THE UNIVERSITY *of* TEXAS
HEALTH SCIENCE CENTER AT HOUSTON
SCHOOL *of* HEALTH INFORMATION SCIENCES

Image Compression

For students of HI 5323
“Image Processing”

Willy Wriggers, Ph.D.
School of Health Information Sciences

http://biomachina.org/courses/processing/09.html

Transforms

Transforms

- A transform is a change in the numeric representation of a signal that preserves all of the signal's information
- Transforms can be thought of as a change of coordinates into some coordinate system (basis set)
  - An alternate form of expressing the vector/signal
- They all have the same basic form:
  1. Choose your basis functions
  2. Get the weights using inner product of signal and basis functions
  3. Reconstruct by adding weighted basis functions

Example: The Fourier Transform

Basis functions: complex harmonics

$$e^{i2\pi st} \quad \text{or} \quad e^{i2\pi sn/N}$$

Transform (calculating the weights of each basis function):

$$F(s) = \int_{-\infty}^{\infty} f(t) e^{-i2\pi st} dt$$

Inverse transform (putting together the weights):

$$f(t) = \int_{-\infty}^{\infty} F(s) e^{i2\pi st} ds$$

Various Transforms

Transform	**Basis Functions**	**Good for…**
Fourier	Sines and Cosines (Complex harmonics)	Frequency analysis, Convolution
Cosine	Cosines	Frequency analysis (but not convolution)
Haar	Square pulses of different widths and offsets	Binary data
Slant	Ramp signals of different slopes and offsets	First-order changes
Wavelets	Various	Time/frequency analysis

Discrete Cosine Transform

Discrete Cosine Transform (DCT)

- The **discrete cosine transform** (DCT) is a discrete Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. It is equivalent to a DFT of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even).

Basis functions: real-valued cosines

$$\alpha(s)\cos\left[\frac{(2n+1)s\pi}{2N}\right]$$

where

$$\alpha(0)=\sqrt{1/N} \quad \text{and} \quad \alpha(s)=\sqrt{2/N} \ \text{ for } 0<s<N$$

Discrete Cosine Transform (DCT)

Transform:

$$G_c[s] = \alpha(s)\sum_{t=0}^{N-1} g[t]\cos\left[\frac{(2t+1)s\pi}{2N}\right]$$

Inverse transform:

$$g[t] = \sum_{s=0}^{N-1} \alpha(s) G_c[s]\cos\left[\frac{(2t+1)s\pi}{2N}\right]$$

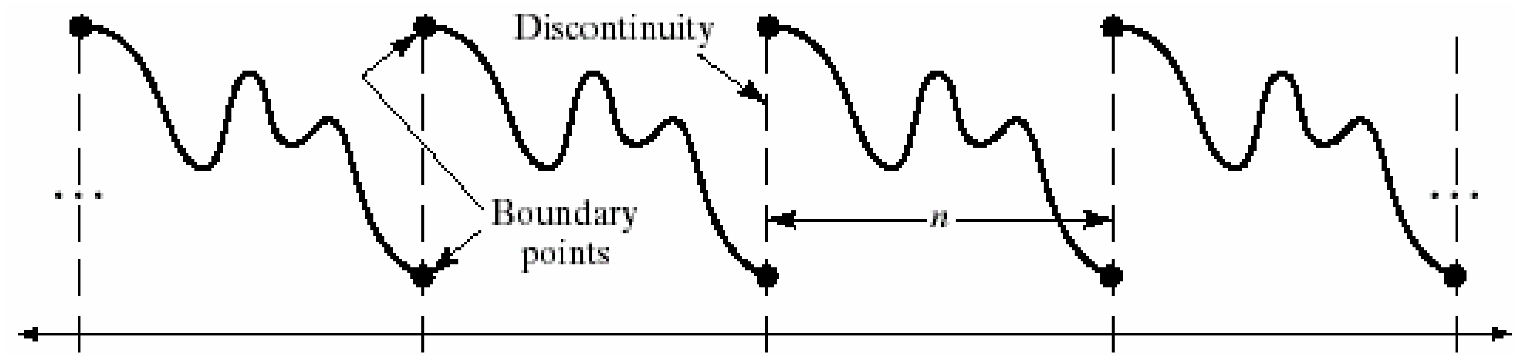
Treat signal as alternating-periodic.

Real-valued transform!

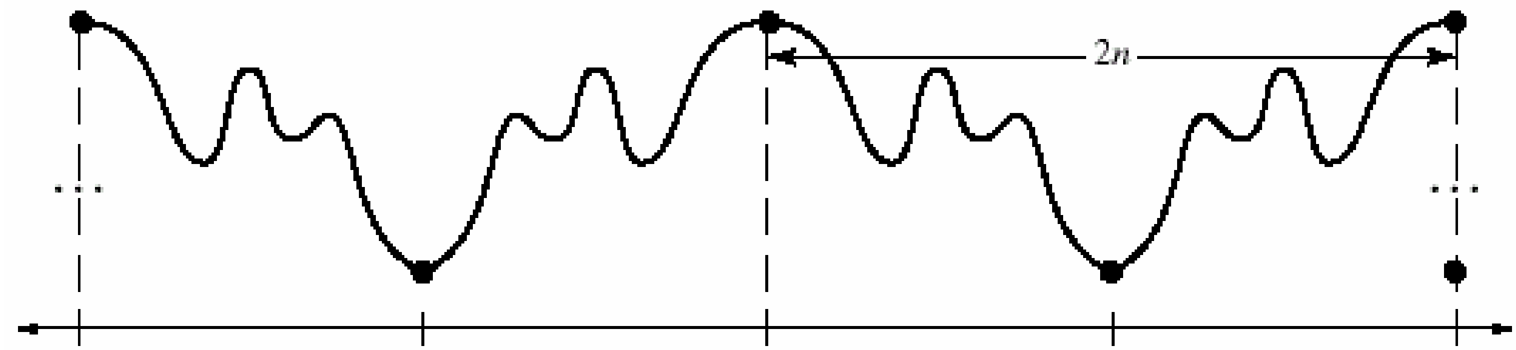
Discrete Cosine Transform (cont.)

Uses alternating periodicity and ~2x larger unit cell

FT



DCT



DCT in 2D

Basis functions: real-valued cosines

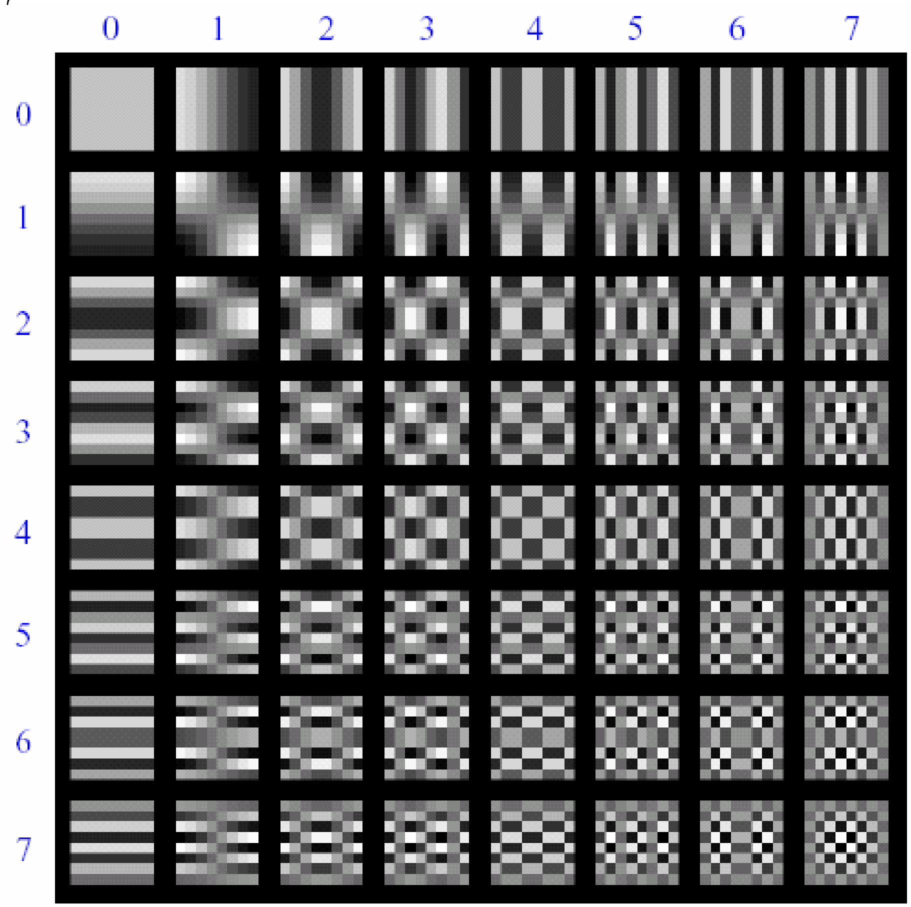
$$g(x,y,u,v) = h(x,y,u,v)$$
$$= \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

where

$$\alpha(u) = \begin{cases} \sqrt{1/N} & \text{if } u = 0 \\ \sqrt{2/N} & \text{otherwise} \end{cases}$$

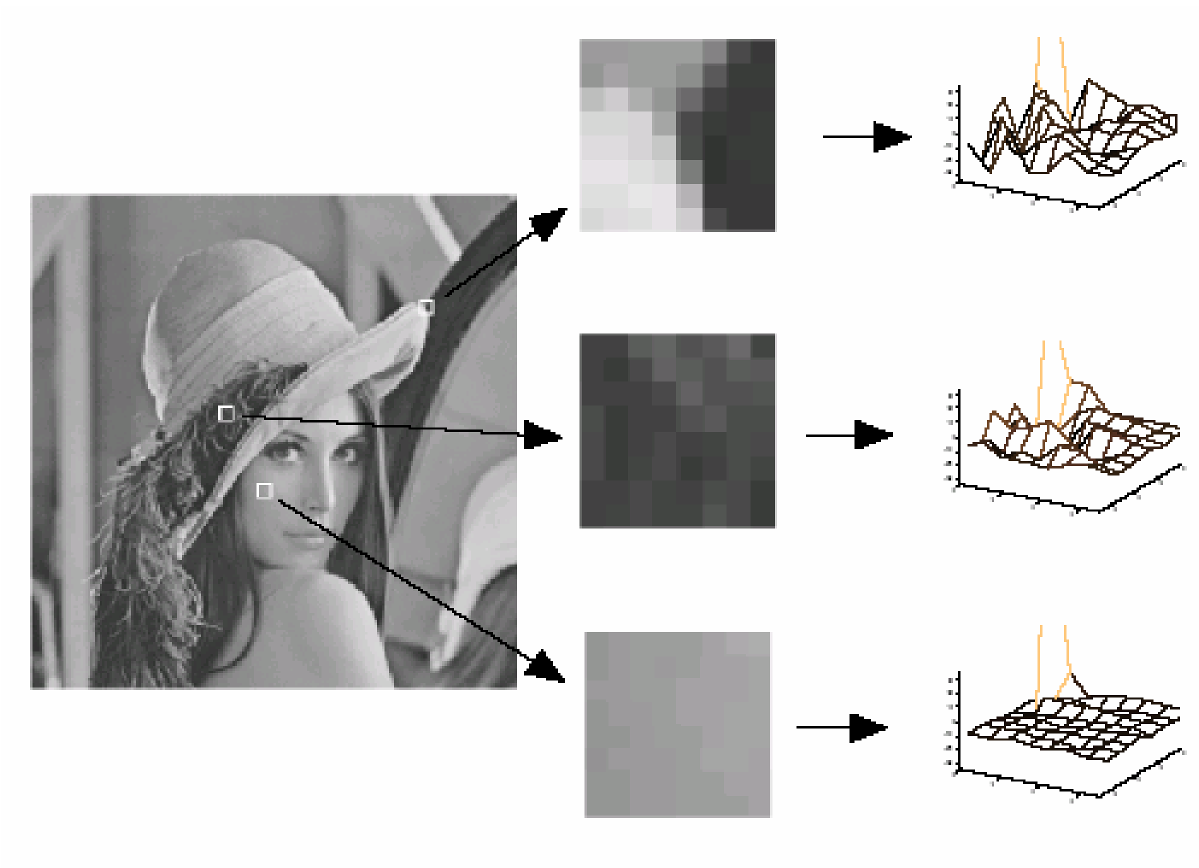
Use of DCT in JPEG Compression

- What linear combination of 8x8 DCT basis functions produces an 8x8 block in the image?



Results (JPEG Example)

- Discrete cosine transform (DCT) on 8x8 blocks

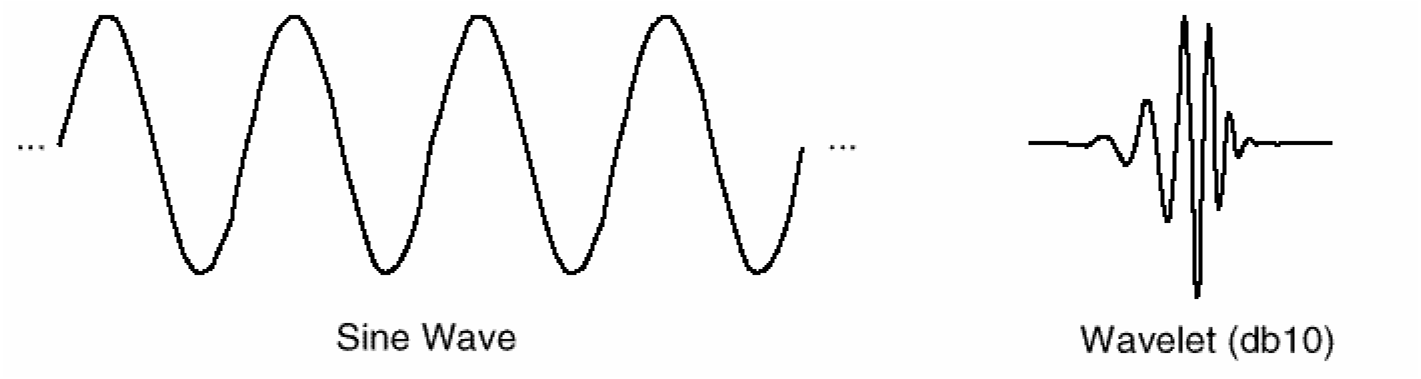


Wavelet Transform

Co-joint Representations

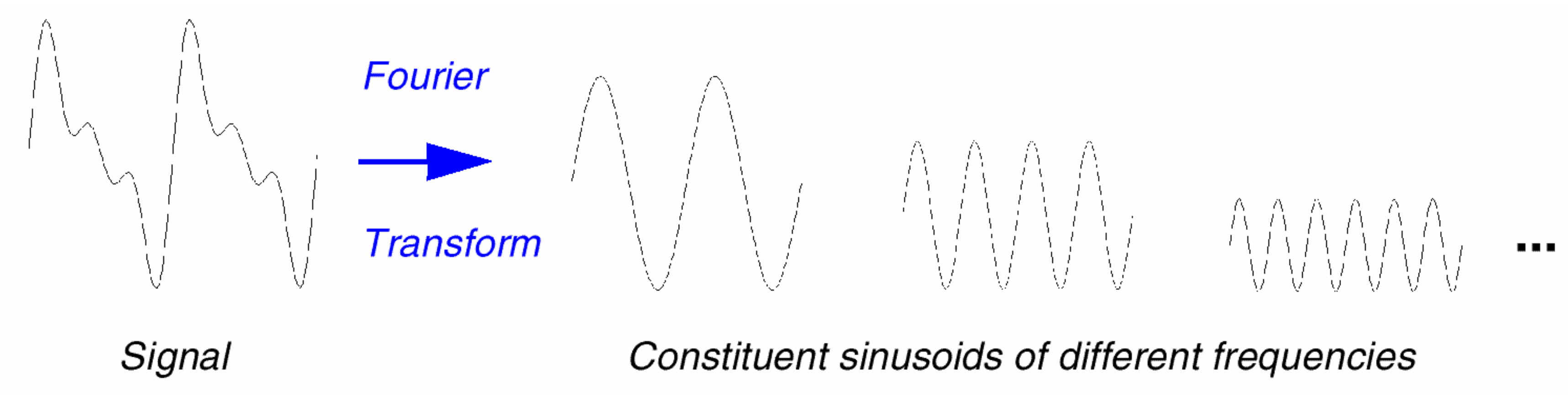
- Signals are pure time/space domain — no frequency component
- Sinusoidal (Fourier, DCT) transforms are pure frequency domain — no spatial component
- Wavelets and other co-joint representation are:
  - Somewhat localized in space
  - Somewhat localized in frequency
- Accuracy in the spatial domain is inversely proportional to accuracy in the frequency domain

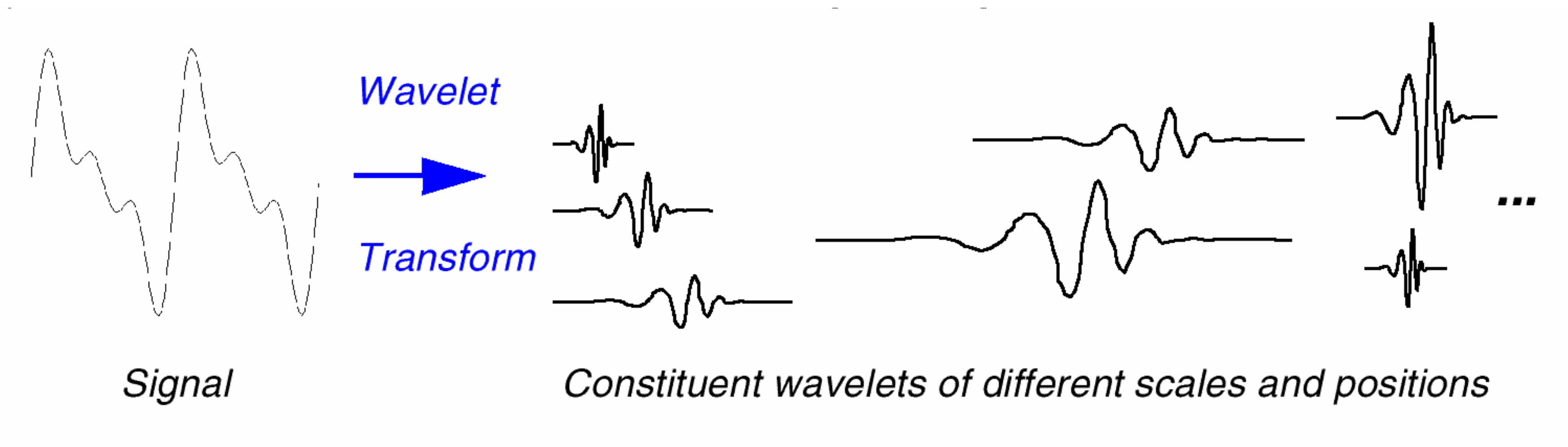
Sinusoidal vs. Wavelet



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

Sinusoidal vs. Wavelet





Wavelet Transform

- The wavelet transform is a tool for carving up functions, operators, or data into components of different frequency, allowing one to study each component separately.

