Image Compression

CS 6640
School of Computing
University of Utah

G&W, 3rd Ed., Ch 8

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

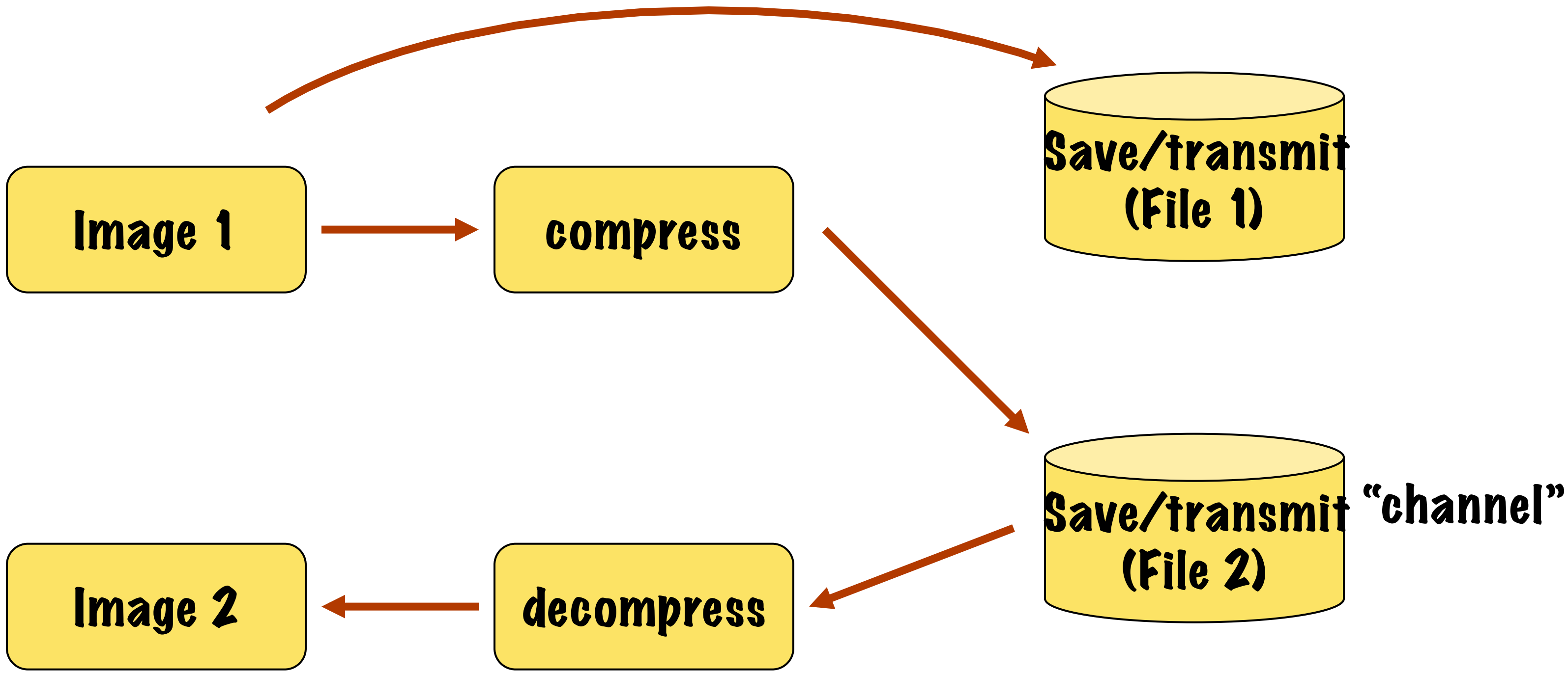


Image1 == Image2 -> "lossless" <- reduces redundant info

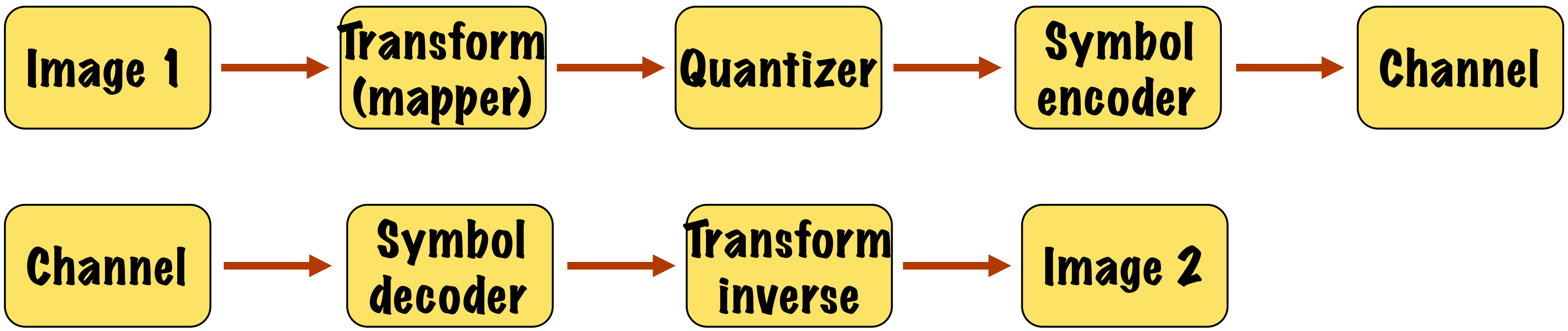
Image1 != Image2 -> "lossy" <- tries to reduce redundant & irrelevant info

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

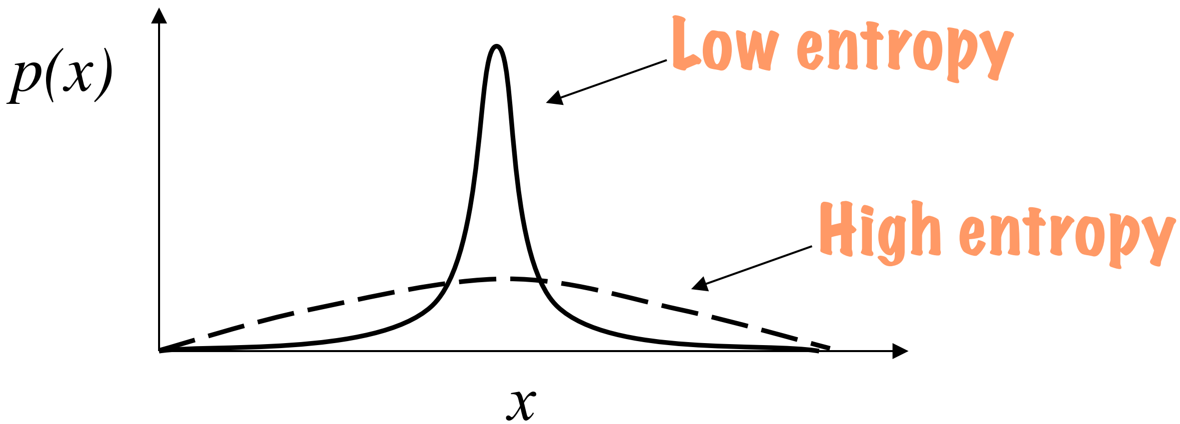


- Note: can have special source or channel modules
  - Account for specific properties of image/ application
  - Account for specific properties of channel (e.g. noise)

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

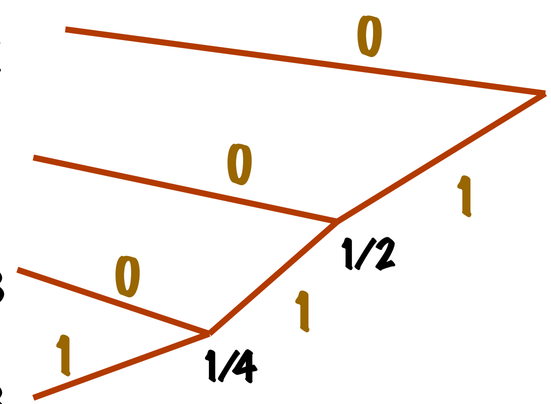
Strategy (optimal)

- 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

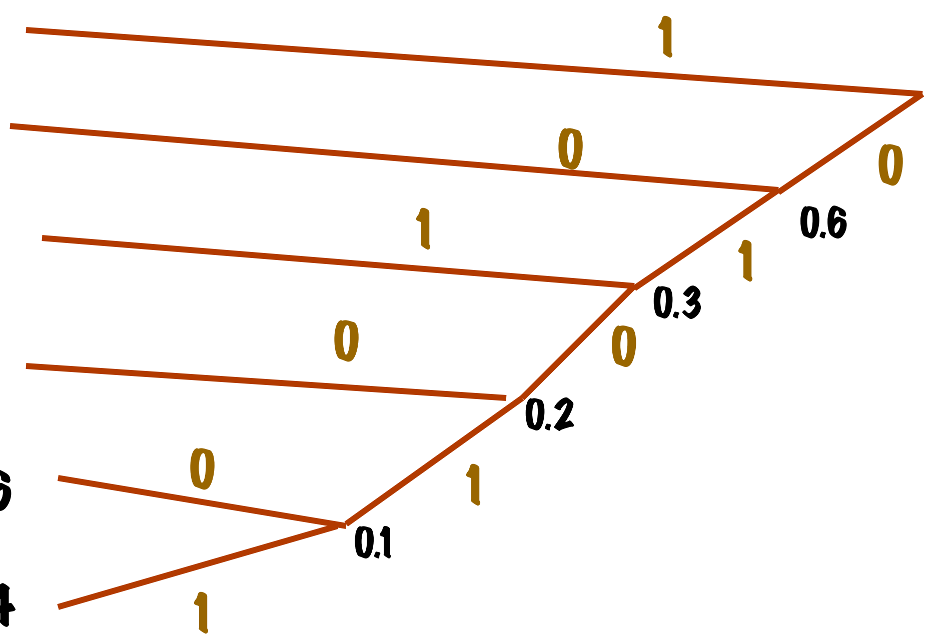
- Input: sumbols and probabilities
- Output: variable length symbol table
  - Coded/decoded one at a time
- Tree

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 sequences of symbols
      - Greedy
  - Can decompress w/out table (first occurance not replaced)

LZW 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		

LZW Decompress

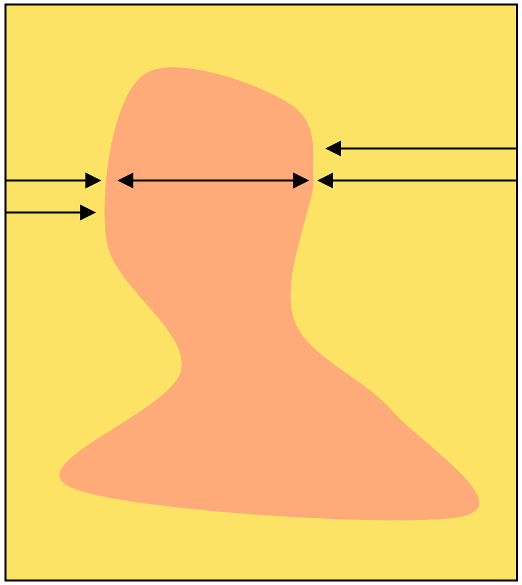
```
read a character k;
  output k;
  w = k;
  while ( read a character k )
 /* k could be a character or a code. */
     {
       entry = dictionary entry for k;
       output entry;
       add w + entry[0] to dictionary;
       w = entry;
     }
```

