

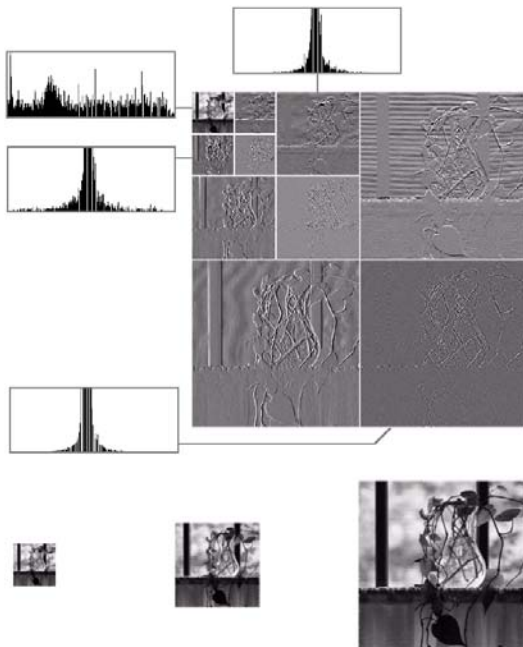
Image Compression

Part 2

- Outline
 - Wavelet-based image compression
 - Wavelets versus DCT
 - Wavelet families
 - Level of decomposition
 - Quantization
 - Wavelet baseline compression
 - Embedded zerotree coding (EZC)
 - Set-partitioning in hierarchical trees (SPIHT)
 - Compression standards
 - JPEG
 - JPEG2000
 - MPEG
 - Modifications of DCT-based coders (AGU and AGU-MHV)

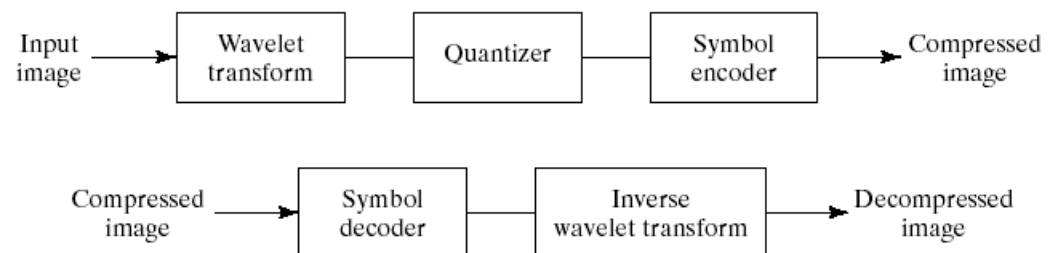
Wavelet-based compression

- Efficient decorrelation through multiscale decomposition with filter-banks
- Horizontal, vertical and diagonal details with zero mean and Laplacian-like distributions
- Many of the transform coefficients carry little visual information – susceptible for efficient coding
 - Intra-scale and inter-scale dependencies (correlations) to be exploited



Main difference with other transform coders is that there is no sub-image division. The localization is an inherent property of the wavelet transform

FIGURE 8.39 A wavelet coding system:
(a) encoder;
(b) decoder.



Wavelet versus DCT Coding

Cr1=34

Cr2=67

Cr1=34

Cr2=67

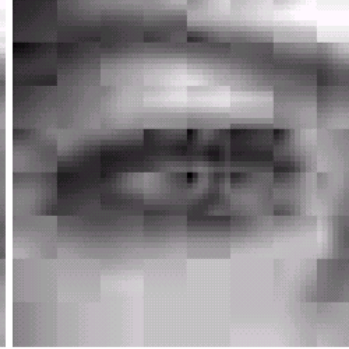
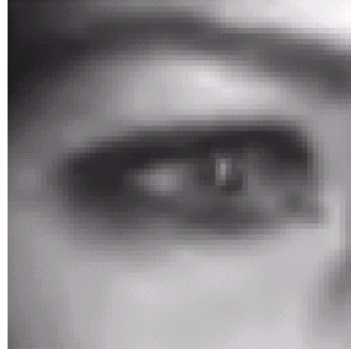
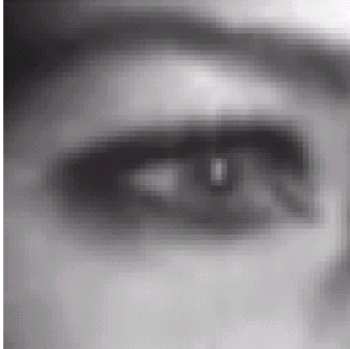
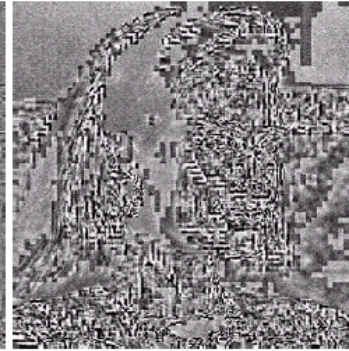
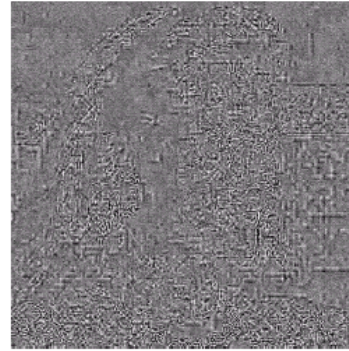
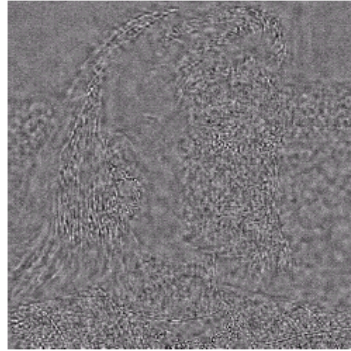
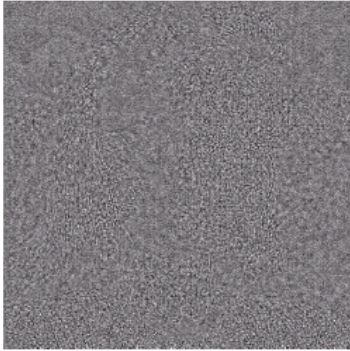
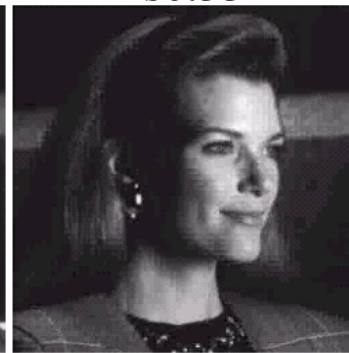
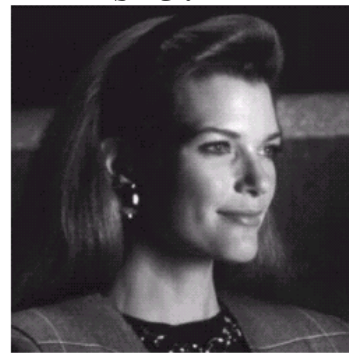
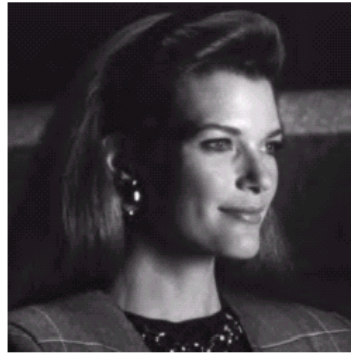
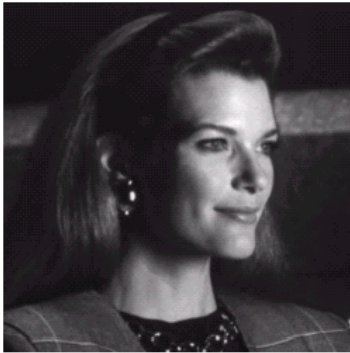
rms=2.29

