Multimedia Software Systems CS4551

Wavelet-Based Coding

Introduction

- Decomposing the input signal into components that are easier to deal with allows us to apply coding technique suitable for each component and improve compression performance.
  - Recall DCT
- Wavelet Transform
  - Seeks to represent a signal with good resolution in both time and frequency, by using a set of basis functions called wavelets.

Wavelet Transform

- There are two types of wavelet transforms: the continuous wavelet transform (CWT) and the discrete wavelet transform (DWT).
- **The Discrete Wavelet Transform (DWT)**
  - Operates on discrete samples of the input signal.
  - Similar to DCT or DFT
  - E.g Harr Wavelet Transform (Simplest wavelet transform) – form averages and differences of a sequence of float values

Wavelet Transform Example - Discrete Haar Wavelet Transform

- Suppose we are given the following input sequence,

  $\{x_{n,i}\} = \{10, 13, 25, 26, 29, 21, 7, 15\}$

  where $i \in [0..7]$ and $n=3$ is the level of a pyramid.

- Consider the transform that replaces the original sequence with its pairwise average $x_{n-1,i}$ and difference $d_{n-1,i}$ defined as follows:

$$x_{n-1,i} = \frac{x_{n,2i} + x_{n,2i+1}}{2}$$
$$d_{n-1,i} = \frac{x_{n,2i} - x_{n,2i+1}}{2}$$

The averages and differences are applied only on consecutive *pairs* of input sequences whose first element has an even index. Therefore, the number of elements in each set $\{\boldsymbol{x_{n-1,i}}\}$ and $\{\boldsymbol{d_{n-1,i}}\}$ is exactly half of the number of elements in the original sequence.

Wavelet Transform Example - Discrete Haar Wavelet Transform

- Form a new sequence having length equal to that of the original sequence by concatenating the two sequences $\{x_{n-1,i}\}$ and $\{d_{n-1,i}\}$.
- From the input $\{x_{n,i}\}$, the resulting sequence $\{x_{n-1,i}, d_{n-1,i}\}$ is below:

$\{x_{n,i}\} = \{10, 13, 25, 26, 29, 21, 7, 15\}$, $n=3$ and $i=0..7$

to

$\{x_{n-1,i}, d_{n-1,i}\} = \{11.5, 25.5, 25, 11, -1.5, -0.5, 4, -4\}$, $n-1=2$ and $i=0..3$

Wavelet Transform Example - Discrete Haar Wavelet Transform

- This sequence has exactly the same number of elements as the input sequence — the transform did not increase the amount of data.
- Since the first half of the above sequence contain averages from the original sequence, we can view it as a **coarser approximation** to the original signal.
- The second half of this sequence can be viewed as the **details** or approximation errors of the first half.

Reconstruction

- It is easily verified that the original sequence can be reconstructed from the transformed sequence using the relations:

$$x_{n,2i} = x_{n-1,i} + d_{n-1,i}$$
$$x_{n,2i+1} = x_{n-1,i} - d_{n-1,i}$$

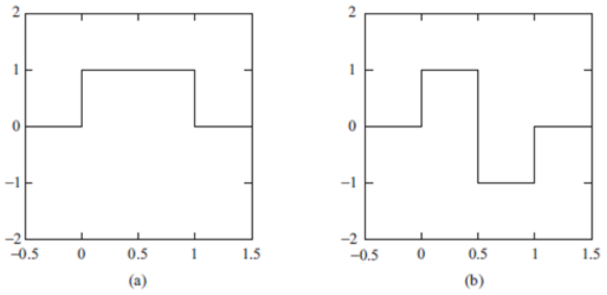
Discrete Haar Wavelet Transform

- At each level, less information will be retained in the beginning elements of the transformed signal sequence. When we reach pyramid level 0, we end of with the sequence average stored in the first element.

$\{x_{n,i}\}$ = {10, 13, 25, 26, 29, 21, 7, 15}, n=3 and i=0..7
$\{x_{n-1,i}, d_{n-1,i}\}$= {11.5, 25.5, 25, 11, -1.5,-0.5,4,-4}, n-1 and i=0..3
$\{x_{n-2,i}, d_{n-2,i}\}$= {18.5,18, -7,7, -1.5,-0.5,4,-4}, n-2 and i=0..1
$\{x_{n-3,i}, d_{n-3,i}\}$= {18.25,0.25, -7,7, -1.5,-0.5,4,-4}, n-3 and i=0

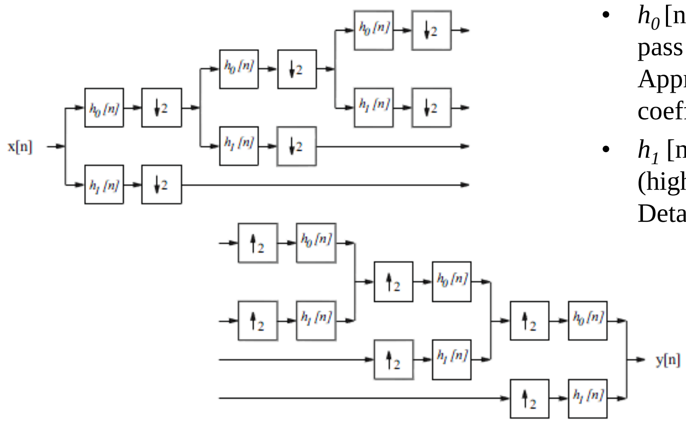
Discrete Haar Wavelet Transform

- Averaging and differencing are carried out by appying a so-called **scaling** function and **wavelet** function along the signal.



Haar Transform: (a) scaling function, (b) wavelet function.

