Image Compression

CS 6640
School of Computing
University of Utah

Compression

- What
  - Reduce the amount of information (bits) needed to represent image
- Why
  - Transmission
  - Storage
  - Preprocessing...

Redundant & Irrelevant Information

- "Your wife Helen will meet you at O'Hare Airport in Chicago at 5 minutes past 6pm tomorrow night"
- Irrelevant or redudant can depend on context
  - Who is receiving the message?

Compression Model

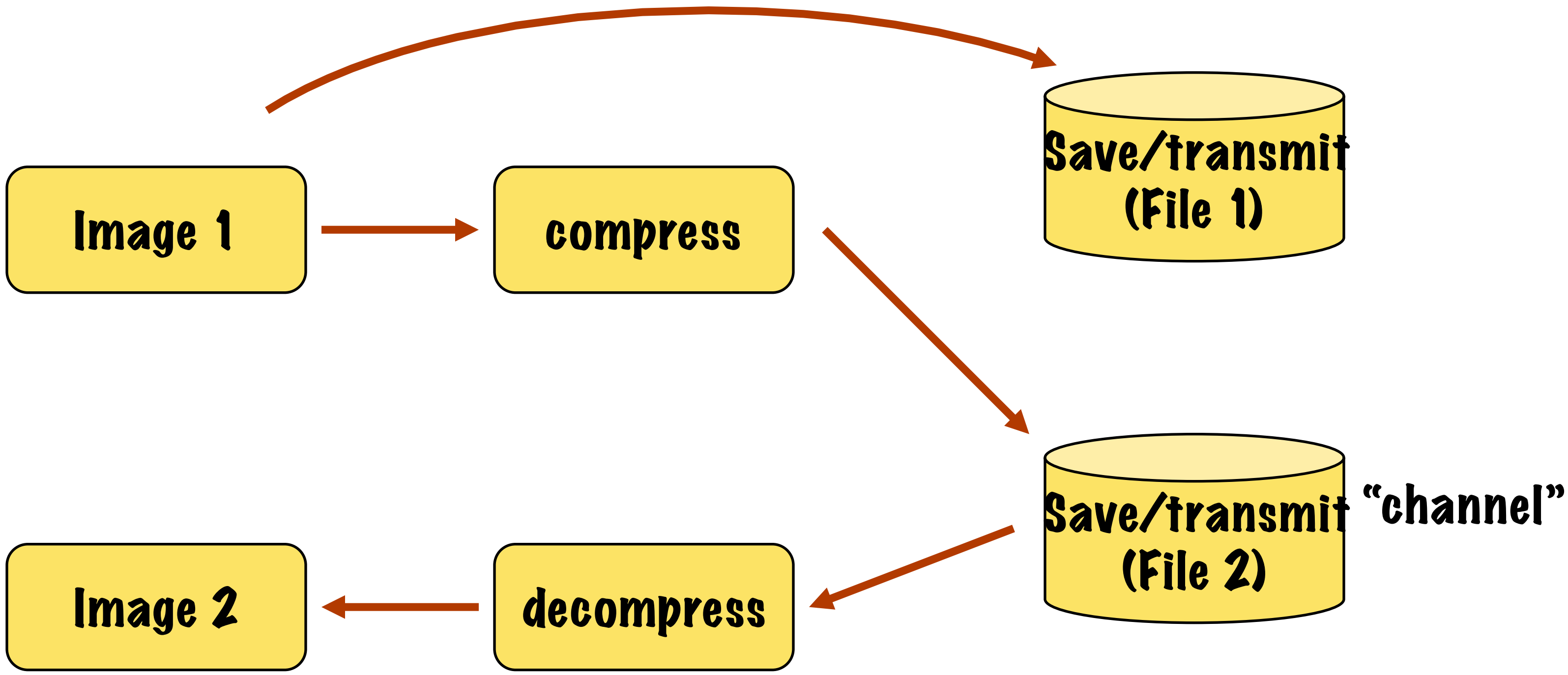


Image1 == Image2 -> "lossless" <- reduces redundant info

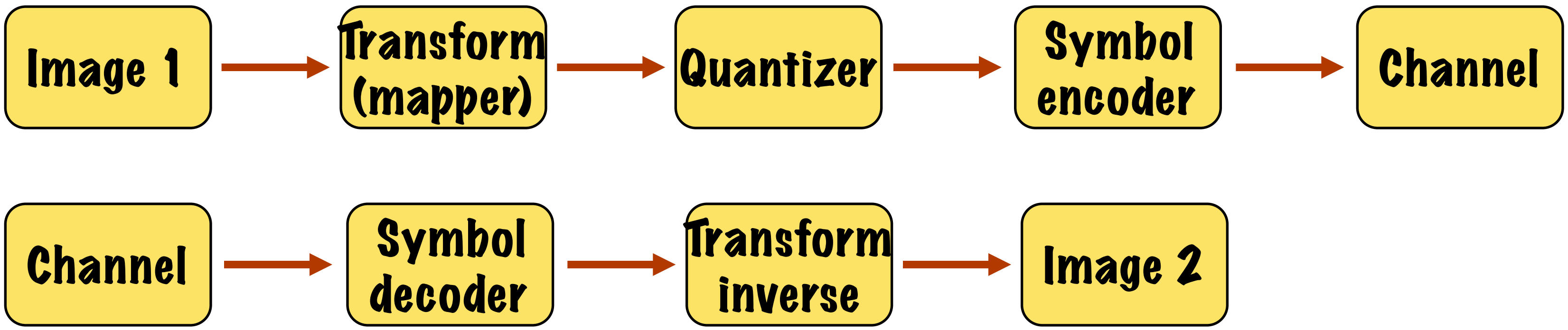
Image1 != Image2 -> "lossy" <- tries to reduce redundant & irrelevant info

Size(File1)/Size(File2) -> "compression ratio"

Redundancy

- Coding redundancy
  - More bits than necessary to create unique codes
- Spatial/geometric redundancy
  - Correlation between pixels
  - Patterns in image
- Psychopysical redundancy (irrelevancy?)
  - Users cannot distinguish
  - Applies to any application (no affect on output)