rms=2.96

rms=3.42

rms=6.33

W
A
V
E
L
E
T



D
C
T

Wavelet versus DCT Coding



$Cr=108;$
 $rms=3.72;$

$Cr=167$
 $rms=4.73$

a b
c d
e f

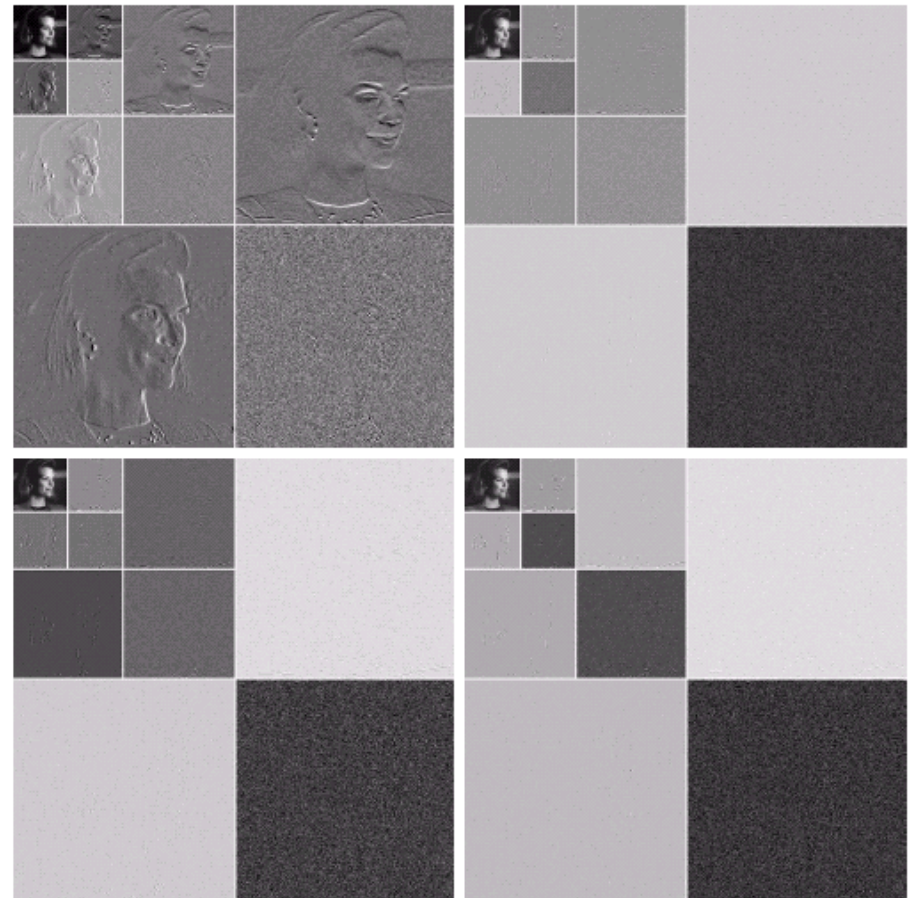
FIGURE 8.41 (a), (c), and (e) Wavelet coding results with a compression ratio of 108 to 1; (b), (d), and (f) similar results for a compression of 167 to 1.