Block Diagram of 1D Wavelet Transform

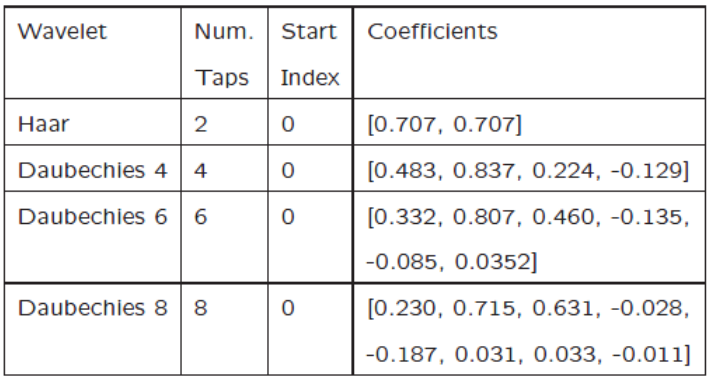


- h_0[n] : scaling (low-pass filter). Approximation coefficients
- h_1 [n] : wavelet (high-pass filter) Detail coefficients

Wavelet Filters

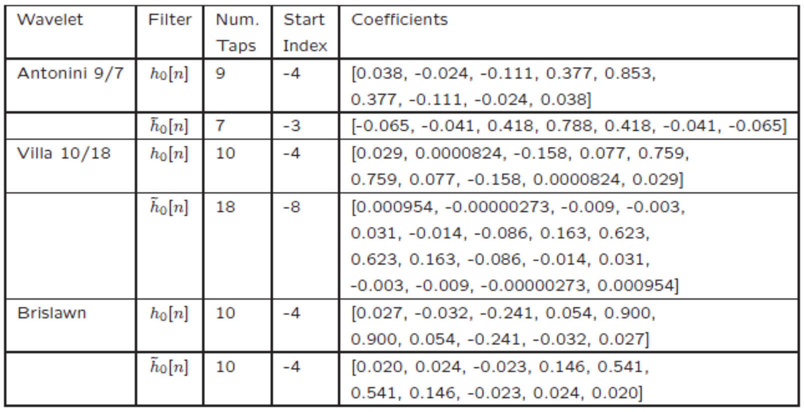
- The filterbank implementation of wavelets can be interpreted as computing the wavelet coefficients of a discrete set of child wavelets for a given mother wavelet.
  - Discrete Haar wavelet has mother wavelet [1 , − 1]. Then the dilated, reflected, and normalized version of this wavelet is h_1 [n] = 1 /sqrt(2) [− 1 , 1], which is the highpass decomposition filter for the discrete Haar wavelet transform.
- For orthonormal wavelets, the forward transform and its inverse are transposes of each other and the analysis filters are identical to the synthesis filters.
- Without orthogonality, the wavelets for analysis and synthesis are called "biorthogonal". The synthesis filters are not identical to the analysis filters. We denote them as $\tilde{h}_0[n]$ and $\tilde{h}_1[n]$.

Wavelet Filters - Orthogonal



Wavelet	Num. Taps	Start Index	Coefficients
Haar	2	0	[0.707, 0.707]
Daubechies 4	4	0	[0.483, 0.837, 0.224, -0.129]
Daubechies 6	6	0	[0.332, 0.807, 0.460, -0.135, -0.085, 0.0352]
Daubechies 8	8	0	[0.230, 0.715, 0.631, -0.028, -0.187, 0.031, 0.033, -0.011]

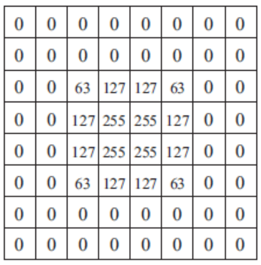
Wavelet Filters - Biorthogonal



Wavelet	Filter	Num. Taps	Start Index	Coefficients
Antonini 9/7	$h_0[n]$	9	-4	[0.038, -0.024, -0.111, 0.377, 0.853, 0.377, -0.111, -0.024, 0.038]
	$\tilde{h}_0[n]$	7	-3	[-0.065, -0.041, 0.418, 0.788, 0.418, -0.041, -0.065]
Villa 10/18	$h_0[n]$	10	-4	[0.029, 0.0000824, -0.158, 0.077, 0.759, 0.759, 0.077, -0.158, 0.0000824, 0.029]
	$\tilde{h}_0[n]$	18	-8	[0.000954, -0.00000273, -0.009, -0.003, 0.031, -0.014, -0.086, 0.163, 0.623, 0.623, 0.163, -0.086, -0.014, 0.031, -0.003, -0.009, -0.00000273, 0.000954]
Brislawn	$h_0[n]$	10	-4	[0.027, -0.032, -0.241, 0.054, 0.900, 0.900, 0.054, -0.241, -0.032, 0.027]
	$\tilde{h}_0[n]$	10	-4	[0.020, 0.024, -0.023, 0.146, 0.541, 0.541, 0.146, -0.023, 0.024, 0.020]

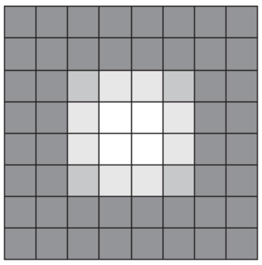
2D Discrete Haar Wavelet Transform

- Apply 1D transform to the rows and columns of the 2D input (e.g 8x8 input image).



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	63	127	127	63	0	0
0	0	127	255	255	127	0	0
0	0	127	255	255	127	0	0
0	0	63	127	127	63	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

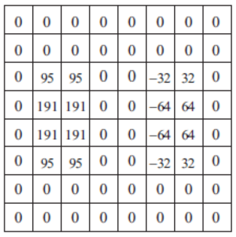
(a)



(b)

Input image for the 2D Haar Wavelet Transform.
(a) The pixel values. (b) Shown as an 8 × 8 image.

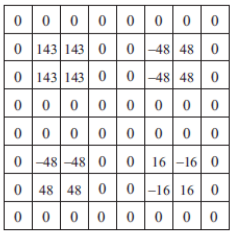
2D Discrete Haar Wavelet Transform



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	95	95	0	0	–32	32	0
0	191	191	0	0	–64	64	0
0	191	191	0	0	–64	64	0
0	95	95	0	0	–32	32	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

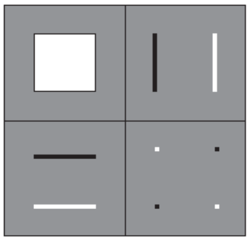
Intermediate output of the 2D Haar Wavelet Trans- form.

after performing the wavelet transform on the rows



0	0	0	0	0	0	0	0
0	143	143	0	0	–48	48	0
0	143	143	0	0	–48	48	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	–48	–48	0	0	16	–16	0
0	48	48	0	0	–16	16	0
0	0	0	0	0	0	0	0

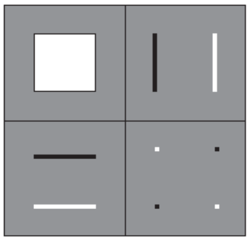
Output of the first level of the 2D Haar Wavelet Transform.