Transform Coding Standard Strategy

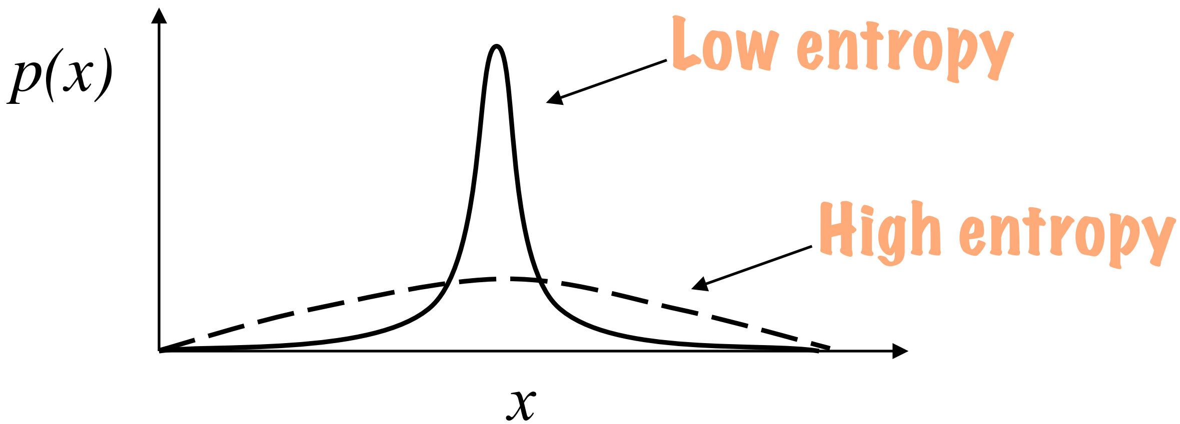


- Note: can have special source or channel modules
  - Account for specific properties of image/ application
  - Account for specific properties of channel (e.g. noise)

Fundamentals

- Information content of a signal -> entropy

$$E\{-\log P(j)\} = -\sum_j P(j) \log P(j)$$



- Lower bound on #bits need to unambiguously represent a sequence of symbols

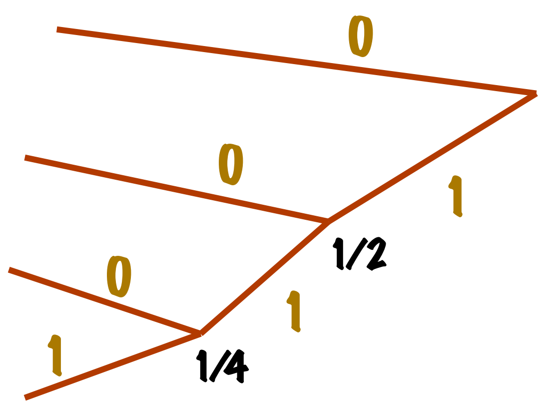
Strategy (optimal)

- Variable-Length Codes
- Devote fewer bits to those symbols that are most likely
  - More generally -> sequences of symbols
- Where do the statistics come from?
  - A-priori knowledge
  - The signal itself (send dictionary)
  - Ad hoc schemes

Huffman Coding

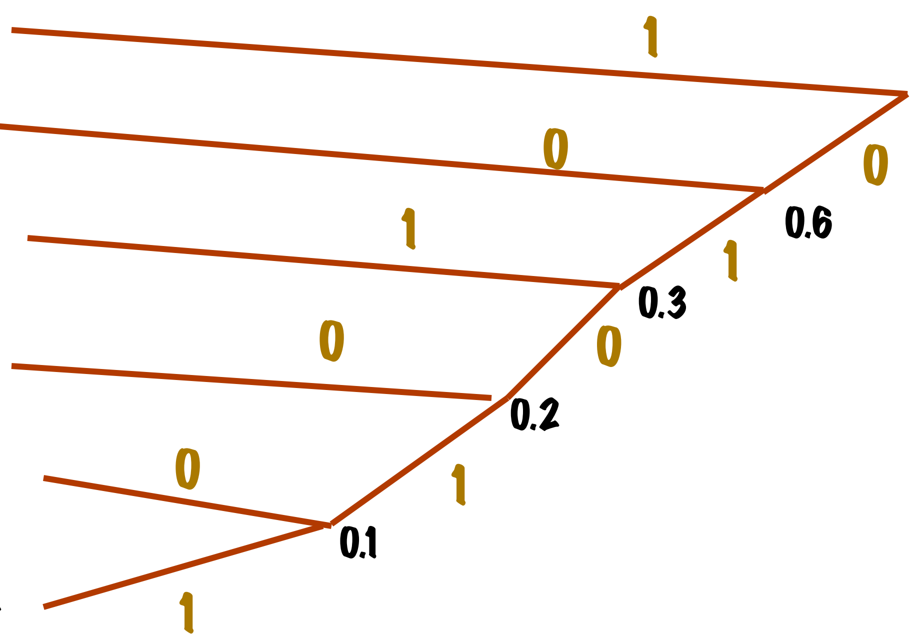
- Input: sumbols and probabilities
- Output: variable length symbol table
  - Coded/decoded one at a time
- Tree

Symbol	Pr()	Code
A1	1/2	0
A2	1/4	10
A3	1/8	110
A4	1/8	111



Huffman Coding

Symbol	Pr()	Code
A2	0.4	1
A6	0.3	00
A1	0.1	011
A4	0.1	0100
A3	0.06	01010
A5	0.04	01011



Fixed Length Codes

- Dictionary with strategy to capture special structure of data
- Example: LZW (Lempel-Ziv-Welch)
  - Start with basic dictionary (e.g. grey levels)
  - As new sequences of symbols are encountered add them to dictionary
    - Hope: encode frequently occuring <u>sequences</u> of symbols
      - Greedy
  - Can decompress w/out table (first occurance not replaced)

LSW Compress

```
w = NIL;
while ( read a character k )
    {
      if wk exists in the dictionary
       w = wk;
      else
        add wk to the dictionary;
        output the code for w;
        w = k;
    }
```