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

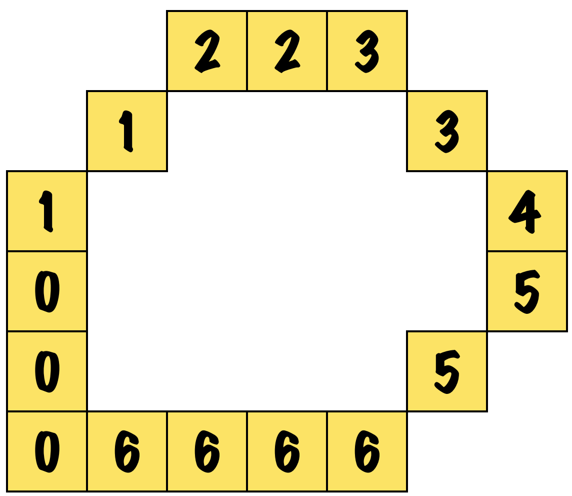
Run Length Enoding (RLE)

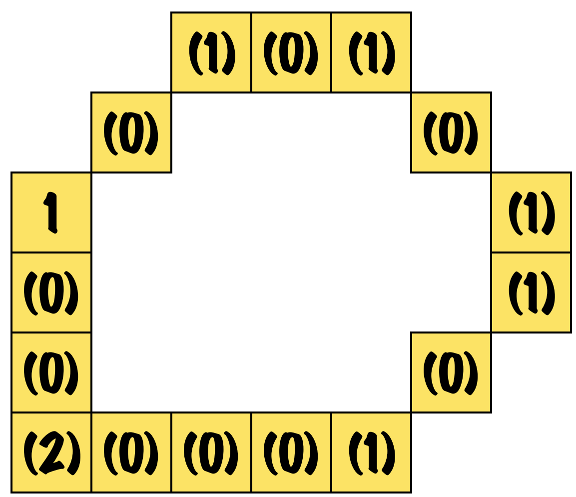
- Good for images with few, discrete color values
- Assumption: images have homogeneous regions
- Strategy
  - Row-major order
  - Encode value of “run” and it’s length
  - Can combine with symbol encoder
- Issues
  - How homogeneous is the data?
  - Is there enough continuity in rows?



RLE For 2D

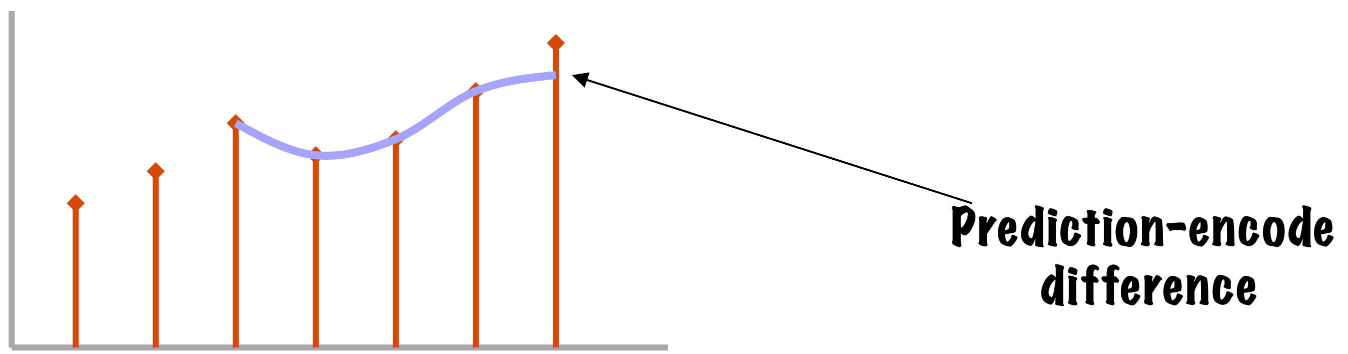
- Complex -> lots of strategies
- Trace contours surrounding regions
- Encode contours using a incremental scheme with a differential strategy (to improve statistics)





Predictive Coding

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

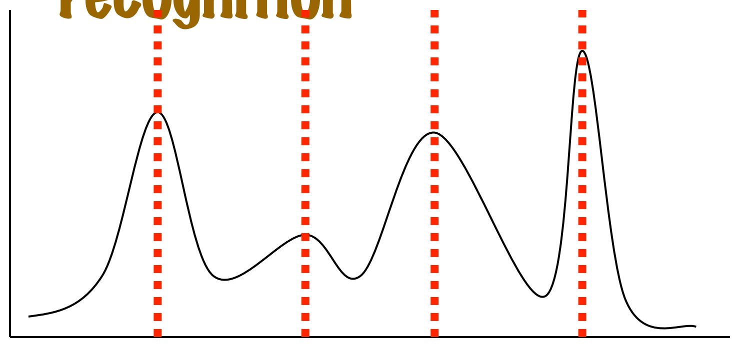


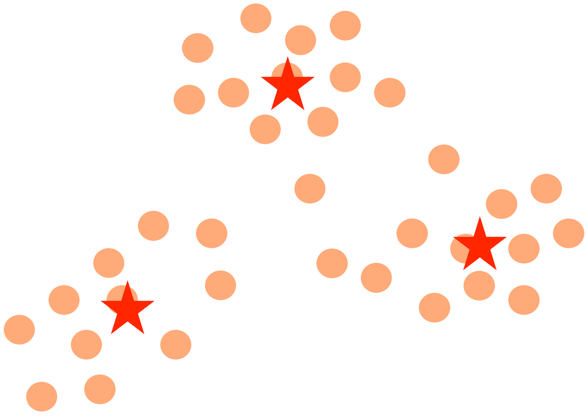
Lossy Compression

- 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

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

- 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)

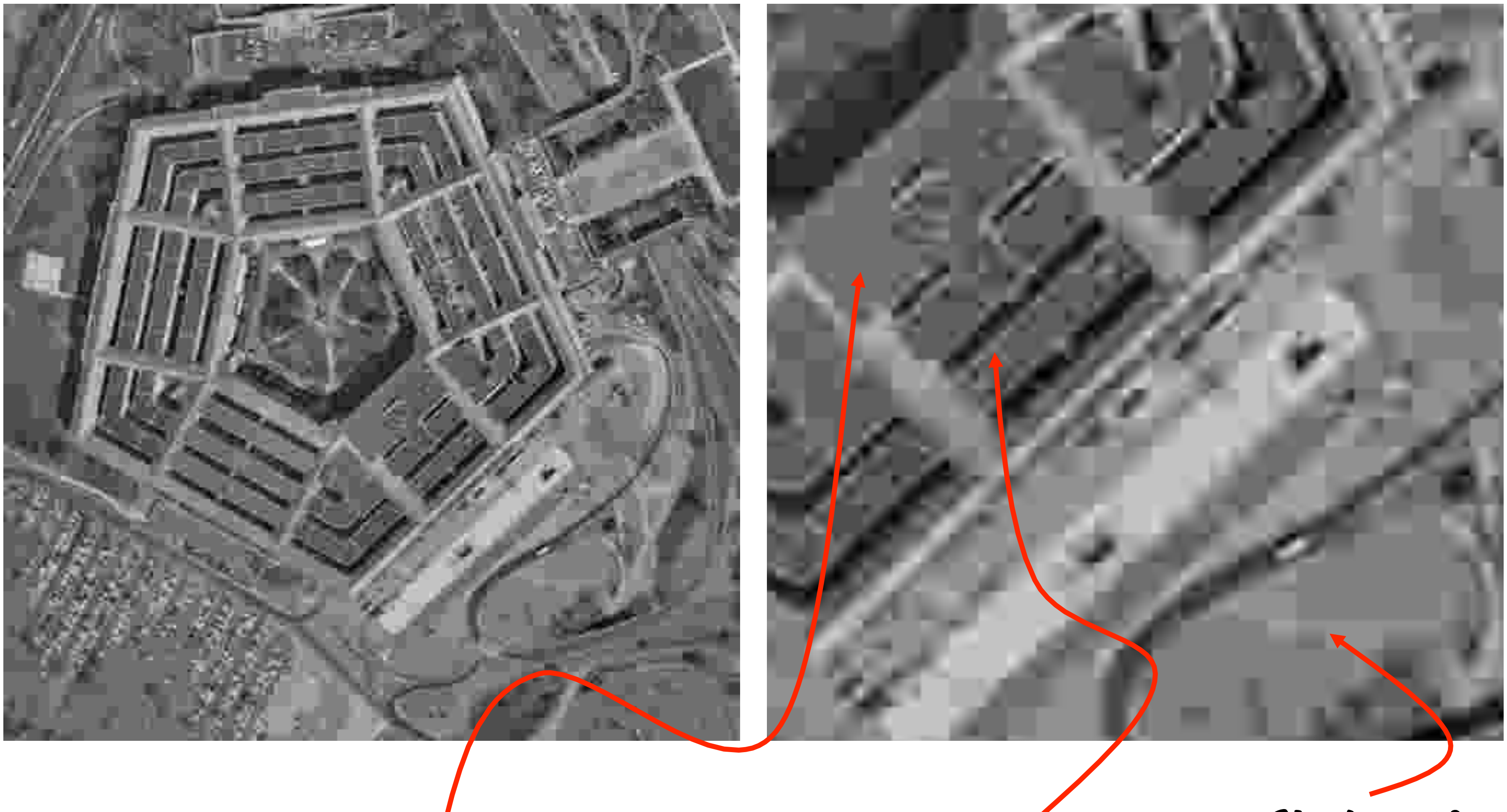
$$\begin{bmatrix} 0 & 1 & 5 & 6 & 14 & 15 & 27 & 28 \\ 2 & 4 & 7 & 13 & 16 & 26 & 29 & 42 \\ 3 & 8 & 12 & 17 & 25 & 30 & 41 & 43 \\ 9 & 11 & 18 & 24 & 31 & 40 & 44 & 53 \\ 10 & 19 & 23 & 32 & 39 & 45 & 52 & 54 \\ 20 & 22 & 33 & 38 & 46 & 51 & 55 & 60 \\ 21 & 34 & 37 & 47 & 50 & 56 & 59 & 61 \\ 35 & 36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

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



Loss of high frequencies

Ringing

Block artifacts

Other Transformations

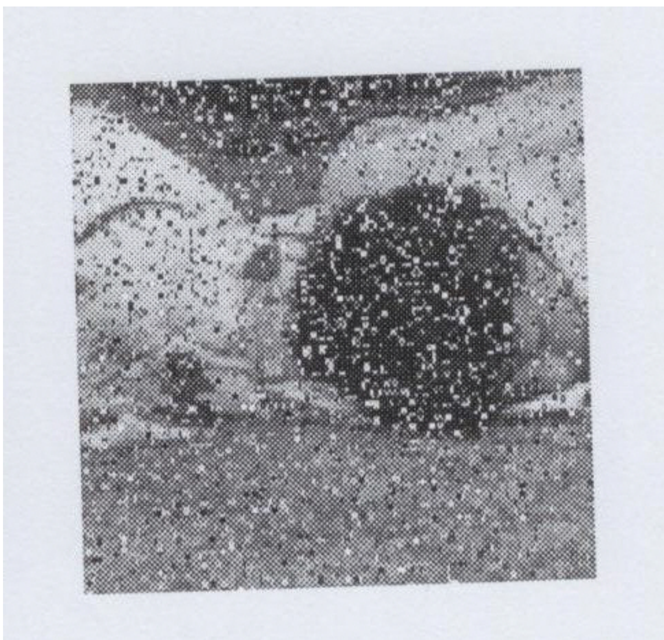
- Sub-band coding
  - Band-pass transformations that partition the Fourier domain into pieces
  - Convolve with those filters and take advantage of sparse structure
    - Hopefully many values near zero (quantization)
- Wavelets
  - Multiscale filters
  - Like subband filters but typically other properties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

List of topics

- Why transform?
- Why wavelets?
- Wavelets like basis components.
- Wavelets examples.
- Fast wavelet transform .
- Wavelets like filter.
- Wavelets advantages.