A simple graphical illustration of Wavelet Transform.

2D Discrete Haar Wavelet Transform

- The transform naturally divide the 2D image into four quadrants.
  - Upper left quadrant: the averaged coefficients from both the horizontal and vertical passes. It is low-pass-filtered version of the original image.
  - Upper right quadrant: vertical averages of the horizontal differences. It represents vertical edge information of the original image.
  - Lower left quadrant: vertical differences of the horizontal averages. It represents horizontal edge information of the original image.
  - Lower right quadrant: differences from both the horizontal and vertical passes. It represents diagonal edges.

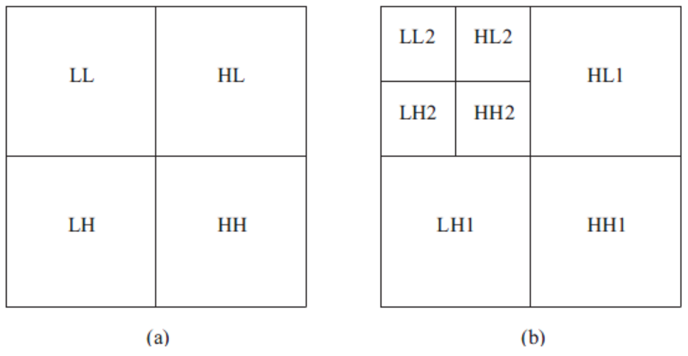


A simple graphical illustration of Wavelet Transform.

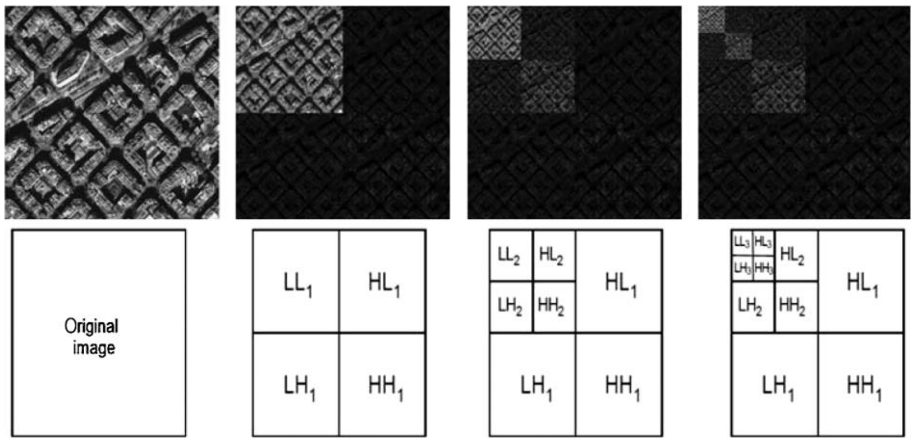
2D Wavelet Transform

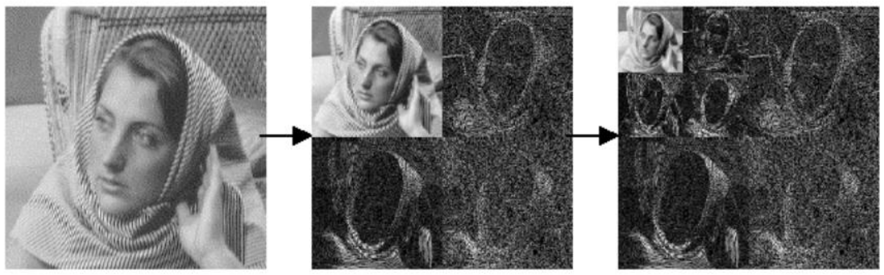
- For an *N* by *N* input image, the two-dimensional DWT proceeds as follows:
  - **Convolve each row** of the image with $h_0[n]$ and $h_1[n]$, discard the odd numbered columns of the resulting arrays, and concatenate them to form a transformed row.
  - After all rows have been transformed, **convolve each column** of the result with $h_0[n]$ and $h_1[n]$. Again discard the odd numbered rows and concatenate the result.
- After the above two steps, one stage of the DWT is complete. The transformed image now contains four subbands LL, HL, LH, and HH, standing for low-low, high-low, etc.
- The LL subband can be further decomposed to yield yet another level of decomposition. This process can be continued until the desired number of decomposition levels is reached.

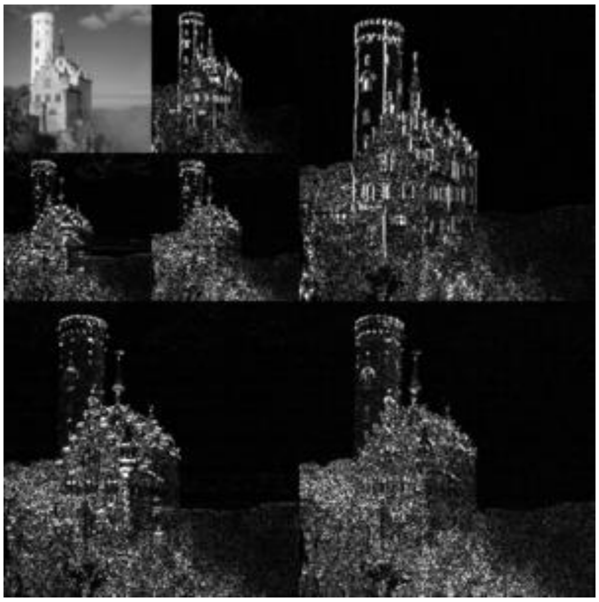
2D Wavelet Transform



The two-dimensional discrete wavelet transform
(a) One level transform, (b) two level transform.