^WED^WE^WEE^WEB^WET

w	k	output	index	symbol
NIL	^			
^	W	^	256	^W
W	E	W	257	WE
E	D	E	258	ED
D	^	D	259	D^
^	W			
^W	E	256	260	^WE
E	^	E	261	E^
^	W			
^W	E			
^WE	E	260	262	^WEE
E	^			
E^	W	261	263	E^W
W	E			
WE	B	257	264	WEB
B	^	B	265	B^
^	W			
^W	E			
^WE	T	260	266	^WET
T	EOF	T		

LSW Decompress

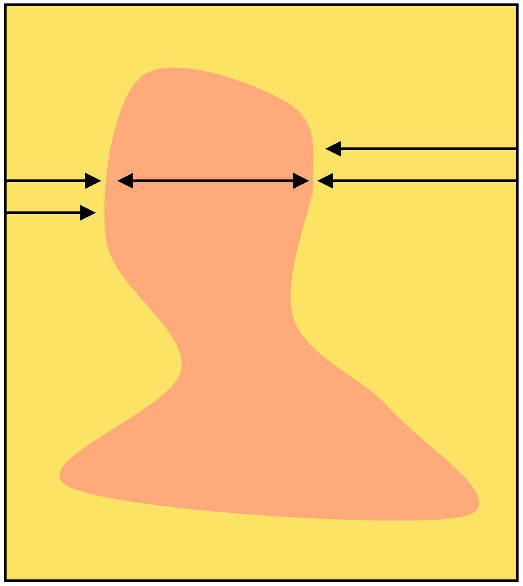
```
read a character k;
  output k;
  w = k;
  while ( read a character k )
 /* k could be a character or a code. */
      {
        entry = dictionary entry for k;
        output entry;
        add w + entry[0] to dictionary;
        w = entry;
      }
```

WED<256>E<260><261><257>B<260>T

w	k	output	index	symbol
	^	^		
^	W	W	256	^W
W	E	E	257	WE
E	D	D	258	ED
D	<256>	^W	259	D^
<256>	E	E	260	^WE
E	<260>	^WE	261	E^
<260>	<261>	E^	262	^WEE
<261>	<257>	WE	263	E^W
<257>	B	B	264	WEB
B	<260>	^WE	265	B^
<260>	T	T	266	^WET

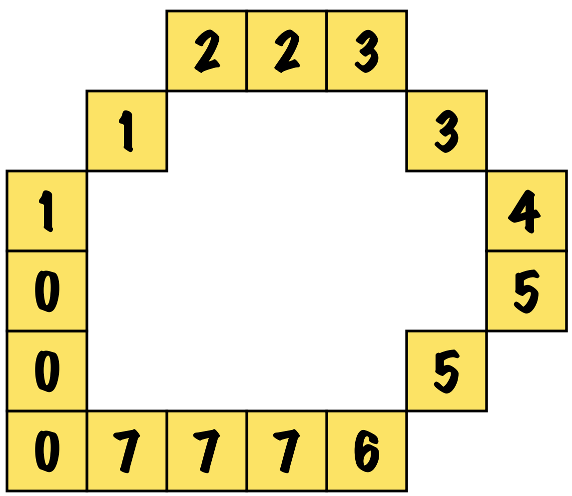
Run Length Enoding (RLE)

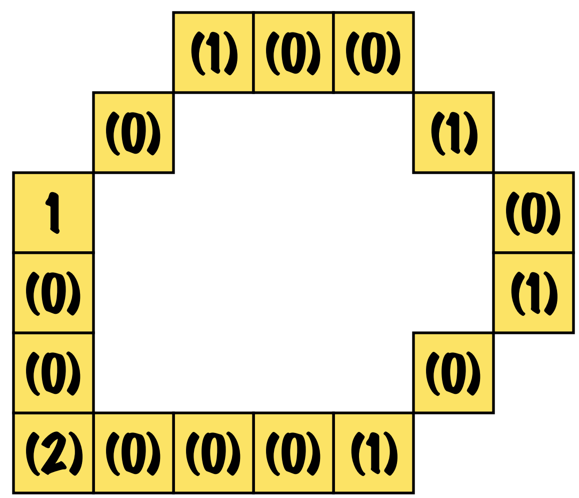
- Good for images with few, discrete color values
- Assumption: images have homogeneous regions
- Strategy
  - Row-major order
  - Encode value of "run" and it's length
  - Can combine with symbol encoder
- Issues
  - How homogeneous is the data?
  - Is there enough continuity in rows?



RLE For 2D

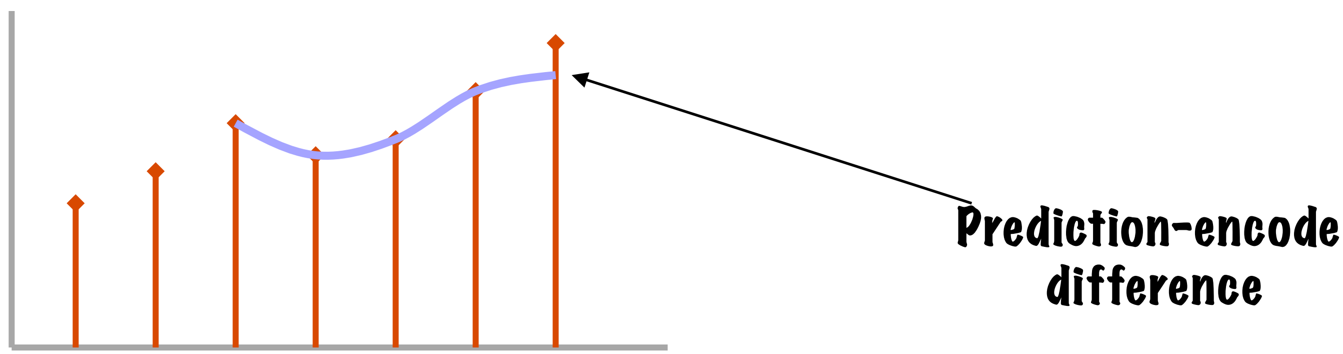
- Complex -> lots of strategies
- Trace contours surrounding regions
- Encode contours using a incremental scheme with a differential strategy (to improve statistics)





