      remove $\mathcal{S}$ from $\text{LIS}_{l_y(\mathcal{S})+l_x(\mathcal{S})}$
      if $|\mathcal{S}| = 1$
        output sign of $\mathcal{S}$
        append $\mathcal{S}$ to LSP
      else
        Code($\mathcal{S}$)

procedure Code($\mathcal{S}$)
  $\{\mathcal{S}_1, \mathcal{S}_2\}$ = Partition($\mathcal{S}$)
  if $\mathcal{S}_1 \neq \emptyset$
    append $\mathcal{S}_1$ to $\text{LIS}_{l_y(\mathcal{S}_1)+l_x(\mathcal{S}_1)}$
    Process($\mathcal{S}_1$)
  append $\mathcal{S}_2$ to $\text{LIS}_{l_y(\mathcal{S}_2)+l_x(\mathcal{S}_2)}$
  Process($\mathcal{S}_2$)

procedure Partition($\mathcal{S}$)
  if $x(\mathcal{S}) \geq y(\mathcal{S})$
    split $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2$:
      $\mathcal{S}_1$: size $y(\mathcal{S}) \times \lfloor x(\mathcal{S})/2 \rfloor$
      $\mathcal{S}_2$: size $y(\mathcal{S}) \times \big(x(\mathcal{S}) - \lfloor x(\mathcal{S})/2 \rfloor\big)$
      $l_x(\mathcal{S}_1), l_x(\mathcal{S}_2) \leftarrow l_x(\mathcal{S}) + 1$
      $l_y(\mathcal{S}_1), l_y(\mathcal{S}_2) \leftarrow l_y(\mathcal{S})$
  else
    split $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2$:
      $\mathcal{S}_1$: size $\lfloor y(\mathcal{S})/2 \rfloor \times x(\mathcal{S})$
      $\mathcal{S}_2$: size $\big(y(\mathcal{S}) - \lfloor y(\mathcal{S})/2 \rfloor\big) \times x(\mathcal{S})$
      $l_y(\mathcal{S}_1), l_y(\mathcal{S}_2) \leftarrow l_y(\mathcal{S}) + 1$
      $l_x(\mathcal{S}_1), l_x(\mathcal{S}_2) \leftarrow l_x(\mathcal{S})$
```

DDWT-BISK uses adaptive arithmetic coding for set significance. Specifically, when set $\mathcal{S}$ is split into sets $\mathcal{S}_1$ and $\mathcal{S}_2$, and it happens that $\mathcal{S}_1$ is insignificant, then it is known that $\mathcal{S}_2$ is significant, and thus significance state $\Gamma_n(\mathcal{S}_2)$ is not coded to the bitstream. Otherwise, the coding of $\Gamma_n(\mathcal{S}_2)$ is conditioned on the fact that $\mathcal{S}_1$ was significant. The contexts used for set-significance coding within the DDWT-BISK algorithm are as follows:

```
$c(\mathcal{S}_1) \leftarrow$ CONTEXT_S1
if $\mathcal{S}_1 = \emptyset$ or $\Gamma_n(\mathcal{S}_1) = 0$
  $c(\mathcal{S}_2) \leftarrow$ CONTEXT_NOCODE
else
  $c(\mathcal{S}_2) \leftarrow$ CONTEXT_S2
```

Above, $c(\mathcal{S}_i)$ denotes the context that will be used to code $\Gamma_n(\mathcal{S}_i)$.

5. RESULTS

In our experiments, we code the grayscale images shown in Table 1 using both the proposed DDWT-BISK coder as well as JPEG2000, widely considered to be the current state of the art for still-image coding. DDWT-BISK uses the anisotropic DDWT decomposition described above in Sec. 2, while JPEG2000 uses a traditional dyadic DWT with the popular 9/7 biorthogonal wavelets. Both coders use 5 levels of decomposition. We measure distortion in terms of peak signal to noise ratio (PSNR) in dB and rate in terms of bits per pixel (bpp). We use the Kakadu[1] Version 5.1 implementation of JPEG-2000, while DDWT-BISK was developed from the QccPack[2] [12] implementation of BISK.

