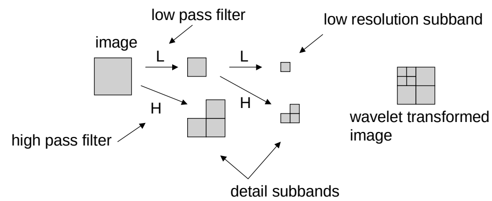
CSE 589
Applied Algorithms
Spring 1999

Wavelet Compression

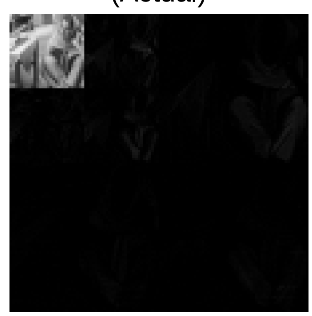
Wavelet Transform

- Wavelet Transform
  – A family of transformations that filters the data into low resolution data plus detail data.



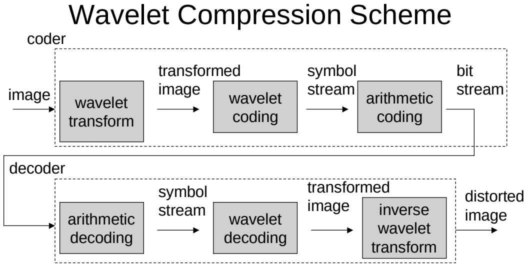
Wavelet Transformed Barbara (Enhanced)

Wavelet Transformed Barbara (Actual)



most of the details are small so they are very dark.

Wavelet Compression Scheme

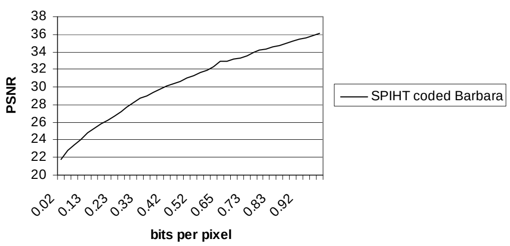


Wavelet coder transmits wavelet transformed image in bit plane order with the most significant bits first. Compression happens when only some of the bit planes are transmitted.

Wavelet Coding Methods

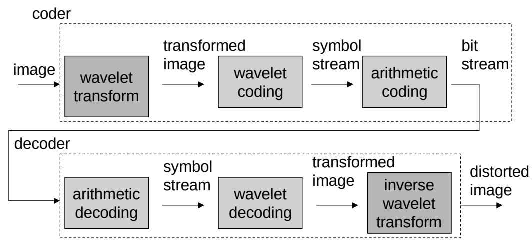
- EZW - Shapiro, 1993
  – Embedded Zero Tree coding.
- SPIHT - Said and Pearlman, 1996
  – Set Partitioning in Hierarchical Trees coding. Also uses “zero trees”.
- ECECOW - Wu, 1997
  – Embedded Conditional Entropy Coding of Wavelet coefficients.
  – Uses arithmetic coding with context.

Rate-Fidelity Curve



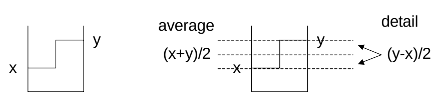
More bit planes of the wavelet transformed image that is sent the higher the fidelity.

Wavelet Transform



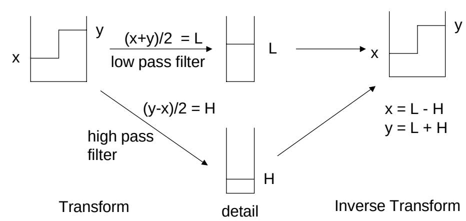
A wavelet transform decomposes the image into a low resolution version and details. The details are typically very small so they can be coded in very few bits.

One Dimensional Haar Transform (1)

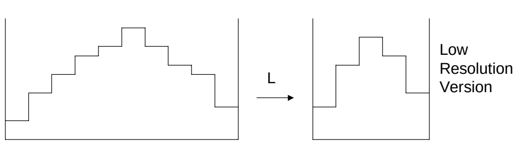


How do we represent
two data points at lower resolution?