(Wavelet slides from Burd Alex, U. of Haifa)

Why transform?



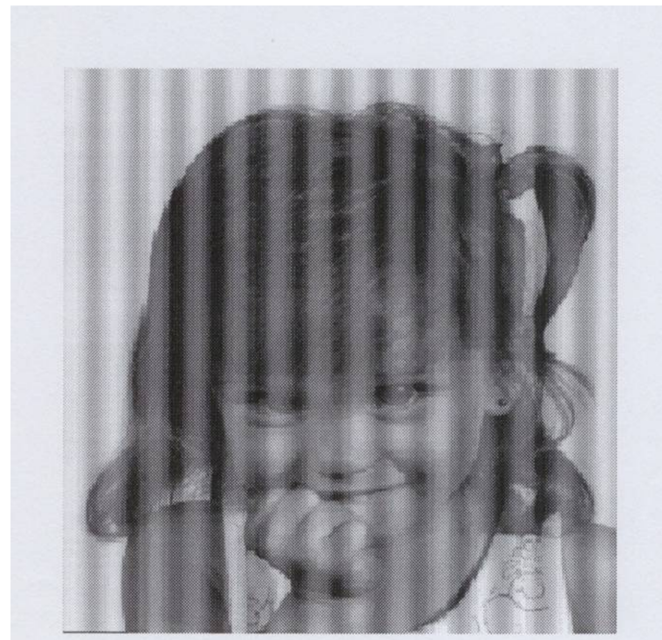
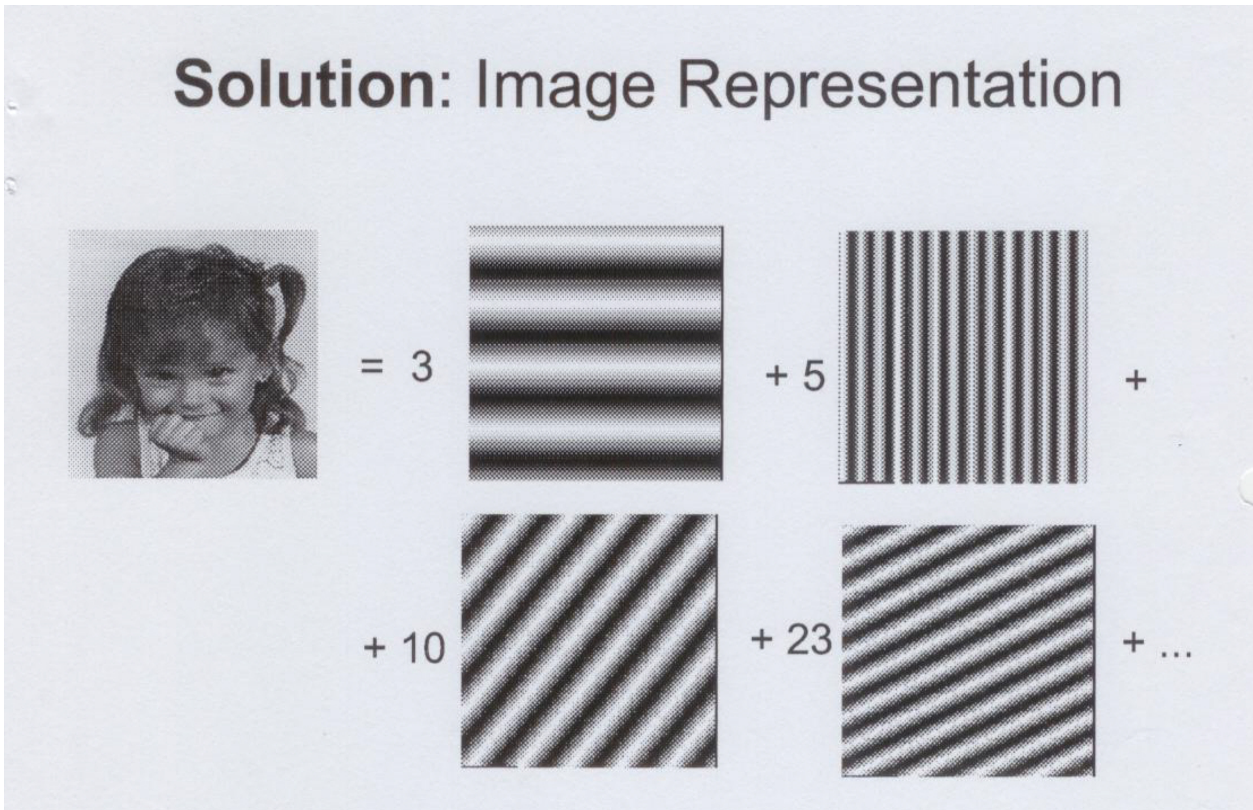
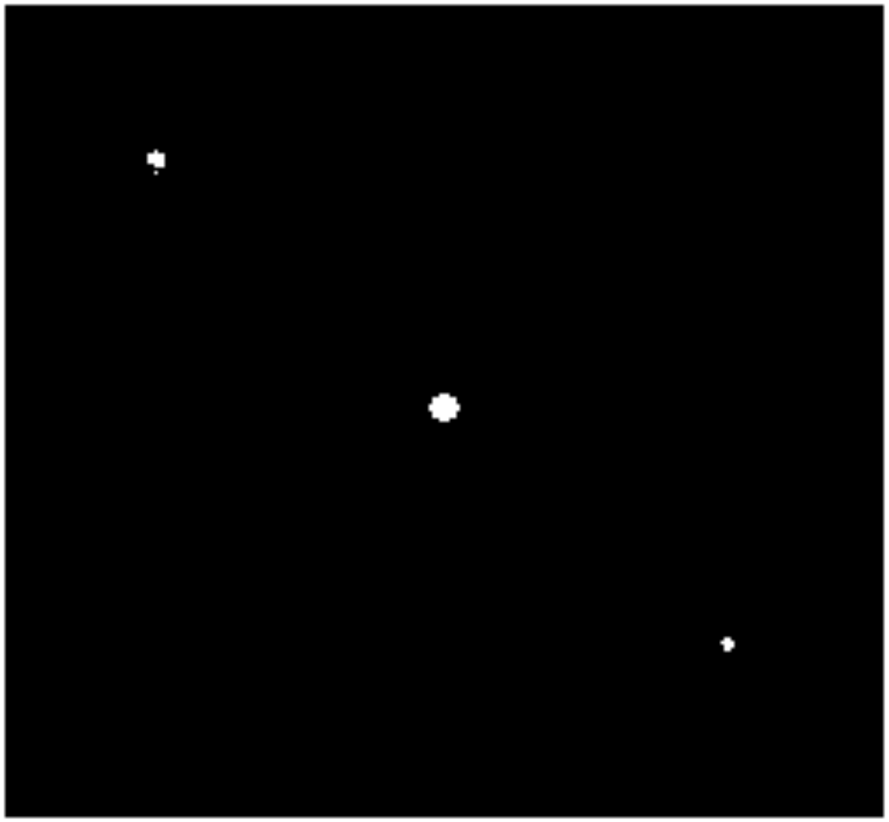


Image representation

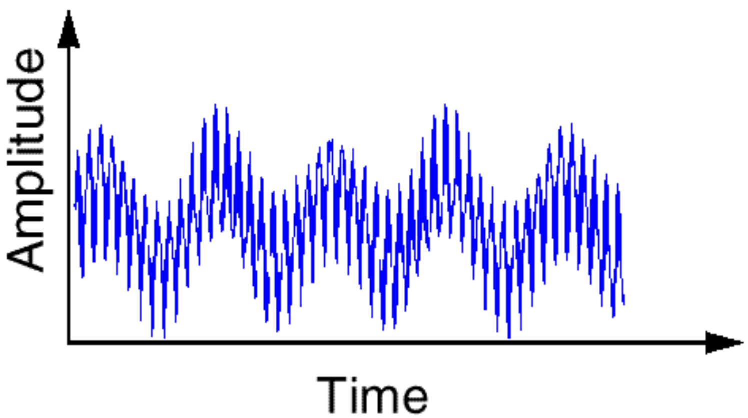


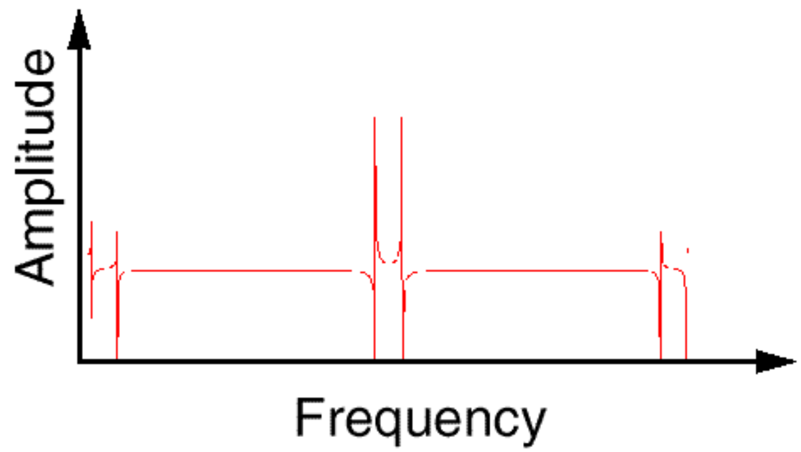
Noise in Fourier spectrum



Fourier Analysis

- Breaks down a signal into **constituent sinusoids** of different frequencies





In other words: Transform the view of the signal from time-base to frequency-base.

What's wrong with Fourier?

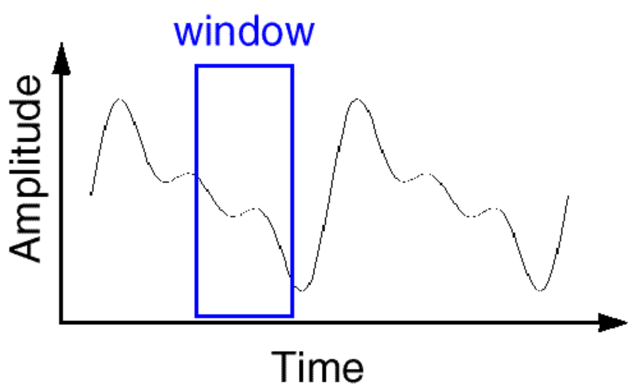
- By using Fourier Transform , we loose the time information : WHEN did a particular event take place ?
- FT can not locate drift, trends, abrupt changes, beginning and ends of events, etc.
- Calculating use complex numbers.