Predictive Coding

- Take advantage of correlations
- Have a simple model that predicts data
  - Encode differences from prediction
  - Residual should be lower entropy

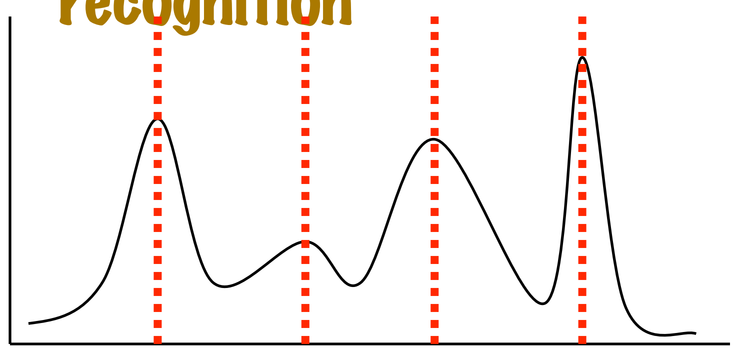


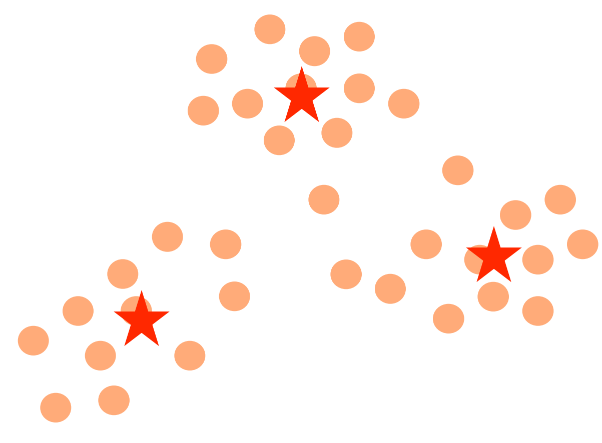
Lossy Compression

- Transforms
  - Move to another representation where "importance" of information is more readily discernable
  - Usually reversible
- Quantization
  - Strategy for reducing the amount of information in the signal
  - Typically not reversible (lossy)

Quantization

- Eliminate symbols that are too small or not important
- Find a small set of approximating symbols (less entropy)
  - Grey level or "vector quantization"
  - Find values that minimize error
  - Related to "clustering" in pattern recognition





Block Transform Coding: JPEG

- International standard (ISO)
- Baseline algorithm with extensions
- Transform: discrete cosine transform (DCT)
  - Encodes freq. Info w/out complex #s
  - FT of larger, mirrored signal
  - Does not have other nice prop. of FT

$$F_u = \alpha(u) \sum_{i=0}^{N-1} f_i \cos\left[\frac{(2i+1)u\pi}{2N}\right]$$

$$F_i = \sum_{u=0}^{N-1} \alpha(u) F_u \cos\left[\frac{(2i+1)u\pi}{2N}\right]$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & u = 0 \\ \sqrt{\frac{2}{N}} & u \neq 0 \end{cases}$$

JPEG Algorithm

- Integer grey-level image broken into 8x8 sub blocks
- Set middle (mean?) grey level to zero (subtract middle)
- DCT of sub blocks (11 bit precision) -> T(u,v)
- Rescale frequency components by Z(u,v) and round

Rescaling

$$\hat{T}(u,v) = \mathrm{round}\left(\frac{\mathrm{T(u,v)}}{\mathrm{Z(u,v)}}\right)$$

- Different scalling matrices possible, but recommended is:

$$Z(u,v) = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Reordering

- DCT entries reordered in zig-zag fashion to increase coherency (produce blocks of zeros)

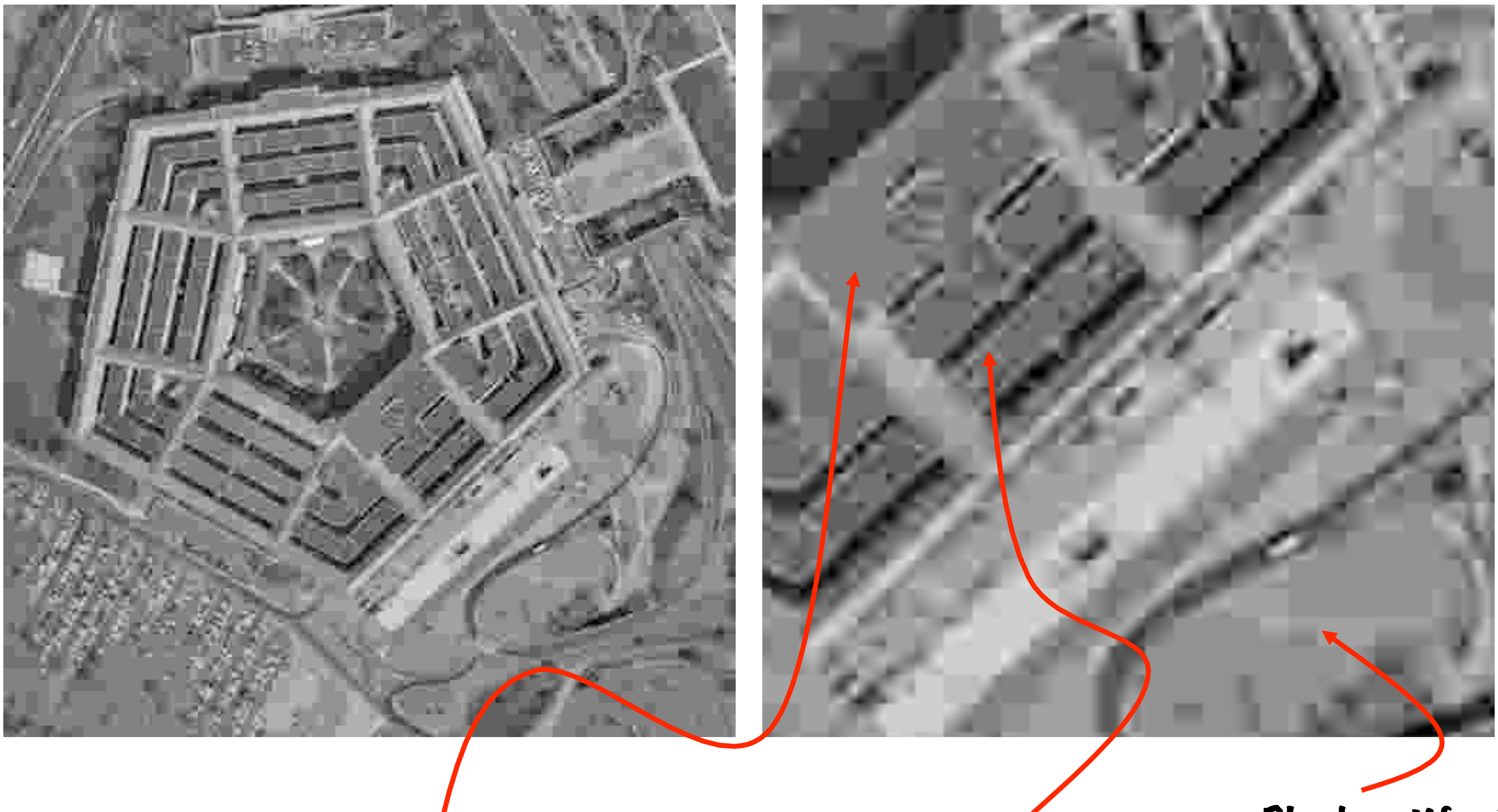
$$
\begin{bmatrix}
0 & 1 & 5 & 6 & 14 & 15 & 27 & 28 \\
2 & 4 & 7 & 13 & 16 & 26 & 29 & 42 \\
3 & 8 & 12 & 17 & 25 & 30 & 41 & 43 \\
9 & 11 & 18 & 24 & 31 & 40 & 44 & 53 \\
10 & 19 & 23 & 32 & 39 & 45 & 52 & 54 \\
20 & 22 & 33 & 38 & 46 & 51 & 55 & 60 \\
21 & 34 & 37 & 47 & 50 & 56 & 59 & 61 \\
35 & 36 & 48 & 49 & 57 & 58 & 62 & 63
\end{bmatrix}
$$