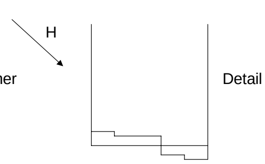
One Dimensional Haar Transform (2)



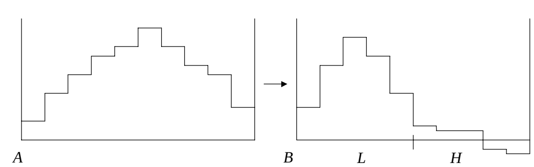
One Dimensional Haar Transform (3)



Note that the low resolution version and the detail together have the same number of values as the original.



One Dimensional Haar Transform (3)

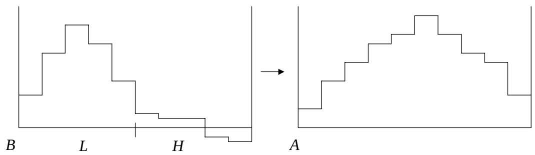


$$B[i] = \frac{1}{2}A[2i] + \frac{1}{2}A[2i+1], \quad 0 \le i < \frac{n}{2}$$

$$B[n/2+i] = -\frac{1}{2}A[2i] + \frac{1}{2}A[2i+1], \quad 0 \le i < \frac{n}{2}$$

L = B[0..n/2-1]
H = B[n/2..n-1]

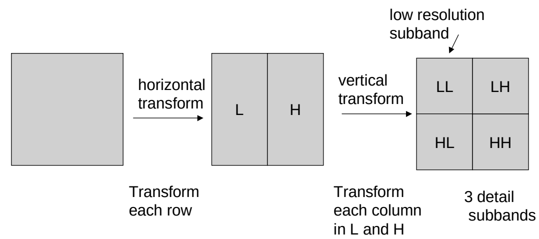
One Dim. Haar Inverse Transform



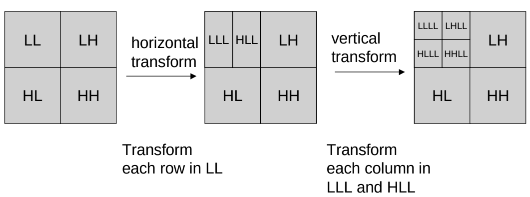
$$A[2i] = B[i] - B[n/2+i], \quad 0 \le i < \frac{n}{2}$$

$$A[2i+1] = B[i] + B[n/2+i], \quad 0 \le i < \frac{n}{2}$$

Two Dimensional Transform (1)

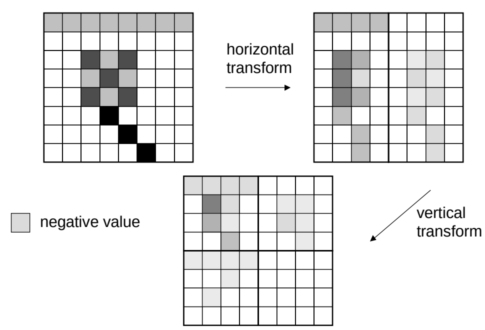


Two Dimensional Transform (1)



2 levels of transform gives 7 subbands.
k levels of transform gives 3k + 1 subbands.

Two Dimensional Haar Transform



Wavelet Transformed Image

2 levels of wavelet transform

1 low resolution subband

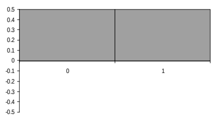
6 detail subbands

Wavelet Transform Details

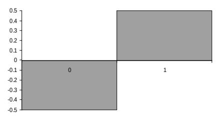
- Conversion to reals.
  - Convert gray scale to floating point.
  - Convert color to Y U V and then convert each to band to floating point. Compress separately.
- After several levels (3-8) of transform we have a matrix of floating point numbers called the wavelet transformed image.
- Image compression does not usually use the Haar filters, but uses the Daubechies 9/7 filters, or other wavelet filters.

Haar Filters

low pass = 1/2, 1/2



high pass = -1/2, 1/2

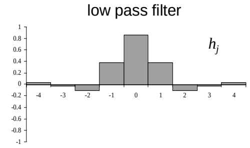


low pass $B[i] = \frac{1}{2}A[2i] + \frac{1}{2}A[2i+1], \quad 0 \le i < \frac{n}{2}$

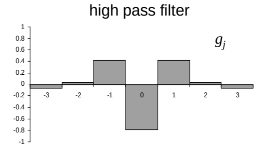
high pass $B[n/2+i] = -\frac{1}{2}A[2i] + \frac{1}{2}A[2i+1], \quad 0 \le i < \frac{n}{2}$