Original image
LL1
HL1
LH1
HH1
LL2
HL2
LH2
HH2
LL3
HL3
LH3
HH3





Embedded Zerotree of Wavelet Coefficients

- How to code the wavelet transform values?
- Embedded Zerotree Wavelet (EZW) algorithm
  - Introduced by Shapiro in 1993
  - Has inspired SPIHT (Set Partitioning in Hierarchical Tree) algorithm
  - Has inspired EBCOT (Embedded Block Coding with Optimized Truncation) algorithm, which is adopted into the JPEG2000 standard.

Motivation for EZW

- Transform Coding Needs "Significance Map" to be sent:
  - At low bit rates a large # of the transform coefficients are quantized to zero => Insignificant Coefficients
  - We'd like to not have to actually send any bits to code these
    - That is... allocate zero bits to the insignificant coefficients
  - But... you need to somehow inform the decoder about which coefficients are insignificant
    - JPEG does this using run-length coding: (Run Length, Next Nonzero)
  - In general... Send a significance map

Quantized Coefficients

64	56	48	32	24	16	0	0
56	48	40	24	16	23	0	0
40	40	30	24	16	8	0	8
32	32	32	24	24	16	0	0
24	24	16	8	0	0	8	0
16	16	8	0	0	8	0	0
0	0	0	8	0	0	0	0
0	0	0	0	0	0	0	0

Significance Map

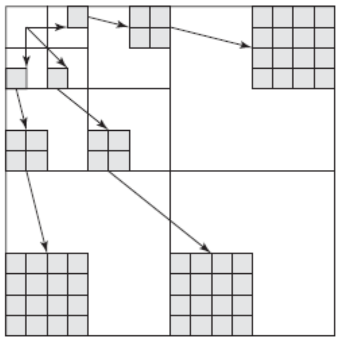
1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	1
1	1	1	1	1	1	0	0
1	1	1	1	0	0	1	0
1	1	1	0	0	1	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0

Embedded Zerotree Wavelet (EZW) algorithm

- Consists of two central components
  - The zero tree data structure
  - Successive approximation quantization

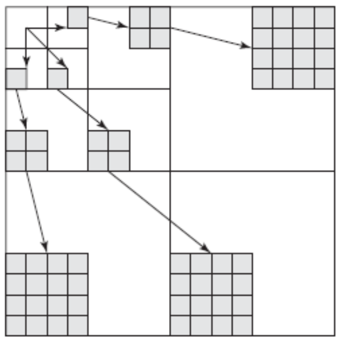
Zerotree Data Structure

- Using the hierarchical wavelet decomposition presented earlier, we can relate every coefficient at a given scale to a set of coefficients at the next finer scale of similar orientation.
- The coefficient at the coarse scale is called the "parent" while all corresponding coefficients are the next finer scale of the same spatial location and similar orientation are called "children".


Parent child relationship in a zerotree.

Zerotree Data Structure

- For a given parent, the set of all coefficients at all finer scales are **descendants**.
- For a given child, the set of all coefficients at all coarser scales are called **ancestors**.


Parent child relationship in a zerotree.

Zerotree Data Structure - Significance

- The EZW algorithm efficiently codes the “significance map” which indicates the locations of nonzero quantized wavelet coefficients.
- A wavelet coefficient *x* is said to be **insignificant** with respect to a given threshold ***T* if |*x*| < *T***.
- The significance map is coded using the zerotree with a four symbol alphabet (the coefficients can be representing by the following four different symbols): **zerotree root, isolated zero, positive significance, negative significance**.

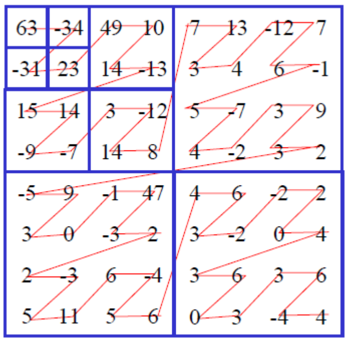
Zerotree Data Structure - Significance

- **The zerotree root**: If the magnitude of a coefficient is less than a threshold T, and all its descendants are less than T, then this coefficient is called zerotree root. And if a coefficient has been labeled as zerotree root, it means that all of its descendants are insignificance, so there is no need to label its descendants.
- **Isolated zero**: If the magnitude of a coefficient that is less than a threshold T, but it still has some significant descendants, then this coefficient is called isolated zero.

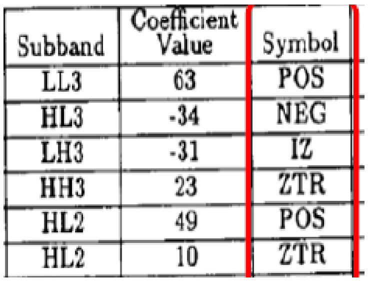
Zerotree Data Structure - Significance

- **Positive significance**: If the magnitude of a coefficient is greater than a threshold T, and also is positive, than it is a positive significant coefficient.
- **Negative significance**: If the magnitude of a coefficient is greater than a threshold T, and also is negative, than it is a negative significant coefficient.

Zerotree Data Structure – Significance Example



63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4



Subband	Coefficient Value	Symbol
LL3	63	POS
HL3	-34	NEG
LH3	-31	IZ
HH3	23	ZTR
HL2	49	POS
HL2	10	ZTR

for T=32

Successive Approximation Quantization (SAQ)

- The goal of embedded coding is to create a bit stream that can be truncated at any point by the decoder AND you get a reconstructed signal that is R-D optimal for the number of bits so far received!
- There are many ways to do this. EZW uses a successive approximation quantization. It links this idea to zerotree coding in a way that allows zerotrees to be highly exploited.
- The SAQ method sequentially applies a sequence of thresholds $T_0, \dots, T_{N-1}$ to determine the significance of each coefficient.