Wavelet compression

Wavelet selection determines the complexity and the compression performance

Different wavelet families: Daubechies (orthogonal, maximal number or vanishing moments), CDF (biorthogonal, spline, symmetrical), symmlets, etc. The reconstruction functions (and the corresponding filters) are the most important

Wavelet	Filter Taps (Scaling + Wavelet)	Zeroed Coefficients
Haar (see Ex. 7.10)	2 + 2	46%
Daubechies (see Fig. 7.6)	8 + 8	51%
Symlet (see Fig. 7.24)	8 + 8	51%
Biorthogonal (see Fig. 7.37)	17 + 11	55%



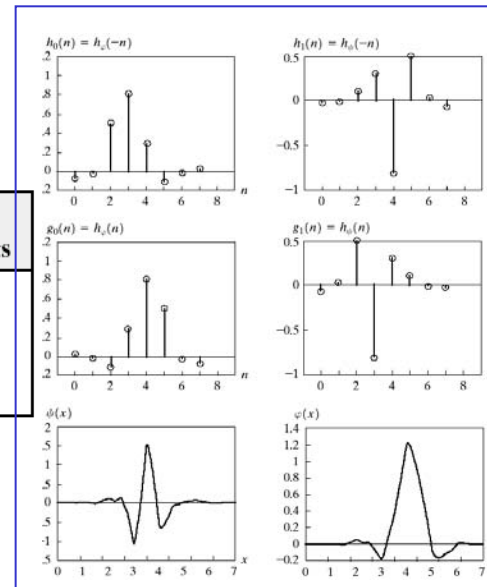
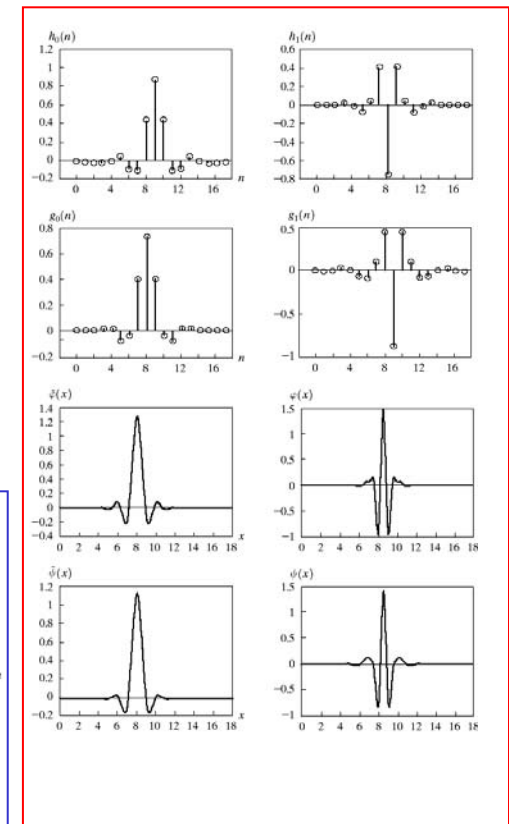
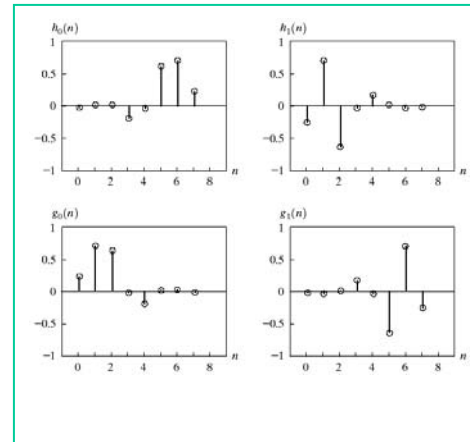
a b
c d

FIGURE 8.42 Wavelet transforms of Fig. 8.23 with respect to (a) Haar wavelets, (b) Daubechies wavelets, (c) symlets, and (d) Cohen-Daubechies-Feauveau biorthogonal wavelets.

Wavelet compression

Wavelet selection determines the complexity and the compression performance

Different wavelet families: Daubechies (orthogonal, maximal number of vanishing moments), CDF (biorthogonal, spline, symmetrical), symmlets, etc. The reconstruction functions (and the corresponding filters) are the most important ones.



Wavelet	Filter Taps (Scaling + Wavelet)	Zeroed Coefficients
Haar (see Ex. 7.10)	2 + 2	46%
Daubechies (see Fig. 7.6)	8 + 8	51%
Symlet (see Fig. 7.24)	8 + 8	51%
Biorthogonal (see Fig. 7.37)	17 + 11	55%

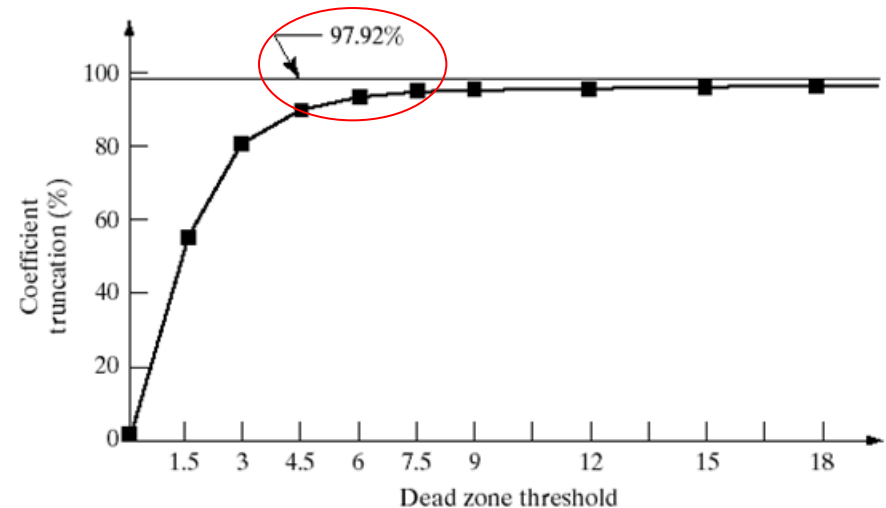
Wavelet compression

- Decomposition level selection: more levels – better decorrelation; however, more computations as well.
- Example: global thresholding of 25

Scales and Filter Bank Iterations	Approximation Coefficient Image	Truncated Coefficients (%)	Reconstruction Error (rms)
1	256 × 256	75%	1.93
2	128 × 128	93%	2.69
3	64 × 64	97%	3.12
4	32 × 32	98%	3.25
5	16 × 16	98%	3.27

TABLE 8.13
Decomposition level impact on wavelet coding the 512 × 512 image of Fig. 8.23.

