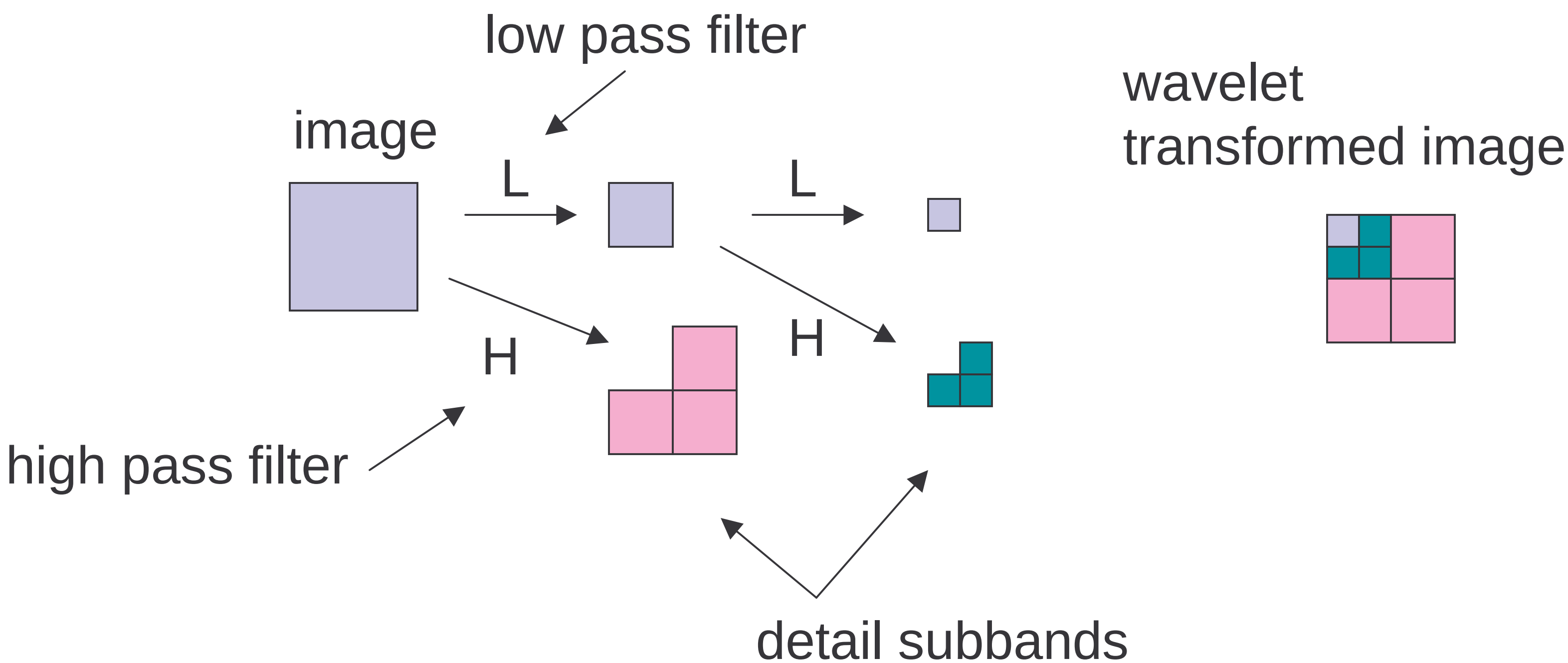
CSE 490 G
Introduction to Data Compression
Winter 2006

Wavelet Transform Coding
PACW

Wavelet Transform

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



Wavelet Transformed Barbara (Enhanced)

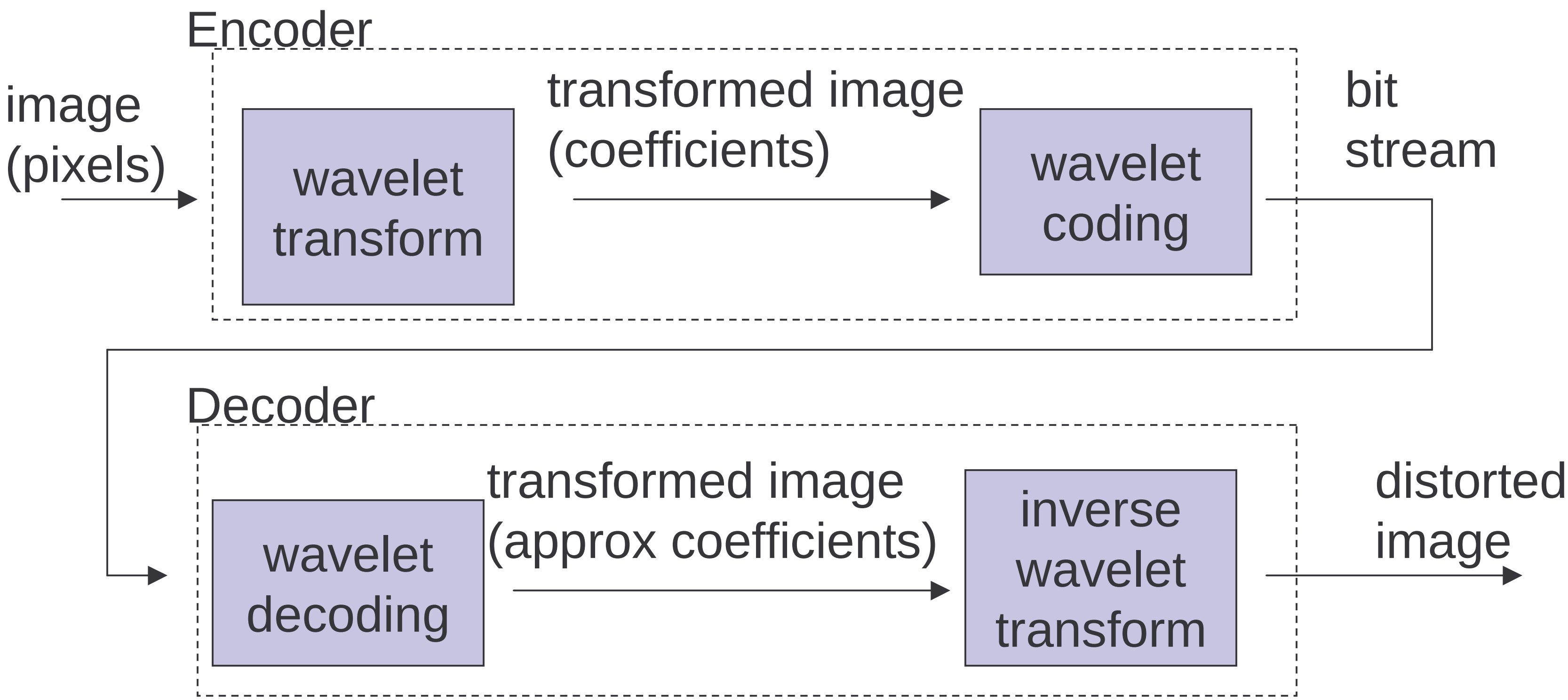
Low resolution subband

Detail subbands

Wavelet Transformed Barbara (Actual)

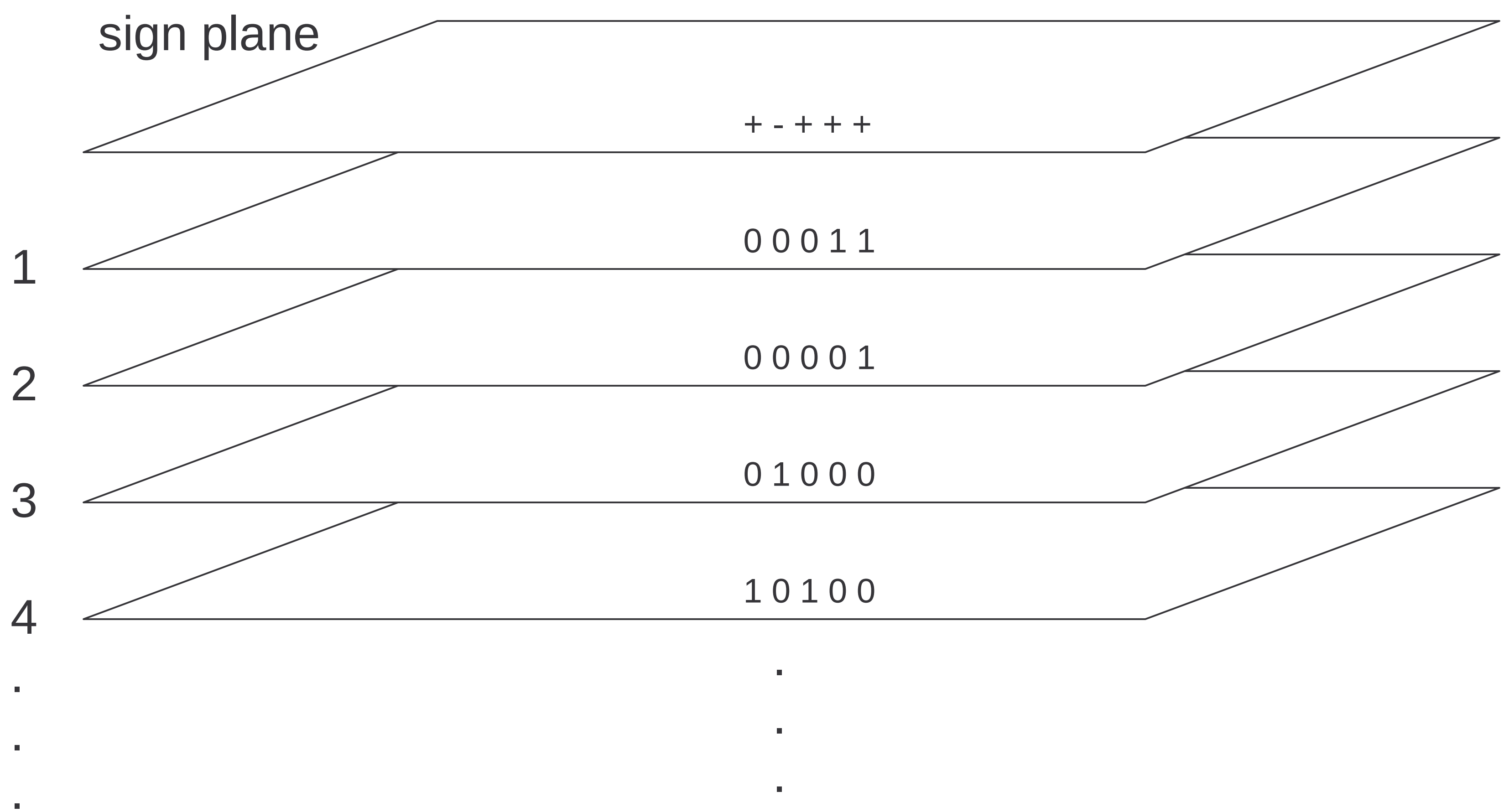
most of the details are small so they are very dark.

Wavelet Transform Compression



Wavelet coder transmits wavelet transformed image in bit plane order with the most significant bits first.