SAQ

- Compute the wavelet transform of the image
- Set a threshold T_0 that that creates of lots of "insignificant values"
  - These give rise to lots of zerotrees
  - Zerotrees efficiently handle significance map problem
  - Send MSB's of significant coefficients
- Then reduce threshold: $T_i+1 = T_i/2$
  - This causes some former insig coeff to become significant -> only have to tell where new significance has occurred
  - For previously significant: refine by sending next finer bit

Threshold Values

- The initial threshold is chosen so that $T_0 = 2^{\lfloor log_2^{max} \rfloor}$ where *max* is the largest coefficient.
- Threshold T_i is reduced to half of the previous threshold such that $T_i = 1/2 * T_{i-1}$

EZW Algorithm

- Compute the wavelet transform of the image
- Set a threshold **T**
- Initialize the **Dominant List** and **Subordinate List.**
- While (total bit budget is not exhausted)
  - Perform **Dominant Pass**
    - Produce significant map (zerotree coded)
    - Update the **Dominant List**
    - Add significant coefficients to **Subordinate List**.
    - Update the Wavelet array
  - Perform **Subordinate Pass**
  - Updated the threshold (reduce it by a factor of two)
- Entropy coding the output using Adaptive Arithmetic Coding.

EZW Algorithm (Expained)

- **Maintain Two Separate Lists**:
  - Dominant List: *coordinates* of coeffs not yet found significant
  - Subordinate List: *magnitudes* of coefficients already found to be significant
- For each threshold, perform two passes: **Dominant Pass** then **Subordinate Pass**.

EZW Algorithm (Expained) – Dominant Pass

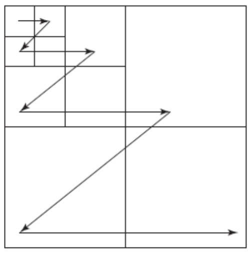
- Coeff's in Dominant List (i.e. currently insignificant) are compared to the threshold T_i to determine their significance.
- The resulting significance map is zero-tree coded.
  - Code significance using four symbols: Zerotree Root (ZTR), Positive Significant (POS), Isolated Zero (IZ) • Negative Significant (NEG)
- For each coeff that has now become significant (POS or NEG)
  - put its magnitude in the Subordinate List (making it eligible for future refinement)
  - remove it from the Dominant List (because it has now been found significant).
  - The coefficient in the wavelet transform array is set to 0 to enable the possibility of the occurrence of a zerotree on future dominant passes at smaller thresholds.

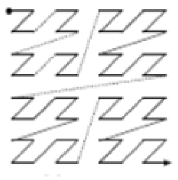
EZW Algorithm (Expained) - Subordinate Pass

- All coefficients on the subordinate list are scanned.
- Halve the quantizer cells:
  - If magnitude of coeff is in upper half of the cell, provide "1"
  - If magnitude of coeff is in lower half of the cell, provide "0"
- *After the completion of the subordinate pass, the magnitudes on the subordinate list are sorted in decreasing order to the extent that the decoder can perform the same sort.*

EZW Scanning Order

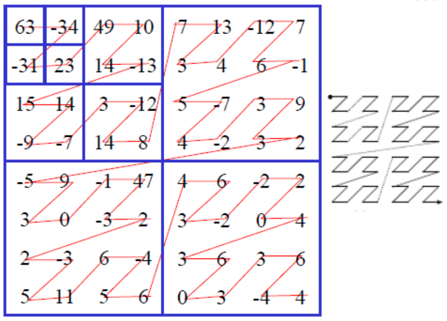
- No child node is scanned before its parent.
- The picture depicts the scanning pattern for a three level wavelet decomposition.


EZW scanning order.



EZW Example - 1

Example of 3-level WT of an 8x8 image

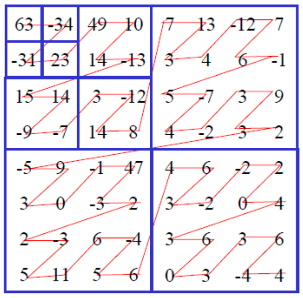


63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

Largest coefficient magn = 63 ➔ $T_0 = 32$
… So after thresholding we have:

63	-34	49	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	47	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

First Dominant Pass



63	-34	49	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	47	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

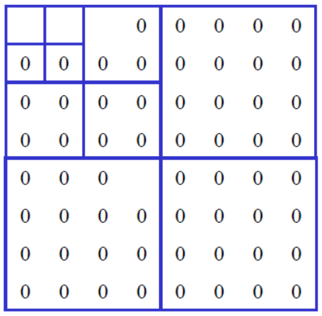
PROCESSING OF FIRST DOMINANT PASS AT THRESHOLD $T = 32$. SYMBOLS ARE POS FOR POSITIVE SIGNIFICANT, NEG FOR NEGATIVE SIGNIFICANT, IZ FOR ISOLATED ZERO, ZTR FOR ZEROTREE ROOT, AND Z FOR A ZERO WHEN THERE ARE NO CHILDREN. THE RECONSTRUCTION MAGNITUDES ARE TAKEN AS THE CENTER OF THE UNCERTAINTY INTERVAL

Comment	Subband	Coefficient Value	Symbol	Reconstruction Value
(1)	LL3	63	POS	48
	HL3	-34	NEG	-48
(2)	LH3	-31	IZ	0
(3)	HH3	23	ZTR	0
	HL2	49	POS	48
(4)	HL2	10	ZTR	0
	HL2	14	ZTR	0
	HL2	-13	ZTR	0
	LH2	15	ZTR	0
(5)	LH2	14	IZ	0
	LH2	-9	ZTR	0
	LH2	-7	ZTR	0
(6)	HL1	7	Z	0
	HL1	13	Z	0
	HL1	3	Z	0
	HL1	4	Z	0
	LH1	-1	Z	0
(7)	LH1	47	POS	48
	LH1	-3	Z	0
	LH1	-2	Z	0

Sequence of Symbols sent… but via Arith. Coding

- Dominant List: *coordinates* of coeffs not yet found significant
- Subordinate List: *magnitudes* of coefficients already found to be significant

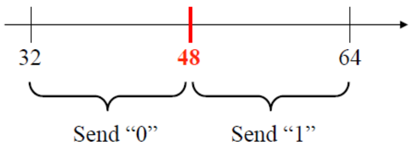
Dominant List Contains
Pointers to all these zeros