- Quantizer design: the largest factor effecting compression. Two techniques applied:
 - Dead zone
 - Scale-adapting quantization interval



Wavelet baseline compression

- Zig – zag scanning of the wavelet coefficients (see Fig 6.6 from [2]) and indexing them
- A coefficient $w(m)$ is defined as *significant* with respect to a given threshold: $|w(m)| \geq T$. It is *insignificant* otherwise.
- Significance map is built: $s(m) = 1$ if $w(m)$ is significant and $s(m) = 0$ otherwise
- The significance map is encoded by a runlength method, i.e. 6 – bit sequences:
 - $0abcde$: run of 0 with length $(abcde)_2$ and $1abcde$: run of 1 with length $(abcde)_2$
- The significant coefficients are encoded, e.g. through Huffman or arithmetic coding
- **Problems:**
 - It is not possible to specify in advance the exact compression ratio (bit budget) or the exact error to be achieved
 - Progressive transmission is not possible

Wavelet compression: Embedded Zerotree Coding

- General idea: establish parent – offspring relationships between coefficients and order them by bit planes in order to transmit the more significant bits first.
 - ‘Embedded’ means that the stream can be truncated at any point.
 - Wavelet trees – set of coefficients from different scales that belong in the same spatial locality (see Fig. 5.36 and Fig.5.37 from [2])
 - Lowest (DC) frequency band forms the *reference signal*
 - Each lower frequency *parent node* has four higher frequency *offspring nodes*. All coefficients below a parent node in the same spatial locality are denoted as *descendants* (quadtree).
 - If a coefficient is insignificant, it is very likely that its offspring and descendants are insignificant as well.
-

Wavelet compression: Embedded Zerotree Coding

- Bit-plain coding
- The procedure can be stopped upon achieving the desired accuracy or the given bit budget

Step 1: Initialize. Choose initial threshold, $T = T_0$, such that *all* transform values satisfy $|w(m)| < T_0$ and at least one transform value satisfies $|w(m)| \geq T_0/2$.

Step 2: Update threshold. Let $T_k = T_{k-1}/2$.

Step 3: Significance pass. Scan through insignificant values using baseline algorithm scan order. Test each value $w(m)$ as follows:

If $|w(m)| \geq T_k$, then

Output sign of $w(m)$

Set $w_Q(m) = T_k$

Else if $|w(m)| < T_k$ then

Let $w_Q(m)$ retain its initial value of 0.

Step 4: Refinement pass. Scan through significant values found with higher threshold values T_j , for $j < k$ (if $k = 1$ skip this step). For each significant value $w(m)$, do the following:

If $|w(m)| \in [w_Q(m), w_Q(m) + T_k]$, then

Output bit 0

Else if $|w(m)| \in [w_Q(m) + T_k, w_Q(m) + 2T_k]$, then

Output bit 1

Replace value of $w_Q(m)$ by $w_Q(m) + T_k$.

Step 5: Loop. Repeat steps 2 through 4.

Wavelet compression: Embedded Zerotree Coding

- Bit-plain coding example
- Given two transform coefficients

- $w(1) = -9.5$
- $w(2) = 42$

Step 1: Initialize. $T_0 = 64$

First loop

Step 2: Update threshold: $T_1 = 32$.

Step 3: Significance pass: Output sign of $w(2)$; Set $w_Q(2) = 32$; $w_Q(1) = 0$

Second loop

Step 2: Update threshold: $T_2 = 16$.

Step 3: Significance pass: No outputs

Step 4: Refinement pass: Output bit 0

Third loop

Step 2: Update threshold: $T_2 = 8$.

Step 3: Significance pass: Output sign of $w(1)$; Set $w_Q(1) = 8$

Step 4: Refinement pass: Output bit 1, Replace $w_Q(2) = 32 + 8 = 40$

$w(1)$		-	0	0	1	1
$w(2)$	+	0	1	0	1	0

Wavelet compression: Embedded Zerotree Coding

- **Zerotree** (see Fig.6.8)
- For a given threshold the tree has insignificant values at each of its locations
- If a zerotree is found it is encoded by one symbol
- Zerotrees appear quite frequently (especially for higher thresholds) in wavelet domain (see Fig. 6.9)
- Replace the significance pass in the bit-plane encoding with the **EZW Step 3**
- During a search through a quadtree, values that were found to be significant at higher thresholds are treated as zeros. All descendants of a root of a zerotree are skipped in the rest of the scanning at this threshold.