Daubechies 9/7 Filters

low pass filter



high pass filter



low pass $B[i] = \sum_{j=-4}^{4} h_j A[2i+j], \quad 0 \le i < \frac{n}{2}$

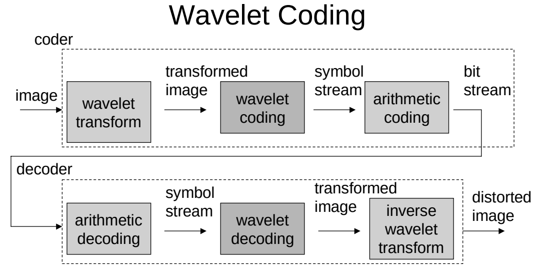
high pass $B[n/2+i] = \sum_{j=-3}^{3} g_j A[2i+j], \quad 0 \le i < \frac{n}{2}$

reflection used near boundaries

Linear Time Complexity of 2D Wavelet Transform

- Let n = number of pixels and let b be the number of coefficients in the filters.
- One level of transform takes time
  - O(bn)
- k levels of transform takes time proportional to
  - $bn + bn/4 + ... + bn/4^{k-1} < (4/3)bn$.
- The wavelet transform is linear time when the filters have constant size.
  - The point of wavelets is to use constant size filters unlike many other transforms.

Wavelet Coding



Wavelet coder transmits wavelet transformed image in bit plane order with the most significant bits first. Compression happens when only some of the bit planes are transmitted.

Normalized Wavelet Transformed Image

- Let B[0..n-1,0..n-1] be the wavelet transformed image.
- Assume $-1 < B[i,j] < 1$ (by normalization)
- Define B[i,j,k], $0 \le i,j < n$ is bit plane k.
- Encode in bit plane order.

B		sign plane		bit plane 1		bit plane 2		bit plane 3	
-.101	-.001	-	-	1	0	0	0	1	1
.010	.110	+	+	0	1	1	1	0	0
				B[*,*,1]		B[*,*,2]		B[*,*,3]	

Significance

- if $2^{-k} \le |B[i,j]|$ then B[i,j] is significant in bit plane k.
- If B[i,j] is insignificant in bit plane k then $|B[i,j]| < 2^{-k}$
- The sign of B[i,j] must be output before B[i,j] becomes significant.

B		sign plane		bit plane 1		bit plane 2		bit plane 3	
-.101	-.001	-	-	1	0	0	0	1	1
.010	.110	+	+	0	1	1	1	0	0

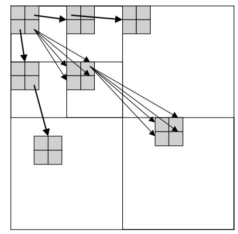
significant

Coding Ideas

- Key coding ideas:
  - The values in first bit plane of the low resolution subband (LL...LL) are very likely significant.
  - The values in the leading bit planes of the detail subbands are likely to be insignificant.
  - Most values in the leading bit planes are insignificant.
- Transmit the wavelet transformed image in bit plane order taking advantage of the high likelihood of insignificant values.

The Zero Tree Method

- Invented by Shapiro, 1993, and refined by Said and Pearlman, 1996.



If a bit plane value in a low resolution subband is insignificant then it is likely that the corresponding values in higher subbands are also insignificant in the same bit plane.

Such groups of insignificant values are called zero trees.

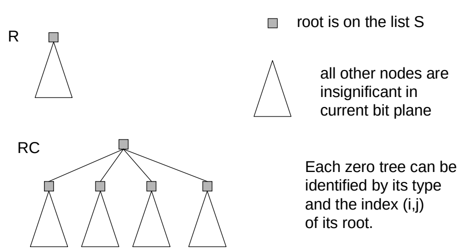
Zero Tree Example

Values in a zero tree are correlated.

SPIHT Coding

- Runs in passes - one for each bit plane.
- Encoder maintains two data structures.
  - S, a list of indices (i,j) such that B[i,j] is declared significant in the current bit plane.
  - Z, a stack of zero trees of two types.
    - rootless (R)
    - root-and-childless (RC)
  - The nodes in a zero tree are insignificant in the current bit plane. (ignore root in R and root and children in RC)

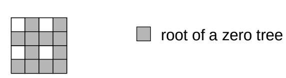
SPIHT Zero Trees



Each zero tree can be identified by its type and the index (i,j) of its root.