- The basic idea of the wavelet transform is to represent any arbitrary function $f(t)$ as a superposition of a set of such wavelets or basis functions.

- These basis functions or baby wavelets are obtained from a single prototype wavelet called the mother wavelet, by dilations or contractions (scaling) and translations (shifts).

Continuous Wavelet Transform

- The Continuous Wavelet Transform (CWT) is defined as the sum over all time of the signal multiplied by scaled, shifted version of the wavelet function: $\Psi_{s,\tau}(t)$

$$\gamma(\tau, s) = \int f(t)\Psi^{*}_{s,\tau}(t)dt$$

where * denotes complex conjugation. This equation shows how a function *f*(t) is decomposed into a set of basis functions $\Psi_{s,\tau}(t)$, called the *wavelets*.

- The variables *s* and τ are the new dimensions, *scale* and *translation (position)*, after the wavelet transform.

Mother Wavelet

- The wavelets are generated from a single basic wavelet $\Psi(t)$, the so-called *mother wavelet*, by scaling and translation:

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right)$$

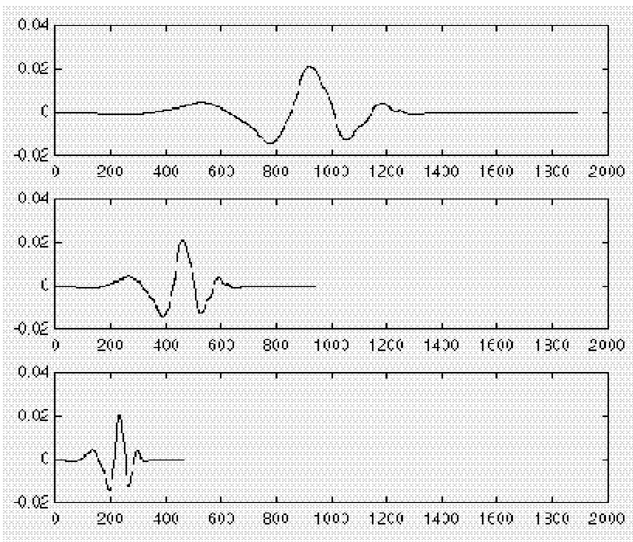
s is the scale factor, τ is the translation factor and the factor $s^{-1/2}$ is for normalization across the different scales.

- It is important to note that in the above transforms the wavelet basis functions can be chosen by the user (if certain mathematical conditions are satisfied, see below).

- This is a difference between the wavelet transform and the Fourier transform, or other transforms.

Scaling

- Scaling a wavelet simply means stretching (or compressing) it:



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

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

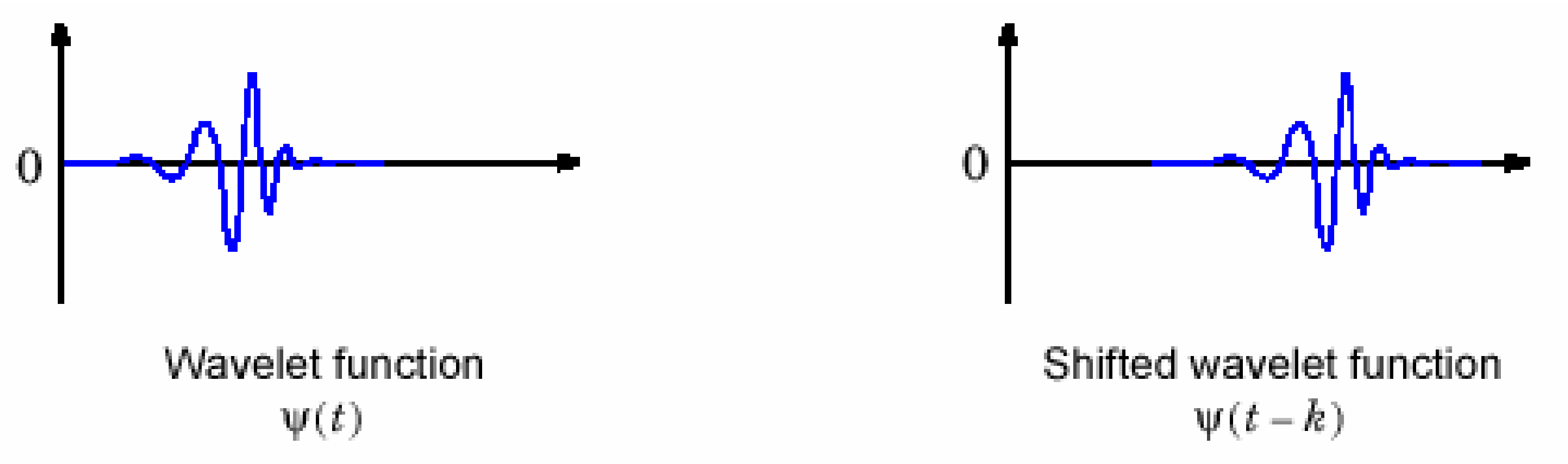
$$f(t) = \Psi(4t) \;\; ; \;\; a = \tfrac{1}{4}$$

- Low scale *a* → compressed wavelet → rapidly changing details → high frequency ω

- High scale *a* → stretched wavelet → slowly changing details → low frequency ω

Shift

- Translating a wavelet simply means delaying (or hastening) its onset.



Wavelet Properties

- The *admissibility constant* must satisfy: $C_\Psi \equiv \int \frac{|\psi(\omega)|^2}{|\omega|} d\omega < +\infty$

  $\psi(\omega)$ stands for the Fourier transform of $\psi(t)$

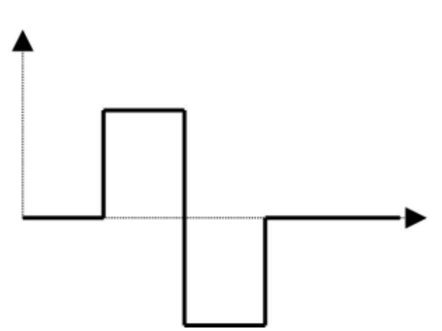
- The admissibility condition implies that the Fourier transform of $\psi(t)$ vanishes at the zero frequency, i.e. $\left. |\psi(\omega)|^2 \right|_{\omega=0} = 0$

- This means that wavelets must have a band-pass like spectrum. This is a very important observation, which we will use later on to build an efficient discrete wavelet transform.
- A zero *DC* (zero frequency) component also means that the average value of the wavelet in the time domain must be zero,

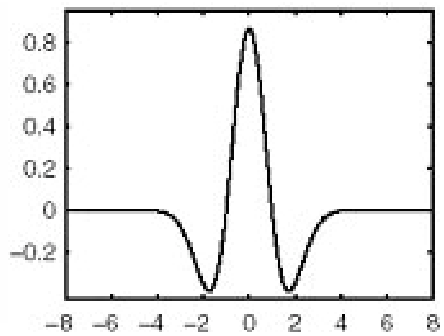
$$\int \psi(t)dt = 0 \longrightarrow$$

$\psi(t)$ must be *AC*, i.e. a *wave*.

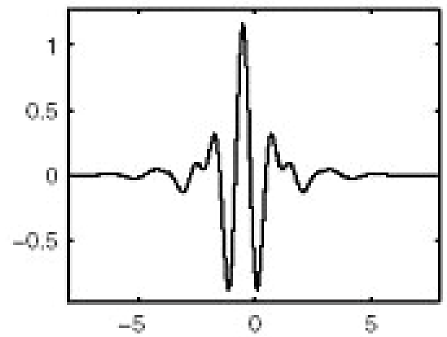
Admissible Mother Wavelet Examples



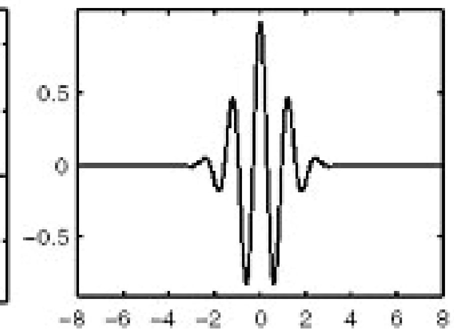
Haar



Mexican Hat



Meyer



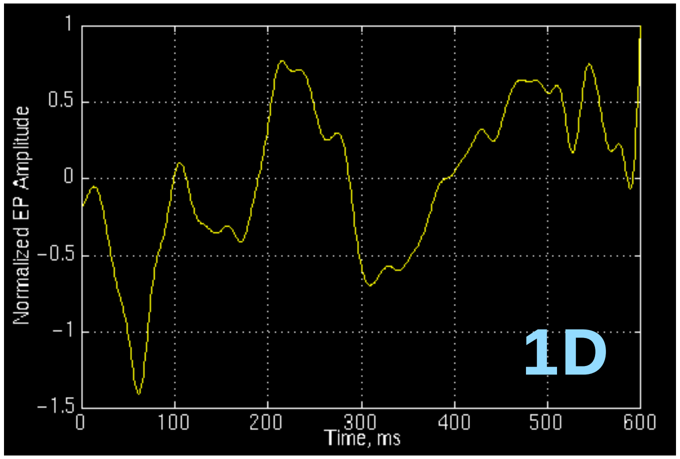
Morlet

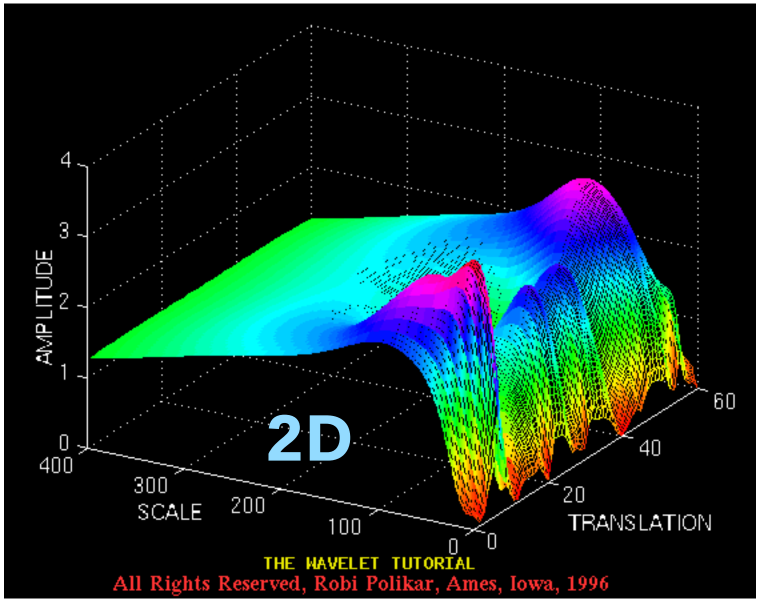
The Inverse CWT

• The original (1D) function can be reconstructed from the 2D CWT with the *inverse CWT*:

$$x(t) = \frac{1}{C_\Psi} \iint \gamma(\tau, s)\psi\left(\frac{t-\tau}{s}\right) d\tau \frac{ds}{|s|^2}$$

• In practice the CWT is not efficient due to the redundancy of the 2D representation (the basis functions are not orthogonal):





Redundancy Removal: Discrete Wavelets

- The discrete wavelet is written as

$$\psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}}\psi\left(\frac{t - k\tau_0 s_0^j}{s_0^j}\right)$$

j and *k* are integers and $s_0 > 1$ is a fixed dilation step. The translation factor τ_0 depends on the dilation step. The effect of discretizing the wavelet is that the time-scale space is now sampled at discrete intervals. We usually choose $s_0 = 2$

- Orthogonality:

$$\int \psi_{j,k}(t)\psi^*_{m,n}(t)dt = \begin{cases} 1 & \text{If } j=m \text{ and } k=n \\ 0 & \text{other} \end{cases}$$

Band-Pass Spectrum

- The wavelet has a band-pass like spectrum

  From Fourier theory we know that compression in time is equivalent to stretching the spectrum and shifting it upwards:

$$F\left\{f(at)\right\} = \frac{1}{|a|} F\left(\frac{\omega}{a}\right)$$

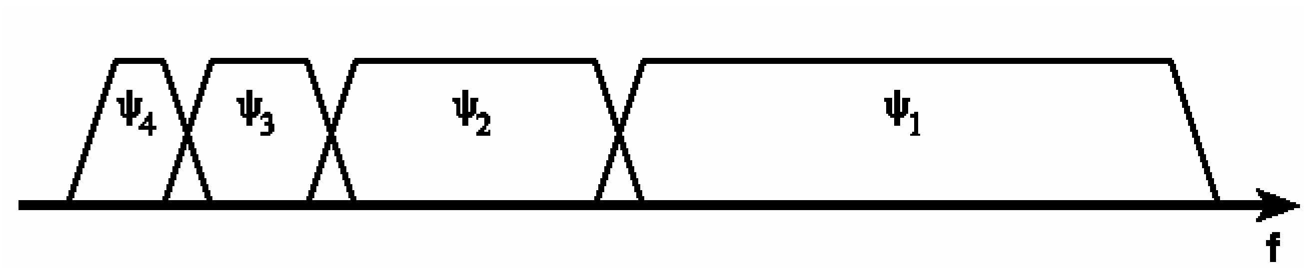
Suppose a=2

This means that a time compression of the wavelet by a factor of 2 will stretch the frequency spectrum of the wavelet by a factor of 2 and also shift all frequency components up by a factor of 2.

Band-Pass Spectrum

• To get a good coverage of the signal spectrum the stretched wavelet spectra should touch each other.

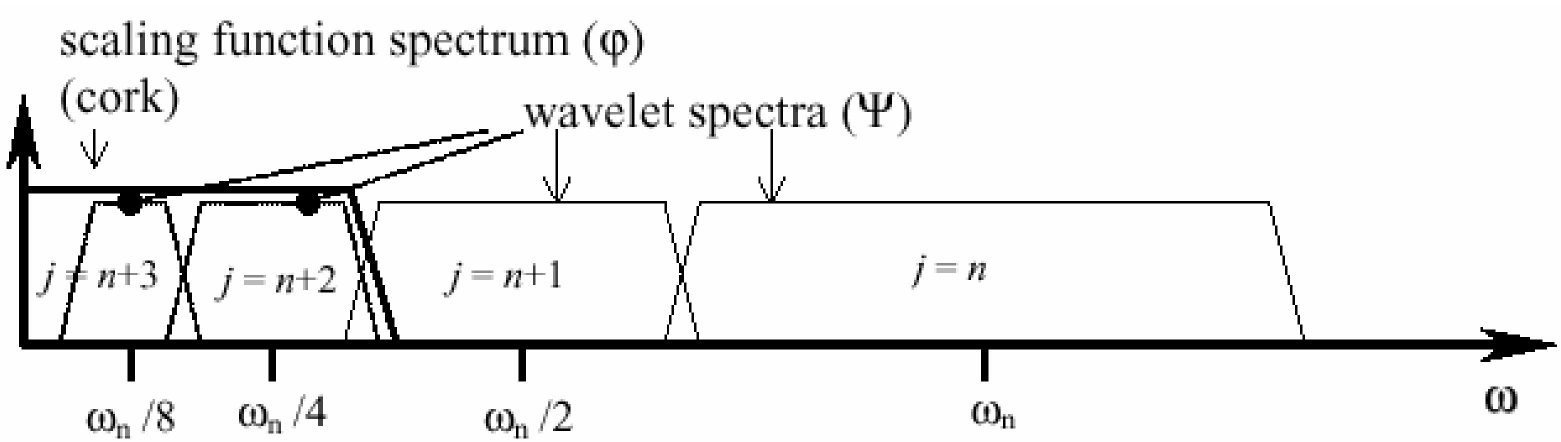
• *Touching wavelet spectra resulting from scaling of the mother wavelet in the time domain.*



• Summarizing, if one wavelet can be seen as a band-pass filter, then a series of dilated wavelets can be seen as a band-pass filter bank.