EZW Step 3: Significance pass. Scan through insignificant values using baseline algorithm scan order. Test each value $w(m)$ as follows:

```
If  $|w(m)| \geq T_k$ , then
  Output the sign of  $w(m)$ 
  Set  $w_Q(m) = T_k$ 
Else if  $|w(m)| < T_k$  then
  Let  $w_Q(m)$  remain equal to 0
  If  $m$  is at 1st level, then
    Output  $I$ 
  Else
    Search through quadtree having root  $m$ 
    If this quadtree is a zerotree, then
      Output  $R$ 
    Else
      Output  $I$ .
```

Wavelet compression: Embedded Zerotree Coding

- **Zerotree** example (see Fig.6.8) :
- First loop: $T_0=64$; $T_1=32$; the coder output is:

+ - I R + RRRRI RRI I I I I + I I

With two values ± 32 for sign-marked symbols

- Second loop: $T_2=16$; the coder output is:

- + RRR - RRRRRRRI I I + I I I I 1 0 1 0

The quantized values are shown in Fig. 6.10

- **Four symbols + - R and I**: coded with 01, 00, 10, and 11. The decoder can infer when the significance pass is complete, then simple bits 0 and 1 follows. Additional arithmetic coding can be applied as well.
- **In the decoder**: round the values to the mid point of the interval they were last found to belong to (i.e. add half of the last threshold value). This reduces the MSE

EZW Step 3: Significance pass. Scan through insignificant values using baseline algorithm scan order. Test each value $w(m)$ as follows:

If $|w(m)| \geq T_k$, then

Output the sign of $w(m)$

Set $w_Q(m) = T_k$

Else if $|w(m)| < T_k$ then

Let $w_Q(m)$ remain equal to 0

If m is at 1st level, then

Output I

Else

Search through quadtree having root m

If this quadtree is a zerotree, then

Output R

Else

Output I .

Wavelet compression: Set Partitioning in Hierarchical Trees (SPIHT)

- **Spatial-orientation tree wavelet (STW)** algorithm: based on state transition model
 - Instead of coding R and I output to mark locations (as in EZW), it uses states I_R , I_V , S_R , and S_V and outputs code for state-transitions such as $I_R \rightarrow I_V$, $S_R \rightarrow S_V$, etc.
- Basis definitions
 - Set $D(m)$: for a given index m in the baseline scan order, if m is either at the first level or at the all-lowpass level, then $D(m)$ is the empty set \emptyset . Otherwise, if m is at the j -th level for $j > 1$, then $D(m) = \{\text{Descendants of index } m \text{ in quadtree with root } m\}$.
 - Significance function
$$S(m) = \begin{cases} \max_{n \in D(m)} |w(n)|, & \text{if } D(m) \neq \emptyset \\ \infty & \text{if } D(m) = \emptyset \end{cases}$$
 - States (see Fig. 6.11)
 - $m \in I_R$ if and only if $|w(m)| < T, S(m) < T$; $m \in I_V$ if and only if $|w(m)| < T, S(m) \geq T$
 - $m \in S_R$ if and only if $|w(m)| \geq T, S(m) < T$ $m \in S_V$ if and only if $|w(m)| \geq T, S(m) \geq T$

Wavelet compression: STW encoding

Step 1: Initialize. Choose initial threshold, $T = T_0$, such that *all* transform values satisfy $|w(m)| < T_0$ and at least one transform value satisfies $|w(m)| \geq T_0/2$. Assign all indices for the L -th level, where L is the number of levels in the wavelet transform, to the *dominant list* (this includes all locations in the all-lowpass subband as well as the horizontal, vertical, and diagonal subbands at the L -th level). Set the *refinement list* of indices equal to the empty set.

Step 2: Update threshold. Let $T_k = T_{k-1}/2$.

Step 3: Dominant pass. Use the following procedure to scan through indices in the dominant list (which can change as the procedure is executed).

Do

 Get next index m in dominant list

 Save old state $S_{\text{old}} = S(m, T_{k-1})$

 Find new state $S_{\text{new}} = S(m, T_k)$

 Output code for state transition $S_{\text{old}} \rightarrow S_{\text{new}}$

 If $S_{\text{new}} \neq S_{\text{old}}$ then do the following

 If $S_{\text{old}} \neq S_R$ and $S_{\text{new}} \neq I_V$ then

 Append index m to refinement list

 Output sign of $w(m)$ and set $w_Q(m) = T_k$

 If $S_{\text{old}} = I_V$ and $S_{\text{new}} = S_R$ then

 Append child indices of m to dominant list

 If $S_{\text{new}} = S_V$ then

 Remove index m from dominant list

Loop until end of dominant list

Wavelet compression: STW encoding

Step 4: *Refinement pass.* Scan through indices m in the refinement list found with higher threshold values T_j , for $j < k$ (if $k = 1$ skip this step). For each value $w(m)$, do the following:

If $|w(m)| \in [w_Q(m), w_Q(m) + T_k]$, then

Output bit 0

Else if $|w(m)| \in [w_Q(m) + T_k, w_Q(m) + 2T_k]$, then

Output bit 1

Replace value of $w_Q(m)$ by $w_Q(m) + T_k$.

Step 5: *Loop.* Repeat steps 2 through 4.

Example (see Fig. 6.8 and Fig. 6.12). Compare the three quadrees in dashed boxes. EZW gives $+IIII$, $-IIII$, and $+RRRR$. STW gives $+S_R$, $-S_R$, and $+S_R$.

Wavelet compression: SPIHT

- Basis definitions

- Set of indices: $D(m)=\{\text{Descendent indices of the index } m\}$; $C(m)=\{\text{Child indices of } m\}$;
 $G(m)=D(m)-C(m)=\{\text{Grandchildren on } m, \text{ i.e. descendants which are not children }\}$; H – indices for the L -th (last) level of WT. This includes all locations in the all-lowpass subband plus three detail subbands at the L -th level.

- Significance: For a given set of indices I :
$$S_T(I) = \begin{cases} 1 & \text{if } \max_{n \in I} |w(n)| \geq T \\ 0 & \text{if } \max_{n \in I} |w(n)| < T \end{cases}$$

- SPIHT keeps track on the states of indices by means of three lists: *list of insignificant sets* (LIS), *list of insignificant pixels* (LIP), and *list of significant pixels* (LSP)

Wavelet compression: SPIHT

Step 1: Initialize. Choose initial threshold T_0 such that *all* transform values satisfy $|w(m)| < T_0$ and at least one value satisfies $|w(m)| \geq T_0/2$. Set LIP equal to H , set LSP equal to \emptyset , and set LIS equal to all the indices in H that have descendants (assigning them all type D).