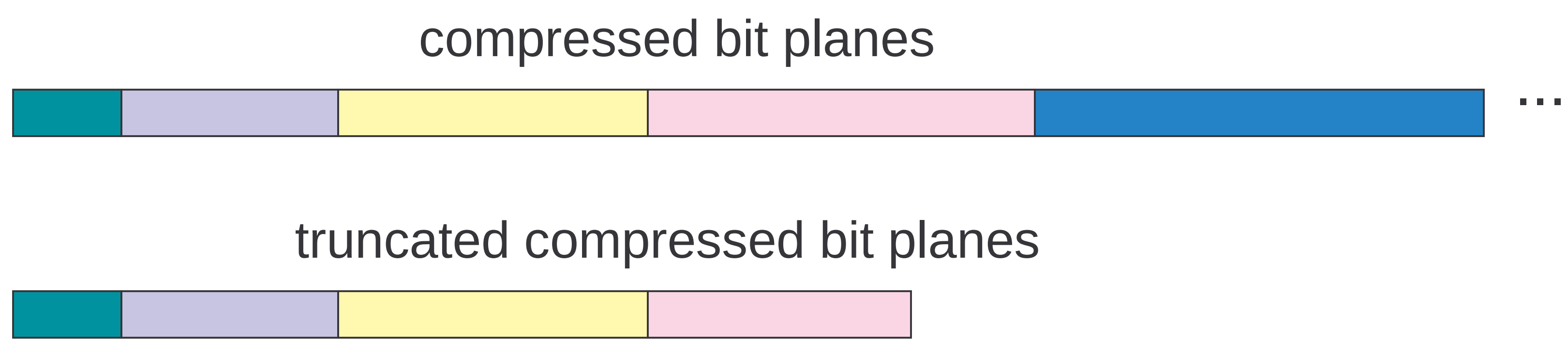
Bit Planes of Coefficients



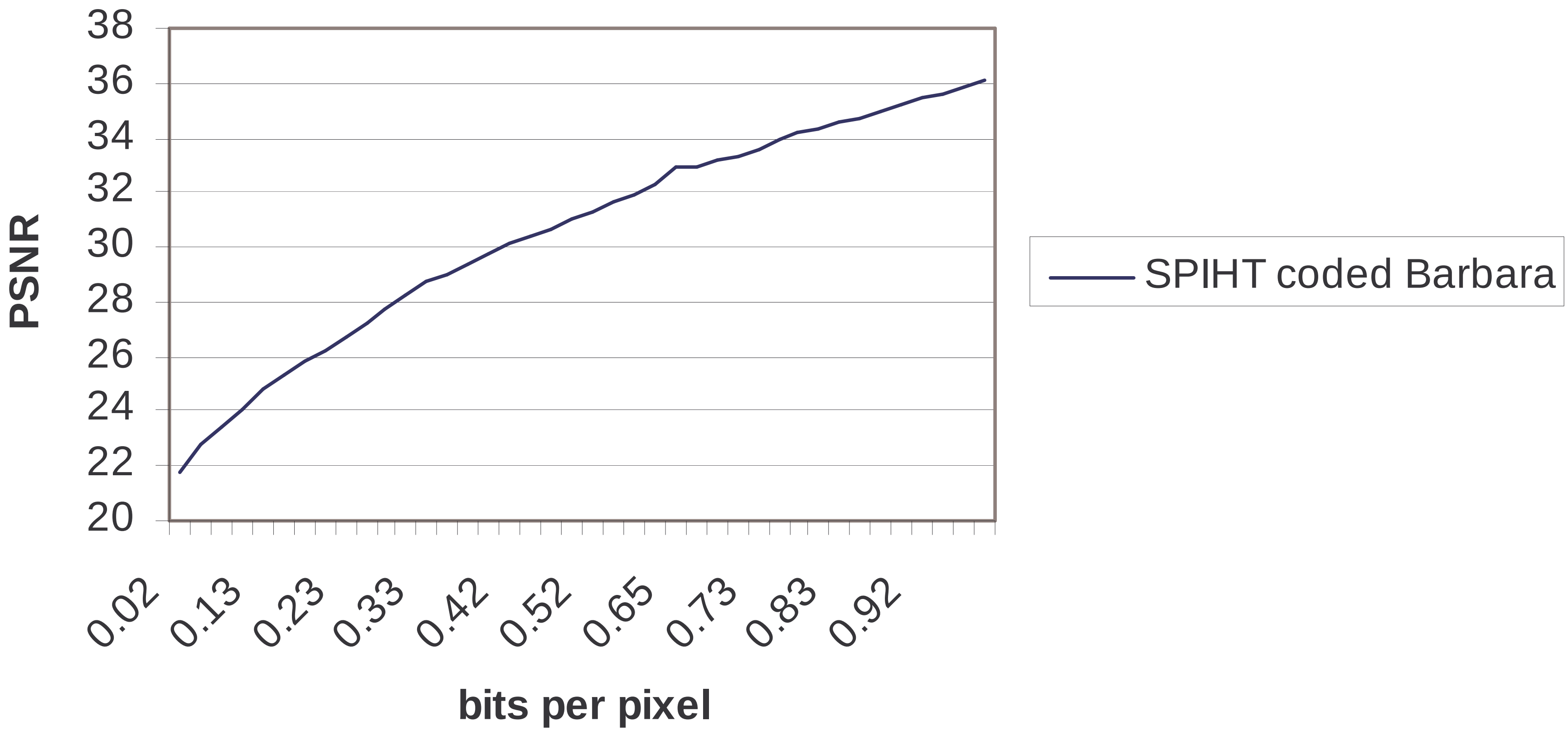
Coefficients are normalized between –1 and 1

Why Wavelet Compression Works

- Wavelet coefficients are transmitted in bit-plane order.
  - In most significant bit planes most coefficients are 0 so they can be coded efficiently.
  - Only some of the bit planes are transmitted. This is where fidelity is lost when compression is gained.
- Natural progressive transmission



Rate-Fidelity Curve

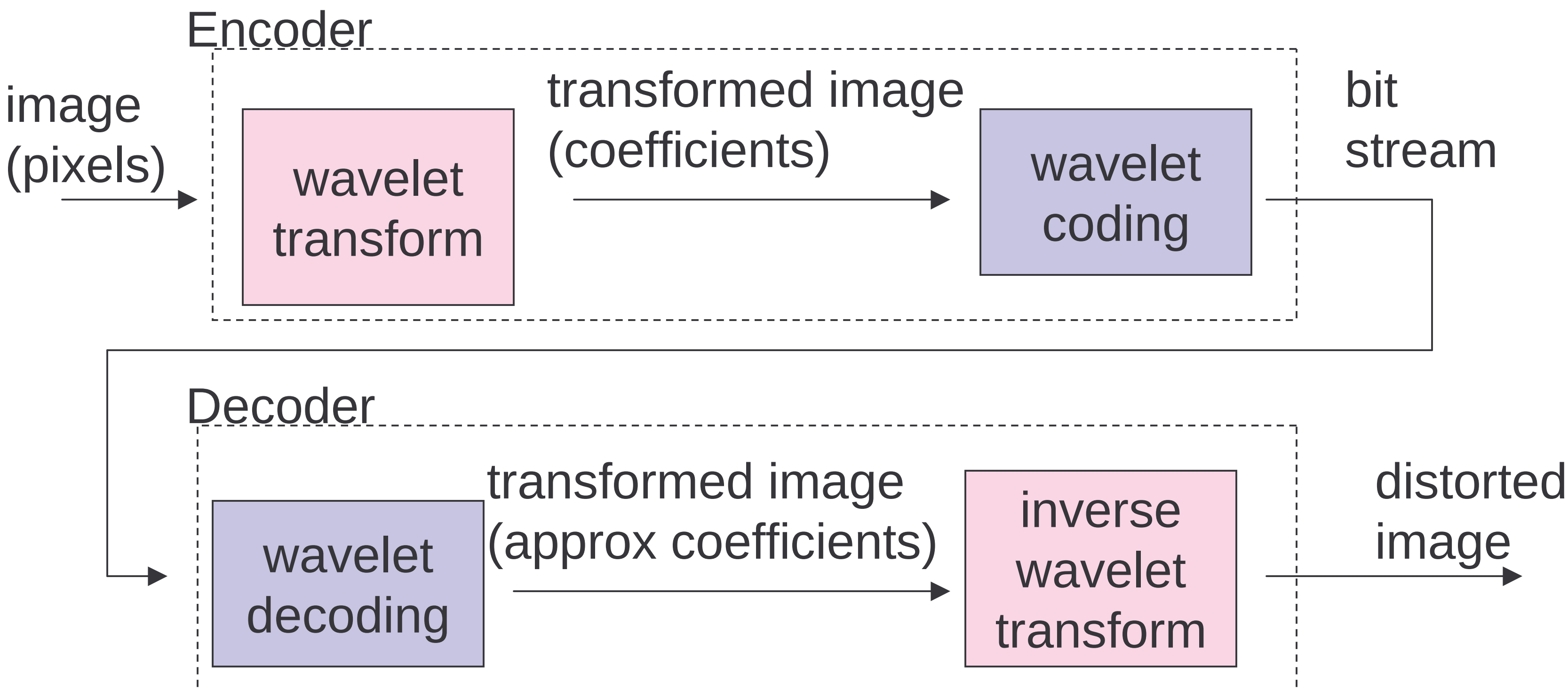


More bit planes of the wavelet transformed image that is sent the higher the fidelity.

Wavelet Coding Methods

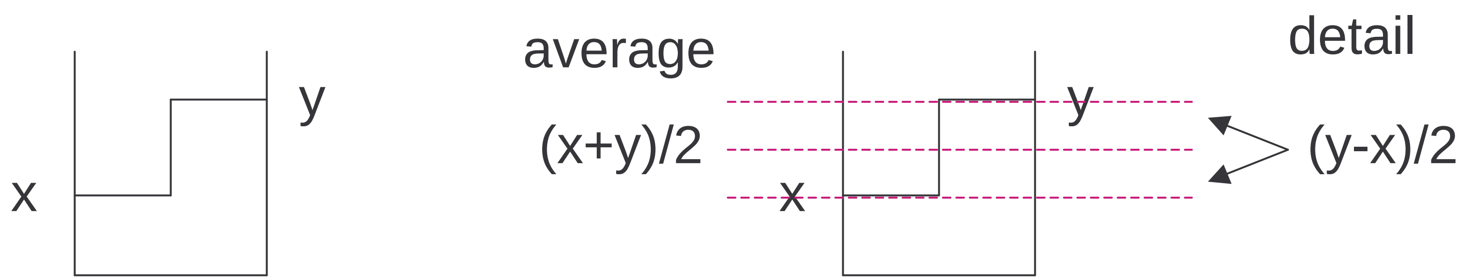
- EZW - Shapiro, 1993
  - Embedded Zerotree coding.
- SPIHT - Said and Pearlman, 1996
  - Set Partitioning in Hierarchical Trees coding. Also uses "zerotrees".
- ECECOW - Wu, 1997
  - Uses arithmetic coding with context.
- EBCOT – Taubman, 2000
  - Uses arithmetic coding with different context.
- JPEG 2000 – new standard based largely on EBCOT
- GTW – Hong, Ladner 2000
  - Uses group testing which is closely related to Golomb codes
- PACW - Ladner, Askew, Barney 2003
  - Like GTW but uses arithmetic coding