The Scaling Function

- How to cover the spectrum all the way down to zero?
- The solution is not to try to cover the spectrum all the way down to zero with wavelet spectra, but to use a cork to plug the hole when it is small enough.
- This cork then has a low-pass spectrum and it belongs to the so-called *scaling function*.



Scaling Function Properties

- We can decompose the scaling function in wavelet components:

$$\varphi(t) = \sum_{j,k} \gamma(j,k)\psi_{j,k}(t)$$

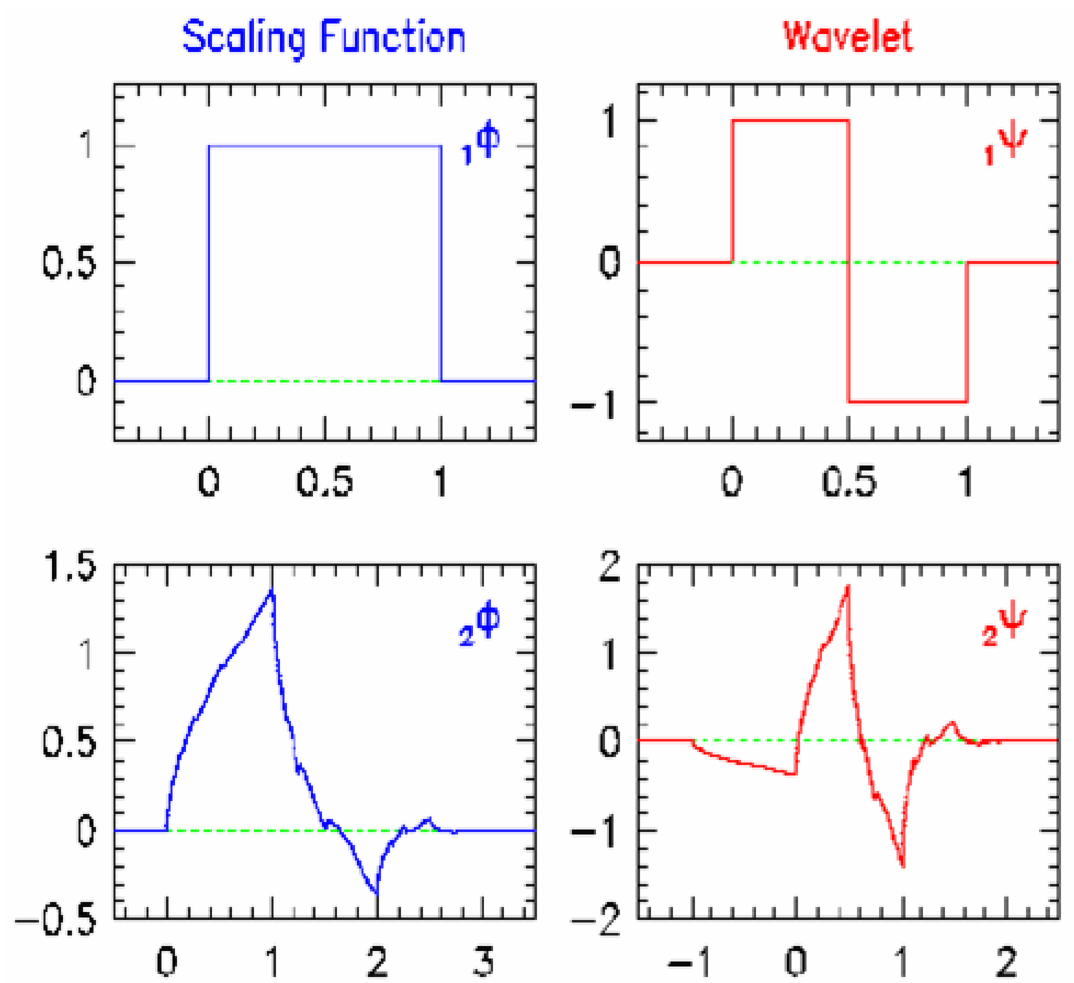
- admissibility condition for scaling functions

$$\int \varphi(t)dt = 1$$

- In practice, scaling functions and wavelets correspond to each other, and choice is not completely free if good reconstruction properties are desired.

Scaling / Wavelet Pairs

- Haar function
- Daubechies function



Daubechies Wavelets

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

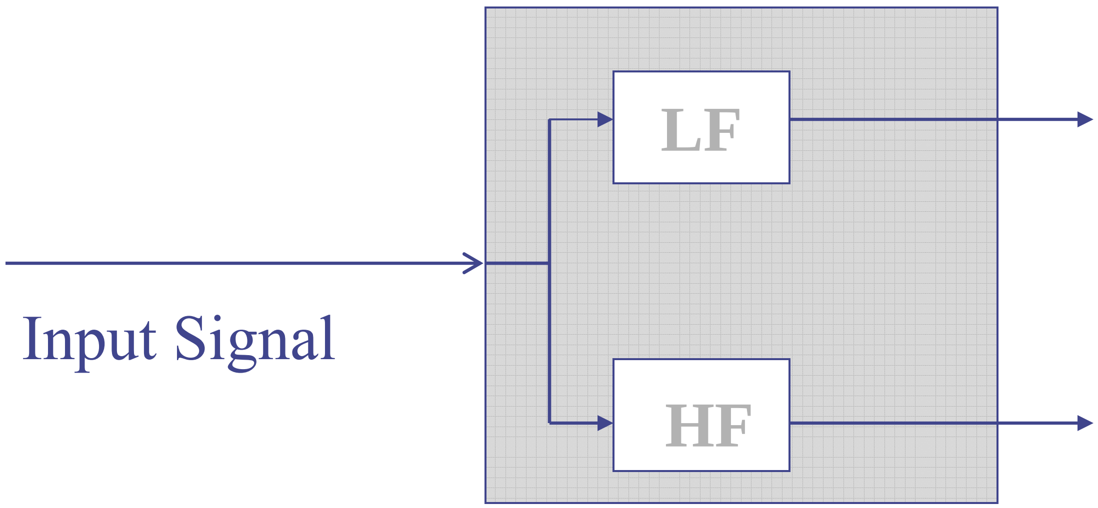
- Compact support
  - finite number of filter parameters / fast implementations
  - high compressibility
  - fine scale amplitudes are very small in regions where the function is smooth / sensitive recognition of structures
- Identical forward / backward filter parameters
  - fast, exact reconstruction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

DWT as a Filter Bank

- Mallat was the first to implement discrete wavelets in a well known filter design called "two channel sub band coder", yielding a *Fast Wavelet Transform* or DWT
- The outputs of the different filter stages are the wavelet- and scaling function transform coefficients.
- The choice of scales and positions based on powers of two -- so-called *dyadic* scales and positions -- yields a very efficient and accurate analysis.

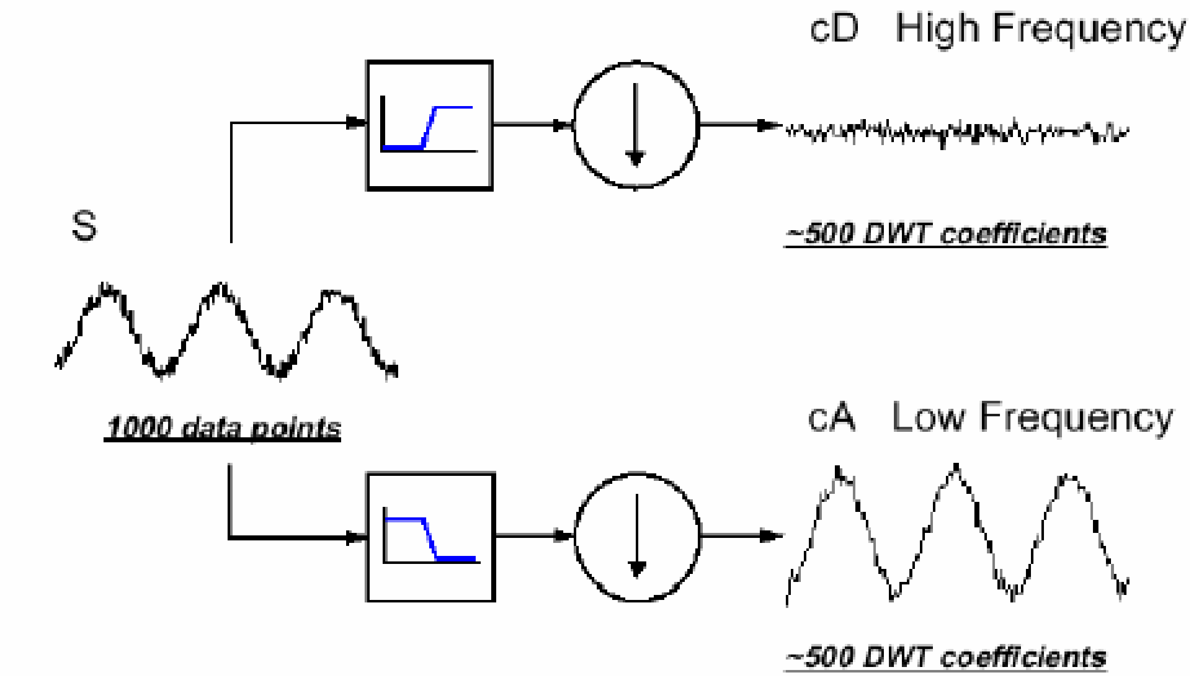
Approximation and Details

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



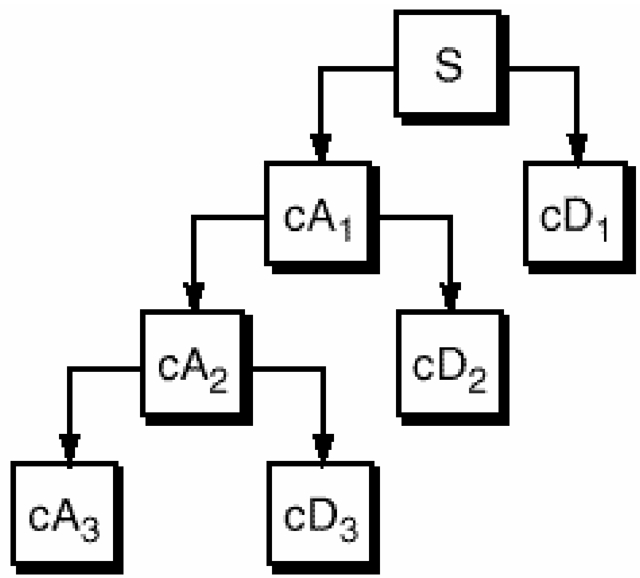
Downsampling

- At each filter the sampling rate remains constant, this is achieved through 2x downsampling of both HF and LF signals:

