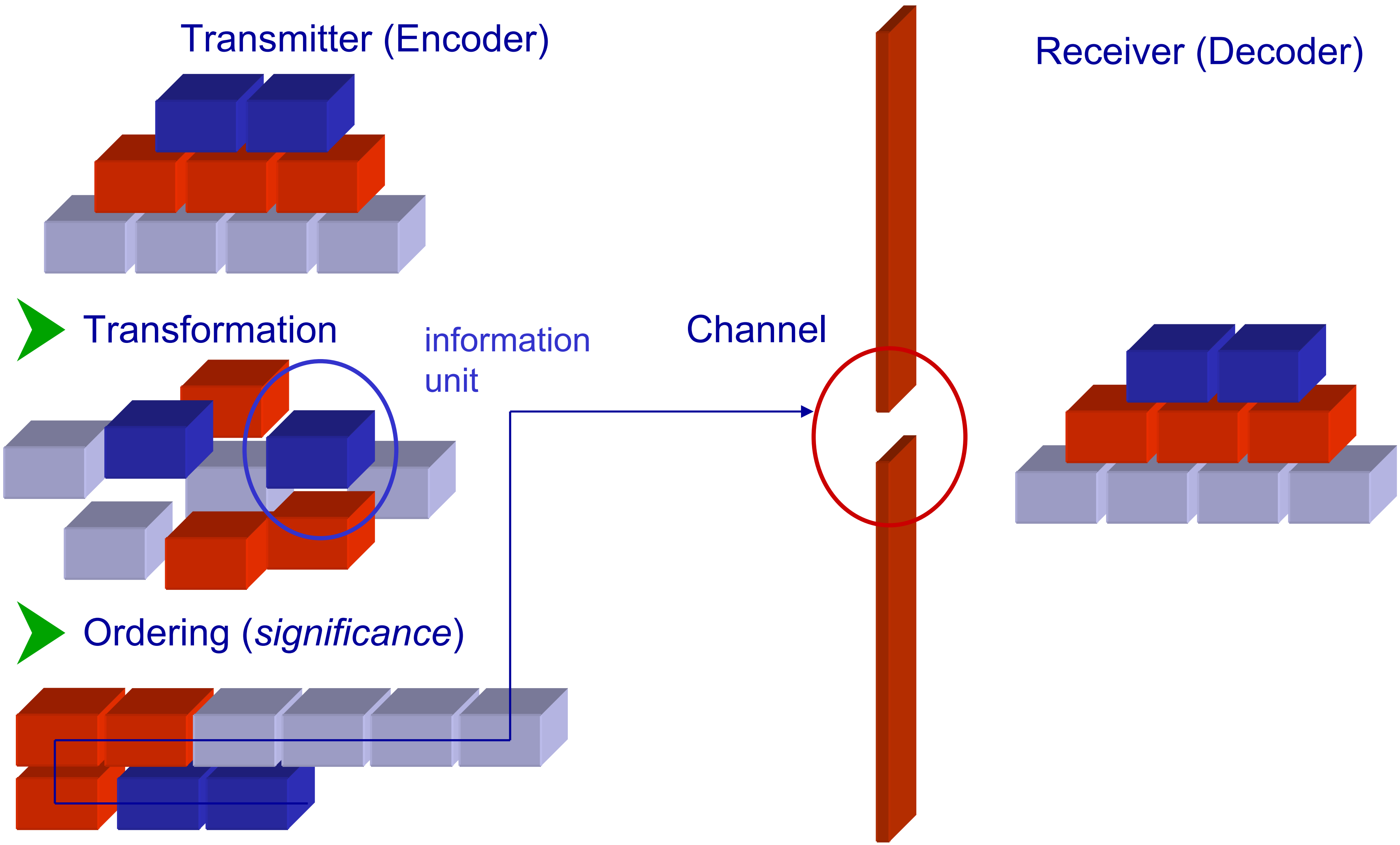
Compression and Coding

Theory and Applications

Part 1: Fundamentals

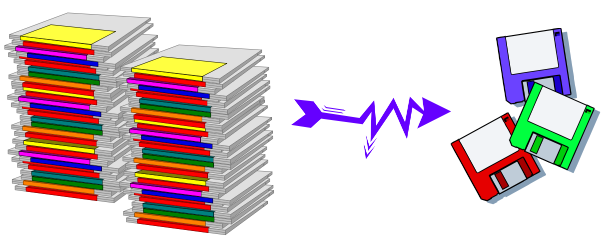
What is the problem?