Wavelet Transform



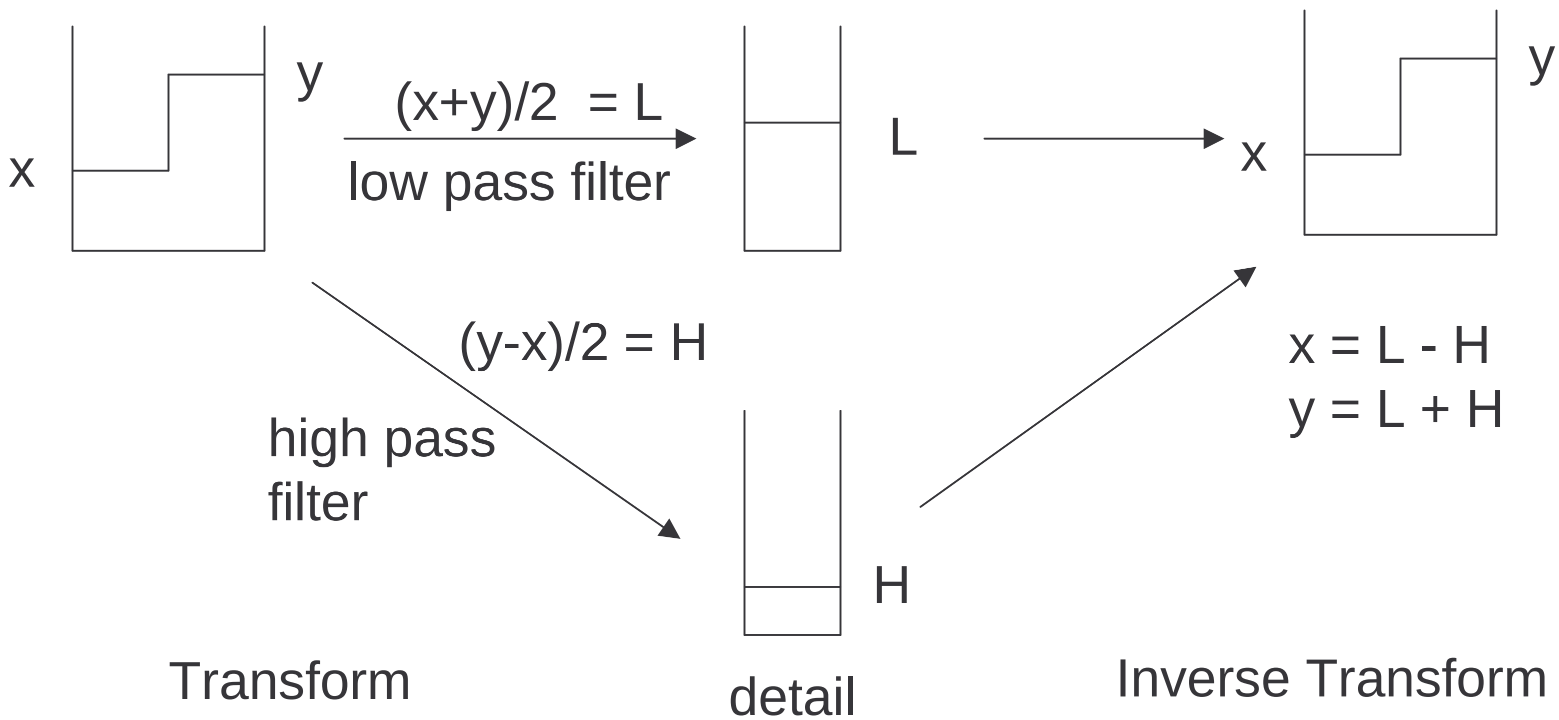
A wavelet transform decomposes the image into a low resolution version and details. The details are typically very small so they can be coded in very few bits.

One-Dimensional Average Transform (1)

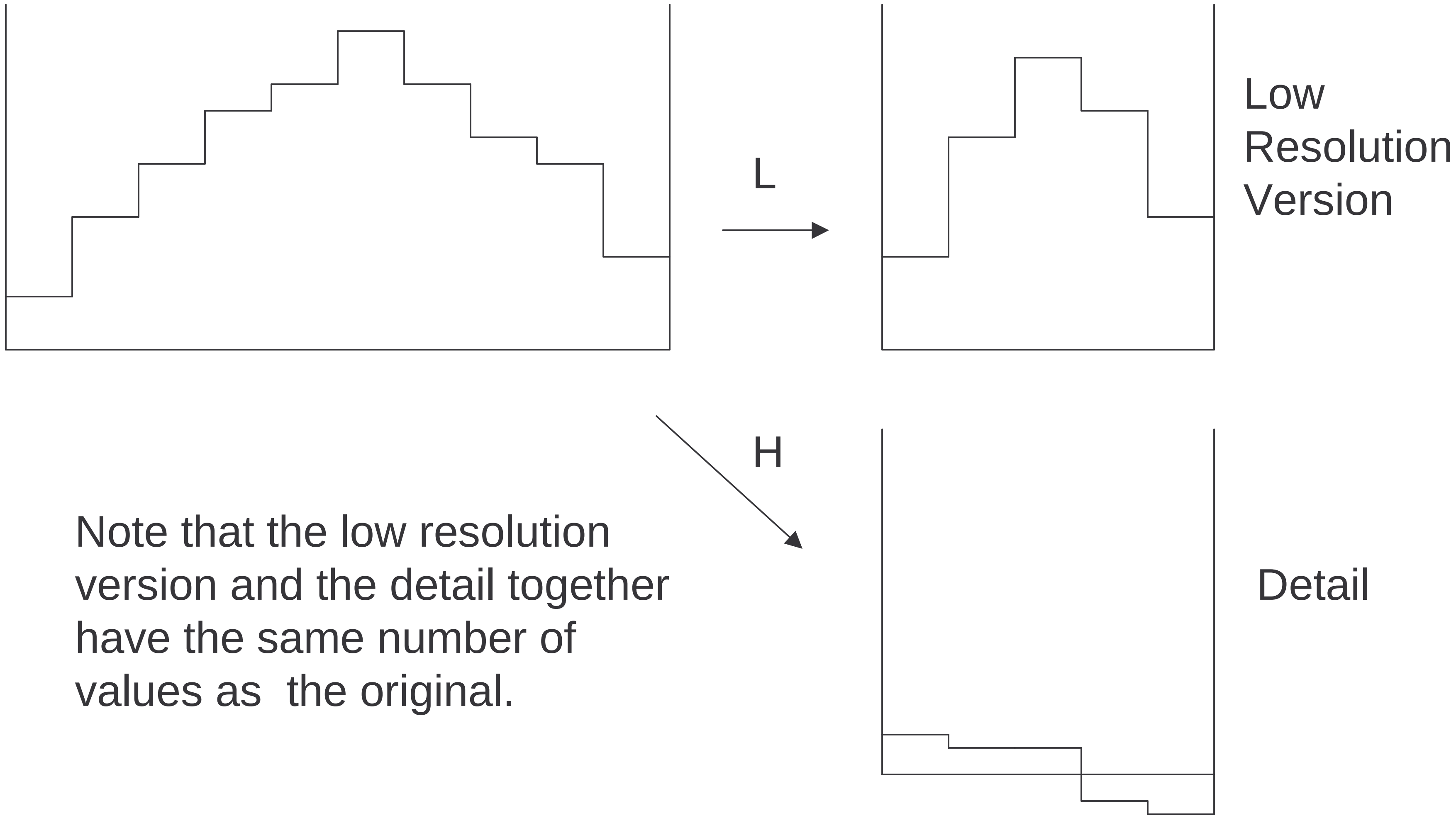


How do we represent
two data points at lower resolution?

One-Dimensional Average Transform (2)

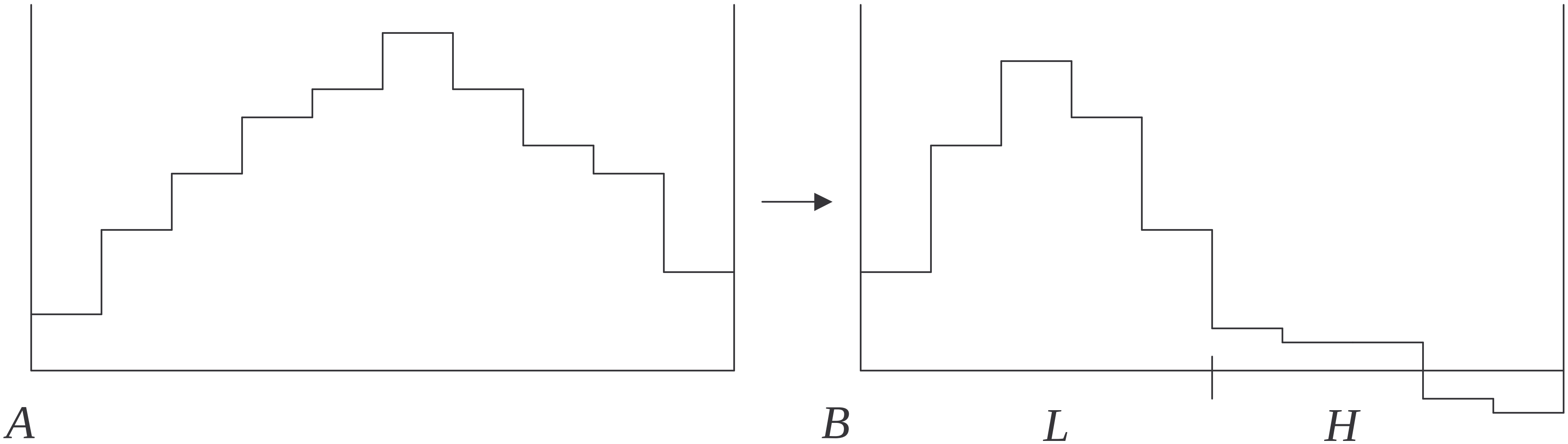


One-Dimensional Average Transform (3)



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

One-Dimensional Average Transform (4)