Coding

- Each sub-block is coded as a difference from previous sub-block
- Zeros are run-length encoded and nonzero elements are Huffman coded
  - Modified HC to allow for zeros

JPEG Example

Compression Ratio ~10:1

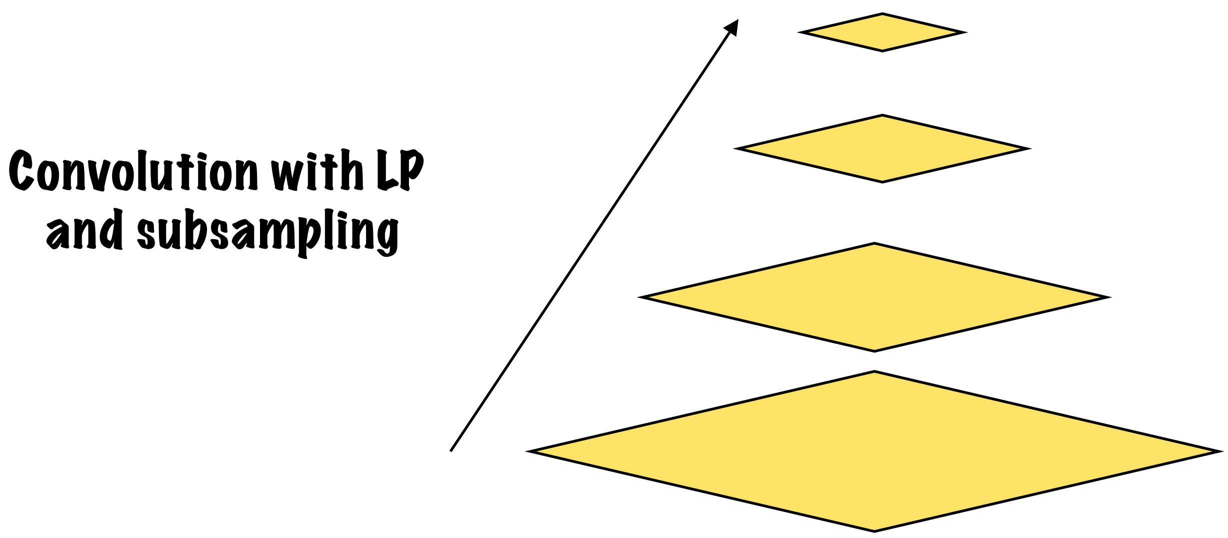


Other Transformations

- Sub-band coding
  - Band-pass transformations that partition the Fourier domain into pieces
  - Convolve with those filters and take advantage of sparse structure
    - Hopefully many values near zero (quantization)
- Wavelets
  - Multiscale filters
  - Like subband filters but typically other properties
    - Eg. Orthogonal (inner between diff filters in bank is zero)

Wavelets as Hierarchical Decomposition

- Image pyramids
  - Represent low-frequency information at coarser scale (less resolution)



Wavelet Example: Harr

Mother wavelet

$$\psi(t) = \begin{cases} 1 & 0 \le t < 1/2, \\ -1 & 1/2 \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Scaling function

$$\varphi(t) = \begin{cases} 1 & 0 \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Orthogonality

$$\int_{-\infty}^{\infty} 2^m \psi(2^{m_1} t - n_1) \psi(2^m t - n) \, dt = \delta(m - m_1) \delta(n - n_1)$$