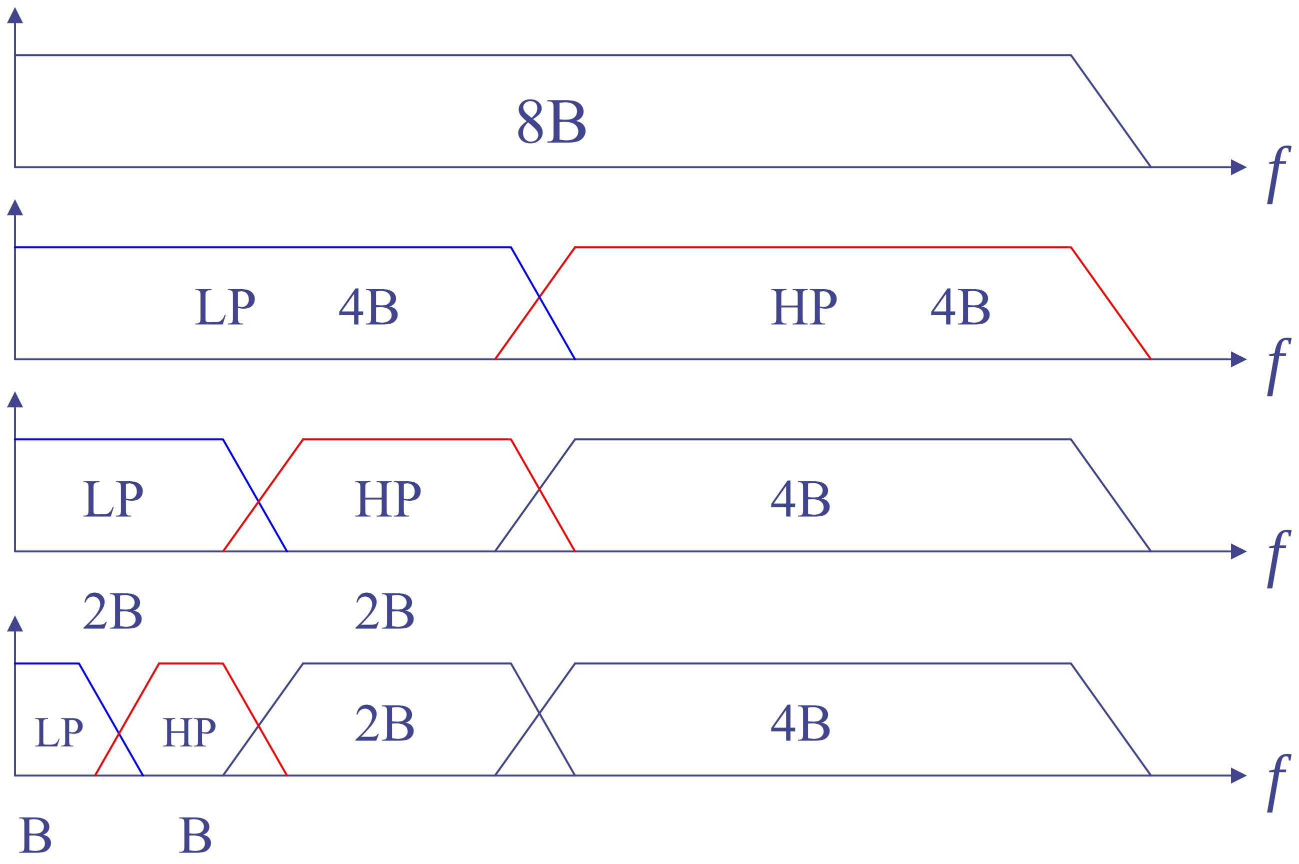


Multi-Level Decomposition

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

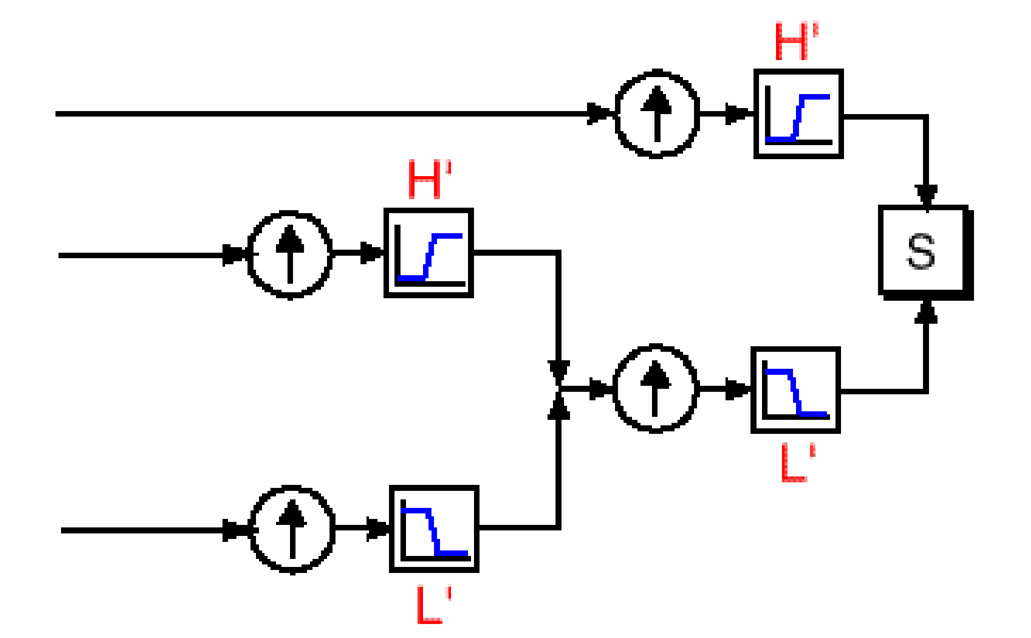


Multi-Level Decomposition



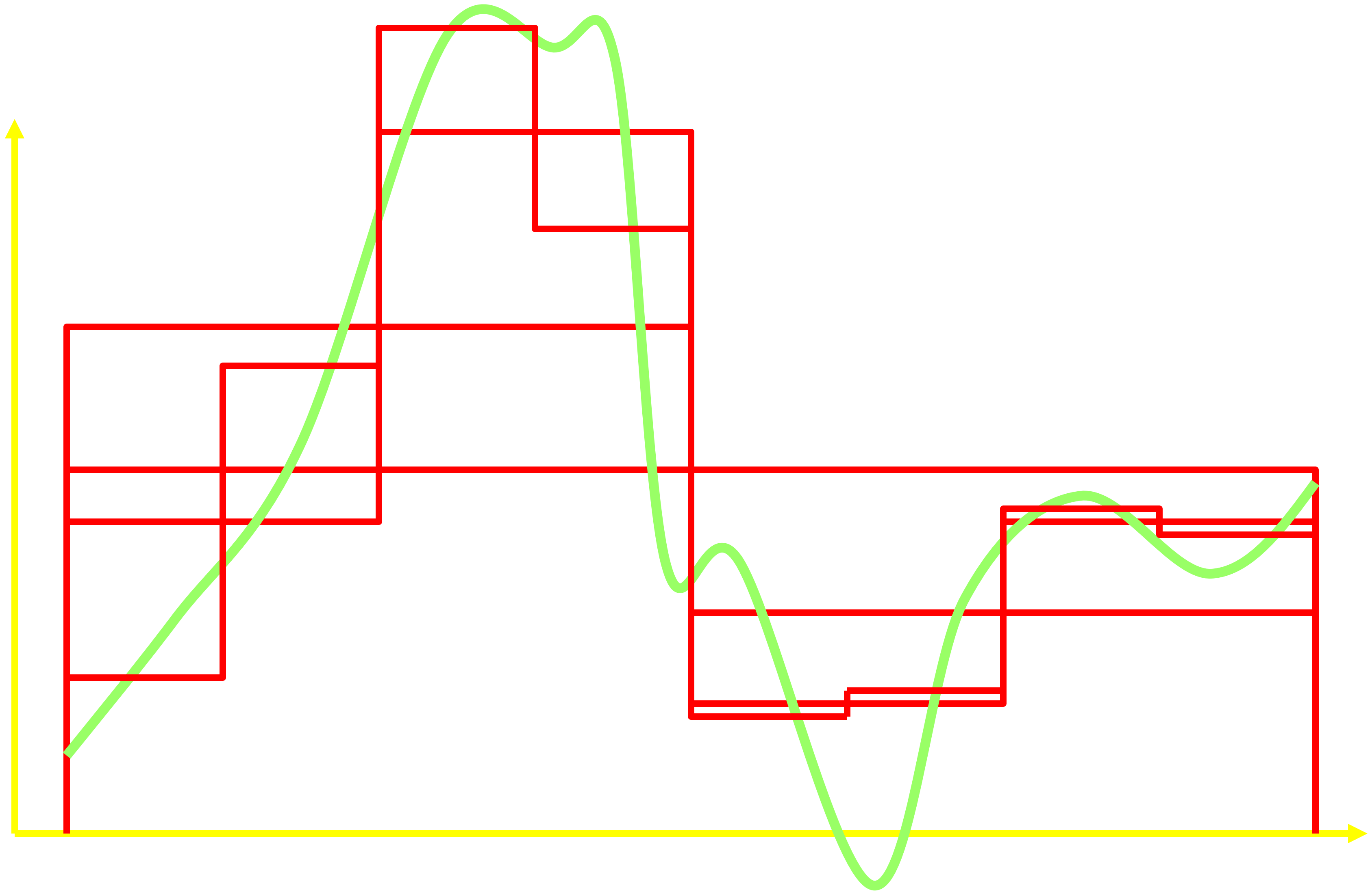
Reconstruction

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

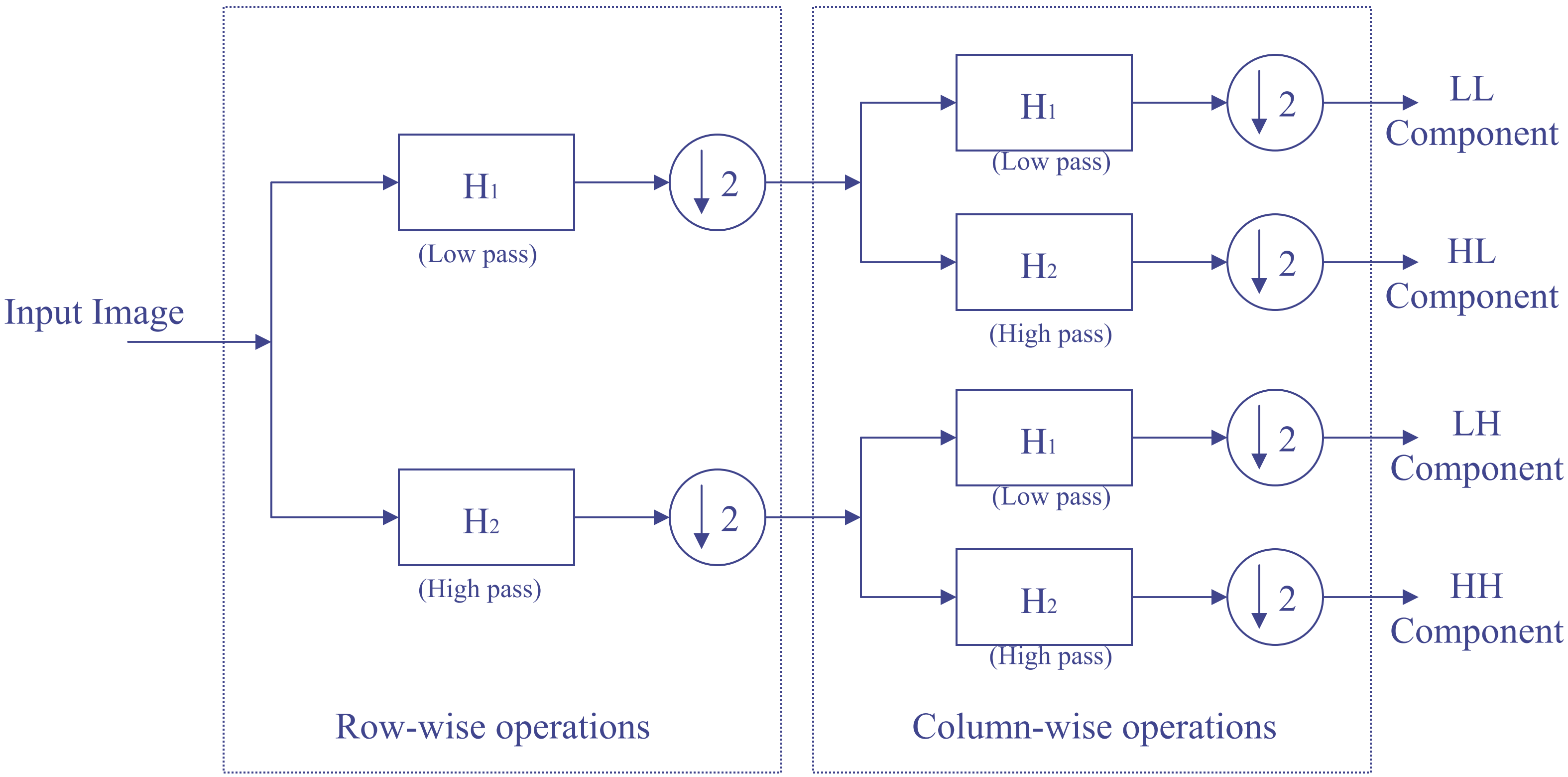


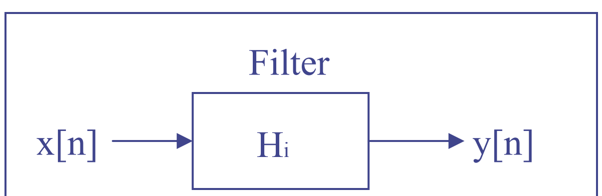
Upsampling
(or interpolation) is done by zero inserting between every two coefficients

Haar DWT in 1D

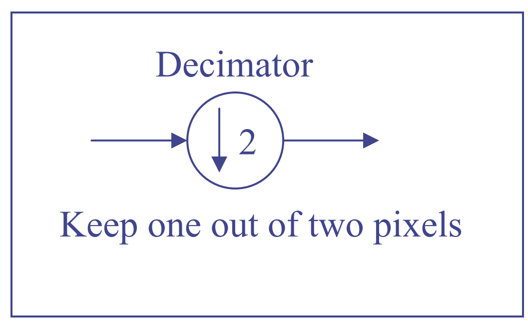


One-Level DWT for 2D Images

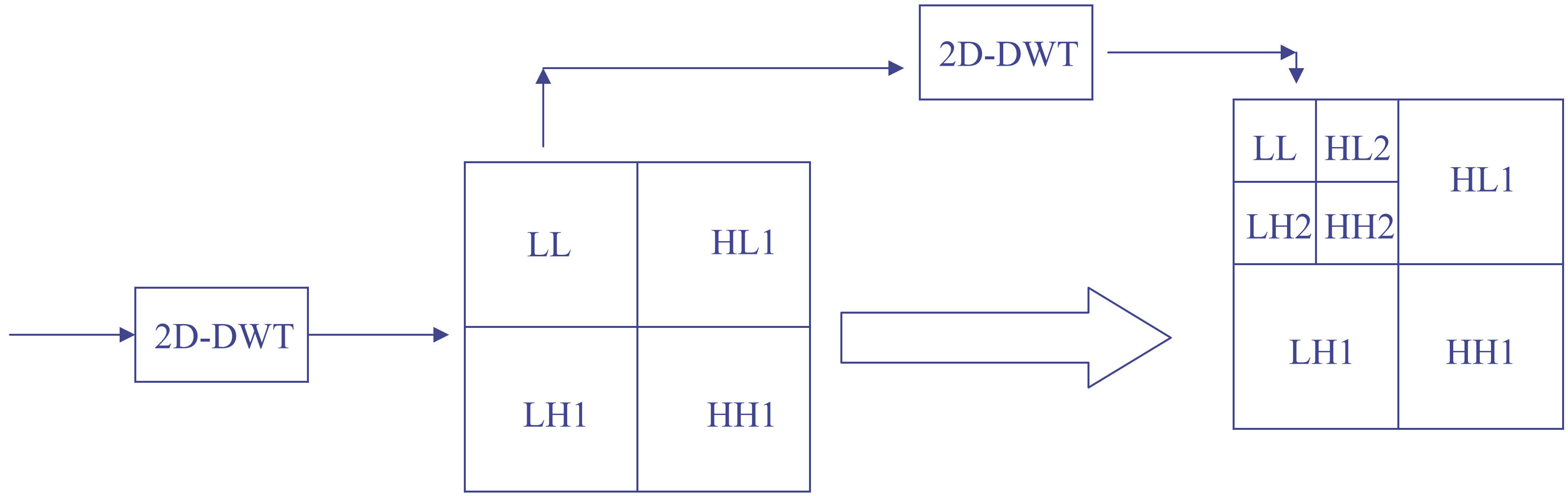


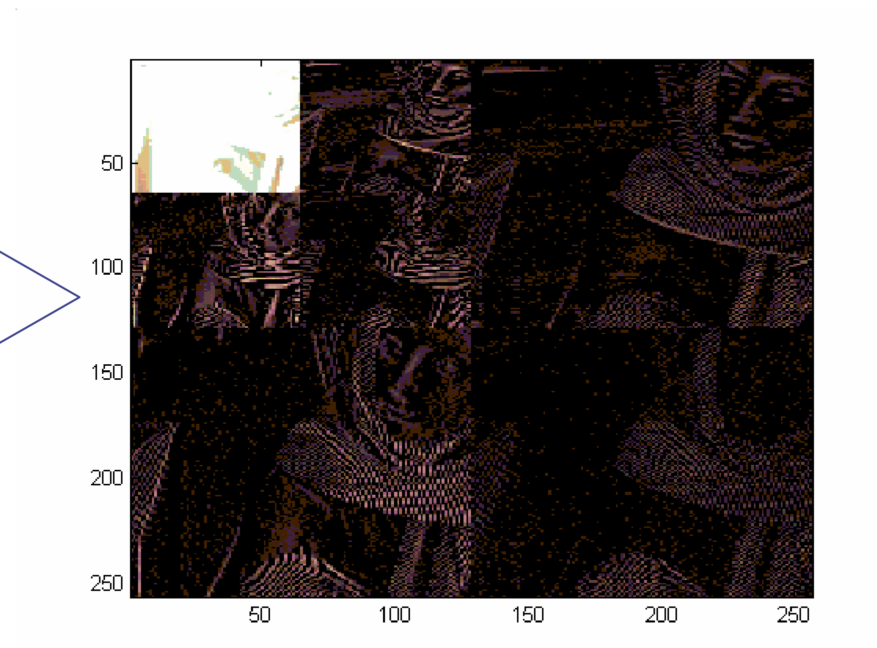


$$y[n] = \sum_{k=0}^{L} x[n-k]h_i[k]$$



Multi-Level DWT for 2D Images





Haar DWT in 2D

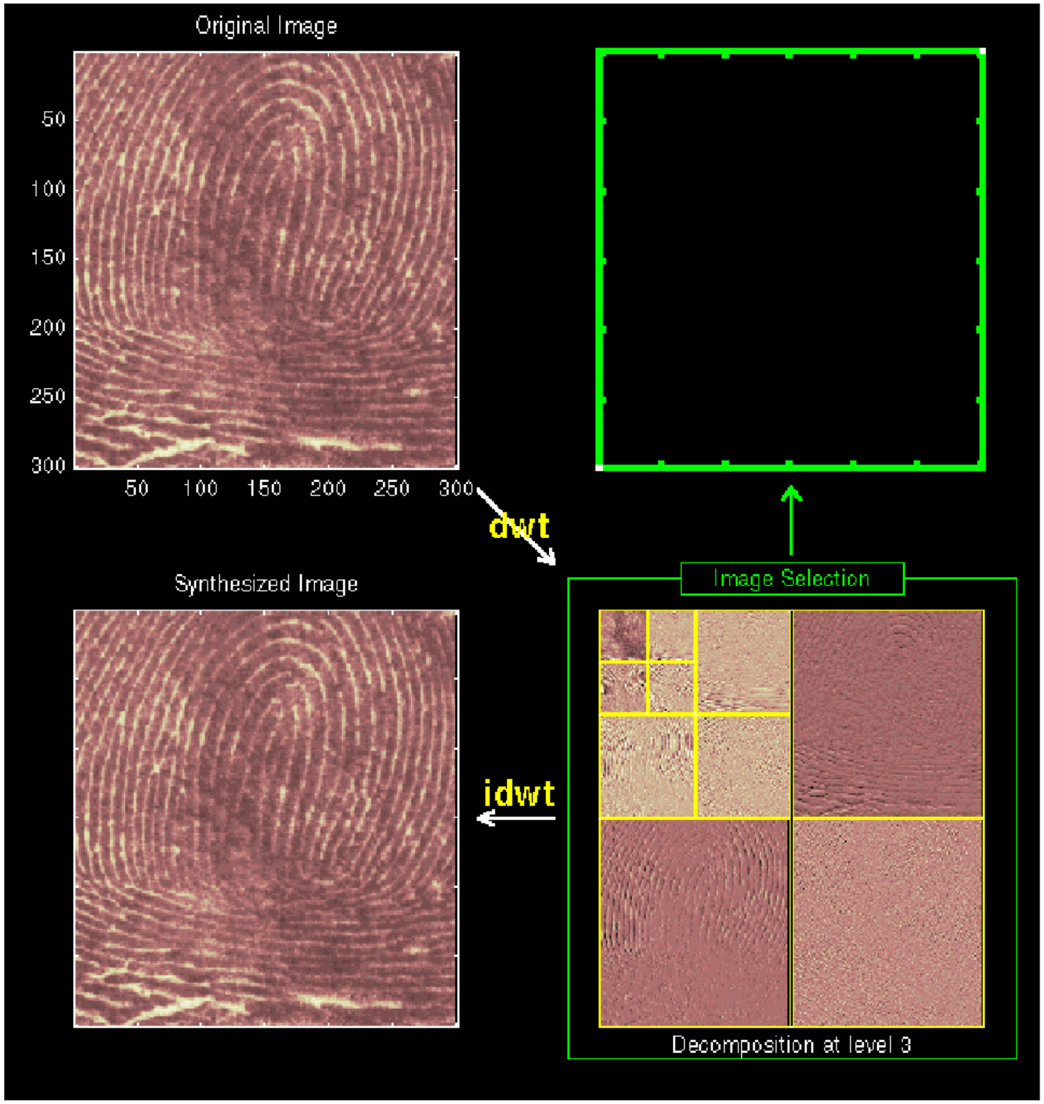
The Haar transform can be used in lossy image compression

Original image

Reconstructed image

The second image was created with the following procedure. A 2D Haar wavelet transformation of the first image was created, 75% of the resulting wavelet data was zeroed out (the 75% of the difference coefficients with the smallest absolute values), and then the second image was reconstructed from the modified wavelet data. The resulting image contains only 25% of the information found in the original, but is still quite recognizable. Zeroed wavelet data compresses well, and this procedure therefore provides a decent lossy compression technique for images.