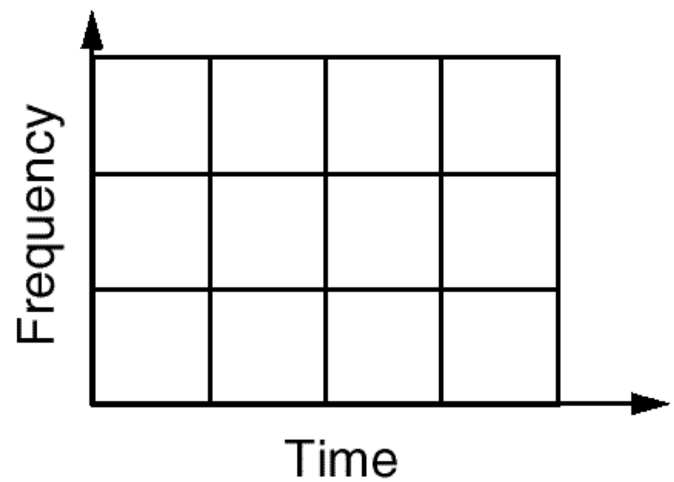
Time and Space definition

- Time – for one dimension waves we start point shifting from source to end in time scale .
- Space – for image point shifting is two dimensional .
- Here they are synonyms .

Short Time Fourier Analysis

- Analyze a small section of a signal
- Denis Gabor (1946) developed ***windowing***: STFT





STFT (or: Gabor Transform)

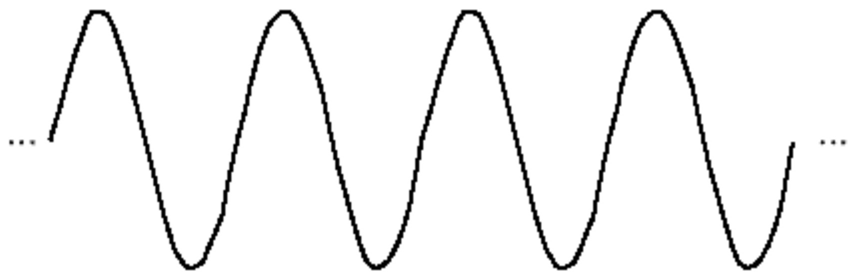
- A compromise between time-based and frequency-based views of a signal.
- both time and frequency are represented in limited precision.
- The precision is determined by the size of the window.
- Once you choose a particular size for the time window – it will be the same for all frequencies.

What's wrong with Gabor?

- Many signals require a more flexible approach - so we can vary the window size to determine more accurately either time or frequency.

What is Wavelet Analysis ?

- And...what is a wavelet...?



Sine Wave

Wavelet (db10)

- A wavelet is a waveform of effectively limited duration that has an average value of zero.

Wavelet's properties

- Short time localized waves with zero integral value.
- Possibility of time shifting.
- Flexibility.

The Continuous Wavelet Transform (CWT)

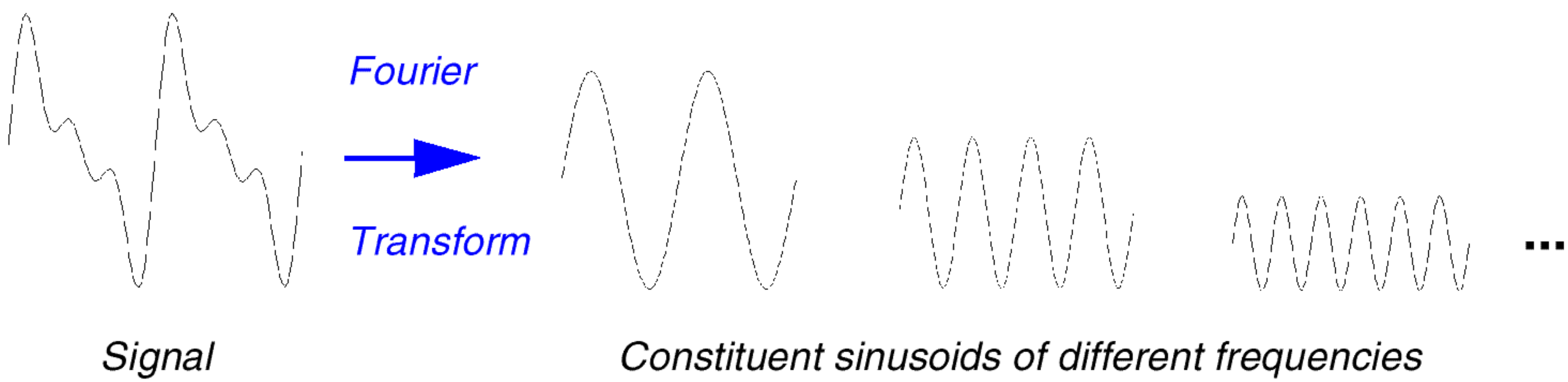
- A mathematical representation of the Fourier transform:

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-iwt} dt$$

- Meaning: the sum over all time of the signal *f(t)* multiplied by a complex exponential, and the result is the Fourier coefficients F(w) .

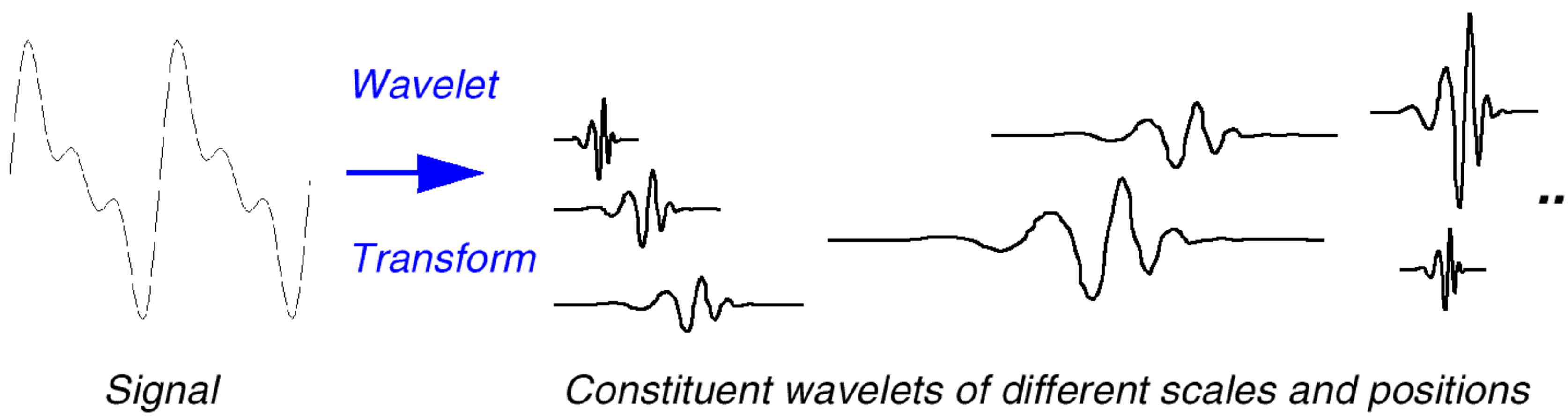
Wavelet Transform

- Those coefficients, when multiplied by a sinusoid of appropriate frequency w, yield the constituent sinusoidal component of the original signal:



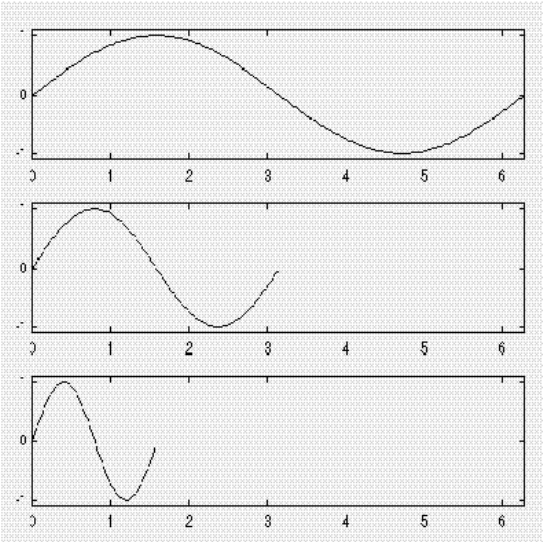
Wavelet Transform

- The result of the CWT are Wavelet coefficients .
- Multiplying each coefficient by the appropriately scaled and shifted wavelet yields the constituent wavelet of the original signal:



Scaling

- Wavelet analysis produces a time-scale view of the signal.
- *Scaling* means stretching or compressing of the signal.
- Like a scale factor (a) for sine waves:



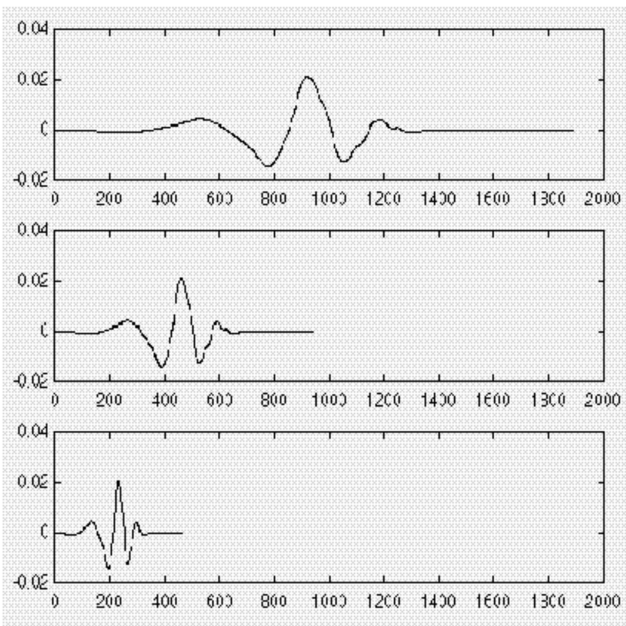
$$f(t) = \sin(t) \quad ; \quad a = 1$$

$$f(t) = \sin(2t) \quad ; \quad a = \tfrac{1}{2}$$

$$f(t) = \sin(4t) \quad ; \quad a = \tfrac{1}{4}$$

Scaling

- Scale factor works exactly the same with wavelets:



$$f(t) = \Psi(t) \quad ; \quad a = 1$$

$$f(t) = \Psi(2t) \quad ; \quad a = \tfrac{1}{2}$$

$$f(t) = \Psi(4t) \quad ; \quad a = \tfrac{1}{4}$$

Wavelet function

$$\Psi_{a,b}(x) = \frac{1}{a}\Psi\left(\frac{x-b}{a}\right)$$

- b - shift coefficient
- a - scale coefficient

$$\Psi_{a,b_x,b_y}(x,y) = \frac{1}{a}\Psi\left(\frac{x-b_x}{a}, \frac{x-b_y}{a}\right)$$