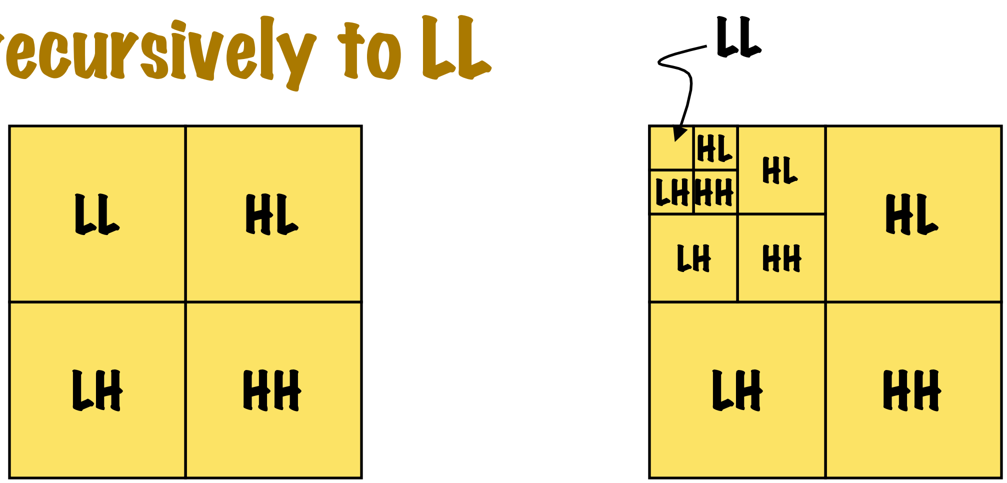
Transformation Matrix

1D signal, discrete, 8 samples ->

1	1	1	1	1	1	1	1
1	1	1	1	−1	−1	−1	−1
1	1	−1	−1	0	0	0	0
0	0	0	0	1	1	−1	−1
1	−1	0	0	0	0	0	0
0	0	1	−1	0	0	0	0
0	0	0	0	1	−1	0	0
0	0	0	0	0	0	1	−1

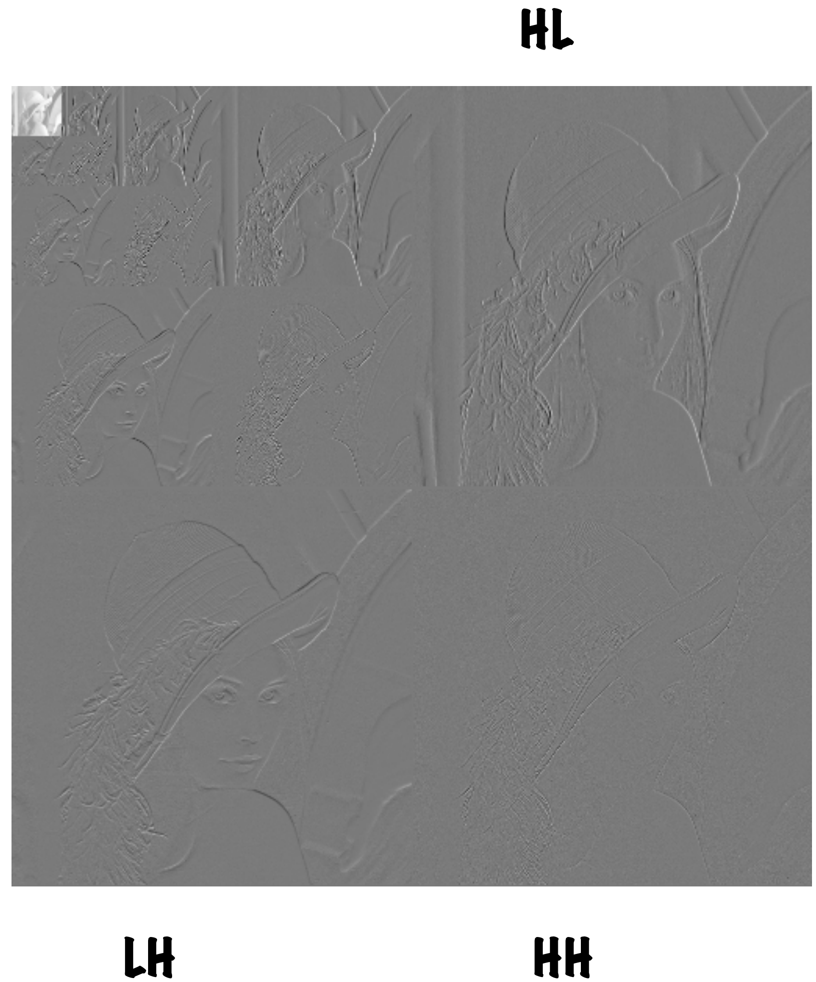
Extending to 2D

- Must take all combinations of wavelet and scaling function at a given scale
  - LL, HL, LH, HH
- Typically organized in blocks, recursively
  - LL is futher decomposed by lower frequency wavelets
  - Apply recursively to LL