Fingerprint Compression

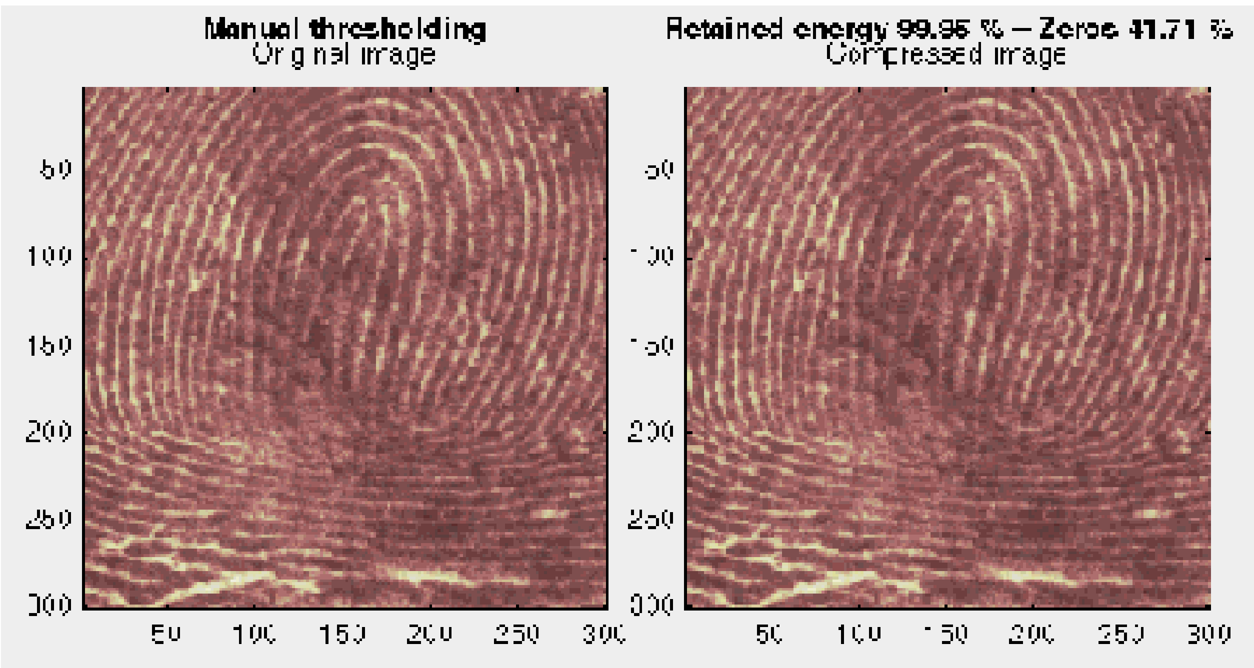


Wavelet:
Haar
Level:3

Motivation:
FBI uses a wavelet technique to compress its fingerprints database.

Fingerprint Compression

Original Image   Compressed Image



Threshold: 3.5
Zeros: 42%
Retained energy: 99.95%

Energy Compaction and Compression Rates

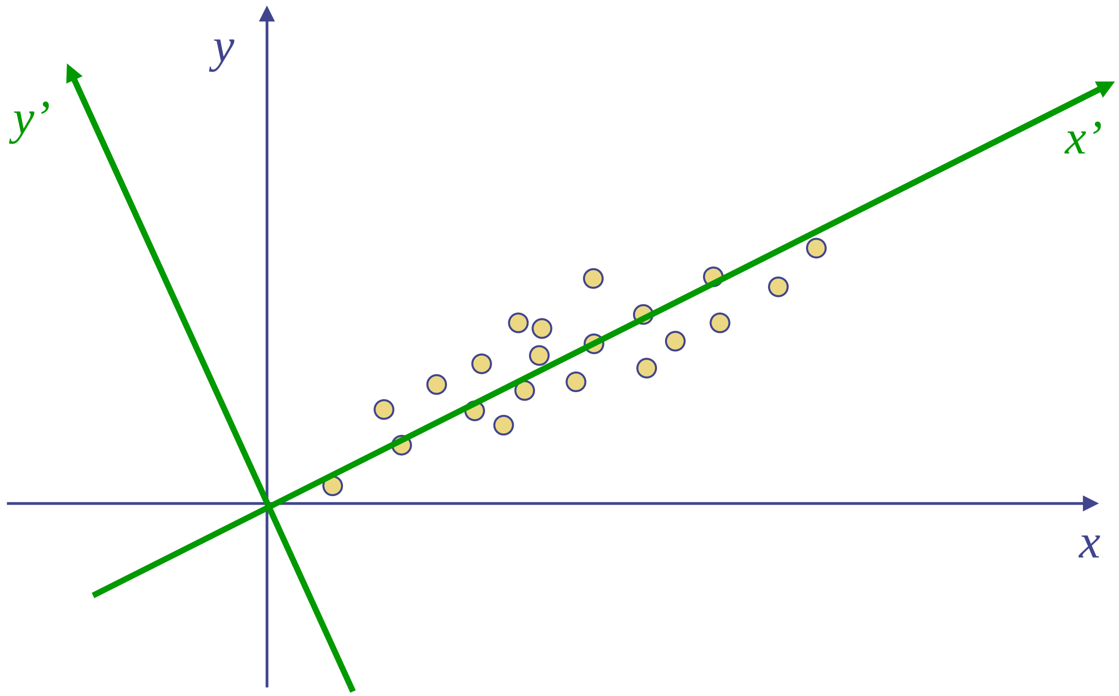
Energy Compaction

Definition: most of the signal information is concentrated in a few low-frequency (or otherwise isolated) components of the transform, approaching the Karhunen-Loève transform, which is optimal in the decorrelation sense.

Karhunen-Loève Transform

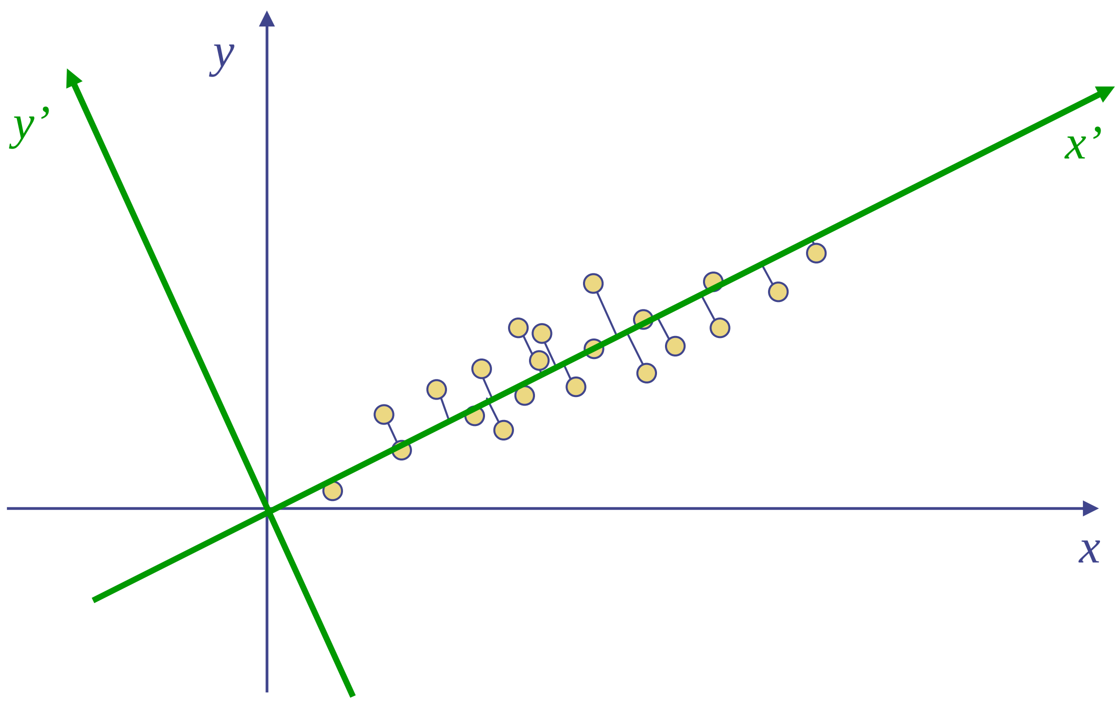
- "Gold standard" for energy compaction
- Basis functions: eigenvectors of covariance matrix
- Idea:
  - Measure statistical properties of the relationship between pixels.
  - Find the "optimal" relationships (eigenvectors).
  - Use these as basis functions.
- Signal/image specific!
- a.k.a. Principal Component Analysis (PCA)

Karhunen-Loève Transform



Given a set of points, find the best line that approximates it – reduce the dimension of the data set…

Karhunen-Loève Transform



… and minimize the sum of distances in the orthogonal direction

Energy Compaction

In practice, most transforms produce a more compact representation than the original image when series is truncated at some point.

Compression in signal/image processing means large part of information content in small part of representation

Representation	Compaction	And/But …
Image	Poor	Easily interpreted
Fourier	Good	Convolution Theorem
Cosine	Better	Fast
PCA	Optimal	Basis functions are signal-specific
Wavelets	Good	Some spatial representation as well

Energy Compaction

- The DCT is often used in signal and image processing, especially for lossy data compression, because it has a strong "energy compaction" property, approaching the Karhunen-Loève transform (which is optimal in the decorrelation sense).
- For example, the DCT is used in JPEG image compression, MJPEG video compression, and MPEG video compression.
- A related transform, the *modified* discrete cosine transform, or MDCT, is used in AAC, Vorbis, and MP3 audio compression.
- For more info and links see http://en.wikipedia.org/wiki/DCT

Signal Compression Rates

Application	Uncompressed	Compressed
Voice: 8 k samples/sec 8 bits/sample	64 kbps	2 - 4 kbps
Audio Conference: 8 k samples/sec 8 bits/sample	64 kbps	16 - 64 kbps
Digital Audio (Stereo): 44.1 k samples/sec 16 bits/sample	1.5 Mbps	128 kbps - 1.5 Mbps
Slow Motion Video: 10 fps 176 × 120 frames 8 bits/pixel	5.07 Mbps	8 - 16 kbps

Signal Compression Rates

Application	Uncompressed	Compressed
Video Conference: 15 fps 352 × 240 frames 8 bits/pixel	30.41 Mbps	64 - 768 kbps
Video File Transfer: 15 fps 352 × 240 frames 8 bits/pixel	30.41 Mbps	384 kbps
Digital Video on CD-ROM: 30 fps 352 × 240 frames 8 bits/pixel	60.83 Mbps	1.5 - 4 Mbps

Signal Compression Rates

Application	Uncompressed	Compressed
DVD / Broadcast Video: 30 fps 720 × 480 frames 8 bits/pixel	248.83 Mbps	3 - 8 Mbps
HDTV: 59.94 fps 1280 × 720 frames 8 bits/pixel	1.33 Gbps	20 Mbps

Information and Coding

Coding

- All information in digital form must be encoded
- Examples:
  - Binary numbers
  - ASCII
  - IEEE floating-point standard
- Coding can be:
  - Simple or complex
  - Loss-less or lossy
  - Efficient or inefficient: in terms of memory (# of bits) and/or computation (# of CPU cycles)

Compression is efficient coding (in terms of memory)

Information

Information is something unknown

More probabilistically, it is something *unexpected*

What's the missing letter?

H E L _ O

Now what letter is missing?

_ A Y

Different letters and/or different contexts convey different amounts of information

Information vs. Data

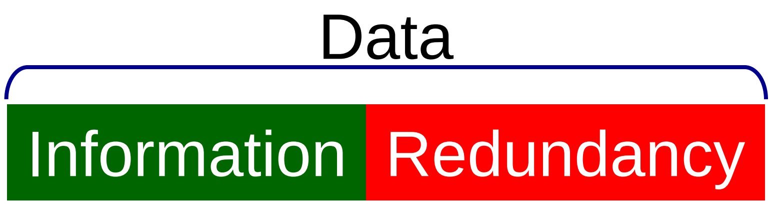
Data: the actual bits, bytes, letters, numbers, etc.

Information: the content

Redundancy: difference between data and information

$$redundancy = data - information$$

Compression: keep the information and remove the redundancy (as much as possible)



Types of Redundancy

Remember:

$$\textit{redundancy} = \textit{data} - \textit{information}$$

In general, there are three types of redundancy:

Coding	Inefficient allocation of bits for symbols
Inter-sample (inter-pixel)	Predictability in the data
Perceptual (visual)	More data than we can hear/see

Types of Compression

Compression algorithms characterized by information preservation:

- Loss-less or information-preserving: No loss of information (text, legal, or medical applications)
- Lossy: Sacrifice some information for better compression (web images)
- Near-lossless: No (or very little) perceptible loss of information (increasingly accepted by legal, medical applications)

Quantifying Compression and Error

- Compression described either using:
  - Compression ratio: popular but less technical
  - Rate: bits-per-symbol (bps) or bits-per-pixel (bpp)
- Distortion (Error) is measured by comparing the compressed-decompressed result $\hat{f}$ to the original f:

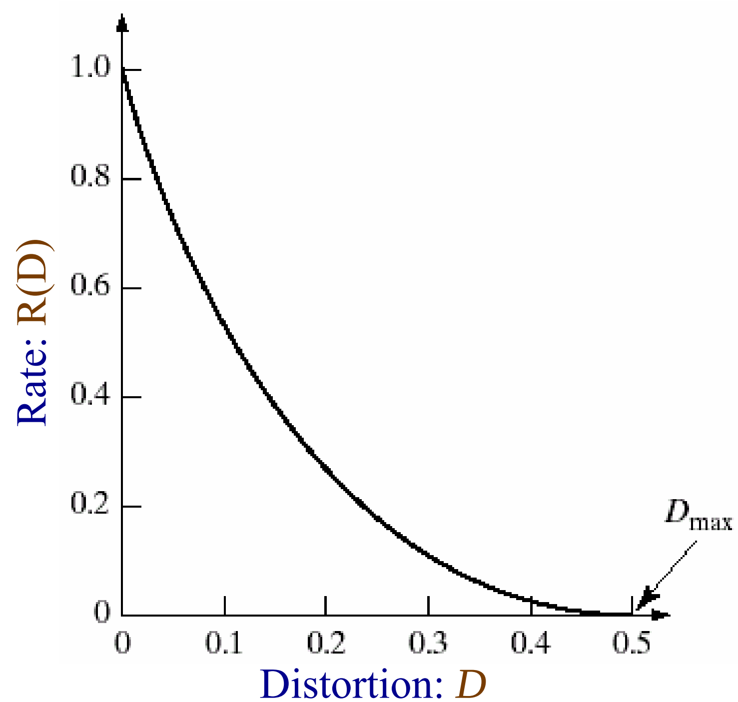
$$error_{rms} = \sqrt{\sum_{y=1}^{M} \sum_{x=1}^{N} \left(\hat{f}(x,y) - f(x,y) \right)^2}$$

$$SNR = \frac{signal}{error} = \frac{\sqrt{\sum_{y=1}^{M} \sum_{x=1}^{M} f(x,y)^2}}{\sqrt{\sum_{y=1}^{M} \sum_{x=1}^{M} \left(\hat{f}(x,y) - f(x,y) \right)^2}}$$

Quantifying Compression and Error

Most lossy algorithms let you trade off accuracy vs. compression

This is described as the *rate distortion curve*: The fewer bits required, the more distorted the signal



Quantifying Information: Entropy

Information content (entropy) of symbol *a* with probability of occurrence *p*(*a*):

$$info_a = -\log_2 p(a) \text{ bits}$$

Examples:

- 8 possible symbols with equal probability (for each symbol *a*):

$$info_a = -\log_2 \frac{1}{8} = 3 \text{ bits}$$

- Symbol *a* with probability $\frac{1}{8}$, with 255 other symbols:

$$info_a = -\log_2 \frac{1}{8} = 3 \text{ bits}$$

Entropy for a Language

The average bits of information (entropy) for a language with n symbols a_1, a_2, …, a_n is:

$$H = \sum_{i=1}^{n} p(a_i) \cdot info_{a_i}$$

$$= -\sum_{i=1}^{n} p(a_i) \cdot \log_2 p(a_i)$$

where $p(a_i)$ is the probability of symbol a_i occurring.

Entropy for a Language: Example

Flip two fair coins and communicate one of three messages: both heads, both tails, one each

Message	Probability	Information
Both heads	1/4	2 bits
Both tails	1/4	2 bits
One each	1/2	1 bit
Weighted Average		1.5 bits

Context

- Information is based on expectation
- Expectation is based on context
- Therefore: *Full analysis of information content must consider context*

Examples of contexts:

- Last *n* letters
- Last *n* values for a time-sampled sequence
- Neighboring pixels in an image

Calculating Information in Context

Without considering context, the information content of symbol a_i is

$$information = -\log_2 p(a_i) \text{ bits}$$

where $p(a_i)$ is the probability of symbol a_i occurring.

Considering context, the information content of the same symbol after sequence $a_0 \ldots a_{i-1}$ is

$$information = -\log_2 p(a_i | a_0 \ldots a_{i-1}) \text{ bits}$$

where $p(a_i | a_0 \ldots a_{i-1})$ is the probability of symbol a_i occurring immediately after symbols $a_0 \ldots a_{i-1}$.