- 2D function

CWT

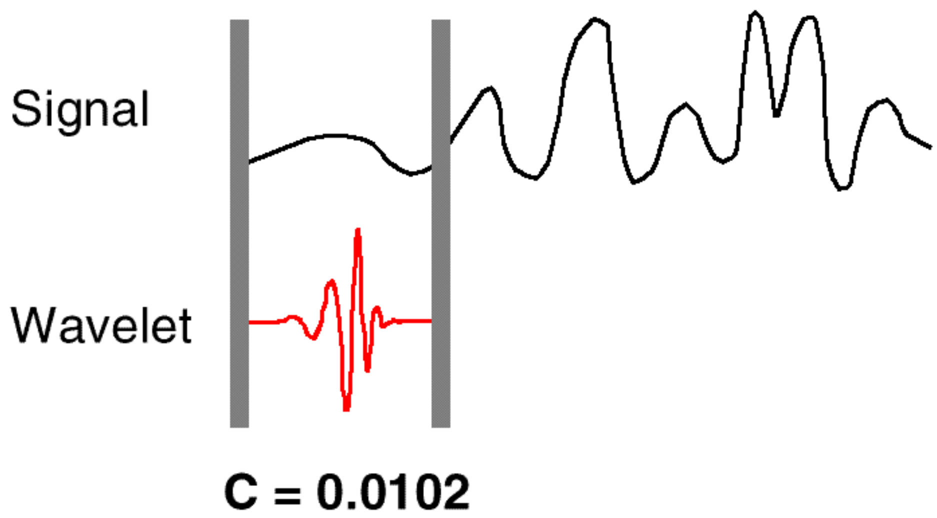
- Reminder: The *CWT* is the sum over all time of the signal, multiplied by scaled and shifted versions of the wavelet function

Step 1:
Take a Wavelet and compare it to a section at the start of the original signal

CWT

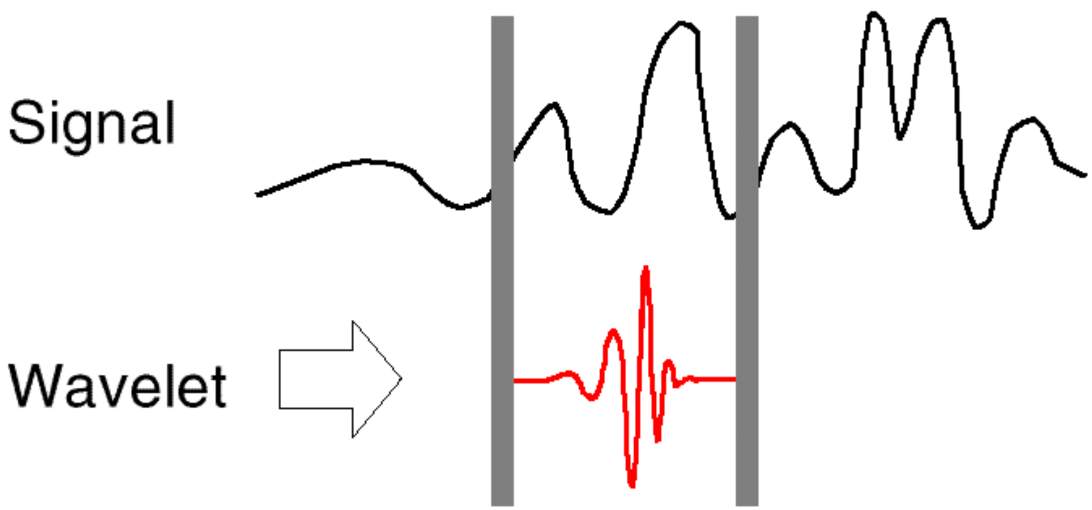
Step 2:

Calculate a number, C, that represents how closely correlated the wavelet is with this section of the signal. The higher C is, the more the similarity.



CWT

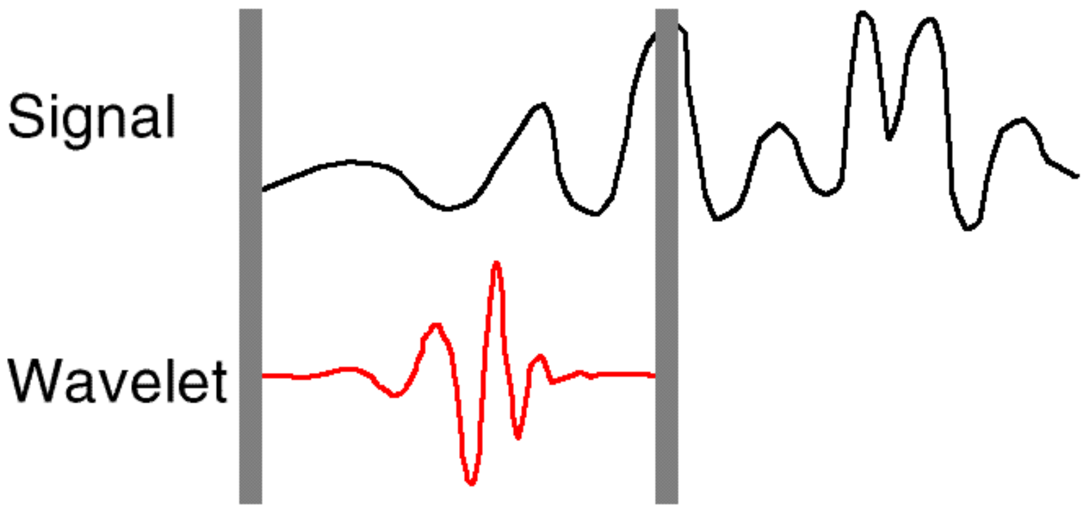
- **Step 3:** Shift the wavelet to the right and repeat steps 1-2 until you've covered the whole signal



amount of shift proportional to scale of wavelet

CWT

- **Step 4:** Scale (stretch) the wavelet and repeat steps 1-3



C = 0.2247

Wavelets examples

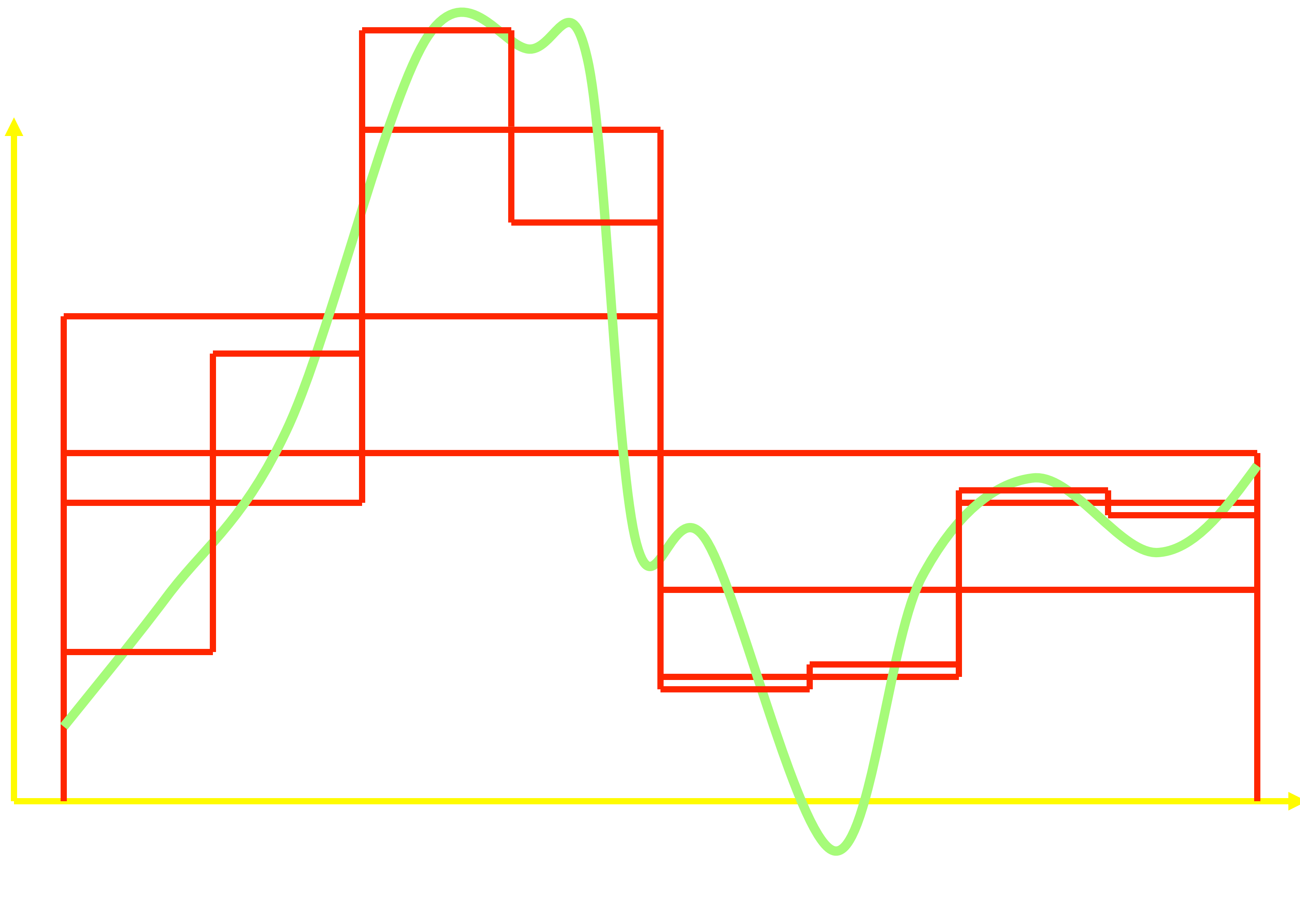
Dyadic transform

- For easier calculation we can to discrete continuous signal.
- We have a grid of discrete values that called dyadic grid .
- Important that wavelet functions compact (e.g. no overcalculatings) .