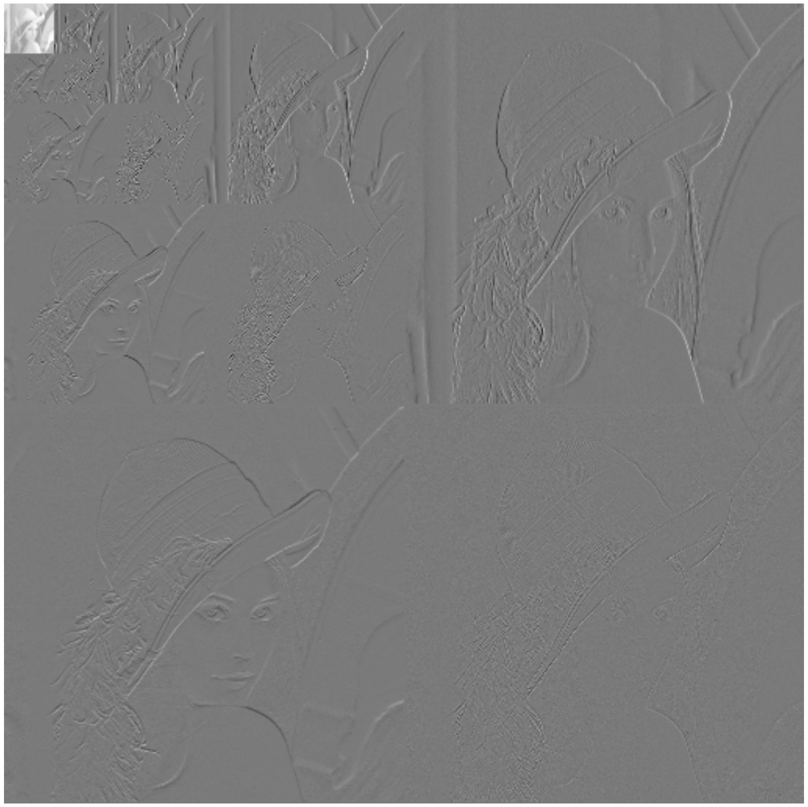


Wavelet Decomposition

Wavelet Decomposition



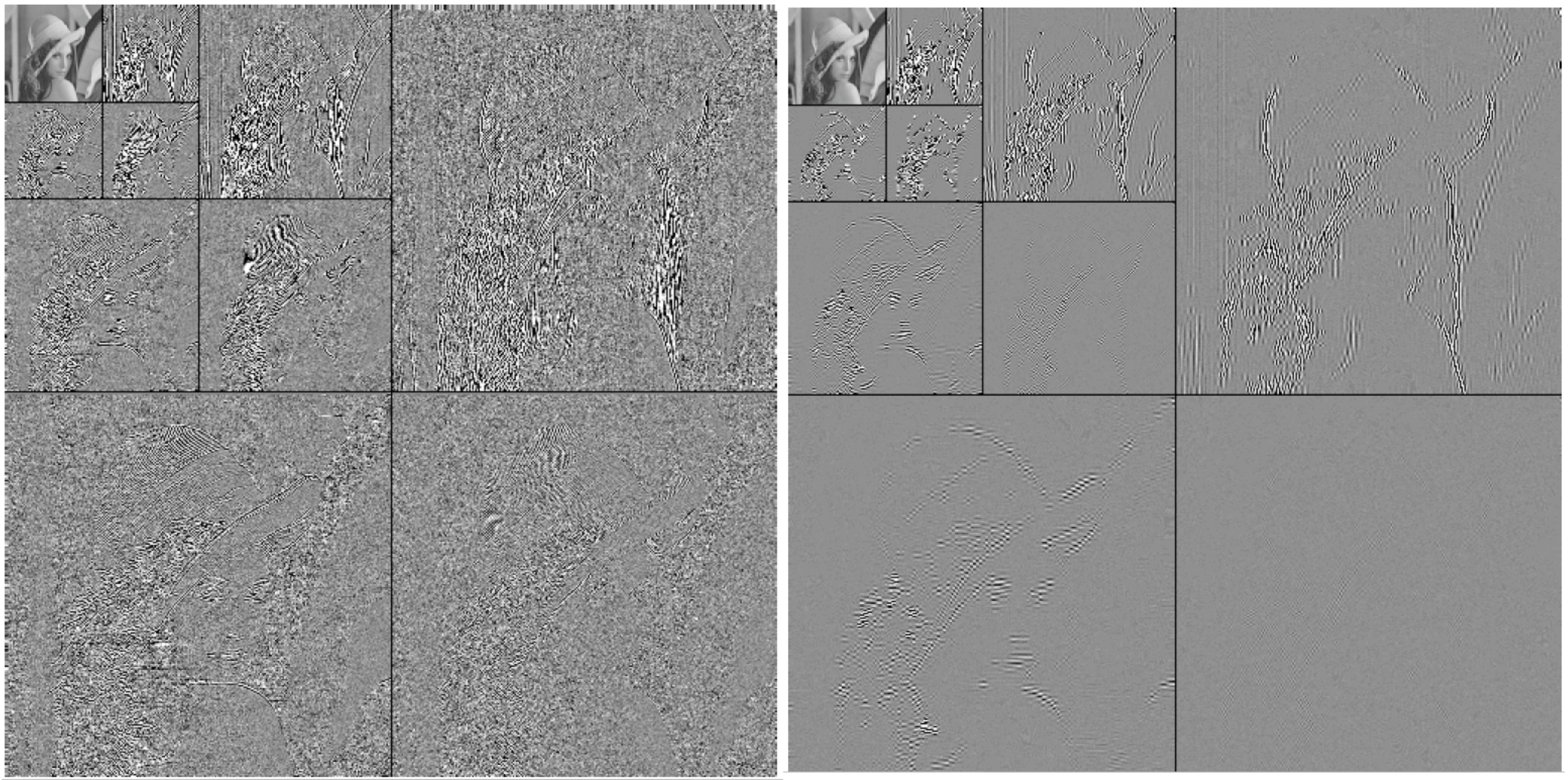
Wavelet Decomposition



Wavelet Compression Algorithm

- Like JPEG, but use DWT instead of DCT
- Steps
  - Transform
  - Weights (emprical)
  - Quantize
  - Entropy (lossless encoding) through RLE, VLC, or dictionary