Initialization of SPIHT

- The lowest subband indices are put into S.
  - If (i,j) in lowest subband then output sign (0 for - and 1 for +) of B[i,j] and put (i,j) into S.
- A stack Z of zero trees is formed using the lowest resolution subband indices as roots.
  - If (i,j) in the lowest subband is a root of a zero tree of type R if i is odd or (i is even and j is odd).



lowest subband

Pass of SPIHT

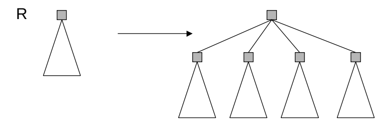
<u>k-th pass</u>
We have list S of significant values and a stack Z of zero trees from the previous pass or the initialization.
<u>Sorting step</u>.

```
while Z is not empty do
    T := pop(Z);
    if T has an index that becomes significant in bit plane k then
        output 1;
        decompose(T);
    else
        output 0;
        push T on Z'
Z := Z'; {At this point all indices in zero trees in Z are insignificant}
```

<u>Refinement step</u>.

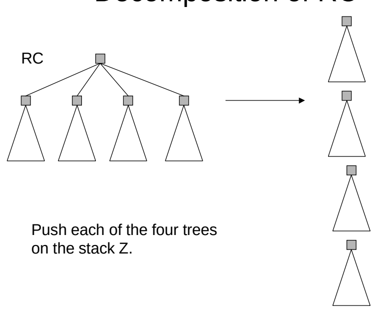
```
for each (i,j) in S output the k-th significant bit, B[i,j,k].
```


Decomposition of R



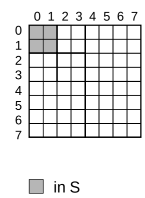
Output the sign (0 for - and 1 for +) of each of the children of the root and put them in S. Push the RC tree on the stack Z. Exception is when tree has no grandchildren. In this case, the tree dies.

Decomposition of RC



Push each of the four trees on the stack Z.

SPIHT Coding Example: Initialization



Initial data structure:

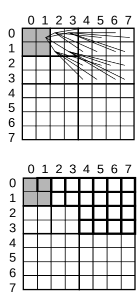
S = (0,0), (0,1), (1,0), (1,1)

Z = (R,0,1), (R,1,0), (R,1,1)

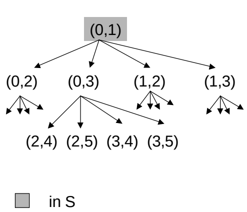
Initial output:
0 1 1 1

sign(0,0) = -
sign(0,1) = +
sign(1,0) = +
sign(1,1) = +

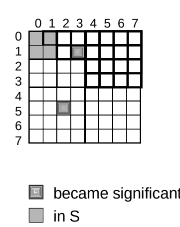
SPIHT Coding Example: Zero Tree



Example of zero tree (R,0,1)



SPIHT Coding Example: Pass 1, Sorting Step (1)



S = (0,0), (0,1), (1,0), (1,1)

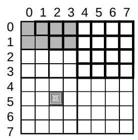
Z = (R,0,1), (R,1,0), (R,1,1)

(R,0,1) is significant
output 1

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3)
output 1101 for signs of these

Z = (RC,0,1), (R,1,0), (R,1,1)

SPIHT Coding Example: Pass 1, Sorting Step (2)



became significant
in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3)

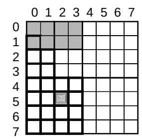
Z = (RC,0,1), (R,1,0), (R,1,1)

(RC,0,1) is not significant
output 0

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3)

Z = (R,1,0), (R,1,1)
Z' = (RC,0,1)

SPIHT Coding Example: Pass 1, Sorting Step (3)



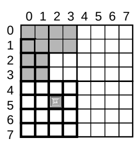
became significant
in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3)
Z = (R,1,0), (R,1,1)
Z' = (RC,0,1)

(R,1,0) is significant
output 1

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1)
output 1100 for signs of these
Z = (RC,1,0), (R,1,1)
Z' = (RC,0,1)

SPIHT Coding Example: Pass 1, Sorting Step (4)



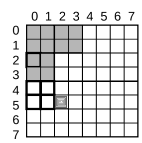
became significant
in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1)
Z = (RC,1,0), (R,1,1)
Z' = (RC,0,1)

(RC,1,0) is significant
output 1

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1)
Z = (R,2,0), (R,2,1), (R,3,0),
(R,3,1), (R,1,1)
Z' = (RC,0,1)

SPIHT Coding Example: Pass 1, Sorting Step (5)