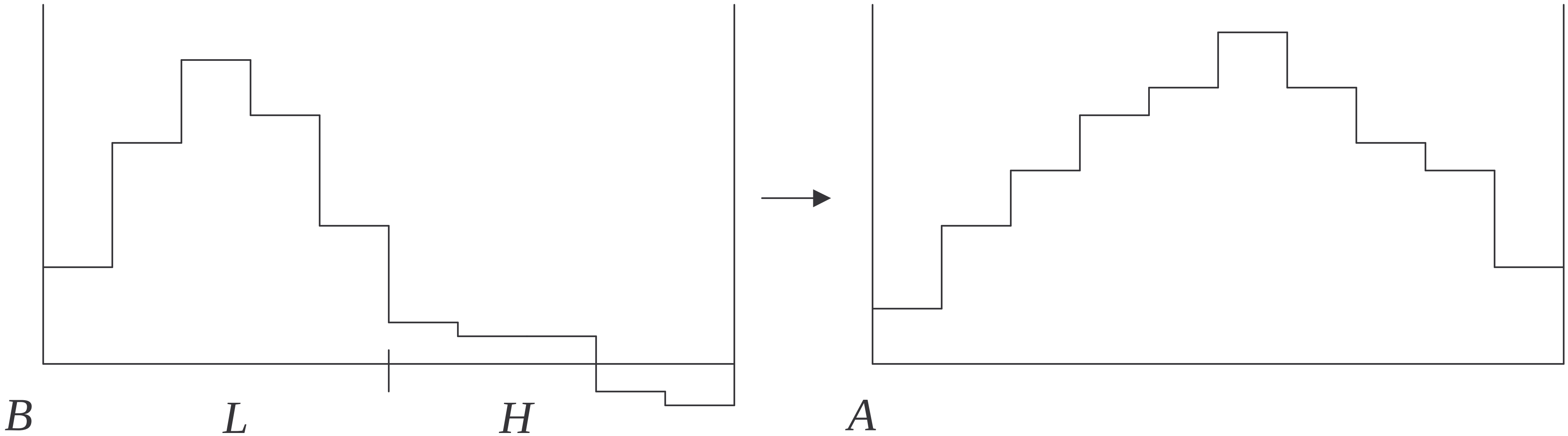


$$B[i] = \frac{1}{2}A[2i] + \frac{1}{2}A[2i+1], \quad 0 \le i < \frac{n}{2}$$

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

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

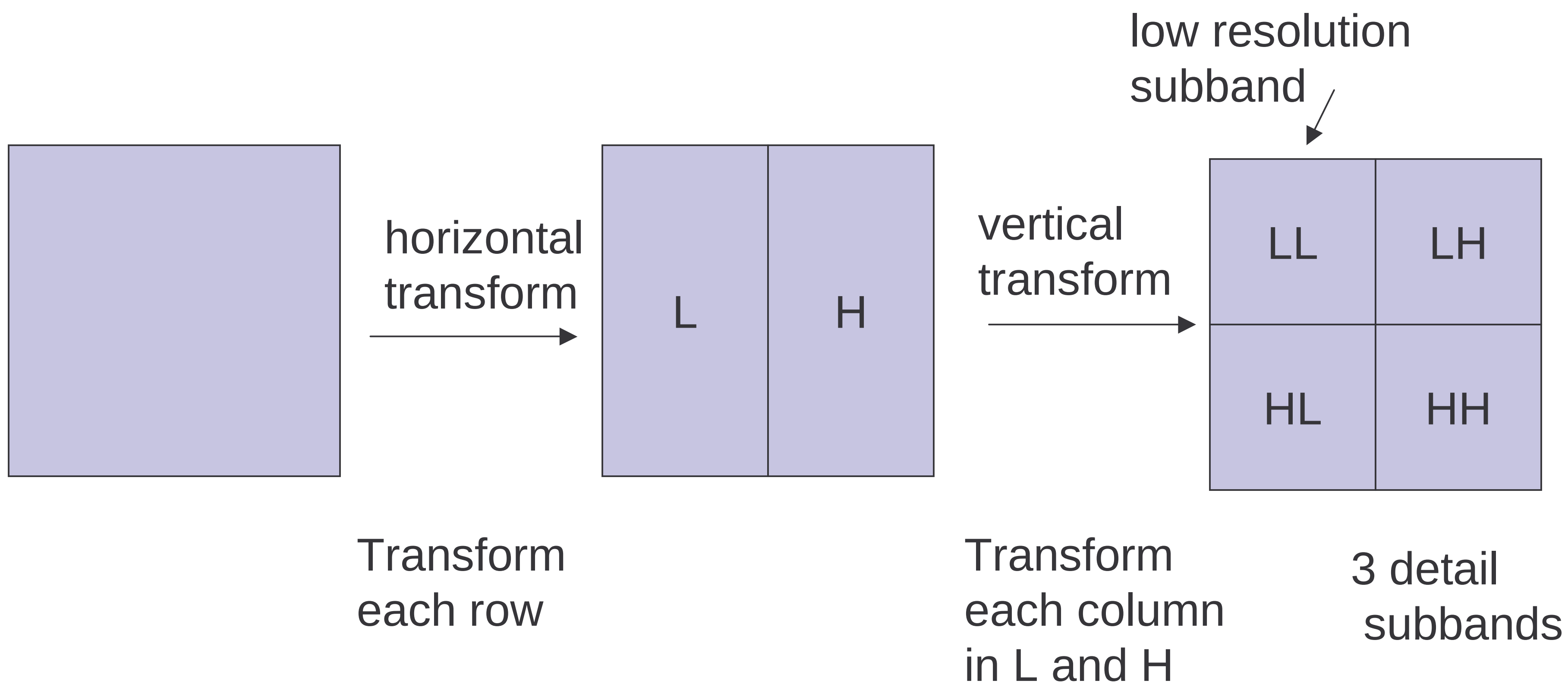
One-Dimensional Average Inverse Transform



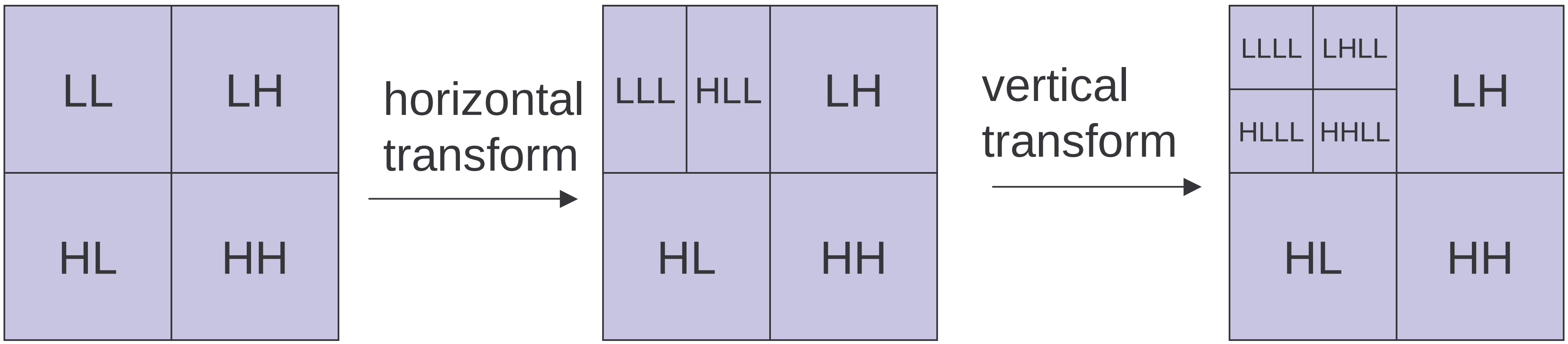
$$A[2i] = B[i] - B[n/2 + i], \quad 0 \le i < \frac{n}{2}$$

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

Two Dimensional Transform (1)



Two Dimensional Transform (1)

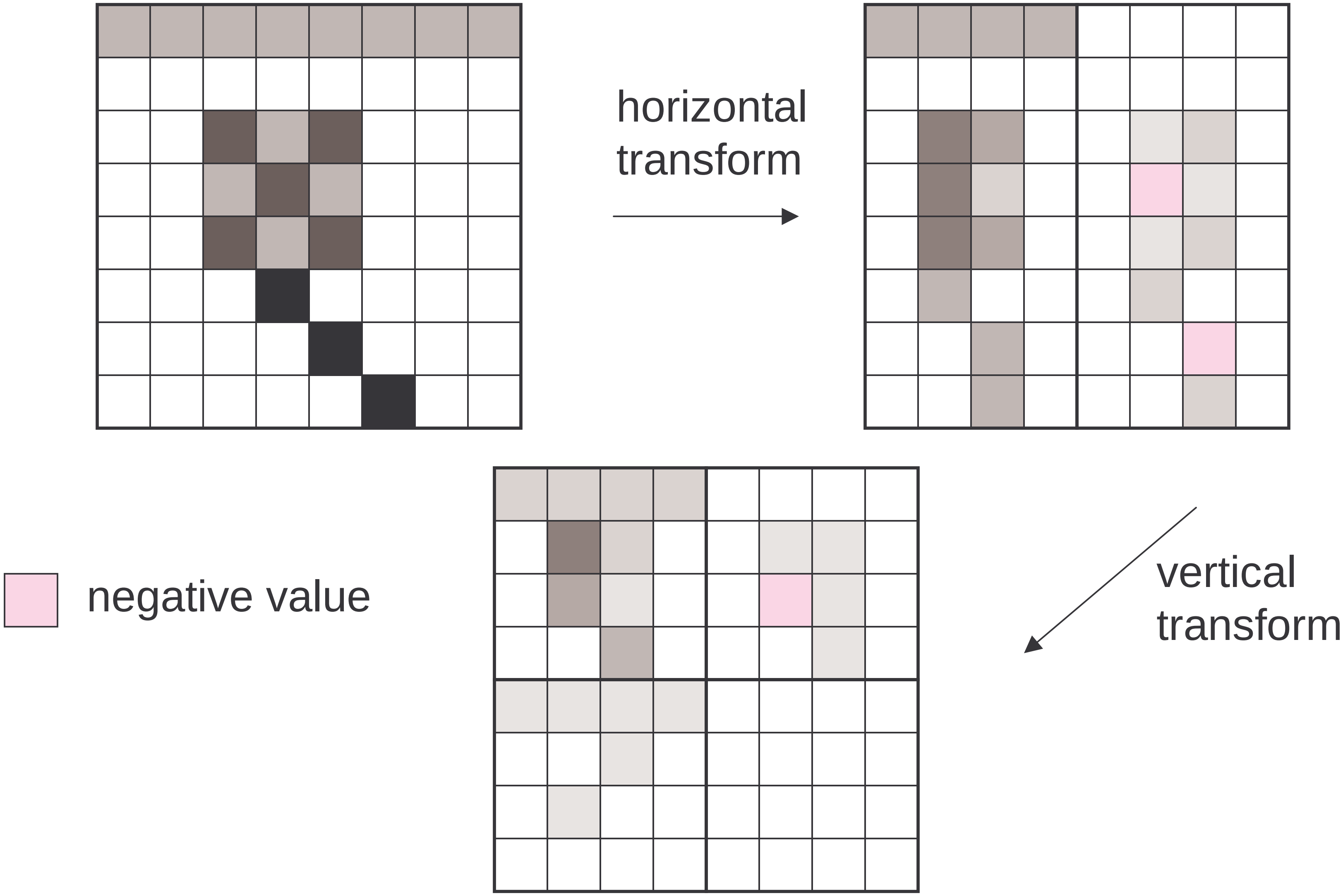


Transform
each row in LL

Transform
each column in
LLL and HLL

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

Two Dimensional Average 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 (coefficients).