Step 2: Update threshold. Let $T_k = T_{k-1}/2$.

Wavelet compression: SPIHT

Step 3: Sorting pass. Proceed as follows:

```

For each  $m$  in LIP do:
  Output  $S_k[m]$ 
  If  $S_k[m] = 1$  then
    Move  $m$  to end of LSP
    Output sign of  $w(m)$ ; set  $w_Q(m) = T_k$ 
Continue until end of LIP
For each  $m$  in LIS do:
  If  $m$  is of type D then
    Output  $S_k[D(m)]$ 
    If  $S_k[D(m)] = 1$  then
      For each  $n \in C(m)$  do:
        Output  $S_k[n]$ 
        If  $S_k[n] = 1$  then
          Append  $n$  to LSP
          Output sign of  $w(n)$ ; set  $w_Q(n) = T_k$ 
        Else If  $S_k[n] = 0$  then
          Append  $n$  to LIP
    If  $G(m) \neq \emptyset$  then
      Move  $m$  to end of LIS as type G
  Else
    Remove  $m$  from LIS
Else If  $m$  is of type G then
  Output  $S_k[G(m)]$ 
  If  $S_k[G(m)] = 1$  then
    Append  $C(m)$  to LIS, all type D indices
    Remove  $m$  from LIS
Continue until end of LIS

```

Notice that the set LIS can undergo many changes during this procedure, it typically does not remain fixed throughout.

Wavelet compression: SPIHT

Step 4: *Refinement pass.* Scan through indices m in LSP found with higher threshold values T_j , for $j < k$ (if $k = 1$ skip this step). For each value $w(m)$, do the following:

If $|w(m)| \in [w_Q(m), w_Q(m) + T_k]$, then

Output bit 0

Else if $|w(m)| \in [w_Q(m) + T_k, w_Q(m) + 2T_k]$, then

Output bit 1

Replace value of $w_Q(m)$ by $w_Q(m) + T_k$.

Step 5: *Loop.* Repeat steps 2 through 4.

Image Compression Standards

- **JPEG** (stands for Joint Photographic Experts Group). Three different coding systems: *baseline coding system*; *extended coding system*, and *lossless independent coding system*
 - Baseline coding system
 - DCT-based: 8x8 block-transform
 - Thresholding-based quantization
 - Zig-zag based reordering to obtain 1-D sequence of quantized coefficients
 - Huffman coding of AC coefficients
 - Difference coding of DC coefficients
-

Image Compression Standards

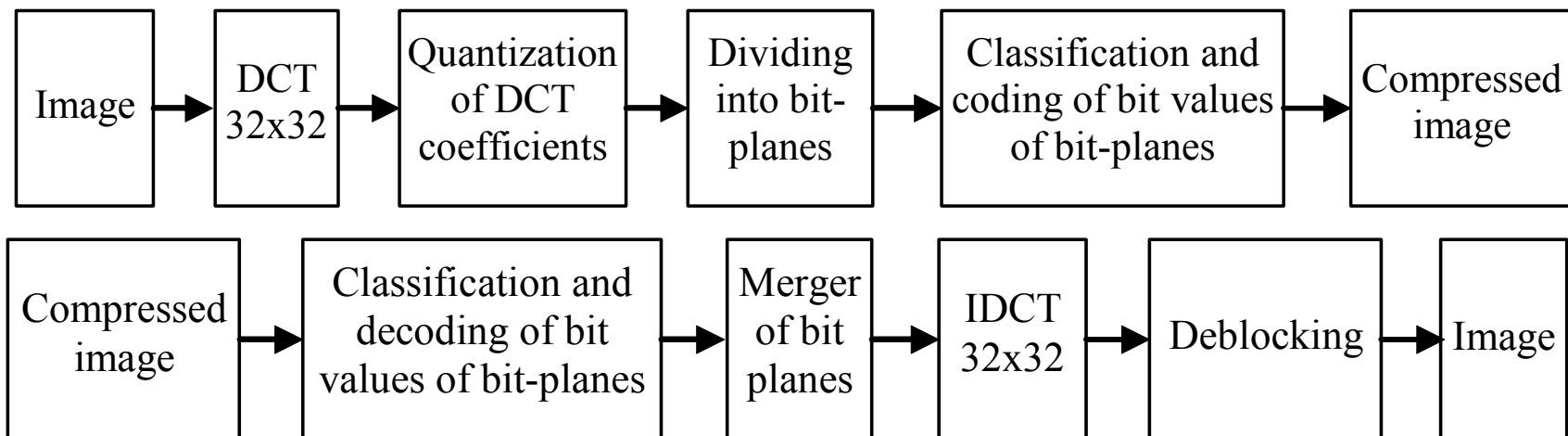
- **JPEG2000** Designed to overcome the drawbacks of JPEG. It offers improved compression efficiency, progressive lossy to lossless performance within a single data stream, etc.
 - Scalability (*compress once but decompress in many ways*)
 - Resolution scalability
 - Distortion (SNR) scalability – full resolution but successively lower quality
 - Spatial accessibility
 - Component accessibility
 - Technology
 - DWT-based
 - Embedded block coding with optimized truncation (EBCOT)
 - Decorrelating ‘color’ transform for multi-component images
 - RGB to YCbCr for lossy or RGB to RCT (reversible color transform) for lossless
-