Entropy Coding

Entropy coding allocates bits per symbol or groups of symbols according to information content (entropy)

- Huffman Coding: Optimal unique coding on a per-symbol basis
- Arithmetic Coding: Encodes a sequence of symbols as an infinite-precision real number (About 2:1 better than Huffman)
- Vector Quantization: Encoding large groups of symbols with (lossy) approximations

The theoretical compression limit of entropy coding is the entropy of the language (set of symbols) itself

Huffman Coding

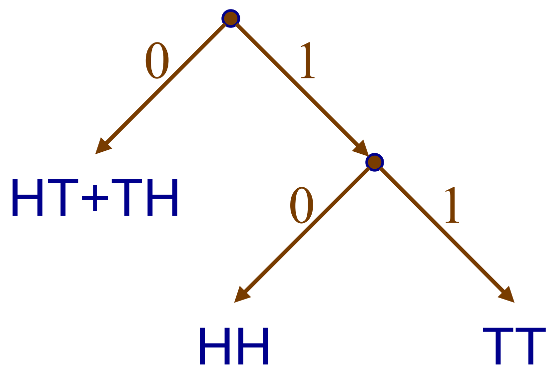
Algorithm for producing variable-length codes based on entropy:

- Sort the symbols according to probability of occurrence
- Repeat the following until there's just one symbol left:
  - Combine the two symbols with the least probability into a single new symbol and add their probabilities
  - Re-sort the symbols (insertion sort of new symbol) according to probability

The "combinations" form a binary tree structure that can be encoded using a single bit for each level

Huffman Coding (cont.)

Message	Probability	Information
Both heads	1/4	2 bits
Both tails	1/4	2 bits
One each	1/2	1 bit

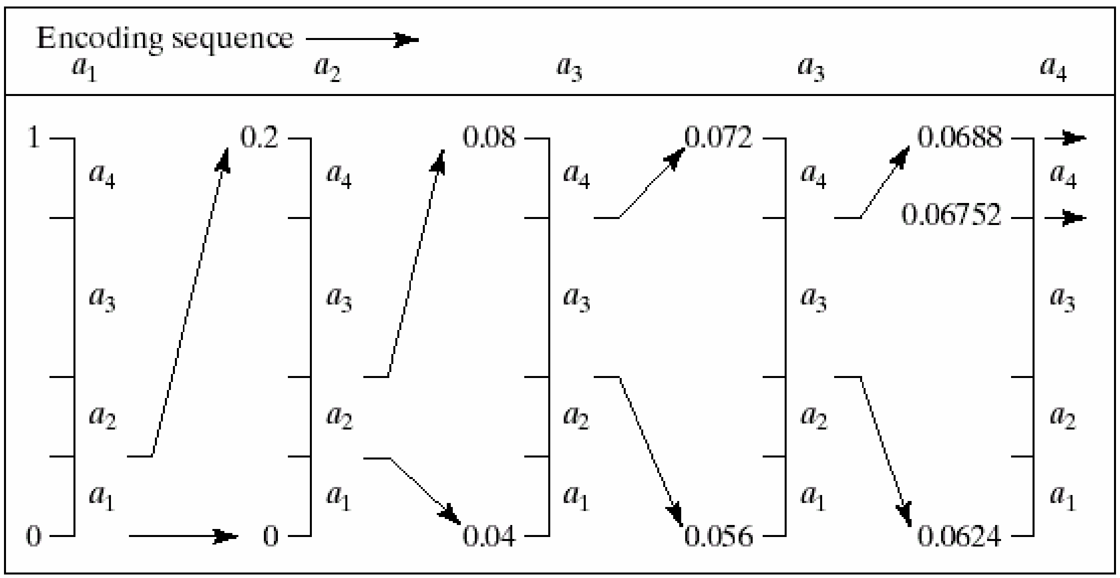


Properties:

- Decoding is just traversing the tree: When you reach a leaf, emit symbol and start a new one
- Prefix property (required for variable-length codes): No symbol's code appears as the beginning of a longer one
- Each symbol is encoded with approximately $\lceil -\log_2 p(a) \rceil$ bits

Arithmetic Coding

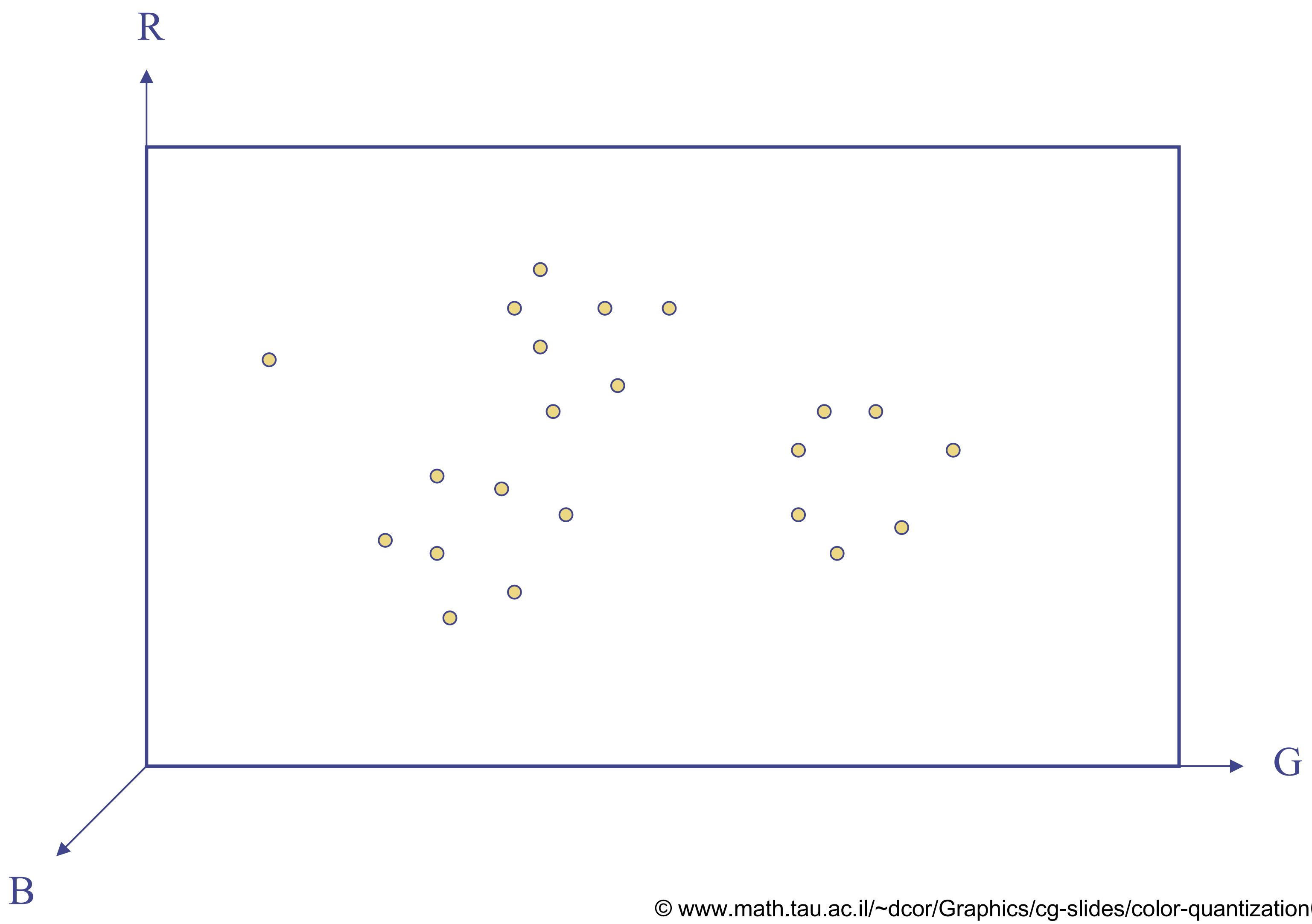
- Huffman encodes symbols one-by-one, so you have to "round up" the bit allocation
- Arithmetic coding doesn't use a one-to-one mapping between symbols and bit patterns
- The entire string is encoded as one (long) real number



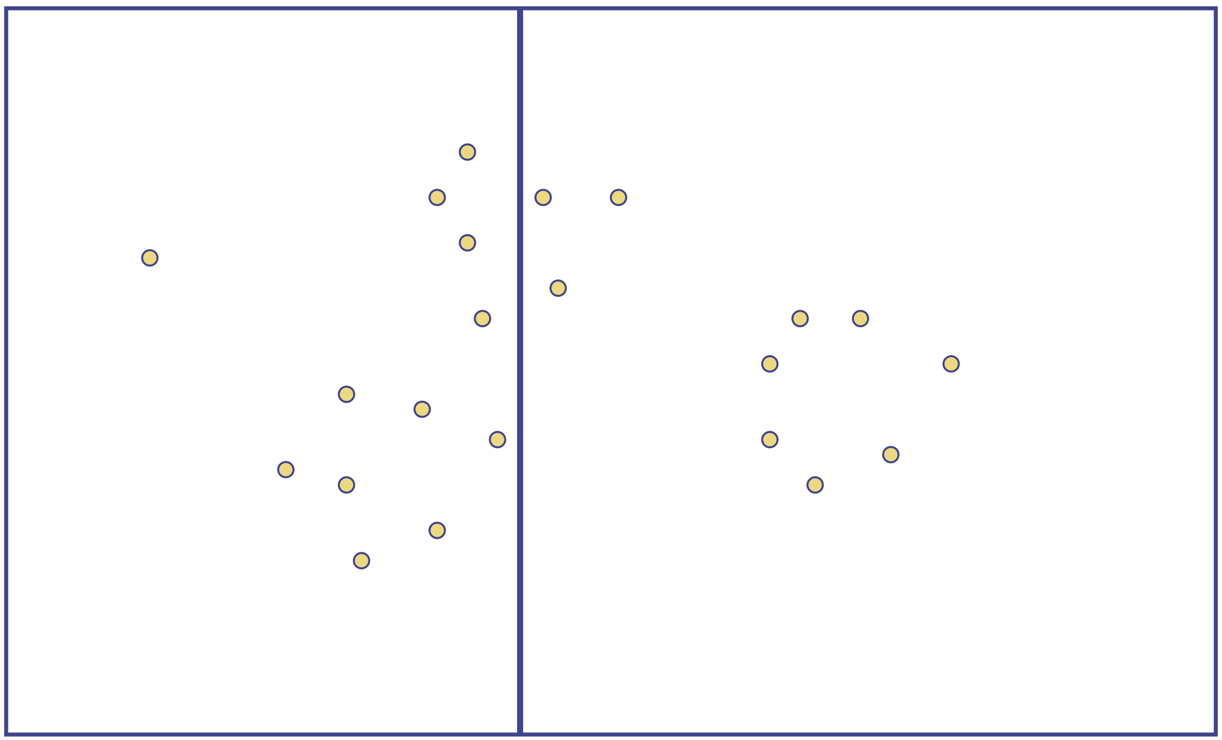
Vector Quantization

- Divide the complex signal into n clusters and their cluster centers
- Transmit only the cluster centers (codebook vectors)—this causes loss but reduces the space of possible values
- Use a unique code for each vector and transmit that code
- Most VQ systems intelligently select the quantization based on the signal content—so you also have to send the codebook (encoding scheme) once