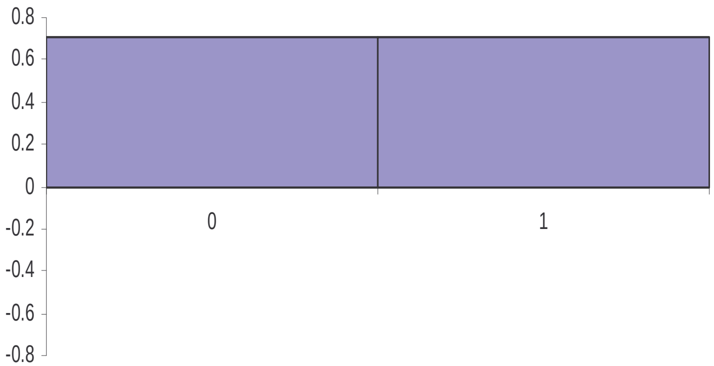
Wavelet Transforms

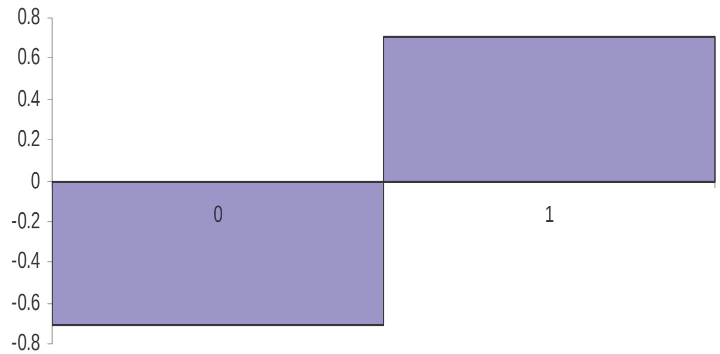
- Technically wavelet transforms are special kinds of linear transformations. Easiest to think of them as filters.
  - The filters depend only on a constant number of values. (bounded support)
  - Preserve energy (norm of the pixels = norm of the coefficients)
  - Inverse filters also have bounded support.
- Well-known wavelet transforms
  - Haar – like the average but orthogonal to preserve energy. Not used in practice.
  - Daubechies 9/7 – biorthogonal (inverse is not the transpose). Most commonly used in practice.

Haar Filters

low pass = $\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$

high pass = $-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$





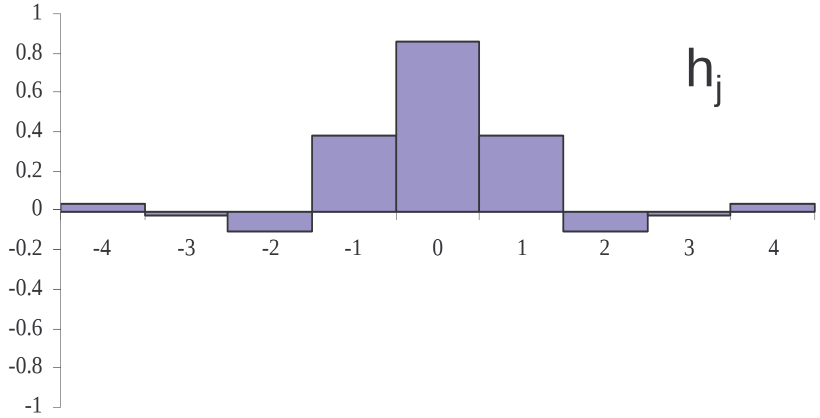
low pass $B[i] = \frac{1}{\sqrt{2}} A[2i] + \frac{1}{\sqrt{2}} A[2i+1], \quad 0 \le i < \frac{n}{2}$

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

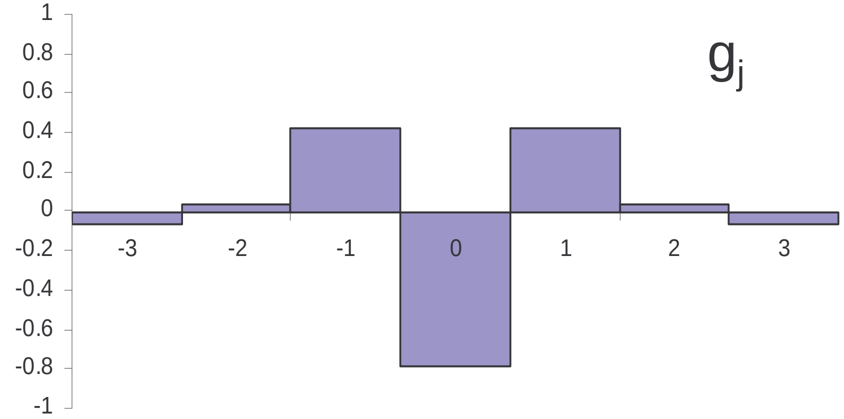
Want the sum of squares of the filter coefficients = 1

Daubechies 9/7 Filters

low pass filter



high pass filter



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

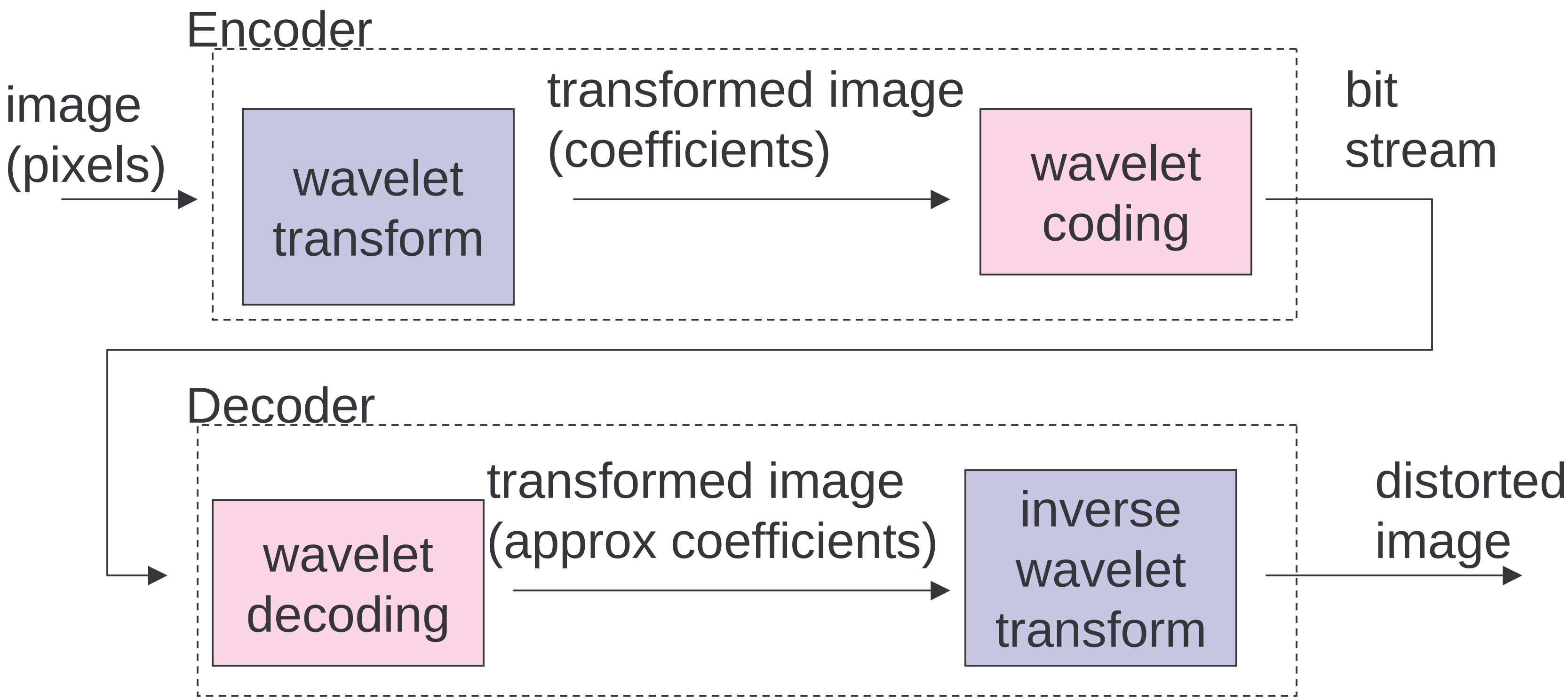
high pass $$B[n/2 + i] = \sum_{j=-3}^{3} g_j A[2i + j], \quad 0 \le i < \frac{n}{2}$$

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 Transform



Wavelet coder transmits wavelet transformed image in bit plane order with the most significant bits first.

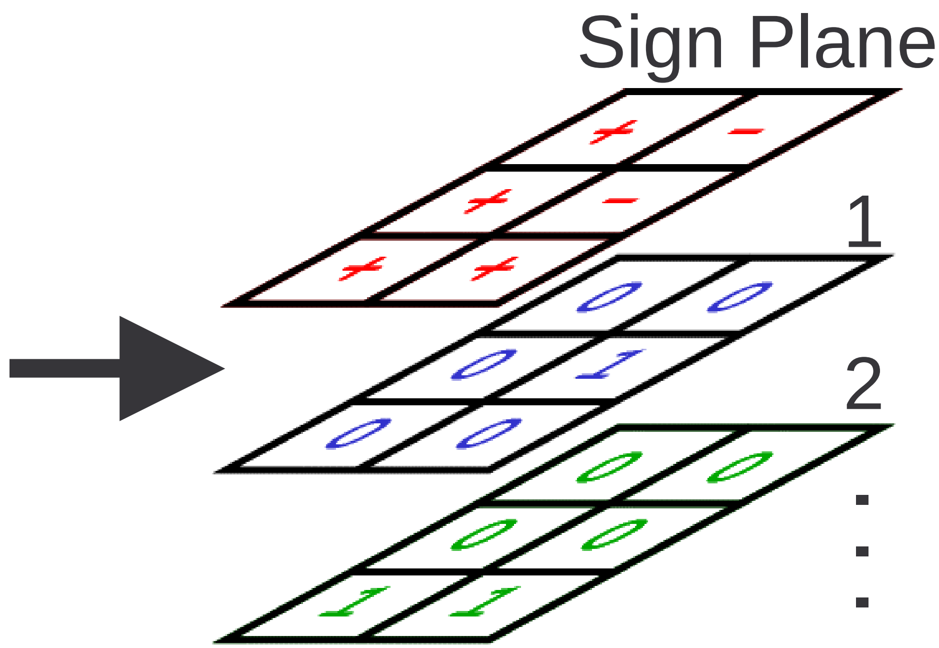
Bit-Plane Coding

- Normalize the coefficients to be between –1 and 1
- Transmit one bit-plane at a time
- For each bit-plane
  - Significance pass: Find the newly significant coefficients, transmit their signs.
  - Refinement pass: transmit the bits of the known significant coefficients.