Image Compression Standards

- DWT
 - Two sets of biorthogonal (symmetrical, with linear phase) filters: (5,3) filterbank and (9,7) filterbank. The numbers shown the length of the lowpass / highpass filters
 - Realization based on the lifting scheme
 - Enables efficient lossy and lossless compression
 - Achieves computational and memory savings
 - Maps integers to integers
 - Embedded block coding
 - DWT subbands are partitioned into smaller 32x32 or 64x64 ‘code-blocks’, each of which is coded independently
 - Scalar ‘dead-zone’ quantizers and bit-plane coding
 - Subsequent arithmetic coding with 18 different adaptive probability models
 - Fractional bit planes
 - Three ‘fractional bit-plane’ passes per bit plane. Most important coefficients coded first.
- Intra-scale prediction based encoding
-

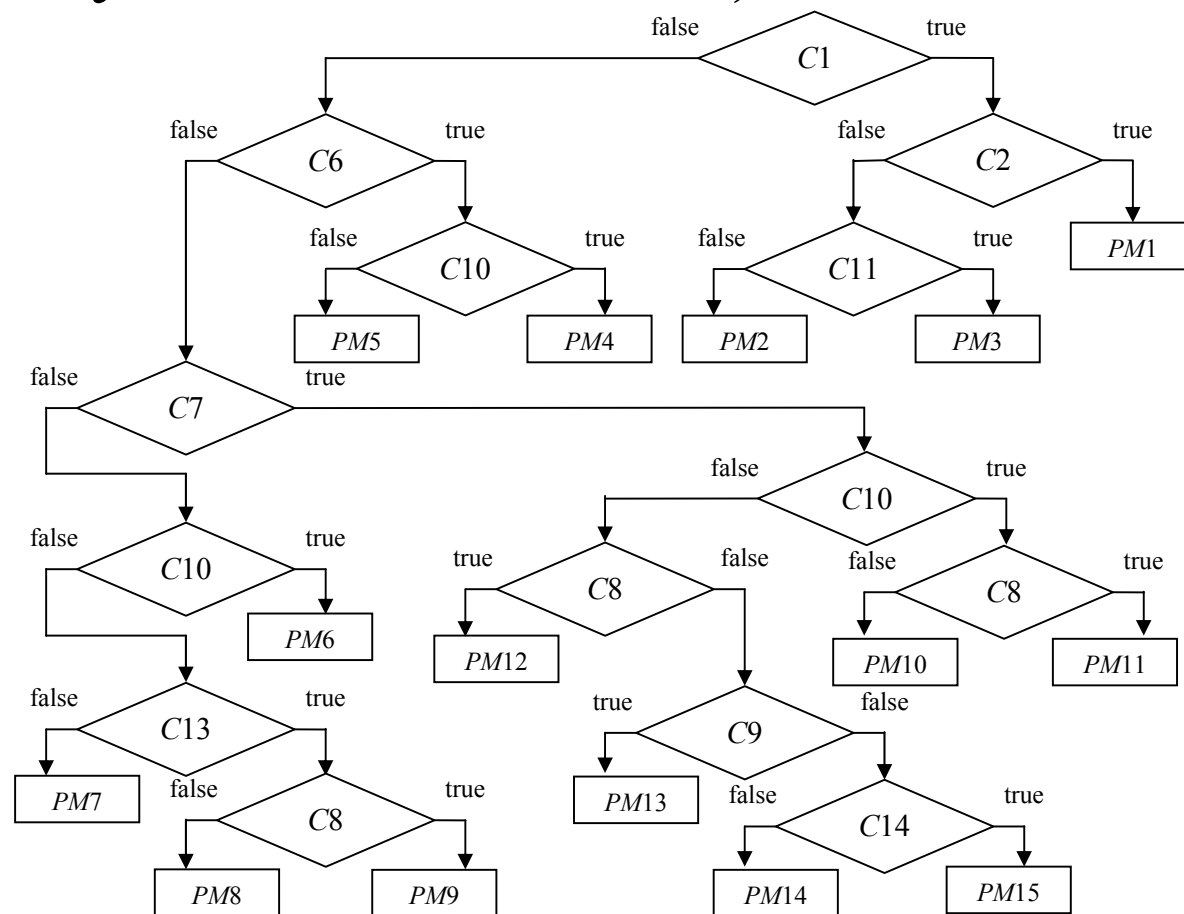
Modifications of DCT-based coders

- Considering DCT as a basis for image compression, it is clear that the block size 8×8 is not the only possible choice
- 32×32 DCT-based AGU coder (see <http://www.cs.tut.fi/~karen/agucoder.htm>) provides the quality of decoded images better than JPEG2000



DCT-based coders (AGU coder)

- Fig. below presents the flowchart of bit value classification by checking conditions (PMX - probability model number X).

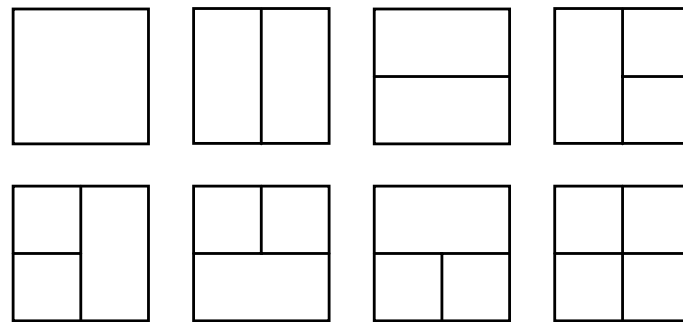


Modifications of DCT-based coders (AGU with variable block size)

- The fact that the use of different block sizes could produce higher PSNR depending upon image characteristics and CR motivates the use of Partition Schemes (PS)
 - PS have been utilized in fractal image compression
 - Horizontal-vertical PS are among most popular ones (simple to optimize)
-

Modifications of DCT-based coders (cont.)

- PS data can be coded in the following way. Note that each square block can be either remained or divided into several other blocks.