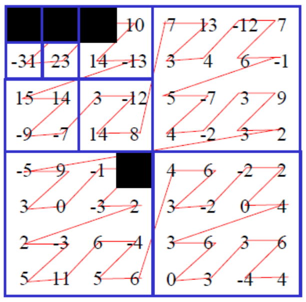
Subordinate List
63
34
49
47

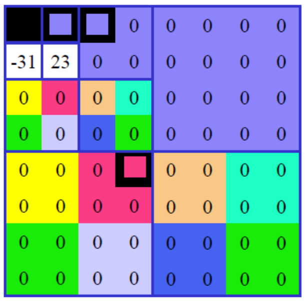
First Subordinate Pass

Now refines magnitude of each element on Subordinate List… Right now we know that each element's magnitude lies in (32,64]



Stream due to 1st Sub. Pass: 1 0 1 0





2nd Dominant Pass

New Thresh: $T_1 = T_0/2 = 16$

Only need to re-visit those on the Sub-Ord List… (not those on the Dom List… which are blacked on WT to the left) ***But*** *previously Significant values can be part of a zerotree!!!*

There is some ambiguity as to if the can be ZT Roots (Shapiro did NOT… some later papers DID)… We allow it here.

Coeff Value	Symbol	Recon Value
xxx	IZ	
xxx	ZTR	
-31	Neg	-24
23	Pos	24
15	ZTR	
14	ZTR	
-9	ZTR	
-7	ZTR	
3	ZTR	
-12	ZTR	
14	ZTR	
8	ZTR	

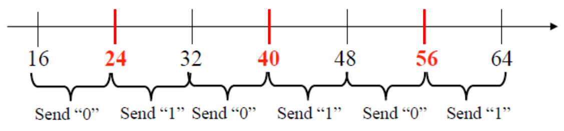
- Dominant List: *coordinates* of coeffs not yet found significant
- Subordinate List: *magnitudes* of coefficients already found to be significant

Dominant List Contains
Pointers to all these zeros

			0	0	0	0	0
		0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0		0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Subordinate List
63
34
49
47
-31
23

2nd Subordinate Pass
Now refines magnitude of each element on Subordinate List…

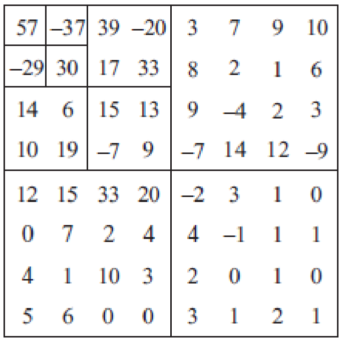


Stream due to 2nd Sub. Pass: 1 0 0 1 1 0

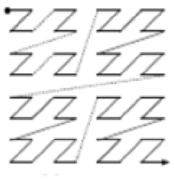
- The algorithm continues like this until the threshold falls below a user specified minimum threshold
- The sequence of symbols (alphabet size of 4) output during each dominant pass is arithmetic coded.

EZW Example - 2

- Consider coefficients of a three-stage wavelet transform used as input to the EZW algorithm.

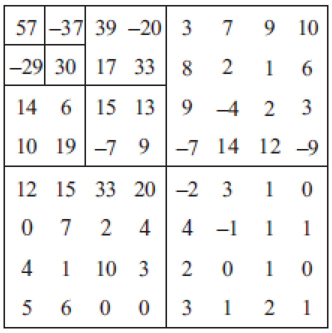


57	–37	39	–20	3	7	9	10
–29	30	17	33	8	2	1	6
14	6	15	13	9	–4	2	3
10	19	–7	9	–7	14	12	–9
12	15	33	20	–2	3	1	0
0	7	2	4	4	–1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1



EZW Example - Encoding

- Use the symbols *p*,*n*, *t*, *z* to denote positive significance, negative significance, zerotree root, and isolated zero.



57	–37	39	–20	3	7	9	10
–29	30	17	33	8	2	1	6
14	6	15	13	9	–4	2	3
10	19	–7	9	–7	14	12	–9
12	15	33	20	–2	3	1	0
0	7	2	4	4	–1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

- Initially, the dominant list contains all the coefficients in the zero tree scanning order.
- The first dominant pass outputs the following symbols with respect to the threshold $T_0 = 32$:

D_0 : *pnzt pttp tztt tttt tttt pttt*

  - The coefficients 57 and -37 are significant. Thus, we output a *p* and a *n* to represent them.
  - The coefficient −29 is insignificant, but contains a significant descendant 33 in LH1. Therefore, it is coded as *z*.
  - The coefficient *30* is also insignificant, and all its descendants are insignificant, so it is coded as *t*.
  - Continue the process for all coeffients
  - Five coefficients found to be significant: 57, -37, 39, 33, and another 33. Since we know that no coefficients are greater than $2T_0 = 64$ and the threshold used in the first dominant pass is 32, all coefficients are in [32, 64).

- The subordinate pass following the dominant pass refines the magnitude of these coefficients by indicating whether they lie in the first half or the second half of the uncertainty interval.

S_0 : 10000

- Now the dominant list contains the coordinates of all the coefficients except those found to be significant.
- The subordinate list contains the values:

{57, 37, 39, 33, 33}

  - Now, we attempt to rearrange the values in the subordinate list such that larger coefficients appear before smaller ones, with the constraint that the decoder is able do exactly the same.
  - The decoder is able to distinguish values from [32, 48) and [48, 64). Since 39 and 37 are not distinguishable in the decoder, their order will not be changed.

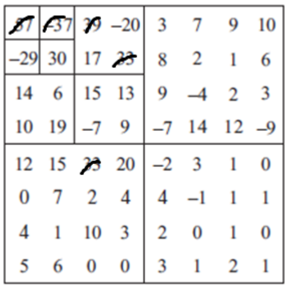
- Before we move on to the second round of dominant and subordinate passes, we need to set the values of the significant coefficients to 0 in the wavelet transform array so that they do not prevent the emergence of a new zerotree.