$$a = 2^j$$

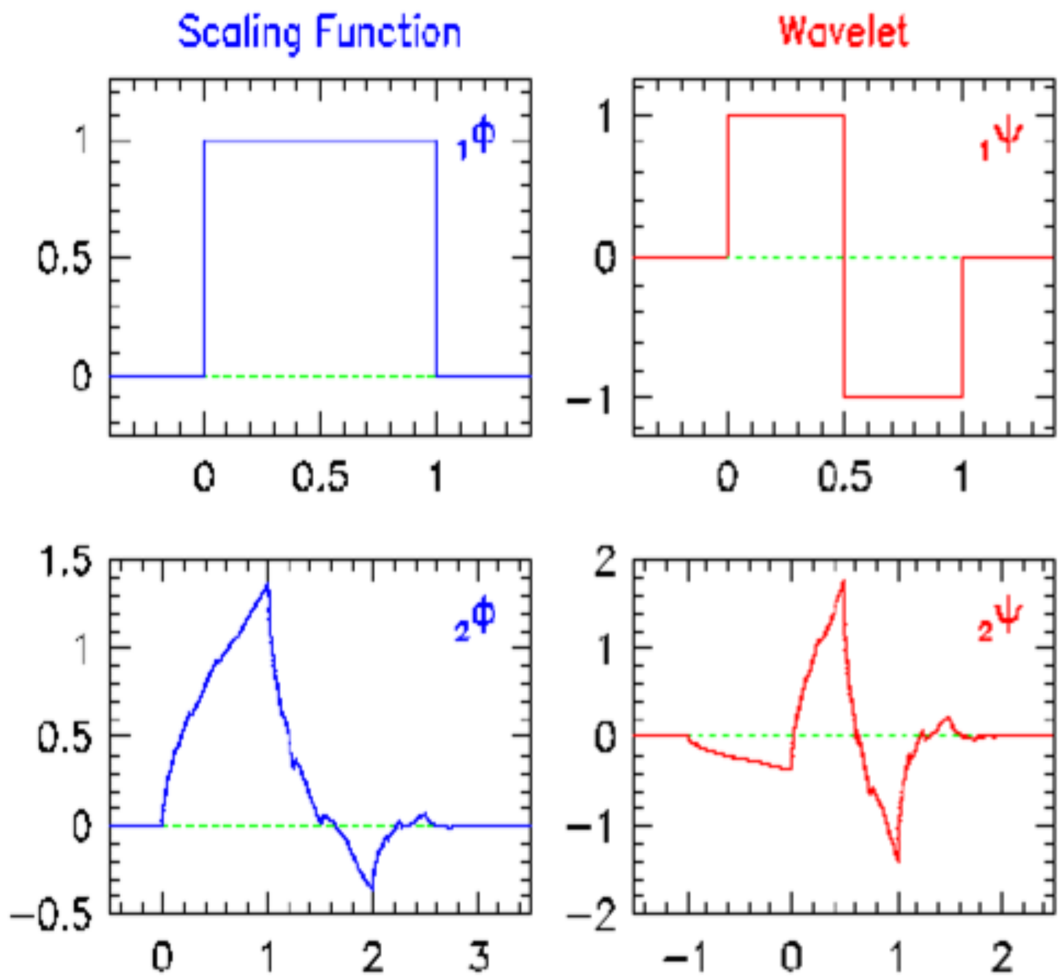
$$b = k2^j$$

Haar transform



Wavelet functions examples

- Haar function
- Daubechies function



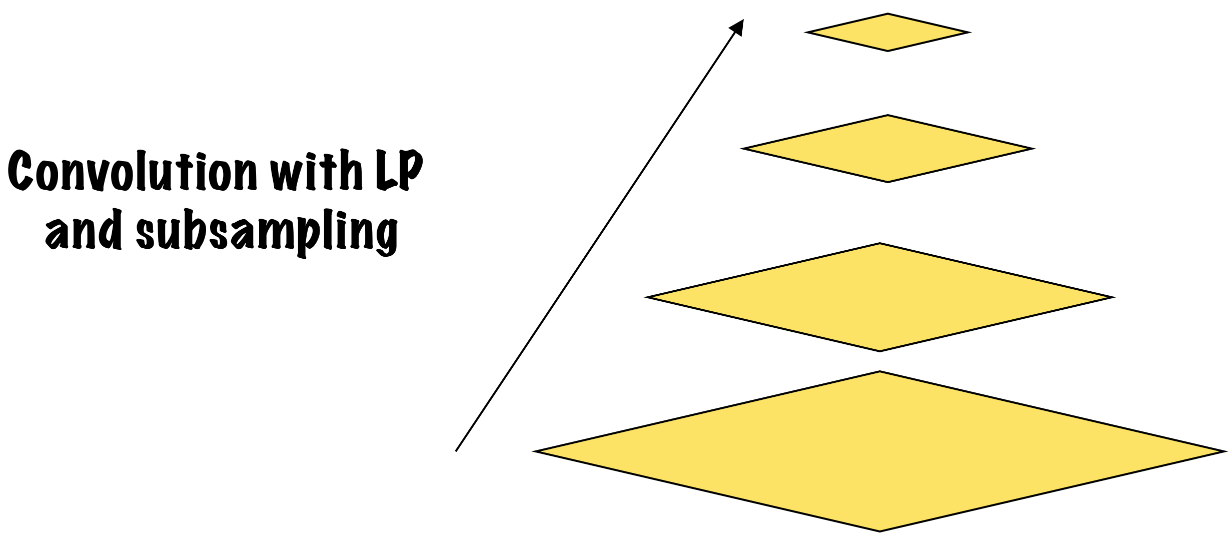
Properties of Daubechies wavelets

I. Daubechies, *Comm. Pure Appl. Math.* 41 (1988) 909.

- Compact support
  - finite number of filter parameters / fast implementations
  - high compressibility
  - fine scale amplitudes are very small in regions where the function is smooth / sensitive recognition of structures
- Identical forward / backward filter parameters
  - fast, exact reconstruction
  - very asymmetric

Wavelets as Hierarchical Decomposition

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

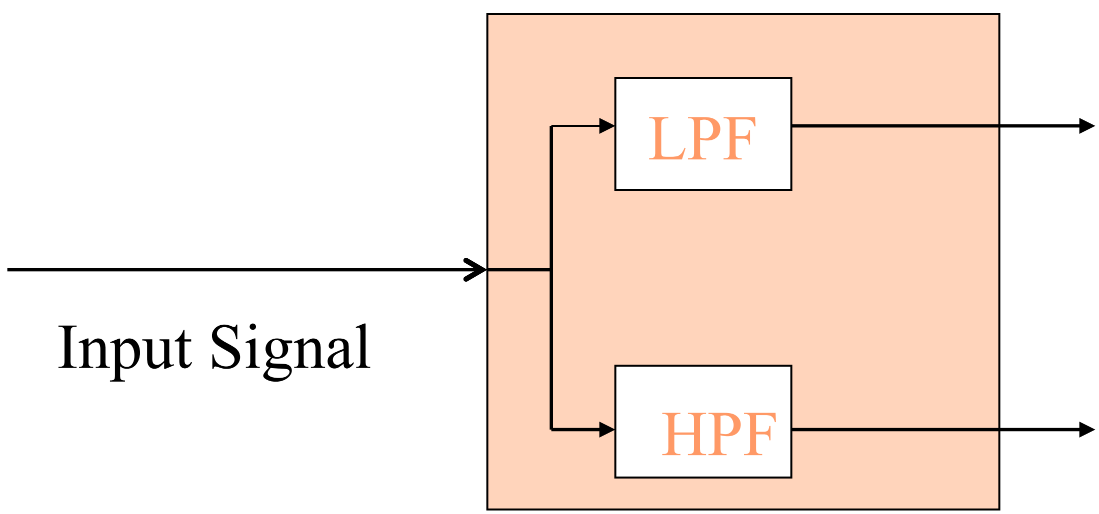


Mallat* Filter Scheme

- Mallat was the first to implement this scheme, using a well known filter design called "two channel sub band coder", yielding a *'Fast Wavelet Transform'*

Approximations and Details:

- Approximations: High-scale, low-frequency components of the signal
- Details: low-scale, high-frequency components

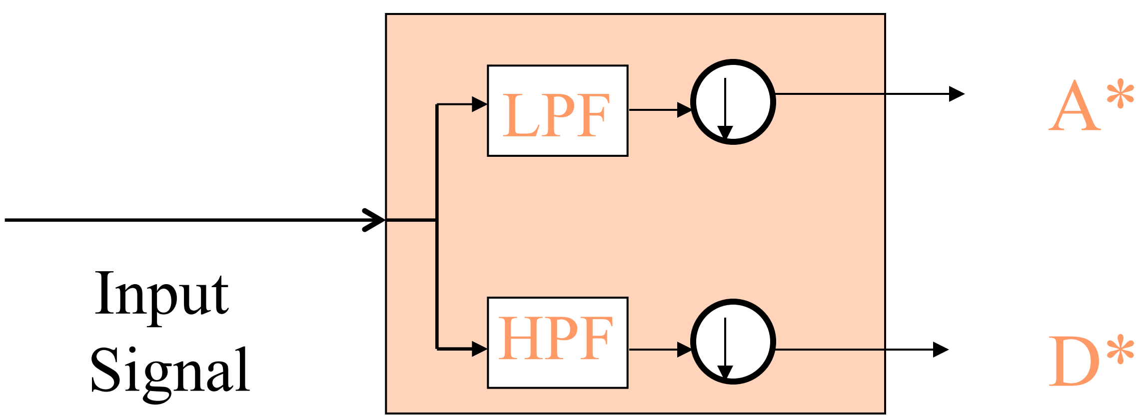


Decimation

- The former process produces <u>twice the data</u> it began with: N input samples produce N approximations coefficients and N detail coefficients.
- To correct this, we *Down sample* (or: *Decimate)* the filter output by two, by simply throwing away every second coefficient.

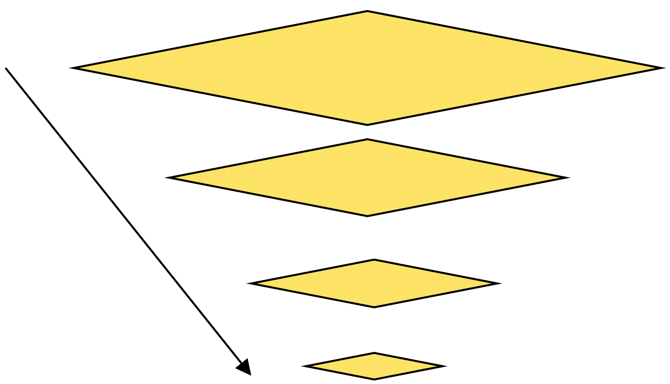
Decimation (cont'd)

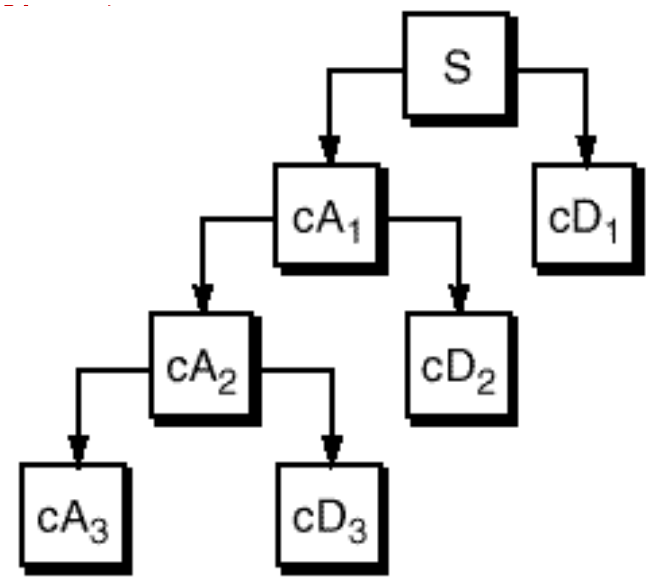
So, a complete one stage block looks like:



Multi-level Decomposition

- Iterating the decomposition process, breaks the input signal into many lower-resolution components: ***Wavelet decomposition***