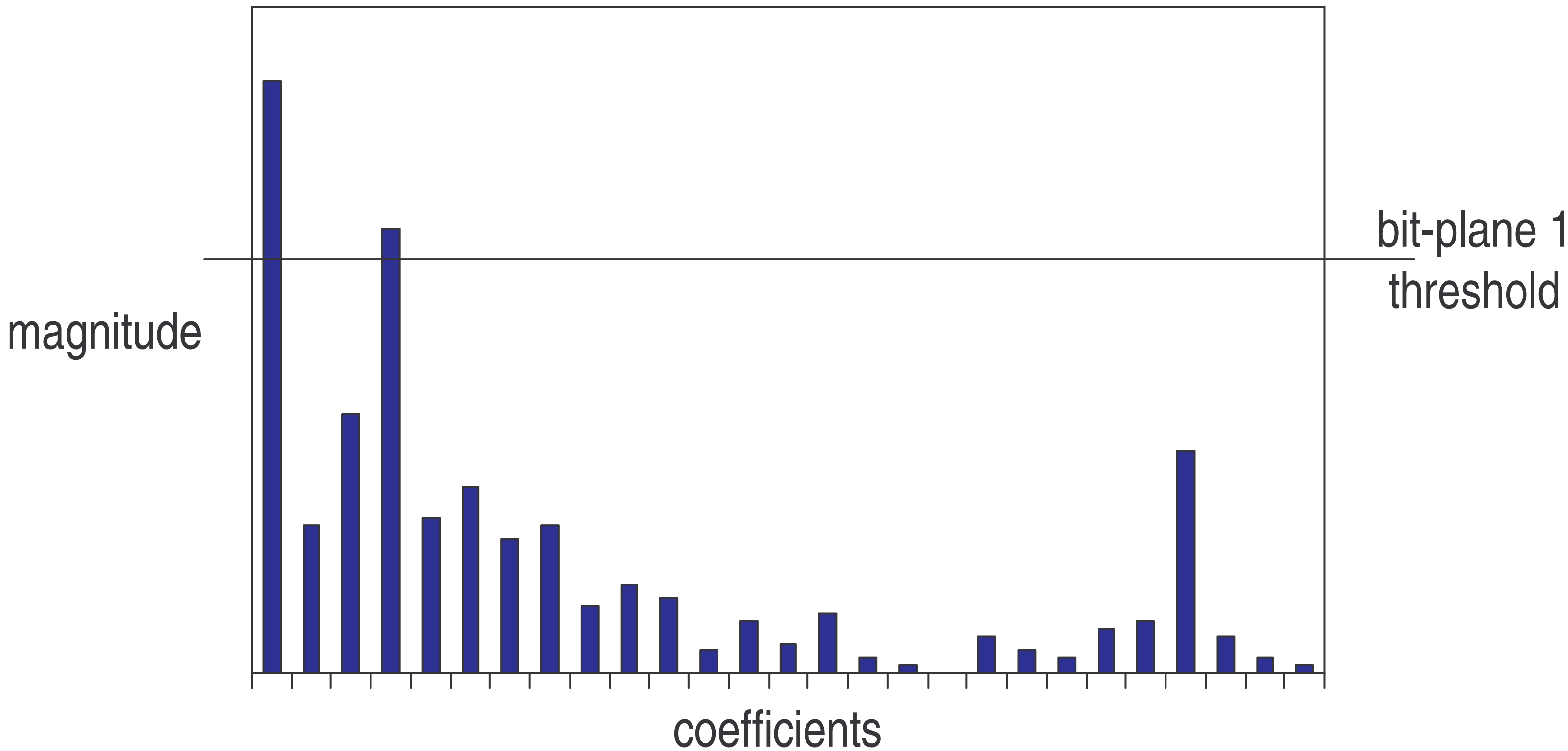
Divide into Bit-Planes

Coefficients

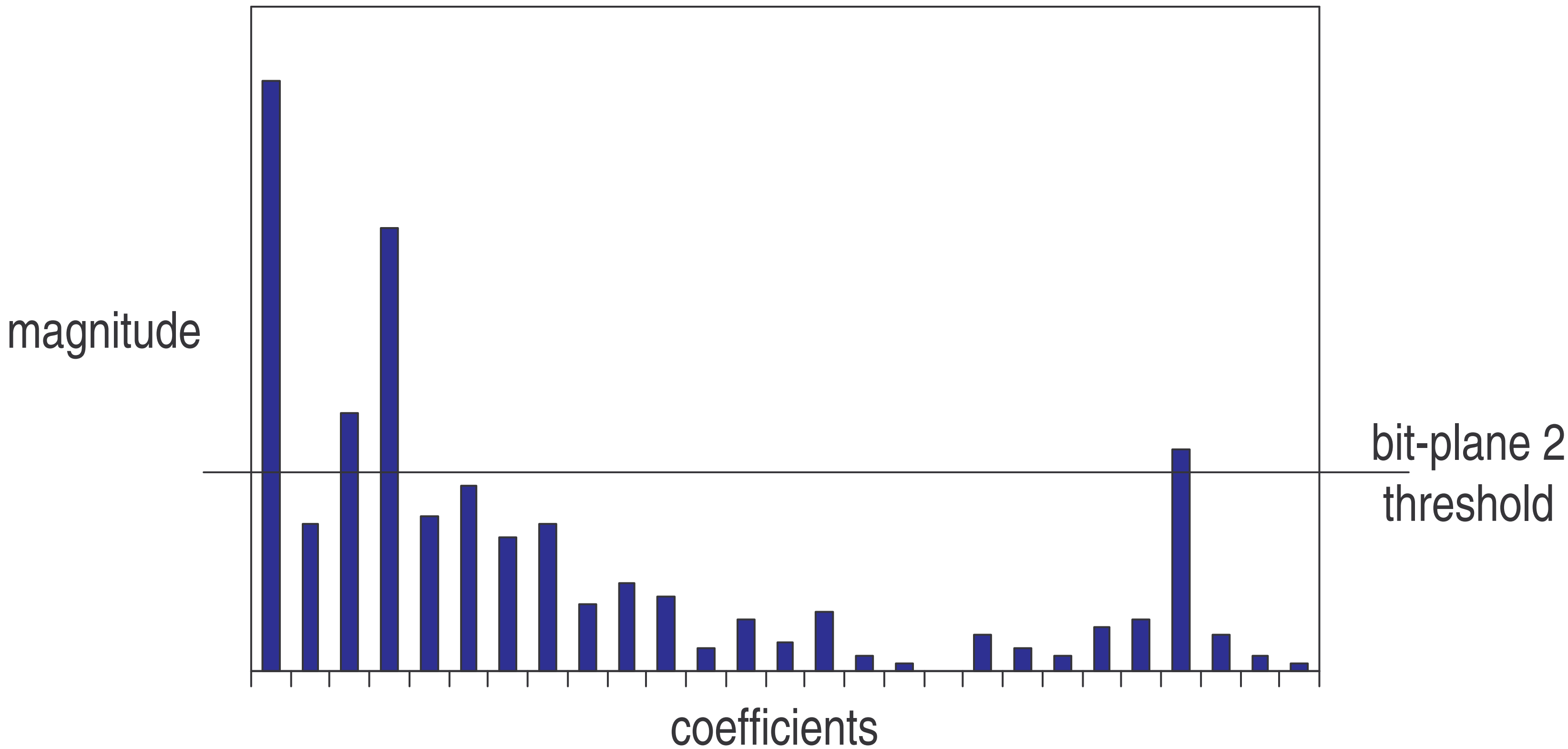
+0010	-0011
+0001	-1000
+0101	+0111



Significant Coefficients



Significant Coefficients



Significance & Refinement Passes

- Code a bit-plane in two passes
  - Significance pass
    - codes previously insignificant coefficients
    - also codes sign bit
  - Refinement pass
    - refines values for previously significant coefficients
- Main idea:
  - Significance-pass bits likely to be 0;
  - Refinement-pass bit are not

Coefficient List

#	value
1	010010010110
2	001011011110
3	000001001001
4	000000010110
5	000100111101
6	000000100101
7	101101110101
8	010010011111
9	001011101101
10	000010100101