- The new threshold for second dominant pass is T_1 = 32/2=16.
- Using the same procedure as above, the dominant pass outputs the following symbols:

  D_1 : *zznp tnpt tztp tttt tttt tttt tptt tttt*
- The subordinate list is now:

  {57, 37, 39, 33, 33, **29, 30, 20, 17, 19, 20**}
- The subordinate pass that follows will halve each of the three current uncertainty intervals [48, 64), [32, 48), and [16, 32).
- The subordinate pass outputs the following bits:

  S_1 : 10000**110000**



~~57~~	~~−37~~	~~39~~	−20	3	7	9	10
−29	30	17	~~33~~	8	2	1	6
14	6	15	13	9	−4	2	3
10	19	−7	9	−7	14	12	−9
12	15	~~33~~	20	−2	3	1	0
0	7	2	4	4	−1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

- The output of the subsequent dominant and subordinate passes are shown below:

*D*2 : *zzzzzzzzptpzpptnttptppttpttpttpnppttttttpttttttttttttttt*

*S*2 : 01100111001101100000110110

*D*3 : *zzzzzzztzpztztnttptttttptnnttttptttpptppttpttttt*

*S*3 : 00100010001110100110001001111101100010

*D*4 : *zzzzzttztztzztzzpttpppttttpttpttnpttptptttpt*

*S*4 : 1111101001101011000001011101101100010010010101010

*D*5 : *zzzztztttztzzzzttpttptttttnptppttttppttp*

EZW - Decoding

- Suppose we only received information from the first dominant and subordinate pass.
- From the symbols in D_0 we can obtain the position of the significant coefficients.
- Then, using the bits decoded from S_0, we can reconstruct the value of these coefficients using the center of the uncertainty interval.

56	-40	40	0	0	0	0	0
0	0	0	40	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	40	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Reconstructed transform coefficients from the first pass.

- If the decoder received only D_0, S_0, D_1, S_1, D_2, and only the first 10 bits of S_2, then the reconstruction is

58	-38	38	-22	0	0	12	12
-30	30	18	34	12	0	0	0
12	0	12	12	12	0	0	0
12	20	0	12	0	12	12	-12
12	12	34	22	0	0	0	0
0	0	0	0	0	0	0	0
0	0	12	0	0	0	0	0
0	0	0	0	0	0	0	0

Reconstructed transform coefficients from D_0, S_0, D_1, S_1, D_2, and the first 10 bits of S_2.

Embedded Zerotree Wavelet (EZW) algorithm

- Effective and computationally efficient for image coding.
- The EZW algorithm addresses two problems:
  - 1. obtaining the best image quality for a given bit-rate, and
  - 2. accomplishing this task in an embedded fashion.
- Using an embedded code allows the encoder to terminate the encoding at any point. Hence, the encoder is able to meet any target bit-rate exactly.
- Similarly, a decoder can cease to decode at any point and can produce reconstructions corresponding to all lower-rate encodings.

JPEG 2000

JPEG 2000 Standard

- A Wavelet-Based New Standard
  - efficient, flexible, interactive
- Targets and features
  - Excellent low bit rate performance without sacrifice performance at higher bit rate
  - Progressive decoding to allow from lossy to lossless
  - Region-of-interest (ROI) coding
  - Error resilience

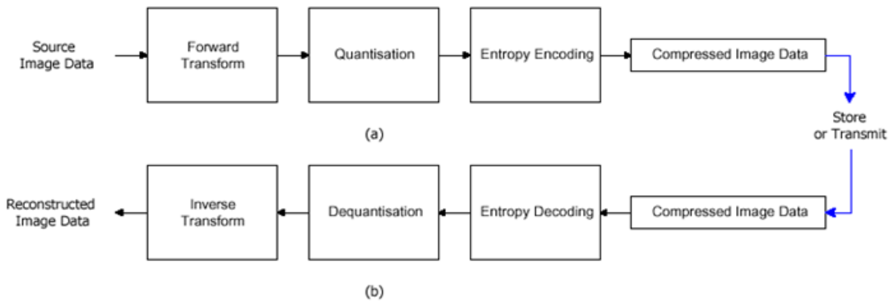


Fig. 1. Block diagrams of the JPEG2000 (a) encoder and (b) decoder.

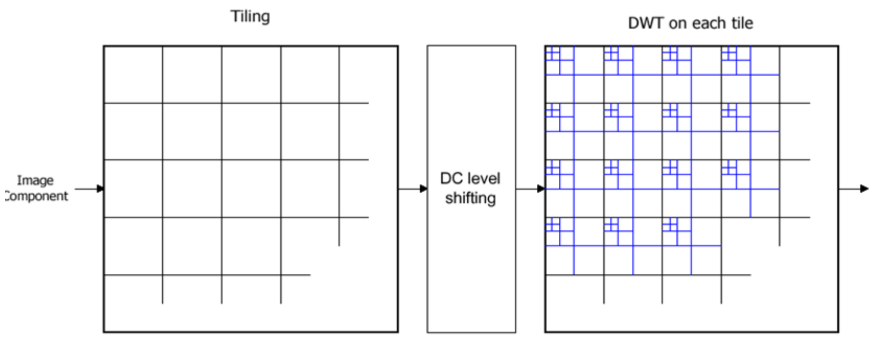


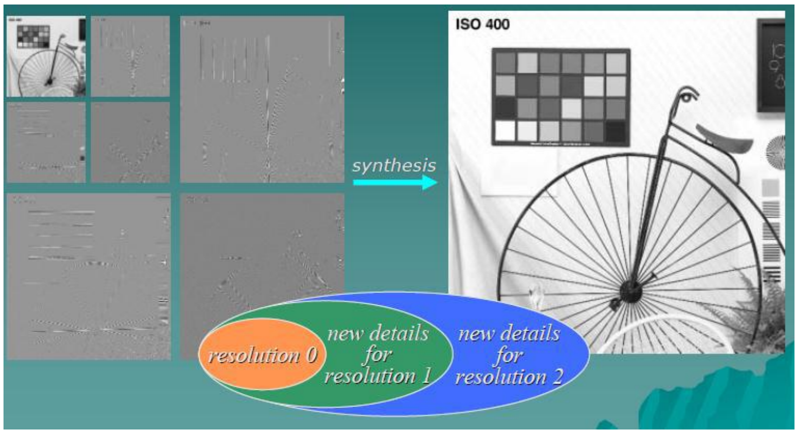
Fig. 2. Tiling, DC level shifting and DWT of each image tile component.