Orthogonality

- For 2 vectors $$\langle v, w \rangle = \sum_n v_n w_n^* = 0$$

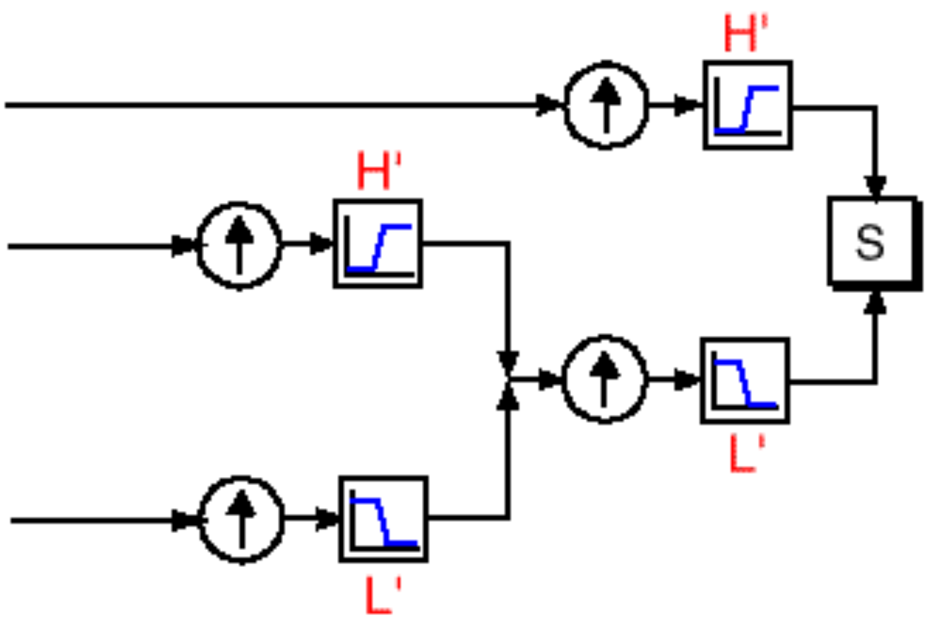
- For 2 functions $$\langle f(t), g(t) \rangle = \int_a^b f(t) g^*(t) \, dt = 0$$

Orthogonal wavelets

- It easier calculation.
- When we decompose some image and calculating zero level decomposition we have accurate values .
- Scalar multiplication with other base function equals zero.

Wavelet reconstruction

- Reconstruction (or synthesis) is the process in which we assemble all components back



Up sampling (or interpolation) is done by zero inserting between every two coefficients

Wavelets like filters

- Relationship of Filters to Wavelet Shape
  - Choosing the correct filter is most important.
  - The choice of the filter determines the shape of the wavelet we use to perform the analysis.