refinement bits

Bit-plane 3

bit plane
1

bpp
.0014

PSNR
15.3

Compressed
size

bit planes
1 – 2

bpp
.0033

PSNR
16.8

Compressed
size

bit planes
1 – 3

bpp
.0072

PSNR
18.8

Compressed
size

bit planes
1 – 4

bpp
.015

533 : 1

PSNR
20.5

Compressed
size

bit planes
1 – 5

bpp
.035

ratio
229 : 1

PSNR
22.2

Compressed size

bit planes
1 – 6

bpp
.118

ratio
68 : 1

PSNR
24.8

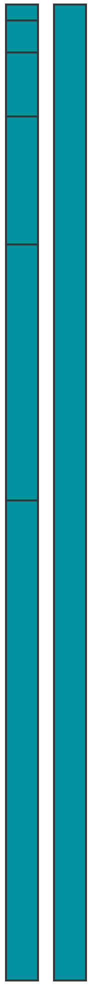
Compressed size

bit planes
1 – 7

bpp
.303

ratio
26 : 1

PSNR
28.7



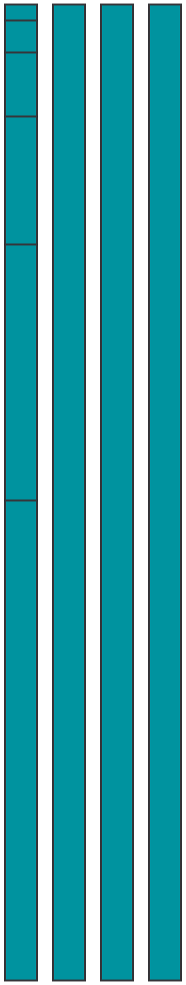
bit planes
1 – 8

bpp
.619

ratio
13 : 1

PSNR
32.9

Compressed size



bit planes
1 – 9

bpp
1.116

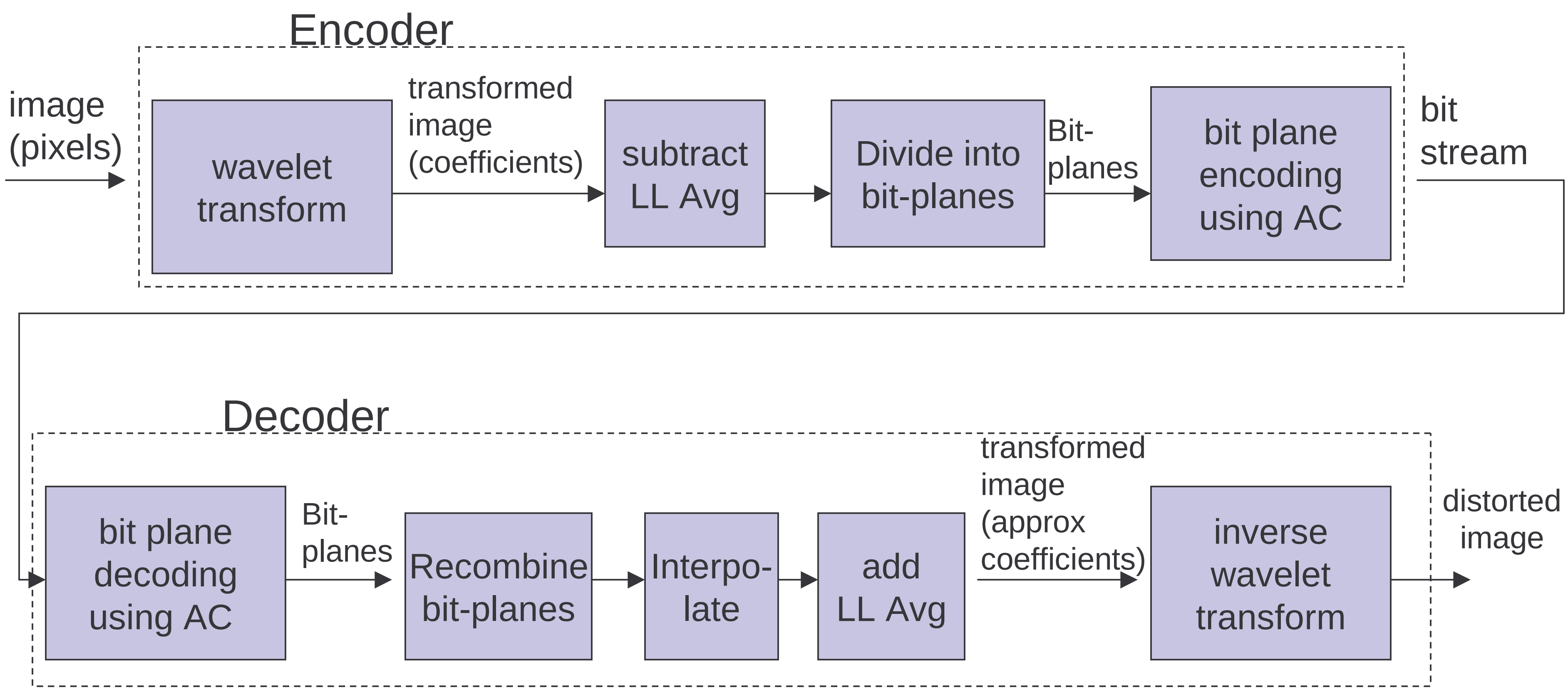
ratio
7 : 1

PSNR
37.5

PACW

- A simple image coder based on
  - Bit-plane coding
    - Significance pass
    - Refinement pass
  - Arithmetic coding
  - Careful selection of contexts based on statistical studies
- Implemented by undergraduates Amanda Askew and Dane Barney in Summer 2003.

PACW Block Diagram



Arithmetic Coding in PACW

- Performed on each individual bit plane.
  - Alphabet is $\Sigma=\{0,1\}$
  - Signs are coded as needed
- Uses integer implementation with 32-bit integers. (Initialize $L = 0$, $R = 2^{32}-1$)
- Uses scaling and adaptation.
- Uses contexts based on statistical studies.

Encoding the Bit-Planes

- Code most significant bit-planes first
- Significance pass for a bit-plane
  - First code those coefficients that were insignificant in the previous bit-plane.
  - Code these in a priority order.
  - If a coefficient becomes significant then code its sign.
- Refinement pass for a bit-plane
  - Code the refinement bit for each coefficient that is significant in a previous bit-plane

Decoding

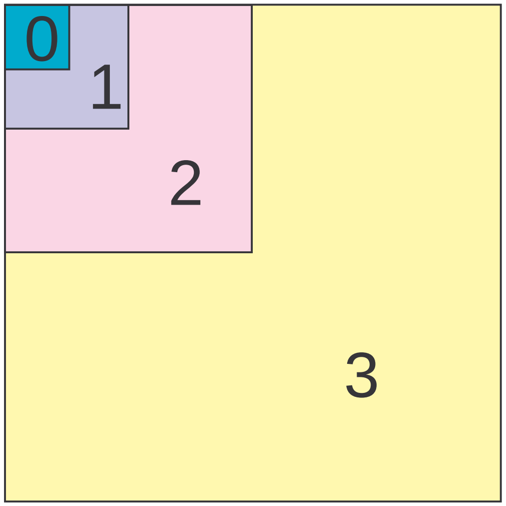
- Emulate the encoder to find the bit planes.
  – The decoder know which bit-plane is being decoded
  – Whether it is the significant or refinement pass
  – Which coefficient is being decoded.
- Interpolate to estimate the coefficients.

Contexts (per bit plane)

- Significance pass contexts:
  - Contexts based on
    - Subband level
    - Number of significant neighbors
  - Sign context
- Refinement contexts
  - 1st refinement bit is always 1 so no context needed
  - 2nd refinement bit has a context
  - All other refinement bits have a context
- Context Principles
  - Bits in a given context have a probability distribution
  - Bits in different contexts have different probability distributions

Subband Level

- Image is divided into subbands until LL band (subband level 0) is less than 16x16
- Barbara image has 7 subband levels



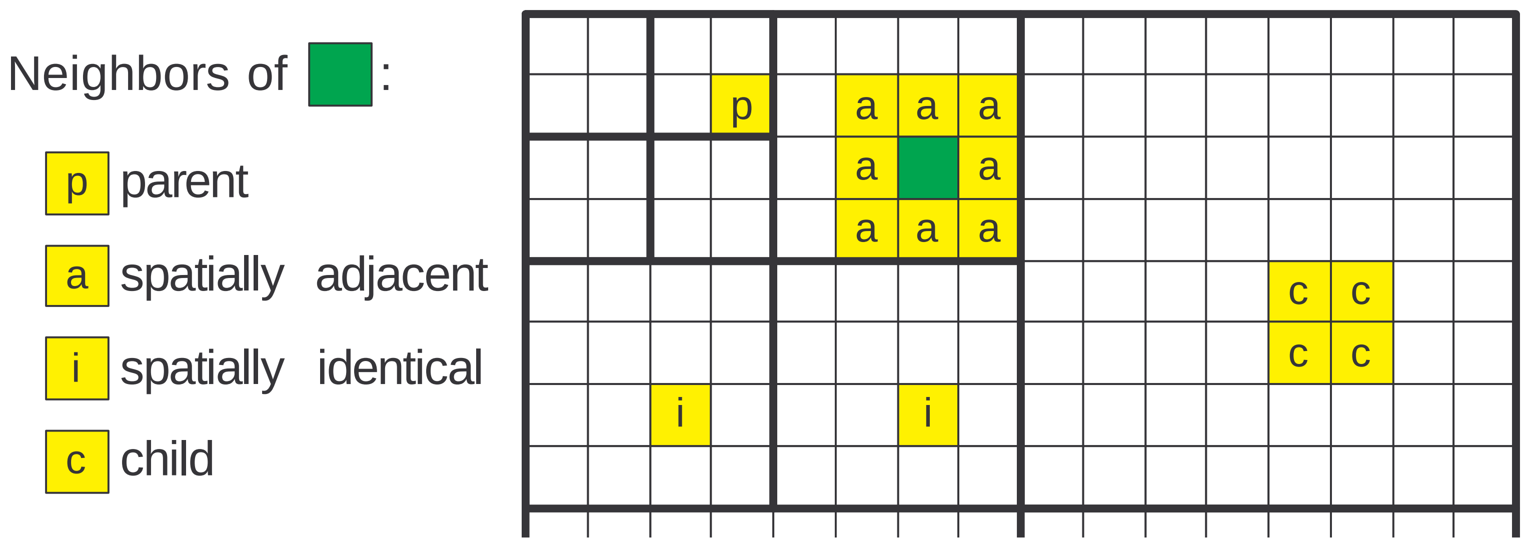
Statistics for Subband Levels

Barbara (8bpp)

Subband Level	# significant	# insignificant	% significant
0	144	364	28.3%
1	272	1048	20.6%
2	848	4592	15.6%
3	3134	23568	11.7%
4	12268	113886	9.7%
5	48282	504633	8.7%
6	190003	2226904	7.8%

Significant Neighbor Metric

- Count # of significant neighbors
  - children count for at most 1
  - 0,1,2,3+



Number of Significant Neighbors

Barbara (8bpp)

Significant neighbors	# significant	# insignificant	% significant
0	4849	2252468	.2%
1	13319	210695	5.9%
2	22276	104252	17.6%
3	30206	78899	27.7%
4	33244	55841	37.3%
5	27354	39189	41.1%
6	36482	44225	45.2%
7	87566	91760	48.8%

Refinement Bit Context Statistics

Barbara (8bpp)

	0's	1's	% 0's
2^{nd} Refinement Bits	146,293	100,521	59.3%
Other Refinement Bits	475,941	433,982	53.3%
Sign Bits	128,145	130,100	49.6%

- Barbara at 2bpp: 2^{nd} Refinement bit % 0's = 65.8%

Context Details

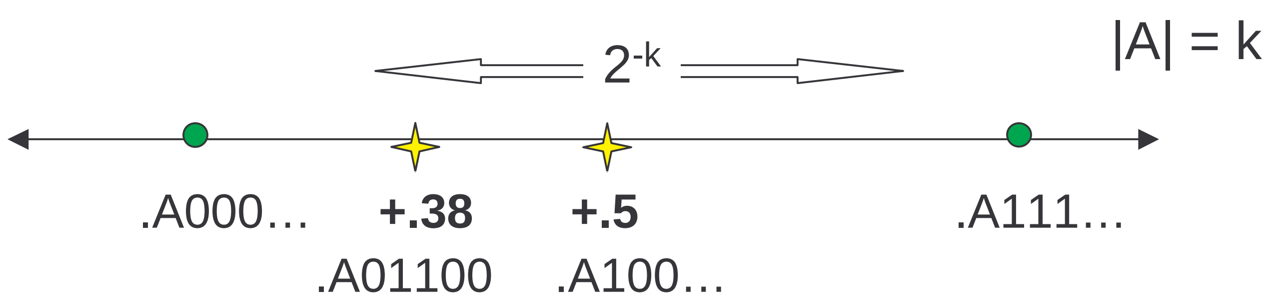
- Significance pass contexts per bit-plane:
  - Max neighbors* num subband levels contexts
  - For Barbara: contexts for sig neighbor counts of 0 - 3 and subband levels of 0-6 = 4*7 = 28 contexts
  - Index of a context.
    - Max neighbors * subband level + num sig neighbors
    - Example num sig neighbors = 2, subband level = 3, index = 4 * 3 + 2 = 14
- Sign context
  - 1 contexts
- 2 Refinement contexts
  - 1st refinement bit is always 1 not transmitted
  - 2nd refinement bit has a context
  - all other refinement bits have a context
- Number of contexts per bit-plane for Barbara = 28 + 1 +2 = 31

Priority Queue

- Used in significance pass to decide which coefficient to code next
  - Goal code coefficients most likely to become significant
- All non-empty contexts are kept in a max heap
- Priority is determined by:
  - # sig coefficients coded / total coefficients coded

Reconstruction of Coefficients

- Coefficients are decoded to a certain number of bit planes
  - .101110XXXXX What should X's be?
  - .101110000… < .101110XXXXX < .101110111…
  - .101110100000 is half-way
- Handled the same as SPIHT and GTW
  - if coefficient is still insignificant, do no interpolation
  - if newly significant, add on .38 to scale
  - if significant, add on .5 to scale