JPEG 2000: A Wavelet-Based New Standard

- For details
  - David Taubman: "High Performance Scalable Image Compression with EBCOT", IEEE Trans. On Image Proc, vol.9(7), 7/2000.
  - JPEG2000 Tutorial by Skrodras @ IEEE Sig. Proc Magazine 9/2001
  - Taubman's book on JPEG 2000

JPEG-2000 Scalability

- Scalable in both SNR and resolution



JPEG-2000 Scalability

- Scalable in both SNR and resolution

Example of resolution progressive bit-stream ordering

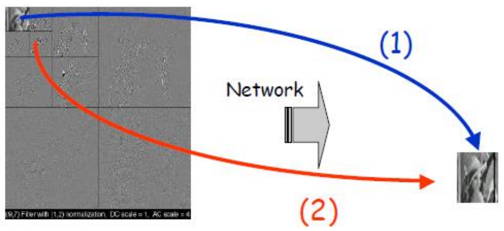
JPEG-2000 Scalability

With JPEG 2000 several levels of resolution or detail are coded in the compressed image file.

When accessing a JPEG 2000 image across a network, a low resolution image (or a full resolution image with low detail) can be extracted from the compressed image stored on the server (1)

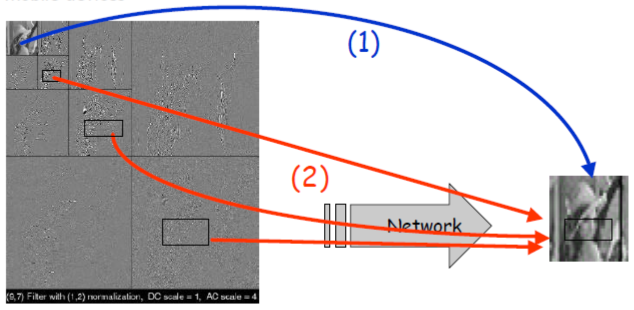
If more quality is desired, more information can be extracted from the compressed file on the server to increase the resolution or detail locally (2)

There is no data redundancy in the compressed image and therefore for this type of application JPEG 2000 is very efficient


(1)
Network
(2)
(9,7) Filter with (1,2) normalization, DC scale = 1, AC scale = 4

JPEG-2000 ROI

- JPEG 2000 also supports selective extraction of image data from a particular region.
- Use of this feature allows areas of interest to be viewed at higher quality while minimizing the amount of data needed to be transferred over the network
- This is particularly useful when viewing images on limited bandwidth networks and when using low power devices with limited display capabilities - such as mobile devices


(1)
(2)
Network
(9,7) Filter with (1,2) normalization, DC scale = 1, AC scale = 4

JPEG2K vs. JPEG

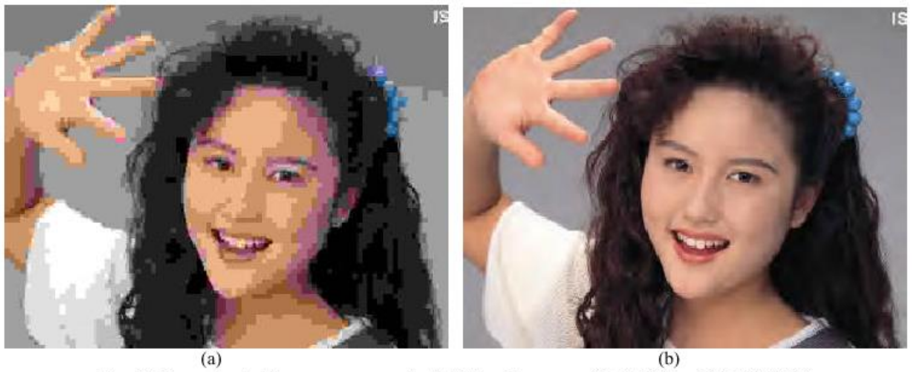


Fig. 20. Reconstructed images compressed at 0.125 bpp by means of (a) JPEG and (b) JPEG2000

JPEG2K vs. JPEG

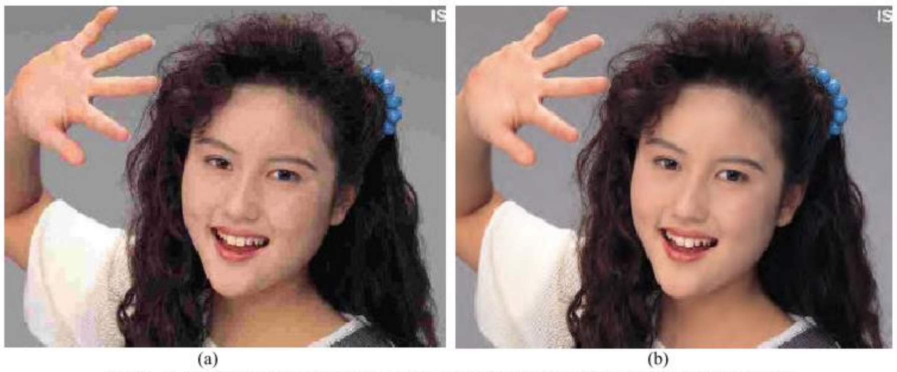


Fig. 21. Reconstructed images compressed at 0.25 bpp by means of (a) JPEG and (b) JPEG2000

JPEG-2000 Performance

- Gain up to about 20% compression performance to the first JPEG standard.
- Applications of JPEG-2000
  - Large images
  - Images with low-contrast edges (e.g., medical images
  - In printers, scanners, facsimile
  - HD satellite images

DCT vs. Wavelet: Which is Better?

- 3dB improvement?
  - Wavelet compression was claimed to have 3dB improvement over DCT-based compression
  - Comparison is done on JPEG Baseline
- Improvement not all due to transforms
  - Main contribution from better rate allocation, advanced entropy coding, & smarter redundancy reduction via zero-tree
  - DCT coder can be improved to decrease the gap

[Ref] "A comparative study of DCT- and wavelet-based image coding", Z. Xiong, K. Ramchandran, M. Orchard, Y-Q. Zhang, IEEE Trans. on Circuits and Systems for Video Tech., v.9, no.5, 8/99, pp692-695.