Median Cut

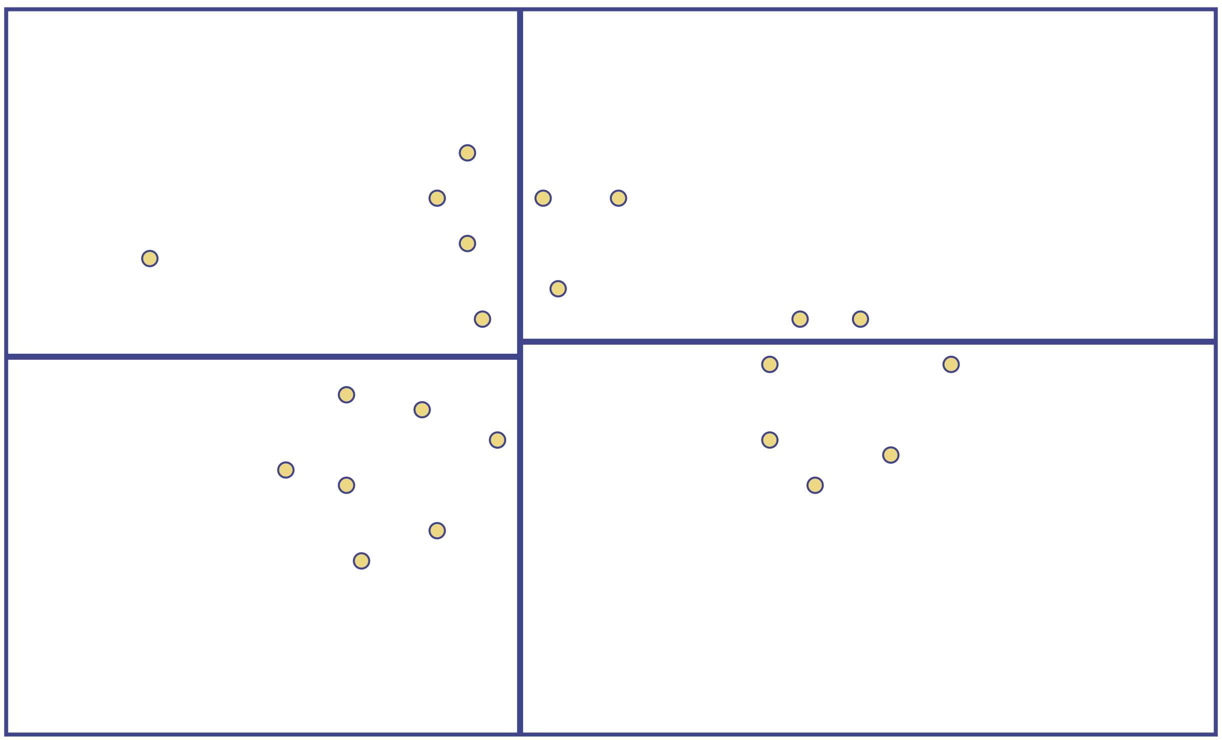


Median Cut

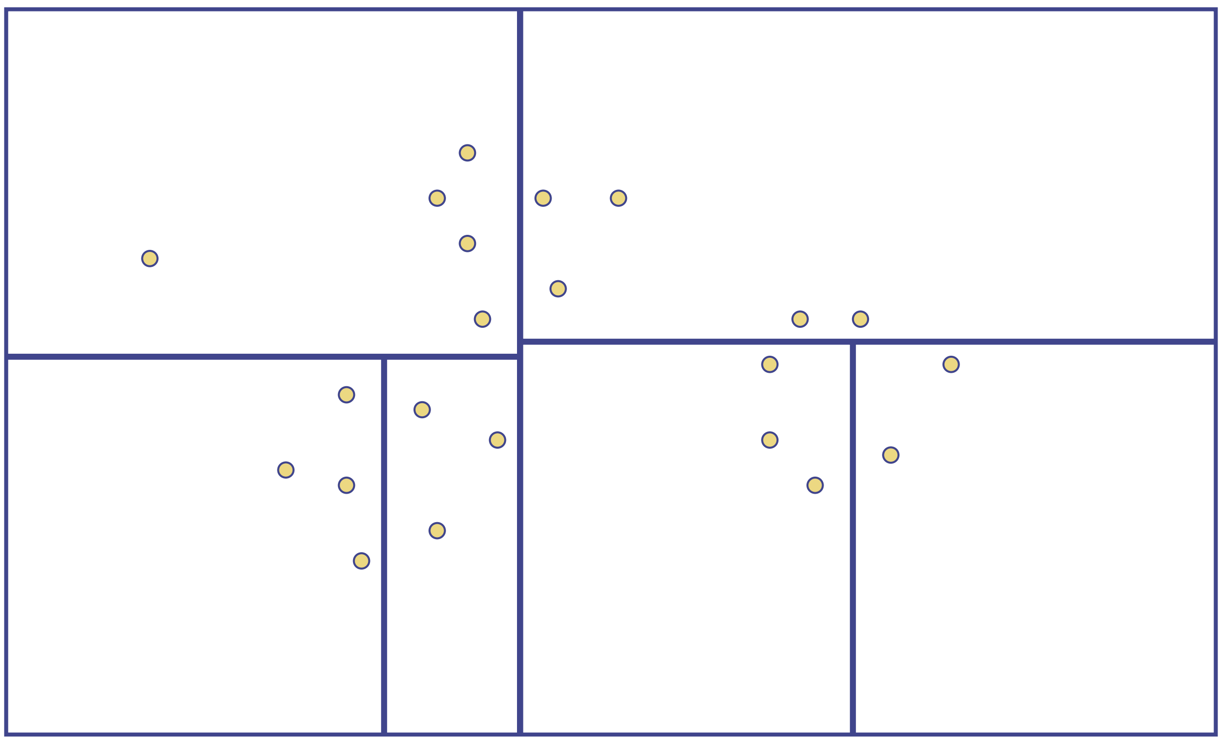


© www.math.tau.ac.il/~dcor/Graphics/cg-slides/color-quantization04.ppt

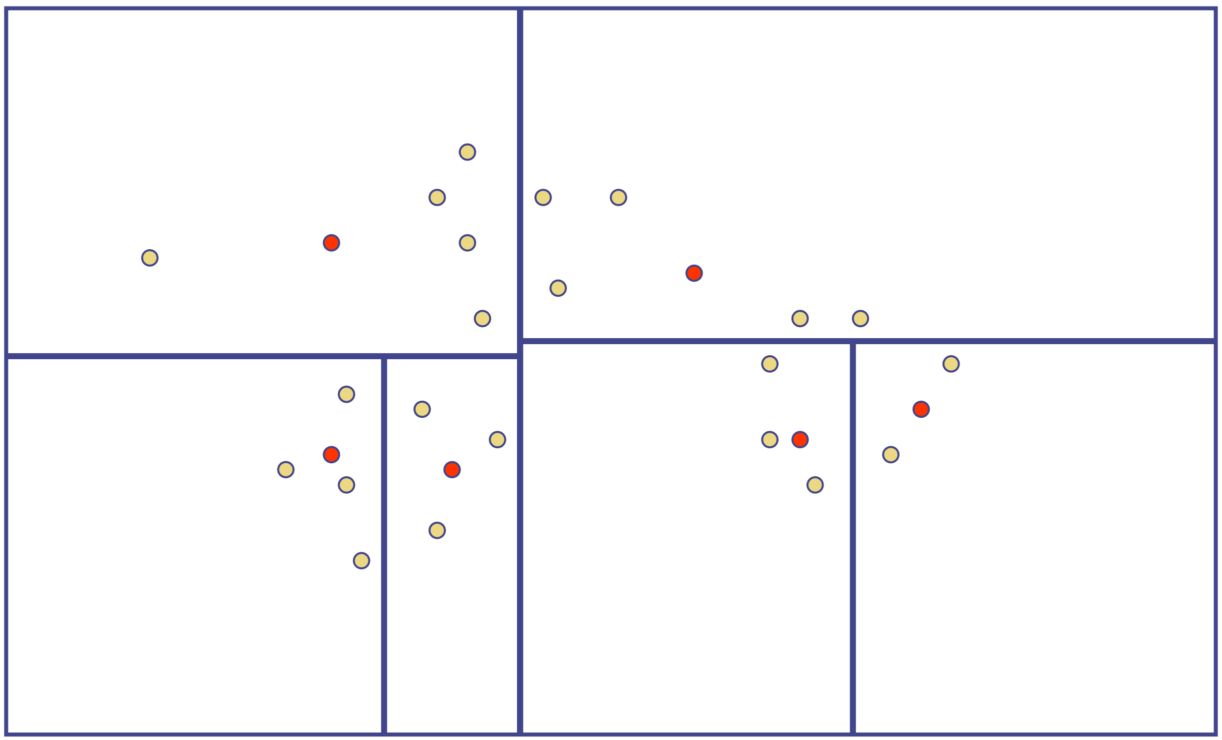
Median Cut



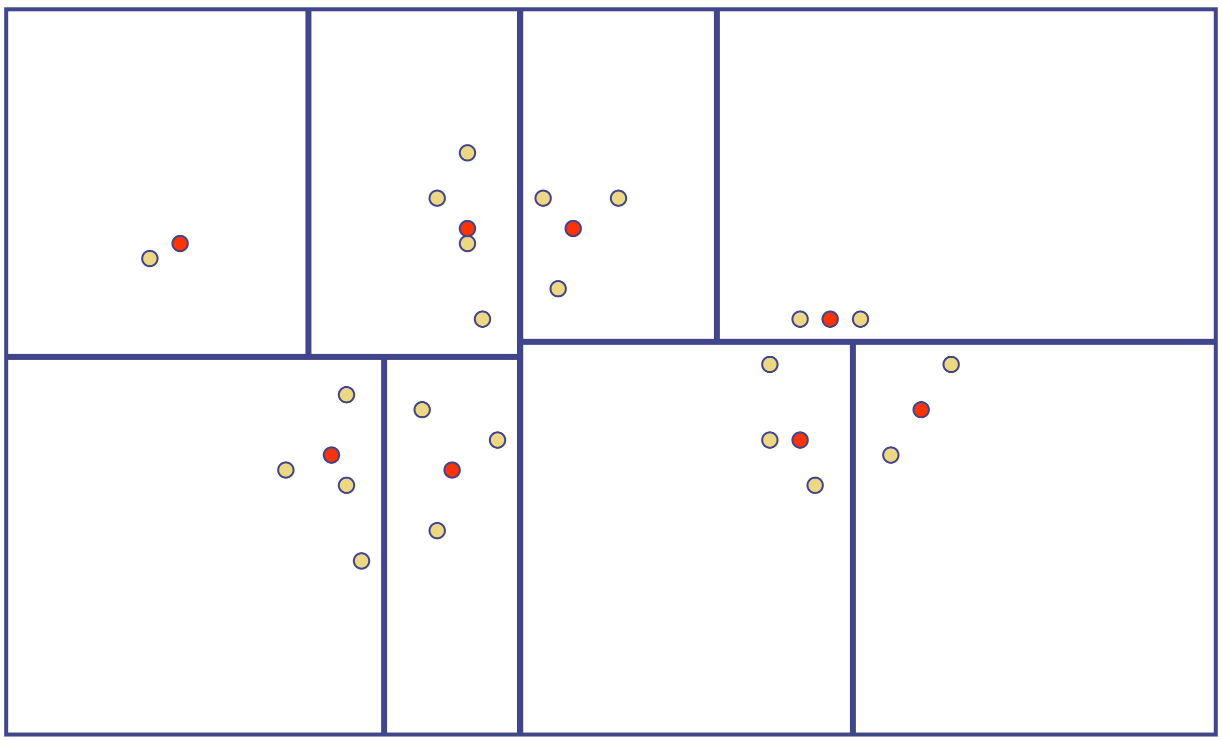
Median Cut



Median Cut



Median Cut



The Median Cut Algorithm

```
Color_quantization(Image, n){

        For each pixel in Image with color C, map C in RGB space;

        B = {RGB space};
        While (n-- > 0) {
                L = Heaviest (B);
                Split L into L1 and L2;
                Remove L from B, and add L1 and L2 instead;
                }

        For all boxes in B do
                assign a representative (color centroid);

        For each pixel in Image do
                map to one of the representatives;
}
```

The Median Cut Algorithm

Is this algorithm image dependent?

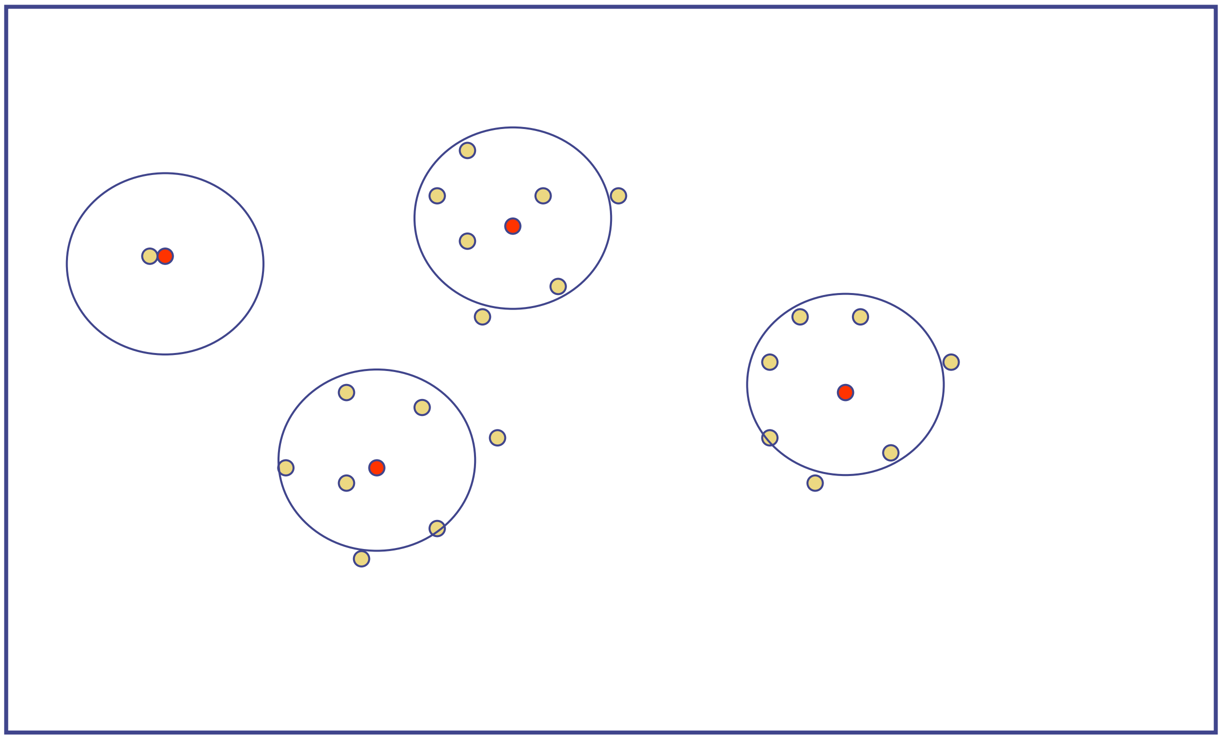
What is the Heaviest(B) box?
Several factors have to be weighed:

- The total number of image colors in the box.
- The total number of DIFFERENT image colors in the box.
- The physical size of the box.

Which representative should be chosen for a given color?

- The representative of the box containing the color.
- The closest representative under some metric.

A Better Solution



k-Means a.k.a. Linde, Buzo & Gray (LBG)

Encoding Distortion Error:

$$E = \sum_{i \text{ (data points)}} \left\| v_i - w_{j(i)} \right\|^2 m_i$$

Lower $E\left(\{w_j(t)\}\right)$ iteratively: Gradient descent $\forall r:$

$$\Delta w_r(t) \equiv w_r(t) - w_r(t-1) = -\frac{\varepsilon}{2} \cdot \frac{\partial E}{\partial w_r} = \varepsilon \cdot \sum_i \delta_{rj(i)} \left(v_i - w_r\right) m_i \; .$$

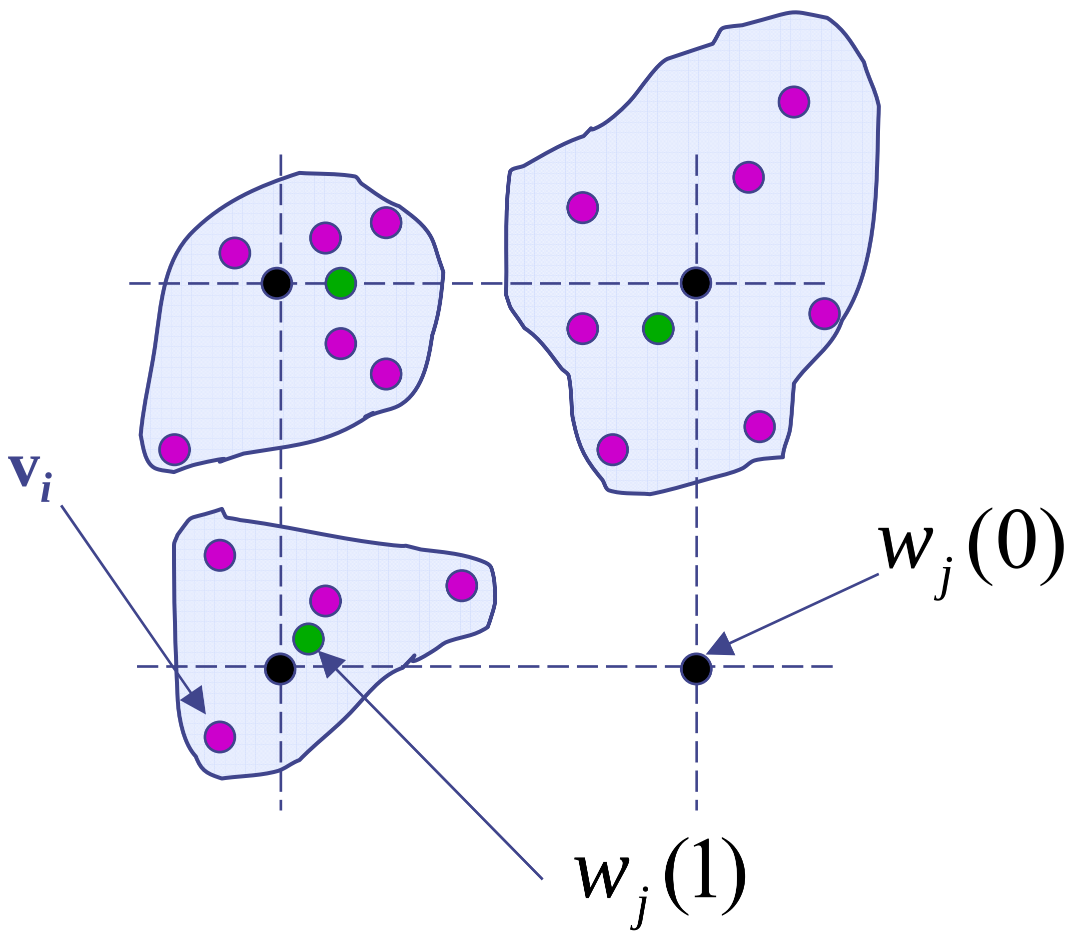
Inline (Monte Carlo) approach for a sequence $v_i(t)$ selected at random according to propability density function m_i :

$$\Delta w_r(t) = \tilde{\varepsilon} \cdot \delta_{rj(i)} \cdot \left(v_i(t) - w_r\right).$$

Advantage: fast, reasonable clustering.
Limitations: depends on initial random positions,
difficult to avoid getting trapped in the many local minima of E

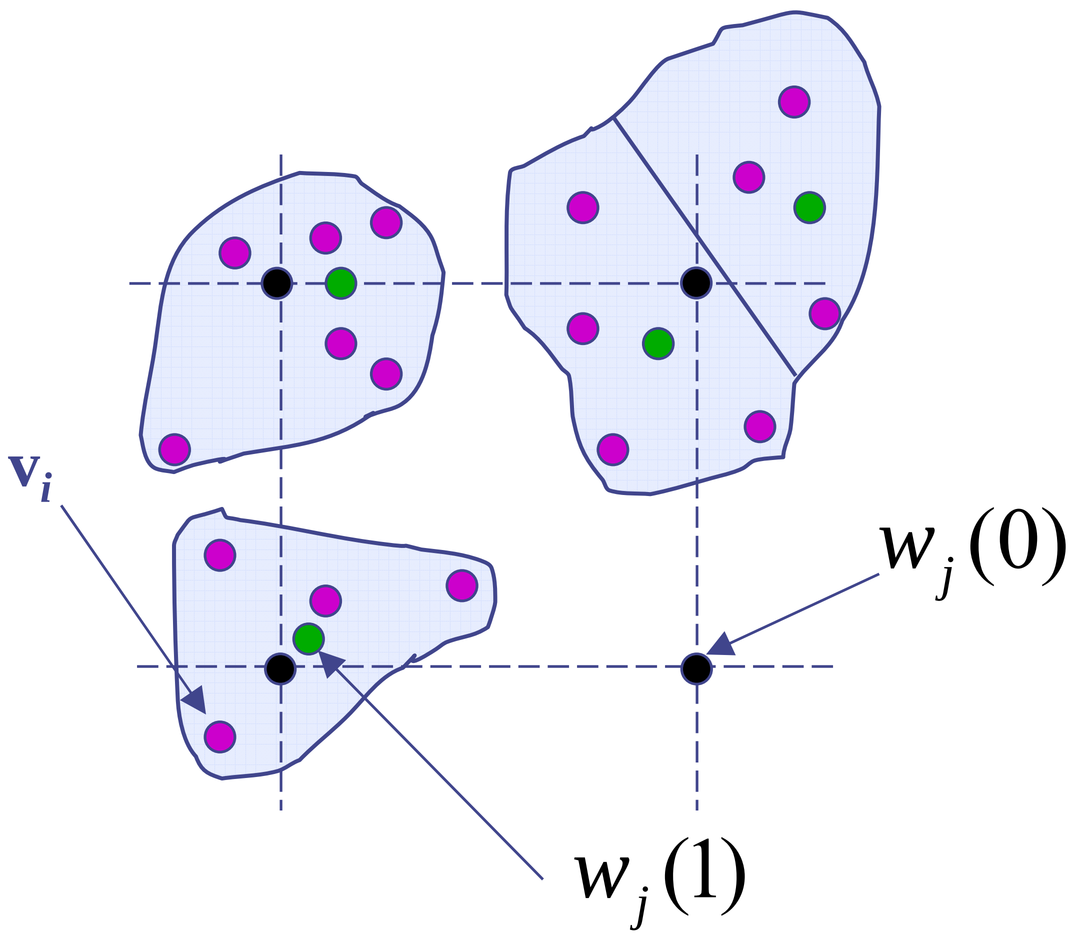
k-Means with Splitting

Generalized Lloyd Algorithm (GLA)