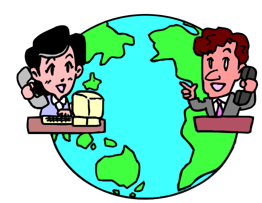
Why is it important?

- The available resources for signal communication and archiving are limited

Compression

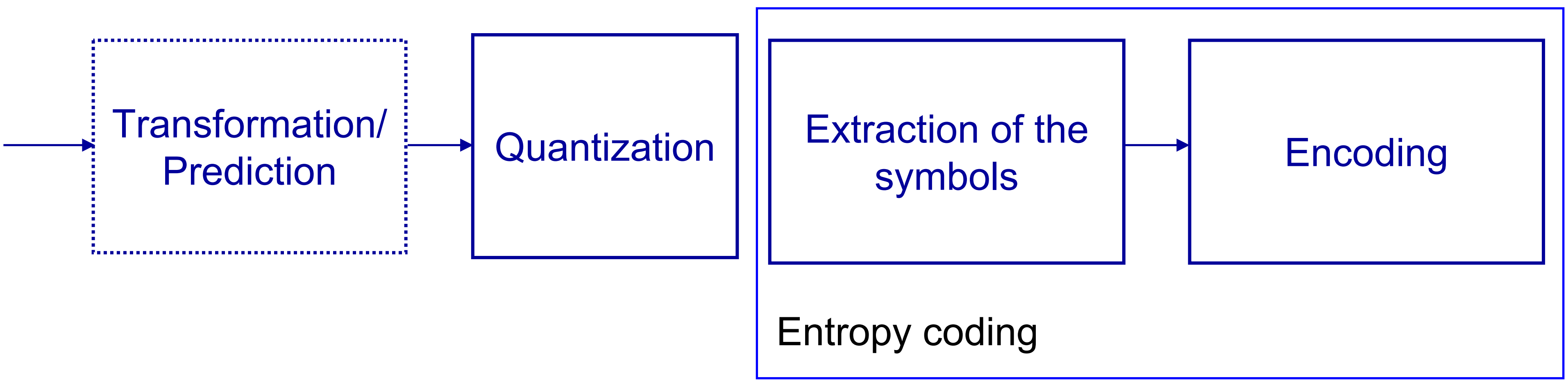


Standardization



Basic steps

- Goal: minimize the amount of resources needed to transmit a source signal from the transmitter to the receiver
- Basic steps:
  - Reduction of the redundancy in the data
    - Transform-based coding
    - Prediction-based coding
  - Translate the resulting information from to a sequence of *symbols* suitable for encoding
  - *Entropy coding* of the sequence of symbols



Basic idea

- Exploit the redundancy among the data samples for an *effective* representation of the data
- Classical coding schemes
  - Look at the data as to set of numbers and reduce the mathematical and/or statistical redundancy among the samples
    - JPEG, MPEG
- Second generation coding schemes
  - Adapt the coding scheme to the different image regions featuring some omogeneity for optimizing the coding gain given the data
    - ROI based coding, JPEG2000
- Model-based coding
  - Look at the data as to perceptual information and exploit the way such information is processed by the sensory system to improve compression

Compression modes

- Lossless
  - The original information can be recovered without loss from the compressed data
  - Low compression factors
    - Less than a factor 3 for natural images
- Lossy
  - The compression process implies the loss of information that cannot be recovered at the decoding
  - Basically due to quantization
  - Very high compression factors
  - Degradation of the perceived quality

$\Rightarrow$ Key point: rate/distortion tradeoff

Information theoretical limits

- Noisy channel coding theorem
  - Information can be transmitted reliably (i.e. without error) over a noisy channel at any source rate, R, below a so-called *capacity C of the channel*