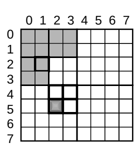


became significant
in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1)
Z = (R,2,0), (R,2,1), (R,3,0),
(R,3,1) (R,1,1)
Z' = (RC,0,1)

(R,2,0) is not significant
output 0
S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1)
Z = (R,2,1), (R,3,0), (R,3,1), (R,1,1)
Z' = (R,2,0),(RC,0,1)

SPIHT Coding Example: Pass 1, Sorting Step (6)

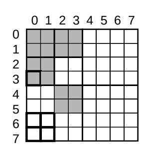


became significant
in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1)
Z = (R,2,1), (R,3,0), (R,3,1), (R,1,1)
Z' = (R,2,0),(RC,0,1)

(R,2,1) is significant
output 1
S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1),
(4,2), (4,3), (5,2), (5,3)
output 1010 for signs of these
Z = (R,3,0), (R,3,1), (R,1,1)
Z' = (R,2,0),(RC,0,1)

SPIHT Coding Example: Pass 1, Sorting Step (7)

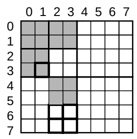


became significant
in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1),
(4,2), (4,3), (5,2), (5,3)
Z = (R,3,0), (R,3,1), (R,1,1)
Z' = (R,2,0),(RC,0,1)

(R,3,0) is insignificant
output 0
S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1),
(4,2), (4,3), (5,2), (5,3)
Z = (R,3,1), (R,1,1)
Z' = (R,3,0), (R,2,0),(RC,0,1)

SPIHT Coding Example: Pass 1, Sorting Step (8)

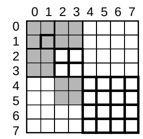


became significant
in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1),
(4,2), (4,3), (5,2), (5,3)
Z = (R,3,1), (R,1,1)
Z' = (R,3,0), (R,2,0),(RC,0,1)

(R,3,1) is insignificant
output 0
S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1),
(4,2), (4,3), (5,2), (5,3)
Z = (R,1,1)
Z' = (R,3,1), (R,3,0), (R,2,0),(RC,0,1)

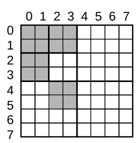
SPIHT Coding Example: Pass 1, Sorting Step (9)



became significant
in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1),
(4,2), (4,3), (5,2), (5,3)
Z = (R,1,1)
Z' = (R,3,1), (R,3,0), (R,2,0),(RC,0,1)
(R,1,1) is insignificant
output 0
S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1),
(4,2), (4,3), (5,2), (5,3)
Z =
Z' = (R,1,1), (R,3,1), (R,3,0),
(R,2,0),(RC,0,1)

SPIHT Coding Example: Pass 1, Refinement Step



in S

S = (0,0), (0,1), (1,0), (1,1),
(0,2), (0,3), (1,2), (1,3),
(2,0), (2,1), (3,0), (3,1),
(4,2), (4,3), (5,2), (5,3)
Z = (R,1,1), (R,3,1), (R,3,0),
(R,2,0),(RC,0,1)

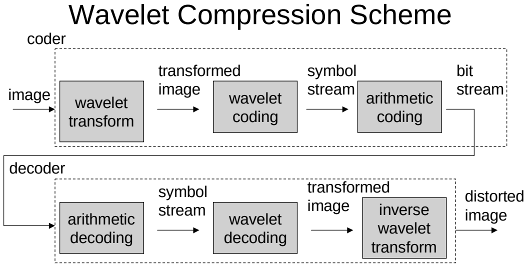
output 1011011011101000
one bit for each member of S.

37 total bits in pass 1 were output.
Initialization was 4 bits.

SPIHT Decoding

- The decoder emulates the encoder.
  - The decoder maintains exactly the same data structures as the encoder.
  - When the decoder has popped the Z stack to examine a zero tree it receives a bit telling it whether the tree is significant. The decoder can then do the right thing.

Wavelet Compression Scheme



9/7 Daubechies wavelet filters for 3 to 8 levels.
SPIHT encoding does well for wavelet coding.
Arithmetic coding adds a small improvement.

Notes on Wavelet Compression

- Currently the best compression available for natural images.
  - Excellent rate-fidelity curve.
  - Encoder and decoder well matched in speed.
  - SPIHT has good time complexity.
  - Wavelet compressed image do not have the blockiness found in VQ and JPEG coded images.
  - Arithmetic code doesn't add much.
- Wavelet compression is very practical
  - JPEG 2000
  - FBI fingerprint data base