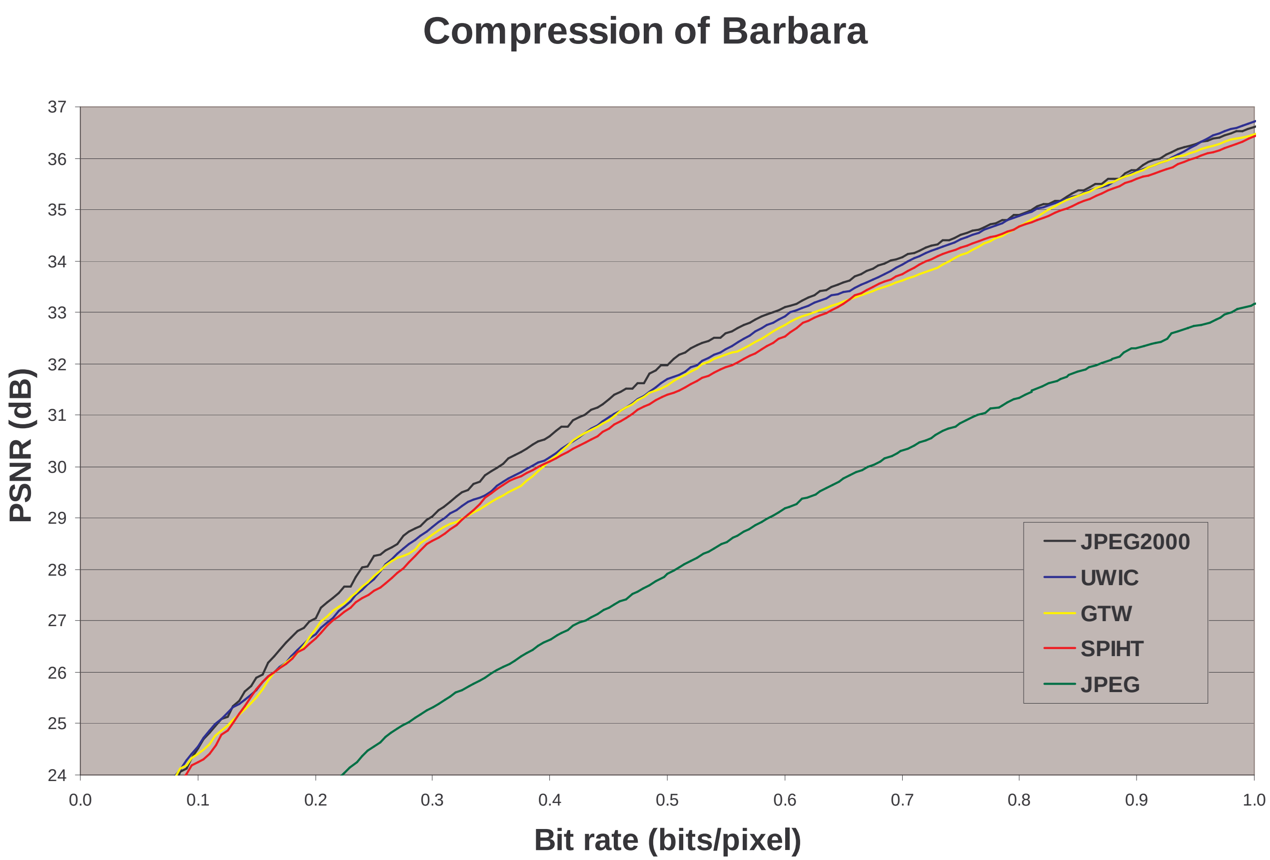
Original Barbara Image

Barbara at .5 bpp (PSNR = 31.68)

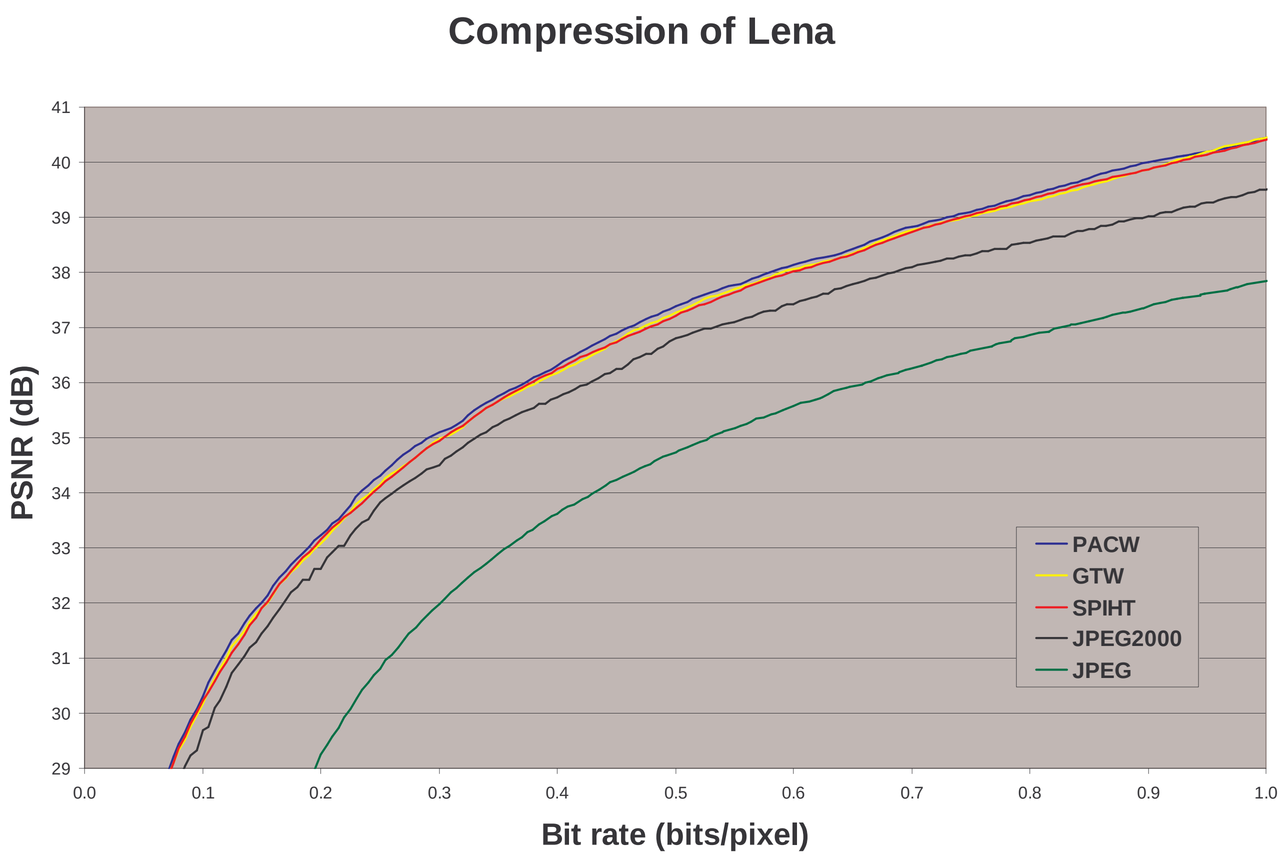
Barbara at .25 bpp (PSNR = 27.75)

Barbara at .1 bpp (PSNR = 24.53)

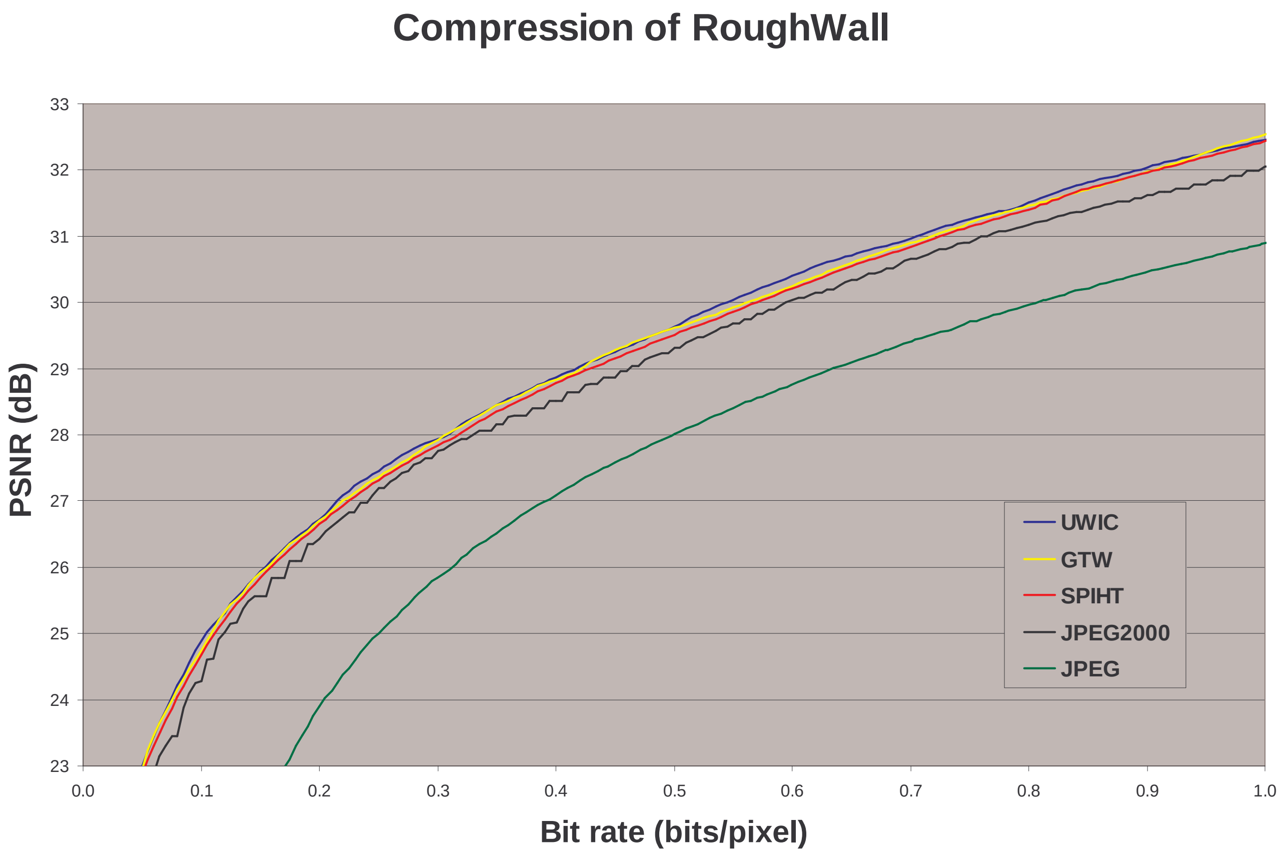
Results


Compression of Barbara
PSNR (dB)
37
36
35
34
33
32
31
30
29
28
27
26
25
24
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Bit rate (bits/pixel)
JPEG2000
UWIC
GTW
SPIHT
JPEG

Results


Compression of Lena
PSNR (dB)
41
40
39
38
37
36
35
34
33
32
31
30
29
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Bit rate (bits/pixel)
PACW
GTW
SPIHT
JPEG2000
JPEG

Results


Compression of RoughWall
PSNR (dB)
Bit rate (bits/pixel)
33
32
31
30
29
28
27
26
25
24
23
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
UWIC
GTW
SPIHT
JPEG2000
JPEG

PACW Notes

- PACW competitive with JPEG 2000, SPIHT-AC, and GTW.
- Developed in Java from scratch by two undergraduates, Dane Barney and Amanda Askew, in 2 months.
- Dane’s final version is slightly different than the one describe here. See his senior thesis.