For the DDWT noise-shaping process described in Sec. 2, we optimize θ^{start} and θ^{stop} for each image and rate independently by exhaustive search over a fixed set of possible threshold values. Specifically, for a given image and a given rate R, we search over all pairs $(\theta^{\text{start}}, \theta^{\text{stop}}) \in \mathcal{A} \times \mathcal{A}$, where $\mathcal{A} = \{8k, k = 1, 2, \ldots, 32\}$, discarding pairs for which $\theta^{\text{stop}} > \theta^{\text{start}}$. For a given pair $(\theta^{\text{start}}, \theta^{\text{stop}})$, the iterative noise-shaping procedure as described in Sec. 2 is run, stepping the threshold down from θ^{start} to θ^{stop}. We then encode the resulting noise-shaped DDWT coefficients using DDWT-BISK at rate R. The pair $(\theta^{\text{start}}, \theta^{\text{stop}})$ yielding the highest PSNR is chosen, and this process is repeated for each image and rate R of interest.

Table 1 tabulates PSNR results for each image at a variety of rates, while Figs. 4 and 5 plot rate-distortion performance for the "barbara" and "mandrill" images, respectively. We see that DDWT-BISK consistently outperforms JPEG2000 at the lower bitrates, delivering, for example, 0.9 and 0.8 dB higher PSNR for "barbara" and "lenna," respectively, at 0.1 bpp. Gains are highest for "barbara," an image well-known for its substantial directional content.

Table 2 presents further investigation for the "barbara" image. Specifically, DDWT-BISK (using the anisotropic DDWT as described in Sec. 2) is compared to JPEG2000 using both a dyadic decomposition structure as well as an anisotropic wavelet-packet decomposition similar to that of Fig. 1 (critically sampled, of course) as supported by Part 2 of the JPEG2000 standard. Additionally, we compare to the original QccPack BISK implementation described in [4] using a traditional critically sampled dyadic DWT with the biorthogonal 9/7 filters. We observe that use of the anisotropic decomposition has little effect on the performance of JPEG2000; additionally, the original BISK implementation is outperformed somewhat by JPEG2000 using both decomposition structures. These observations suggest that the directionality resulting from the DDWT is a primary component to the superior performance of DDWT-BISK.

6. CONCLUSION

In this paper, we modified the BISK coder of [4] to provide efficient coding of DDWT coefficients. Because of the noise-shaping procedure imposed, the significant coefficients are ensured to be distributed rather sparsely throughout the DDWT domain, resulting in spatially coherent regions of insignificant coefficients. The set-partitioning process of the proposed DDWT-BISK coder effectively exploits this high degree of spatial coherency by splitting sets drawn from both DDWT transform trees. In order to increase directionality without further increasing the redundancy of the transform, an anisotropic decomposition structure is used. Experimental results reveal that the increased directionality resulting from the DDWT increases the performance of DDWT-BISK beyond that of the state-of-the-art JPEG2000 for low-bitrate coding, with the strongest gains resulting, as expected, for images such as "barbara" with strong directional content.

[1]http://www.kakadusoftware.com
[2]http://qccpack.sourceforge.net

Acknowledgment

The authors thank Y. Wang and B. Wang for helpful source code and discussions.

REFERENCES

[1] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 123–151, November 2005.

[2] N. G. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *Journal of Applied Computational Harmonic Analysis*, vol. 10, pp. 234–253, May 2001.

[3] J. E. Fowler, "The redundant discrete wavelet transform and additive noise," *IEEE Signal Processing Letters*, vol. 12, no. 9, pp. 629–632, September 2005.

[4] ——, "Shape-adaptive coding using binary set splitting with *k*-d trees," in *Proceedings of the International Conference on Image Processing*, vol. 2, Singapore, October 2004, pp. 1301–1304.

[5] I. W. Selesnick and K. Y. Li, "Video denoising using 2D and 3D dual-tree complex wavelet transforms," in *Wavelets: Applications in Signal and Image Processing X*, M. A. Unser, A. Aldroubi, and A. F. Laine, Eds. San Diego, CA: Proc. SPIE 5207, August 2003, pp. 607–618.