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

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

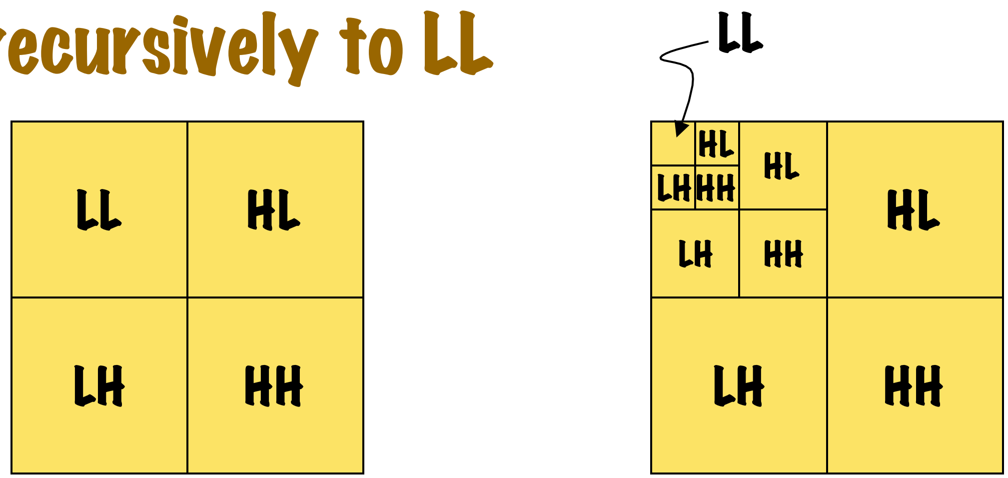
Transformation Matrix

1D signal, discrete, 8 samples ->

$$\begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{matrix}$$

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

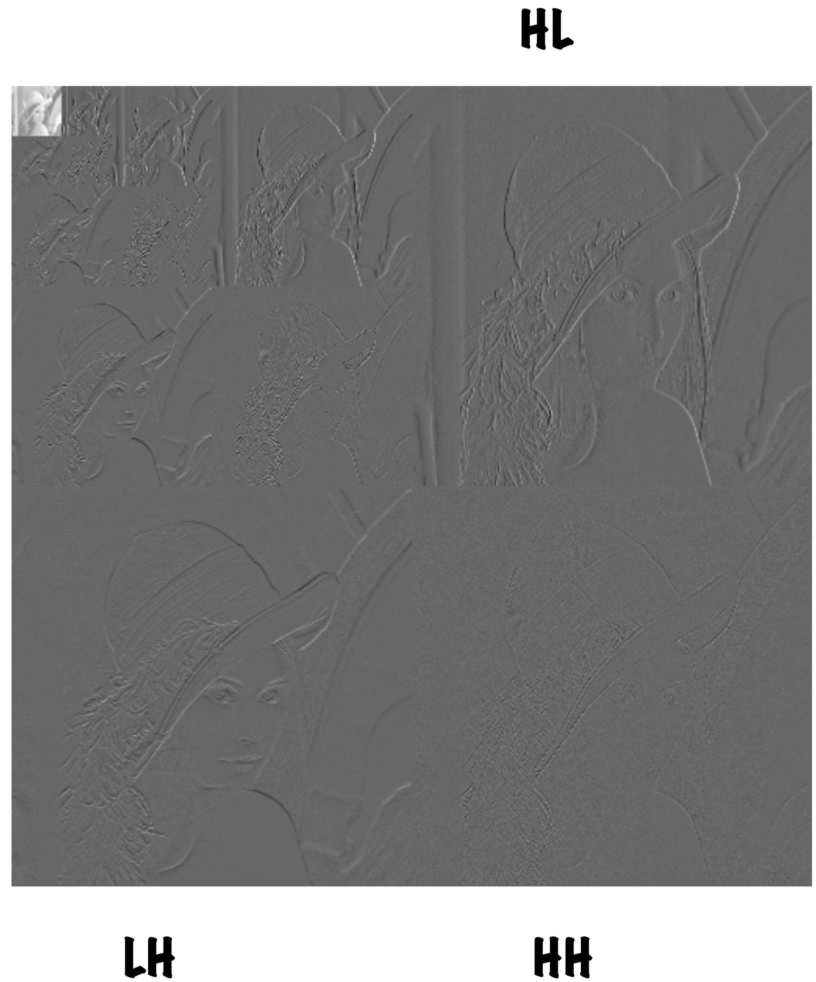


Wavelet Decomposition

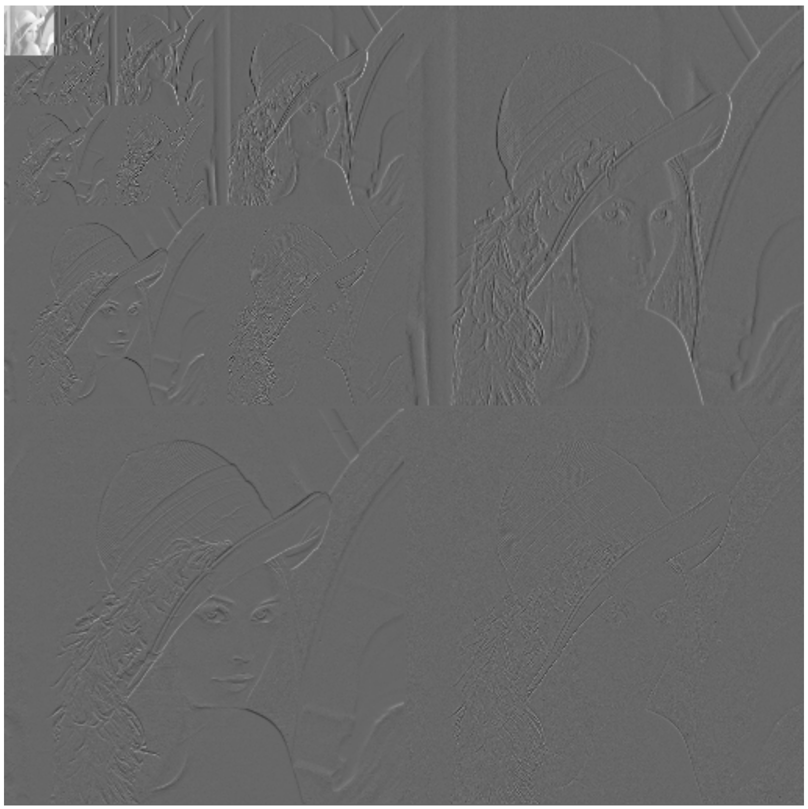
LL HL

LH HH

Wavelet Decomposition



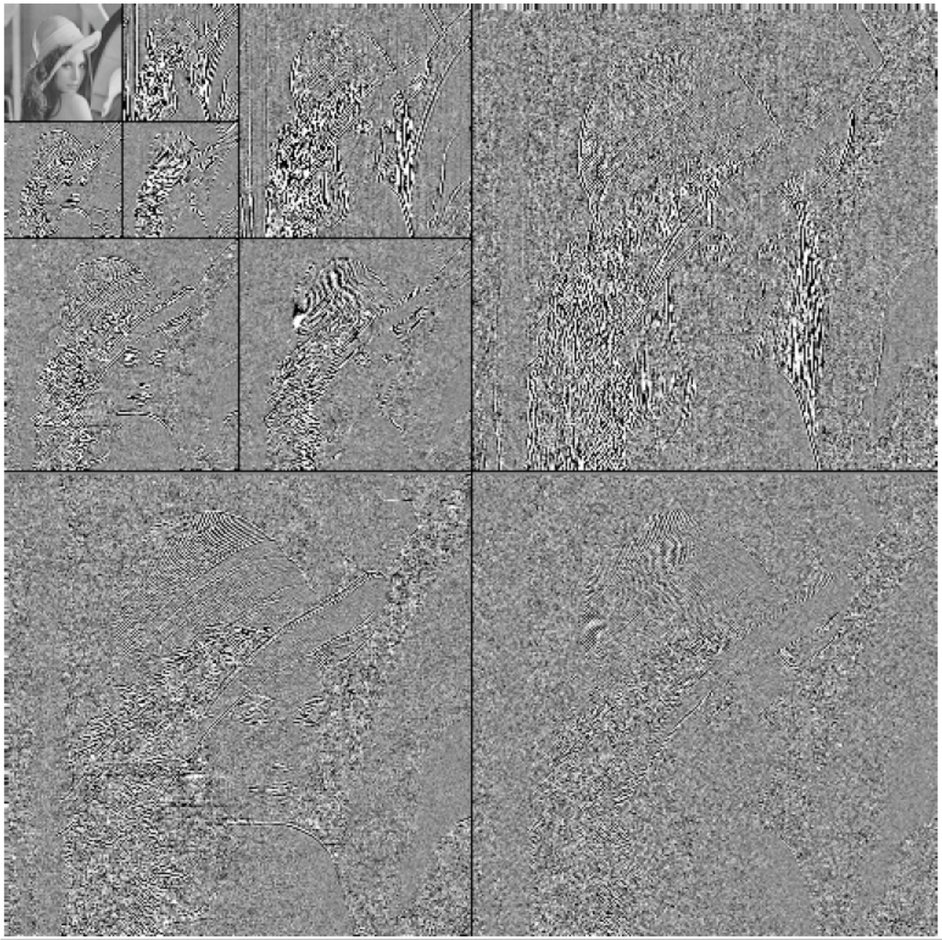
Wavelet Decomposition

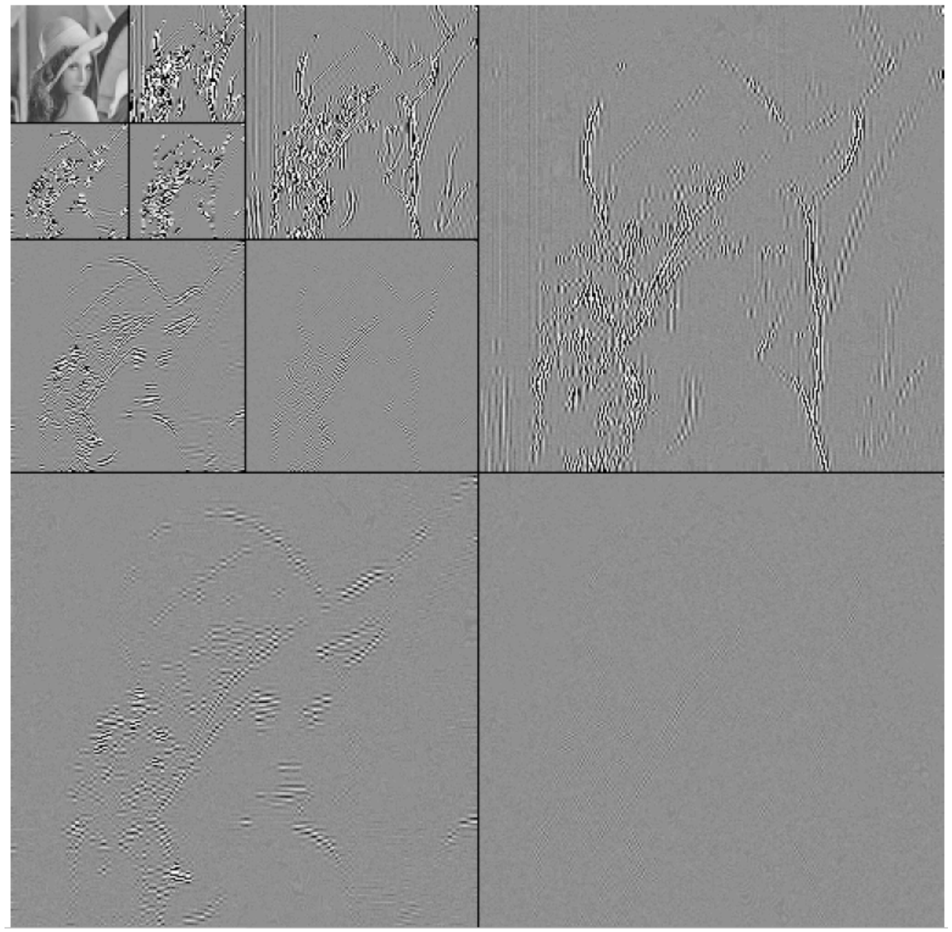


Wavelet Compression Algorithm

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

Wavelet Compression



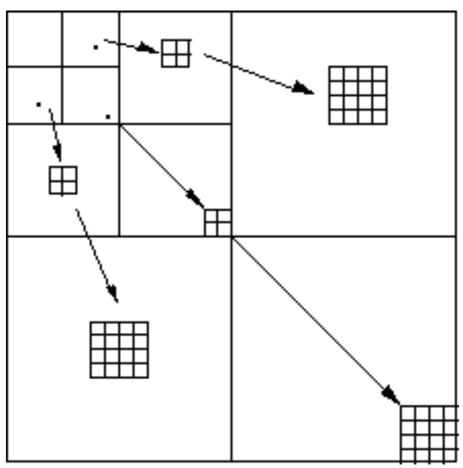


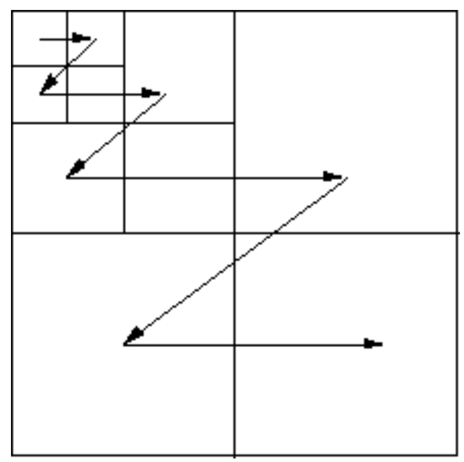
DWT Compression Artifacts

~80:1

Smarter Ways To Encode

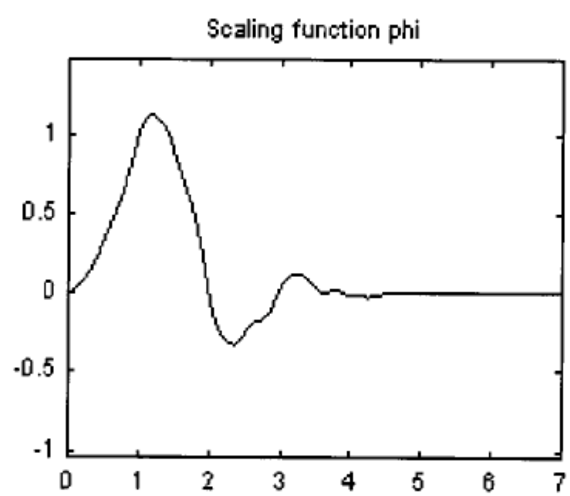
- Embedded zero-tree wavelets (Shapiro 1993)
  - Zeros (threshold) at coarse level likely to be indicative of finer level
  - E.g. edges
  - Continue through levels to hit bit quota

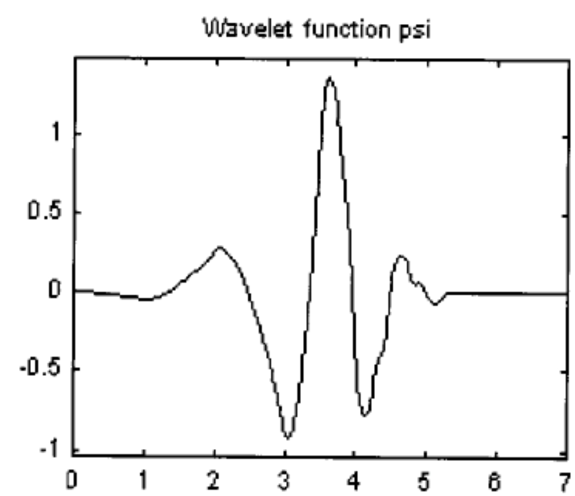




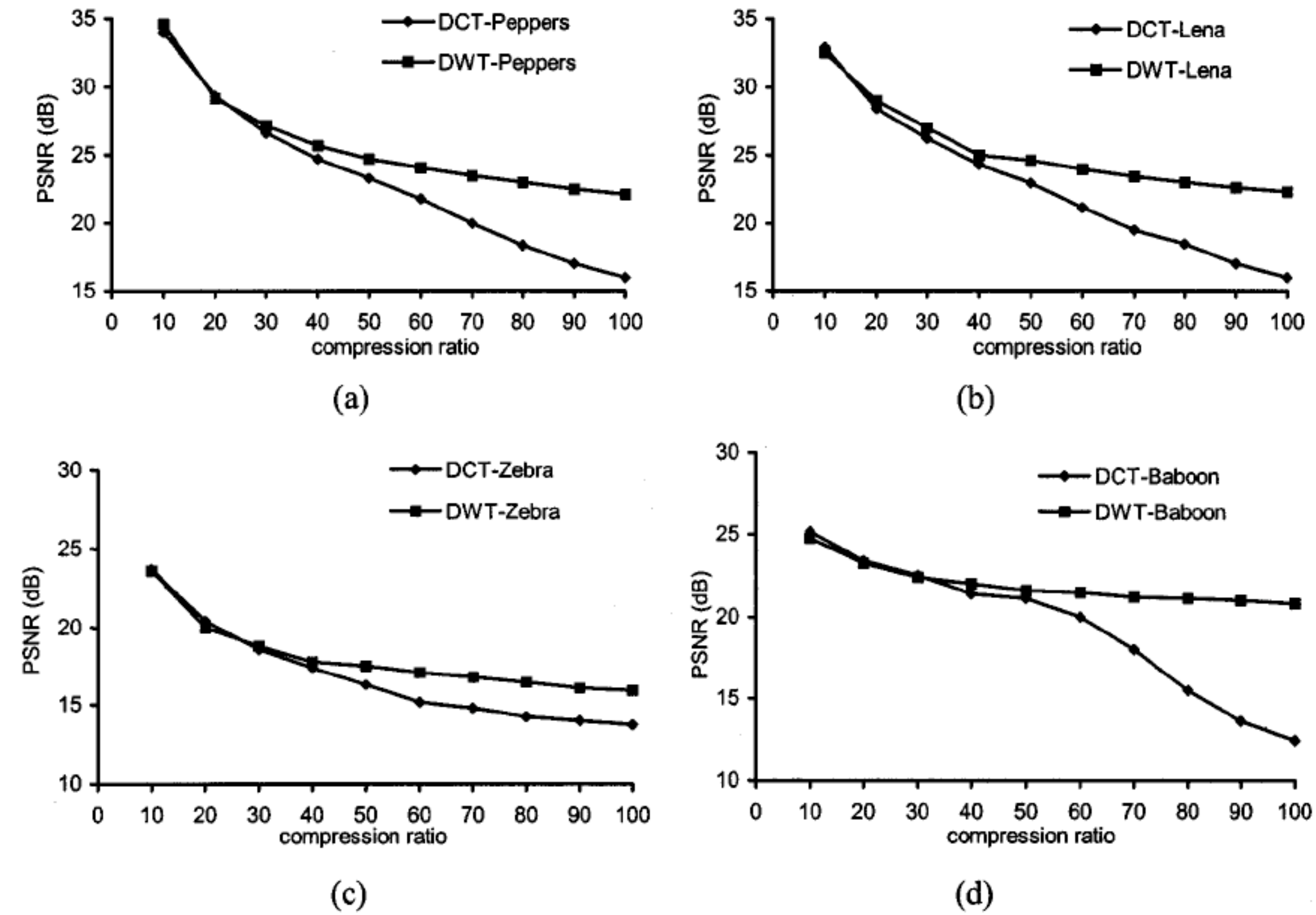
Other Wavelets

- Harr is orthogonal, symmetric, discontinuous
- Daubechies biorthogonal wavelet
  - Continuous, but not symmetric
  - Family of wavelets with parameters
  - JPEG 2000 calls for "Daubechies 9/7 biorthogonal"





Comparisons of Compression



Grgic et al., 2001