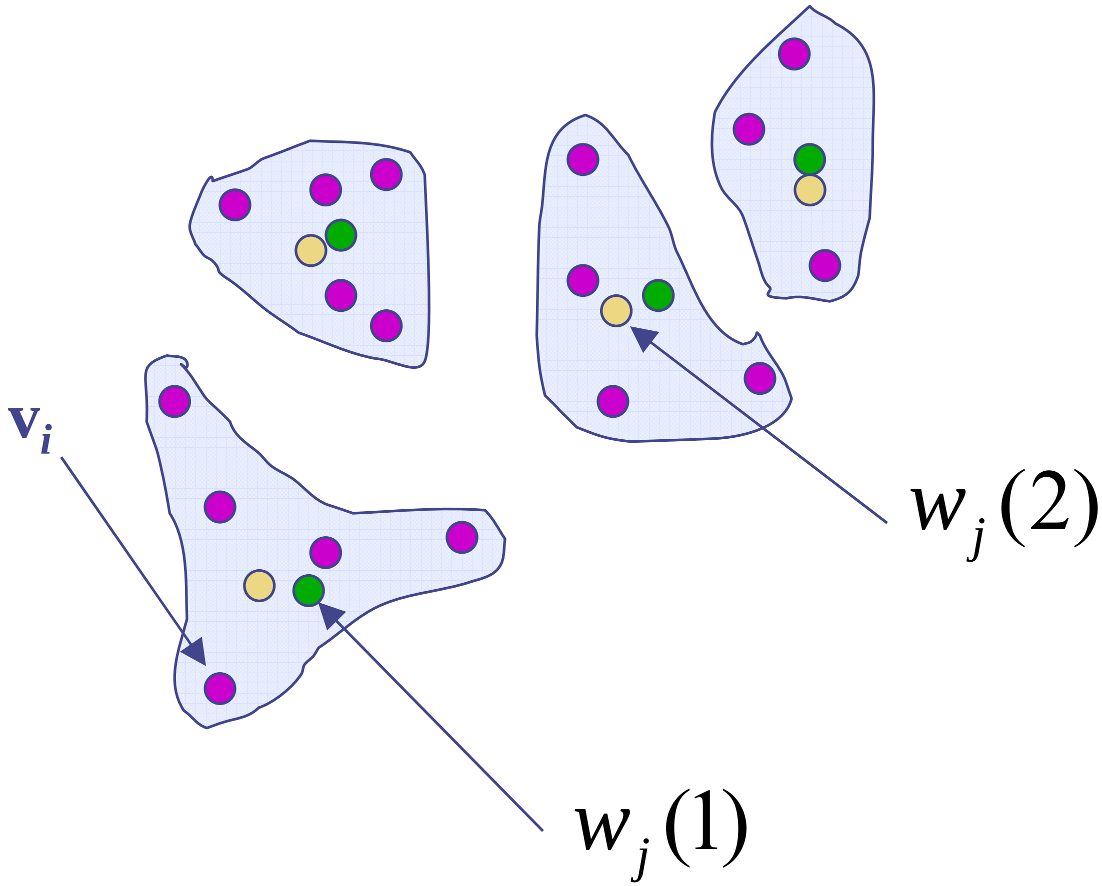
k-Means with Splitting

Generalized Lloyd Algorithm (GLA)



k-Means with Splitting

Generalized Lloyd Algorithm (GLA)



Run-Length Encoding (RLE)

- Encode sequences of identical symbols as (symbol, count) pairs
- Can use fixed-size counts or special prefixes to indicate the number of bits for the count:
  - Fixed: can reduce compression if either too large or too small
  - Variable: overhead for the prefixes
- Can extend to multiple dimensions
  - Encode difference from previous line (hopefully long runs of 0's)
  - Encode using length or markers from previous line
- Useful for binary signals and black-and-white images (or for signal that have only a few possible values)
  - 2-D RLE is used in the CCITT fax standard

Lempel-Ziv-Welch (LZW)

- Basic idea: encode longest possible previously-seen sequence
- Coding stream is mixture of symbols and back-pointers
- Better yet:
  - Keep a “codebook” of previously-seen sequences
  - Store codebook index instead of backwards pointers
- Used in most common text compression algorithms, zip, and the GIF image standard

LZW: Basic Idea

```
codebook = all single symbols
sequence = empty
while (get(symbol))
  if sequence + symbol is in codebook
    sequence += symbol
  else
    output(code for sequence)
    add sequence + symbol to codebook
    sequence = symbol
```

LZW: Example

```
Mary had a little lamb,
little lamb, little lamb.
Mary had a little lamb,
Its fleece was white as snow.
```

Inter-Pixel Redundancy

The basis of inter-sample or inter-pixel redundancy is

- Repetition
- Prediction

Predictive Coding

- Use one set of pixels to predict another
- Predictions:
  - Next pixel is like the last one
  - Next scan line is like the last one
  - Next frame is like the last one
  - Next pixel is the average of the already-known neighbors
- *The error from the prediction* (residual) *hopefully has smaller entropy than the original signal*
- The information used to make the prediction is the context

Predictive Coding

Key: Sender and receiver use the same predictive model

- Sender:
  - Make prediction (no peeking)
  - Send the residual (difference)
- Receiver:
  - Make prediction
  - Add the residual to get the correct value

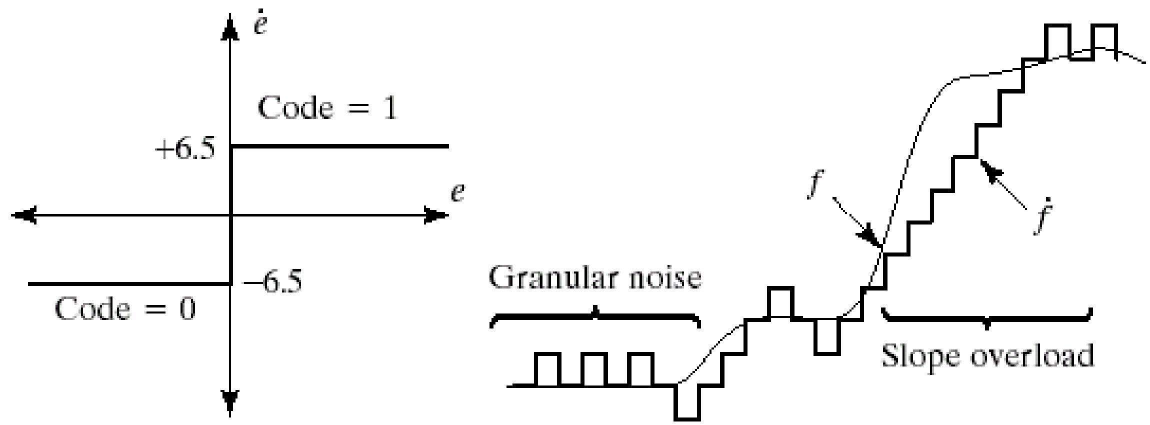
Loss-less: entropy code the residual

Lossy: quantize the residual

Delta Modulation

- Basic algorithm:
  - Prediction: next signal value is the same as the last
  - Residual is the difference (delta) from the previous one
  - Residual is encoded in a smaller number of bits than the original
- Often used in audio systems (phones)
- Problem: limited-range (or limited-precision) delta can cause under/over-shoot

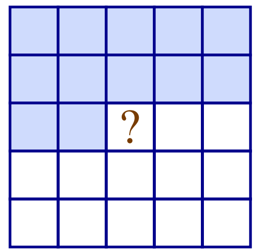
Delta Modulation



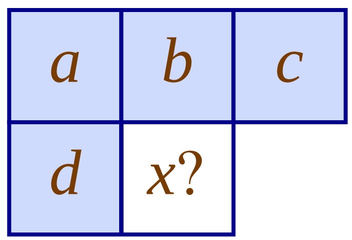
Input		Encoder				Decoder		Error
n	f	$\hat{f}$	e	$\dot{e}$	$\dot{f}$	$\hat{f}$	$\dot{f}$	$[f - \dot{f}]$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	–5.5
2	14	20.5	–6.5	–6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	–5.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Predictive Image Coding

Predict next pixel based on neighbors that have already been seen



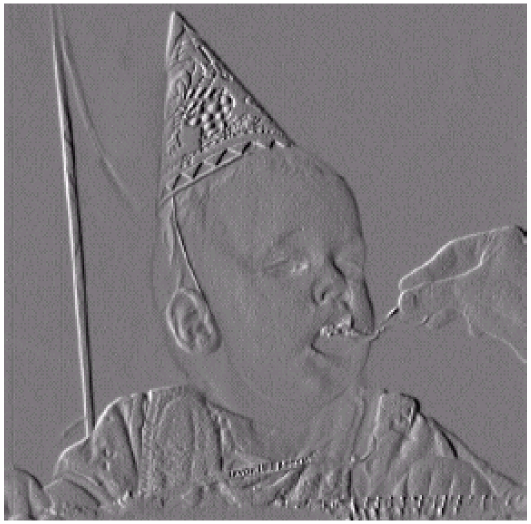
- Simple predictor: average of the four neighbors

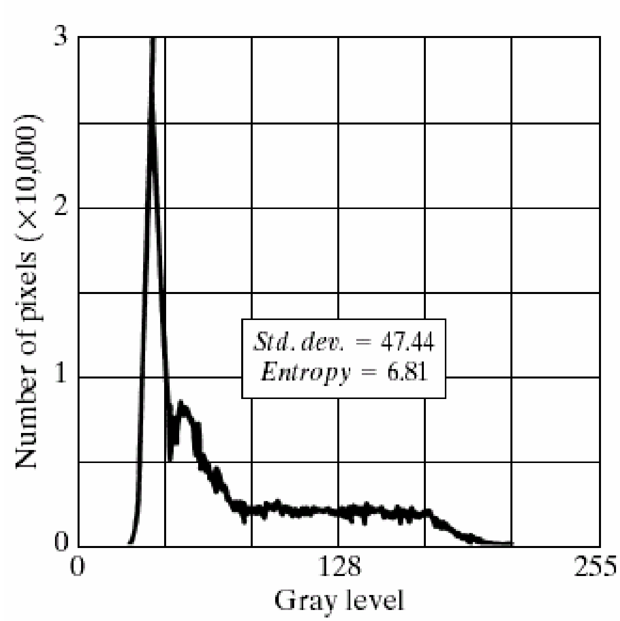


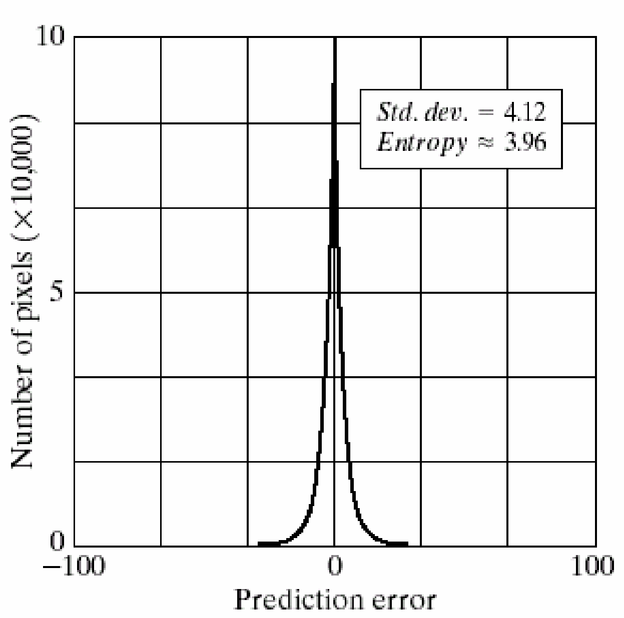
$$x \approx \tfrac{1}{4}(a + b + c + d)$$

- Can use a larger context
- Can quantize (lossy) or entropy code (loss-less) the residual

Predictive Image Coding (cont.)







Perceptual Redundancy

Eye is less sensitive to

- Color
- High Frequencies

So,

- Allocate more bits to intensity than chromaticity
- Allocate more bits to low frequencies than to high frequencies

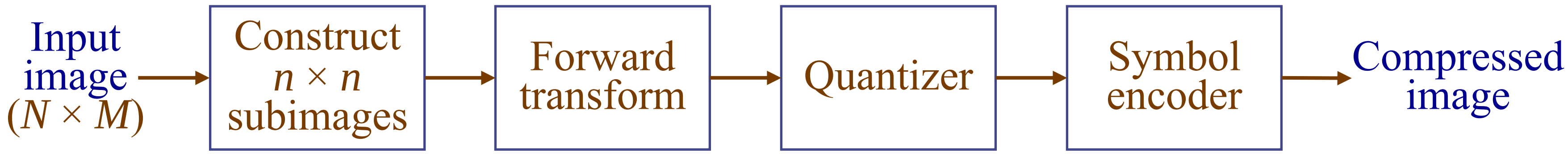
Can play similar tricks with the ear and varying sensitivity to different frequencies (e.g., the "psycho-acoustic model" plays a key role in MP3)

Block Transform Coding

- Use some transform to convert from spatial domain to another (e.g., a frequency-based one)
- Advantage #1: Many transforms "pack" the information into parts of the domain better than spatial representations (e.g. DCT)
- Advantage #2: Quantize coefficients according to perception (e.g., quantize high frequencies more coarsely than low ones)
- Problem: artifacts caused by imperfect approximation in one place get spread across the entire image
- Solution: independently transform and quantize blocks of the image → *block transform encoding*

Transform Coding: General Structure

Encoder

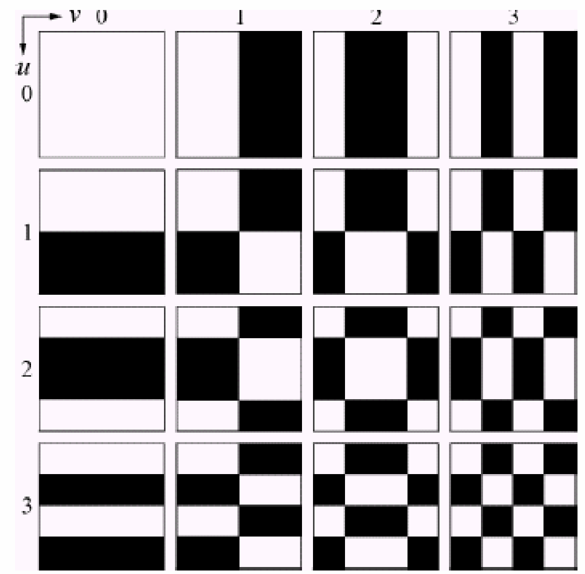


Decoder

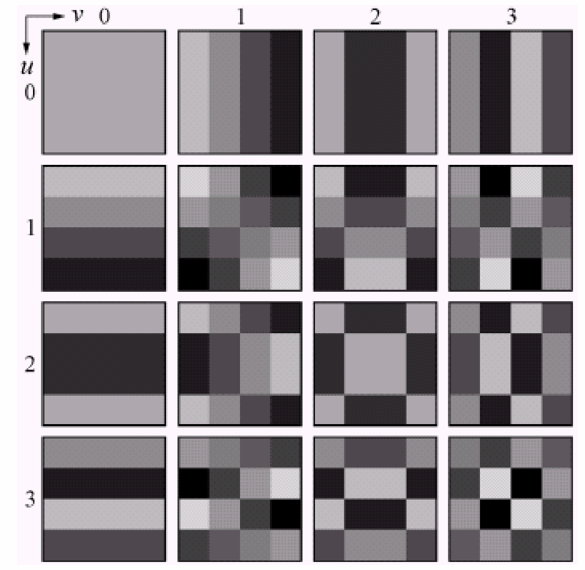
Transform Coding (cont.)

Frequently used basis sets (here: for 4 × 4 blocks):

Walsh-Hadamard



DCT (Cosines)



Applications: GIF and JPEG

GIF

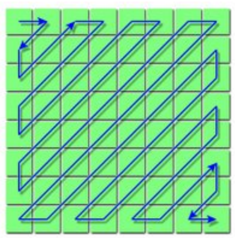
- Graphics Interchange Format.
- Uses 256 (or fewer) distinct colors (unsuitable for photographs).
- Uses only lossless data compression (LZW).
- Large file sizes (unsuitable for photographs on the web).
- Best for sharp transitions in diagrams, buttons, etc, where lossy JPEG compression does poorly.
- LZW royalty disputes historically led to development of successor (PNG), but GIF still dominant on web

JPEG

- Joint Photographic Experts Group
- Lossy compression of photographic images

A photo of a flower compressed with successively lossier compression ratios from left to right.

JPEG Overview

Intensity/Chromaticity	Convert to YCrCb color model and down-sample (allocate fewer bits to) the chromaticity components
8 × 8 block DCT	Energy compaction by converting to frequency representation
Predictively encode DC coefficients	Takes advantage of redundancy in the block averages
Quantize AC coefficients	Many high frequencies become zero!
Zig-zag ordering	 Changes from 2-D to 1-D to group similar frequencies together
Run-length encoding	Collapses long runs of zeros
Entropy encode what's left	Huffman coding to more efficiently encode the RLE sequences (arithmetic coding also allowed in standard)

JPEG Usage

- JPEG is at its best on photographs and paintings of realistic scenes with smooth variations of tone and color.
- In this case it will produce a much higher quality image than other common methods such as GIF which are lossless for drawings and iconic graphics but require severe quantization for full-color images).
- JPEG compression artifacts blend well into photographs with detailed non-uniform textures, allowing higher compression ratios.

JPEG Level

File size

100%

100%

50%

16%

10%

5%

Progressive Methods

Progressive methods allow viewing of the image while downloading:

- Interlaced GIF:
  - Send every 8 scan lines, then every 4, then 2, then all
  - Interpolate intermediate lines until they get there
- Progressive JPEG:
  - Send DC (zero frequency) coefficients
  - Send all lowest-frequency AC coefficients
  - Send successively higher AC coefficients

Resources

WWW:

http://en.wikipedia.org/wiki/Wavelet_transform

http://en.wikipedia.org/wiki/GIF

http://en.wikipedia.org/wiki/JPEG

Textbook:

Kenneth R. Castleman, Digital Image Processing, Chapter 17