[6] N. Kingsbury, "A dual-tree complex wavelet transform with improved orthogonality and symmetry properties," in *Proceedings of the International Conference on Image Processing*, vol. 2, Vancouver, Canada, September 2000, pp. 375–378.

[7] F. Shi, B. Wang, I. W. Selesnick, and Y. Wang, "A new structure of 3-D dual-tree discrete wavelet transform and applications to video denoising and coding," in *Visual Communications and Image Processing*, J. G. Apostolopoulos and A. Said, Eds. San Jose, CA: Proc. SPIE 6077, January 2006, p. 60771C.

[8] B. Wang, Y. Wang, I. Selesnick, and A. Vetro, "Video coding using 3-D dual-tree wavelet transform," *EURASIP Journal on Image and Video Processing*, vol. 2007, 2007, article ID 42761, 15 pages.

[9] J. B. Boettcher and J. E. Fowler, "Video coding using a complex wavelet transform and set partitioning," *IEEE Signal Processing Letters*, vol. 14, no. 9, September 2007, to appear.

[10] T. H. Reeves and N. G. Kingsbury, "Overcomplete image coding using iterative projection-based noise shaping," in *Proceedings of the International Conference on Image Processing*, vol. 3, Rochester, NY, September 2002, pp. 597–600.

[11] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, September 1975.

[12] J. E. Fowler, "QccPack: An open-source software library for quantization, compression, and coding," in *Applications of Digital Image Processing XXIII*, A. G. Tescher, Ed. San Diego, CA: Proc. SPIE 4115, August 2000, pp. 294–301.

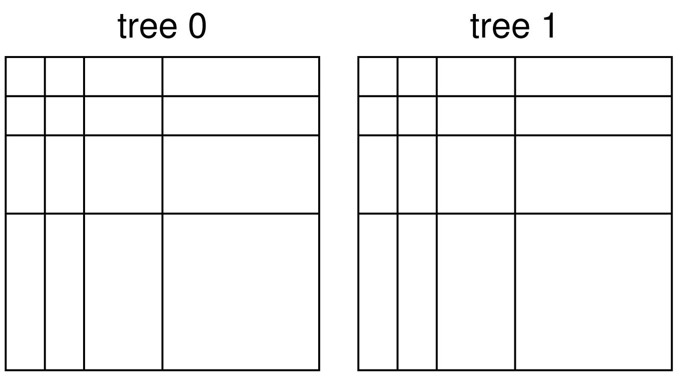

Figure 1: The anisotropic DDWT decomposition consisting of J-scale 1D DDWT decompositions applied separably both horizontally and vertically (illustrated for $J = 3$).

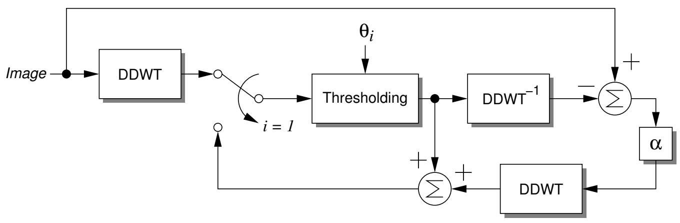

Figure 2: The noise-shaping procedure (adapted from [10]).

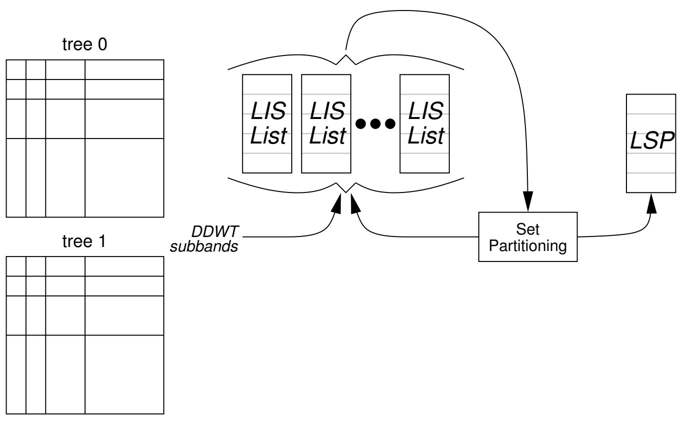

Figure 3: The partitioning procedure for DDWT-BISK.