Wavelet Compression

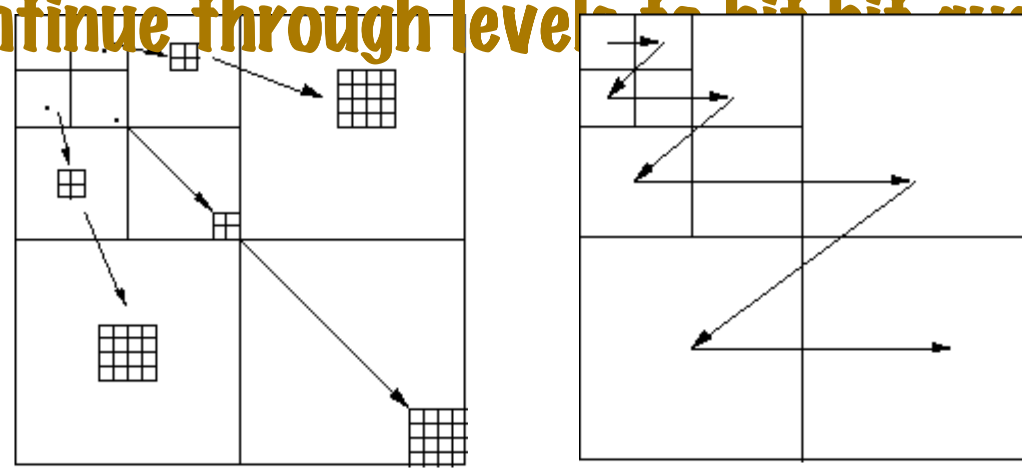


DWT Compression Artifacts

~80:1

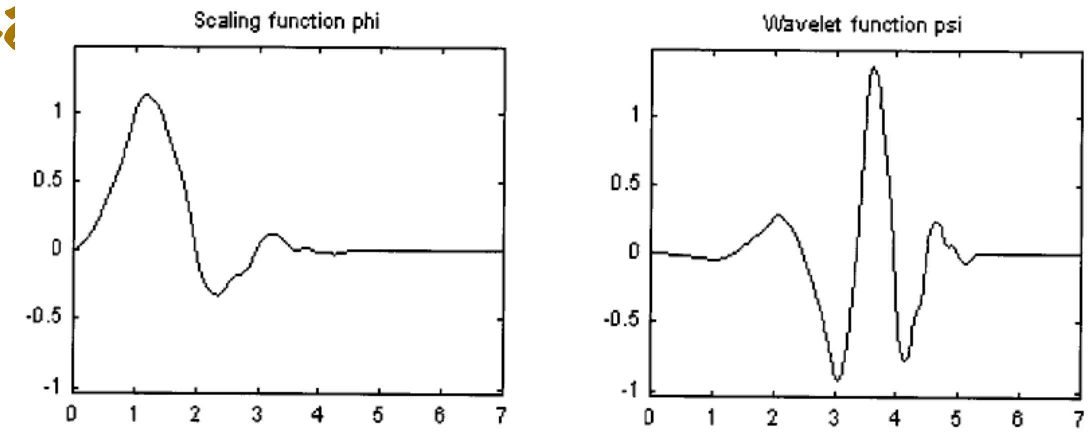
Smarter Ways To Encode

- Embedded zero-tree wavelets (Shapiro 1993)
  - Zeros (threshold) at coarse level likely to be indicative of finer level
  - E.g. edges
  - Continue through levels to hit bit quota

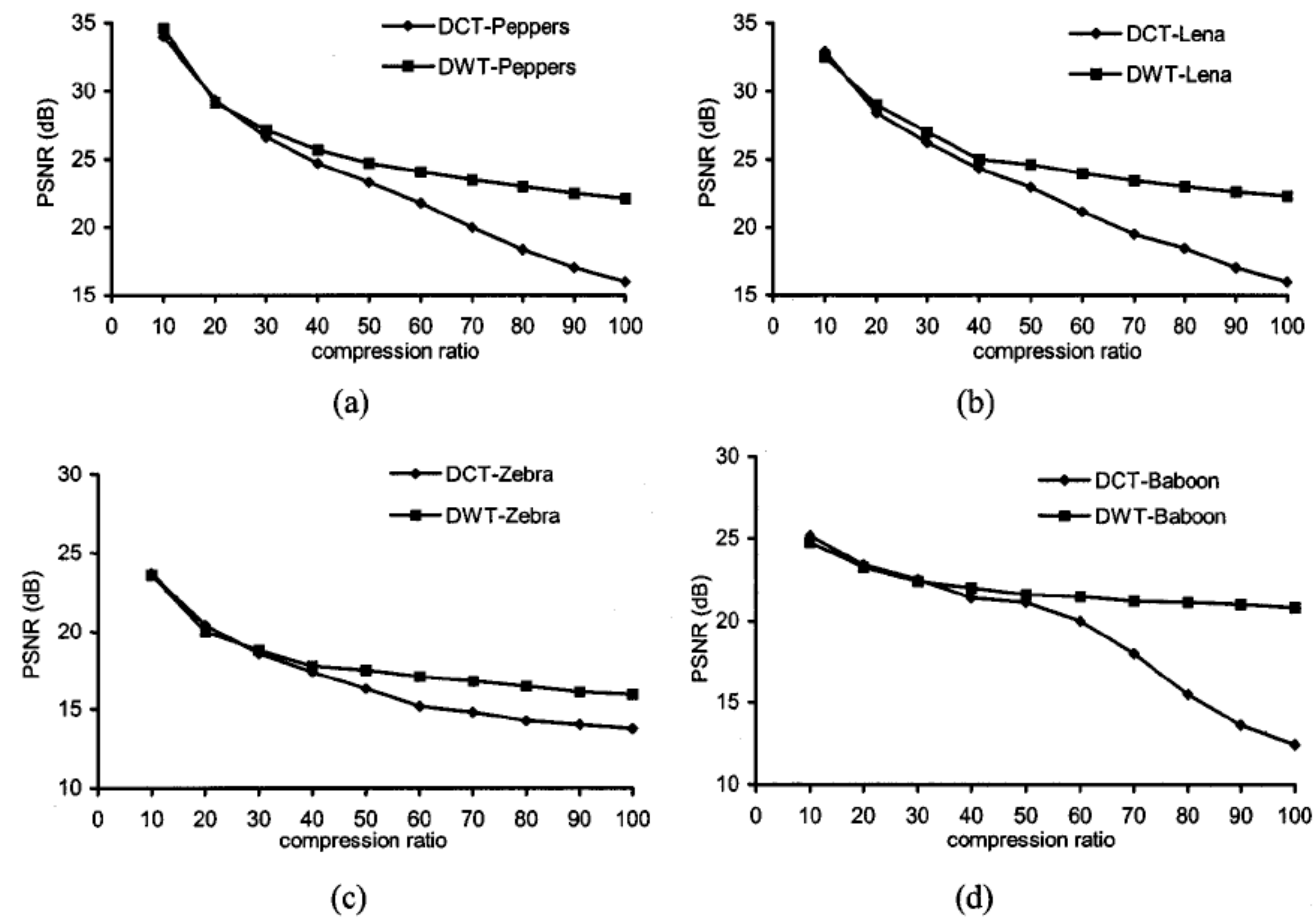


Other Wavelets

- Harr is orthogonal, symmetric, discontinuous
- Daubechies biorthogonal wavelet
  - Continuous, but not symmetric
  - Family of wavelets with parameters
  - JPEG 2000 ca[...] biorthogonal"



Comparisons of Compression



Grgic et al., 2001