$$R<C \text{ for reliable transmission}$$

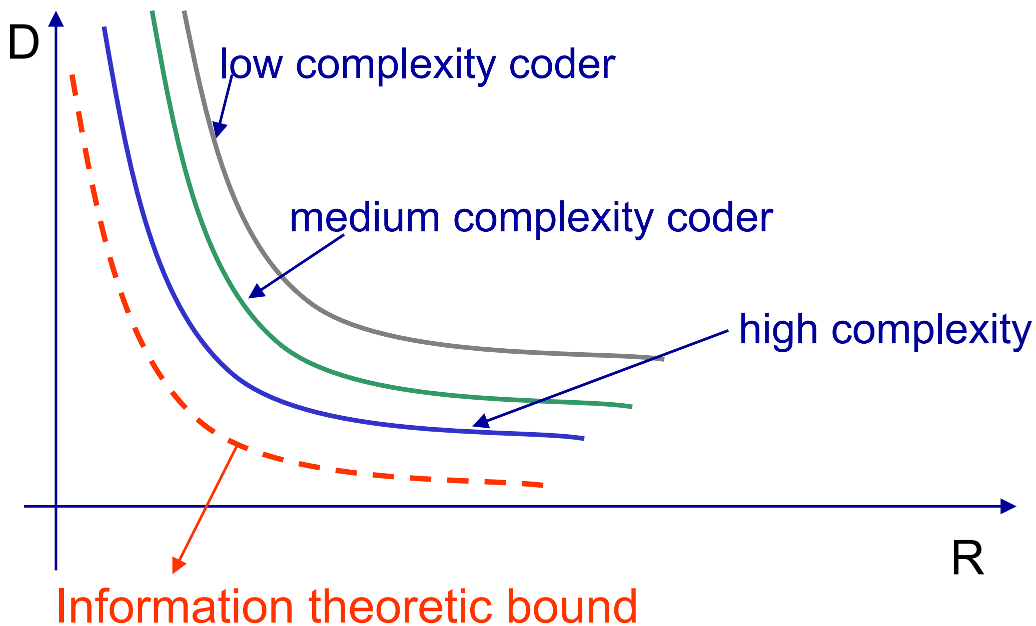
- Source coding theorem
  - There exists a map from the source waveform to the codewords such that for a given distortion D, $R(D)$ bits (per source sample) are sufficient to enable waveform reconstruction with an average distortion that is arbitrarily close to D. Therefore, the actual rate R has to obey:

$$R \geq R(D) \text{ for fidelity given by } D$$

R(D): rate distortion function

Qualitative *R(D)* curves

- *R(D)* curves are monotonically no-increasing
  - Noteworthy points
    - R(0): rate needed for exact reproduction of the source⇔ *entropy* of the source
    - Ropt, Dopt: minimum rate for a given distortion / minimum distortion at a given rate



Entropy Coding

Fundamentals

Information

- Information

  Let X be a Random Variable (RV) and s be a realization of X. Then, the information hold by symbol s can be written as

$$I(s) = -\log_2 p(s)$$

  where $p(s)$ is the probability of the symbol s.

  – $I(s)$ *represents the amount of information carried by the symbol* s.

    - ***p(s)=1*** *→ There is* ***no uncertainty*** *on the expectation on value taken by the RV →* ***no information*** *is conveyed by the knowledge of the actual value of the RV (current realization). This is expressed by the corresponding information being zero →* ***I(s)=0***
    - ***p(s)<< (very small)*** *→ the value s is highly improbable → it corresponds to a* ***rare event*** *→ knowing that the current realization of the RV is equal to s is highly informative, as an indication of a rare event. This is expressed by the corresponding information being very high in value* ***I(s) → infinity***
    - *Summary: symbols that are* ***certain*** *convey* ***no information****, while* ***very improbable*** *symbols are* ***highly informative***

Information

- Discrete time sources
  - Let X be a discrete time ergodic source generating the sequences $\{x_k\}_{\{k=1,K\}}$ of *source symbols*.
    - The sequences are realizations of the RV $\{X\}$
    - The source is *memoryless* if successive samples are *statistically independent*
  - *Information*

$$I_k = -\log_2 p_k = -\log_2 p(x_k)$$
$$p(x_k) = 1 \rightarrow I_k = 0$$
$$p(x_k) << 1 \rightarrow I_k \rightarrow \infty$$

Information

- Relation to uncertainty

  If the K symbols have the same probability $p_k = \frac{1}{K}$

  Then the information is

$$I_k = -\log_2 \frac{1}{K} = \log_2 K$$

  In this case, the *uncertainty* on the expectation is *maximized*, because all the symbols are equally probable.