While optimizing the PS, one has to choose to code a given image fragment as one block (denote such decision as C1) or fragment division into several blocks and their separate coding (C2).

Modifications of DCT-based coders (cont.)

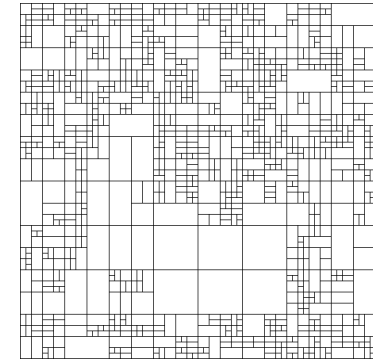
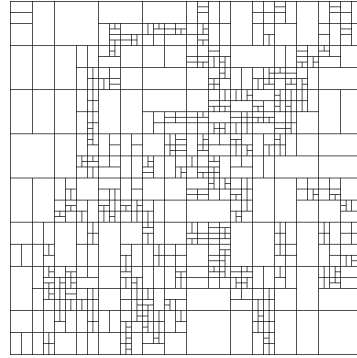
- Denote the size of image coded fragment in case of its compression as one block as S_1 , and as S_2 - size of this fragment in case of its compression as several blocks. Let M_1 and M_2 be corresponding MSE values. The four variants of conditions for the considered measures of coding efficiency are possible, namely:

$$\left\{ \begin{array}{lll} (S_1 \leq S_2), & (M_1 \leq M_2), & V1 \\ (S_1 \geq S_2), & (M_1 \geq M_2), & V2 \\ (S_1 < S_2), & (M_1 > M_2), & V3 \\ (S_1 > S_2), & (M_1 < M_2), & V4 \end{array} \right.$$

Modifications of DCT-based coders (cont.)

- For the variant $V1$ clearly the choice is $C1$ since both measures evidence in its favor (recall that one has to reduce both S and M). Similarly, for the variant $V2$ the choice $C2$ is obviously preferable.
- For variants $V3$ and $V4$ it is necessary to understand what is better – to provide smaller data size with worse compression quality or the larger size for better quality? To give an answer let us introduce additional criterion expressed as: $GR=(S2-S1)/(M1-M2)$. If $GR < GRM$, where GRM is the averaged value GR for entire image, then we propose to undertake the decision $C2$. Otherwise, the decision $C1$ is selected.

Modifications of DCT-based coders (cont.)



- Variable-size DCT-based AGU coder [6] performs even better (see “**High-Quality DCT-Based Image Compression Using Partition Schemes**”, Ponomarenko, N. N.; Egiazarian, K. O.; Lukin, V. V.; Astola, J. T.; IEEE Signal Processing Letters, Vol. 14, Issue 2, Feb. 2007, pp:105 – 108)

Comparative analysis of coding efficiency

	CR=8					CR=16				
	JPEG 2000	AGU 8x8	AGU 16x16	AGU 32x32	AGU MHV	JPEG 2000	AGU 8x8	AGU 16x16	AGU 32x32	AGU MHV
Lenna	40.33	40.32	40.59	40.5	40.85	37.27	37.02	37.48	37.51	37.86
Barbara	38.07	38.34	39.13	39.26	39.91	32.87	33.34	34.38	34.65	35.28
Baboon	29.11	29.47	29.71	29.70	30.27	25.57	25.79	26.07	26.12	26.39
Goldhill	36.54	36.77	37.02	37.03	37.38	33.24	33.29	33.61	33.65	33.81
Peppers	38.17	38.34	38.50	38.33	38.91	35.80	36.61	35.77	35.55	36.04
Patterns	35.79	35.90	37.38	38.55	43.81	28.46	26.92	29.16	30.66	32.51

	CR=32					CR=64				
	JPEG 2000	AGU 8x8	AGU 16x16	AGU 32x32	AGU MHV	JPEG 2000	AGU 8x8	AGU 16x16	AGU 32x32	AGU MHV
Lenna	34.15	33.66	34.44	34.50	34.75	31.02	30.10	31.32	31.52	31.62
Barbara	28.89	29.16	30.30	30.77	31.21	25.87	25.92	26.97	27.55	27.53
Baboon	23.18	23.31	23.60	23.69	23.77	21.68	21.61	21.89	22.01	22.04
Goldhill	30.53	30.49	30.99	31.09	31.22	28.49	28.25	28.84	28.97	29.03
Peppers	33.54	33.14	33.57	33.32	33.95	30.79	30.09	30.97	30.90	31.52
Patterns	22.60	20.75	23.05	25.13	26.39	18.67	17.39	19.62	22.08	22.25

References

- [1] R. Gonzalez and R. Woods, “Digital Image Processing”, 2nd ed.", Prentice-Hall, 2002, www.imageprocessingbook.com
 - [2] K. Rao and P. Yip, “The Transform and Data Compression Handbook”, CRC Press, 2001, http://www.engnetbase.com/ejournals/books/book_summary/summary.asp?id=431
 - [3] J. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients” *IEEE Trans. on Signal Processing*, vol. 41, pp. 3445–3462, 1993.
 - [4] A. Said and W. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees”, *IEEE Trans. on Circuits Syst. Video Tech.*, vol. 6, pp. 243–250, 1996.
 - [5] D. Taubman and M. Marcellin, “JPEG2000: Standard for interactive imaging”, Proceedings of IEEE, Vol. 90, pp. 1336–1357.
 - [6] Ponomarenko, N. N.; Egiazarian, K. O.; Lukin, V. V.; Astola, J. T.; “High-Quality DCT-Based Image Compression Using Partition Schemes”, *IEEE Signal Processing Letters*, Vol. 14, Issue 2, Feb. 2007, pp:105 – 108
-