Table 1: Rate-distortion performance.

PSNR (dB)

Rate (bpp)	"barbara" JPEG 2000	"barbara" DDWT-BISK	"lenna" JPEG 2000	"lenna" DDWT-BISK	"goldhill" JPEG 2000	"goldhill" DDWT-BISK	"mandrill" JPEG 2000	"mandrill" DDWT-BISK
0.1	24.7	**25.6**	29.9	**30.7**	27.8	**28.2**	21.5	**21.8**
0.25	28.4	**29.4**	34.1	**34.3**	30.5	**30.8**	23.3	**23.9**
0.5	32.2	**33.0**	**37.3**	37.1	**33.2**	33.1	25.8	**25.9**
0.75	34.9	**35.5**	**39.0**	38.7	**35.0**	34.8	27.6	**27.8**
1.0	**37.2**	**37.2**	**40.3**	39.9	**36.6**	36.1	**29.3**	29.0

Table 2: Rate-distortion performance for "barbara."

PSNR (dB)

Rate (bpp)	BISK (dyadic)	JPEG 2000 (dyadic)	JPEG 2000 (anisotropic)	DDWT-BISK
0.1	24.3	24.7	24.8	**25.6**
0.25	27.7	28.4	28.5	**29.4**
0.5	31.5	32.2	32.4	**33.0**
0.75	34.3	34.9	34.9	**35.5**
1.0	36.4	**37.2**	37.1	**37.2**

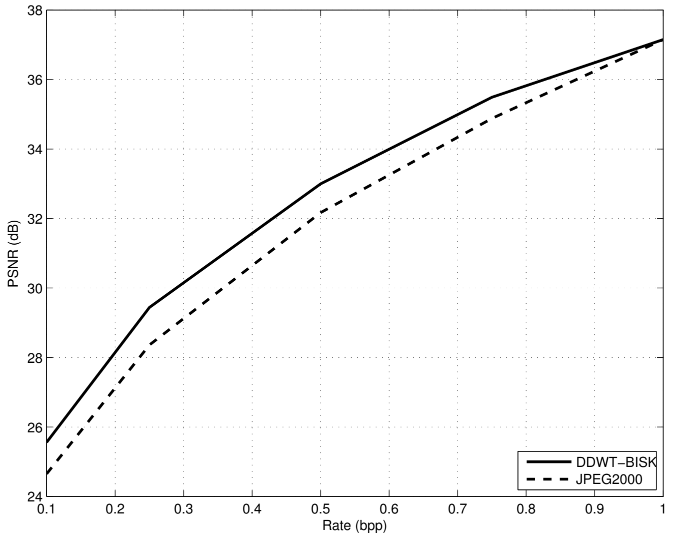

Figure 4: Rate-distortion performance for "barbara."

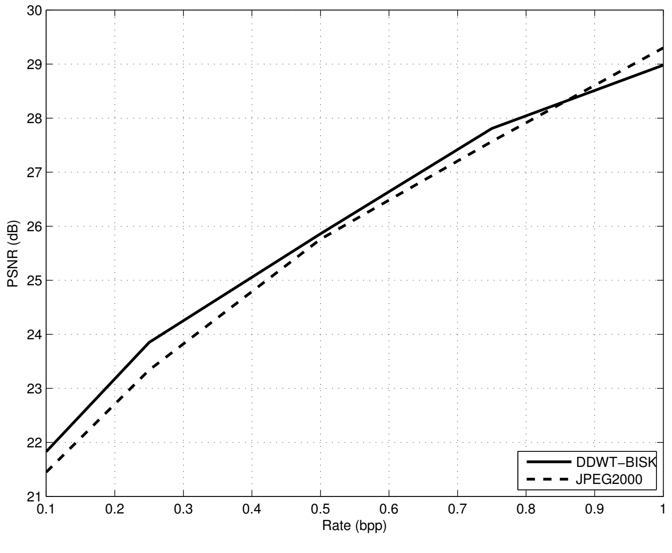

Figure 5: Rate-distortion performance for "mandrill."