  The amount of information is the *same* for all symbols

Same probability ↔ Maximum uncertainty

Entropy

- Entropy

  Let X be a discrete RV: $\{x_k\}_{\{k=1,K\}}$. Then, the *entropy* is defined as

$$H(X) = \sum_{k=1}^{K} p_k I_k = -\sum_{k=1}^{K} p_k \log_2 p_k$$

$$p_k = p(x_k)$$

  - H(X) represents the *average information content of the source* (or the average information conveyed by the RV)
  - Symbols with **same probability** (<u>maximum uncertainty</u>)

$$H(X) = \sum_{k=1}^{K} p_k I_k = \sum_{k=1}^{K} \frac{1}{K} \log_2 K = K \frac{1}{K} \log_2 K = \log_2 K$$

  - It can be shown that this corresponds to the **upper bound**

$$0 \le H(X) \le \log_2 K$$

Entropy

- Summary
  - The entropy represents the average information conveyed by the source RV
    - H(X) is the average information received if one is informed about the value of the RV X has taken
  - The entropy *increases with the degree of uncertainty* on the expectation of the realizations of the RV
    - Equivalently: it is the uncertainty about the source output before one is informed about it
  - All the discrete sources with a ***finite*** number K of possible amplitudes have a finite informational entropy that is no greater than $\boldsymbol{log_2 K}$ **bits/symbol**

$$0 \leq H(X) \leq log_2 K$$

    - The right side equality holds if and only if all probabilities are equal (most unpredictable source)
    - **Due to unequal symbol probabilities and inter-symbol dependencies $H(X)$ will in general be lower than the bound value**

- **Entropy coding** exploits unequal symbol probabilities as well as source memory to realize average bit rates approaching $H(X)$ bits/symbol

Entropy coding

- Goal: Minimize the number of bits needed to represent the values of X.
  - We consider the codes that **associate** to each **symbol** x_k a **binary word** w_k of **length** l_k.
  - A sequence of values produced by the source is coded by aggregating the corresponding binary words.
- Bit-rate
  - The *average* bit-rate to code each symbol emitted by the source is

$$R_X = -\sum_k l_k \log_2 p_k$$

  - Goal: optimize the codewords to **minimize R_X**

Shannon theorem

- The Shannon theorem proves that the entropy is a *lower bound* for the average bitrate R_X of a prefix code
- The *average rate* of a prefix code satisfies

$$R_X \geq H(X) = -\sum_k p_k log_2 p_k$$

Moreover, there exists a prefix code such that

$$R_X \leq H(X)+1$$

  - The lower bound is set by the entropy of the source
  - We cannot do better than reaching the entropy of the source
- Redundancy:

$$R(X)=log_2 K-H(X)$$

Entropy coding policies

- Fix and variable length codes
  - Fix length codes: If $\log_2 K$ is an integer, all symbols could be coded with words of the same length $l_k = \log_2 K$ bits.
  - Variable length codes: the average code length can be reduced by using *shorter* binary codewords for symbols that occur *frequently*.

$$p_k \text{ large} \rightarrow \text{short codewords}$$

$$p_k \text{ small} \rightarrow \text{long codewords}$$

- Variable Length Codes (VLCs)
  - Prefix codes
    - Huffman coding
    - Arithmetic coding

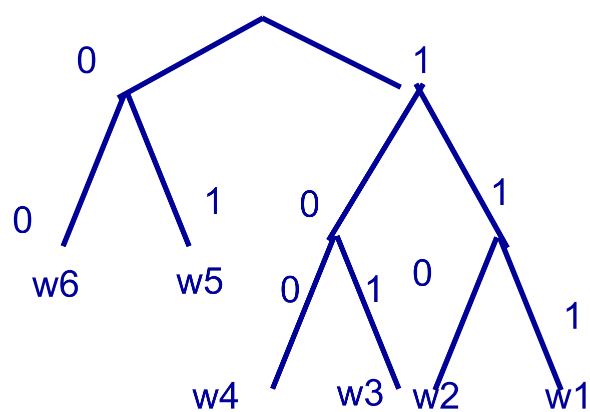
Prefix codes

- To guarantee that any aggregation of codewords is *uniquely* decodable the *prefix condition* imposes that *no codeword may be the prefix (beginning) of another one*
- Example

  {w1=0, w2=10, w3=110, w4=101}

  $\rightarrow$ 1010 can be read as both w2w2 and w4w1: ambiguous!

$\rightarrow$ Prefix codes are constructed by building binary trees



Huffman code

- Optimal prefix code tree
  - rate approaching the lower bound
- Each symbol is represented by a codeword whose length gets longer as the probability of the symbol gets smaller
- Dynamic programming rule that constructs a binary tree from bottom up by successively aggregating low probability symbols

Let us consider K symbols with their probability of occurrence sorted by **increasing** order

$p_k \leq p_{k+1}$

$$\{(x_1,p_1),(x_2,p_2),\ldots,(x_K,p_K)\}$$

we aggregate x_1 and x_2 in a single symbol of probability $p_{12}=p_1+p_2$.

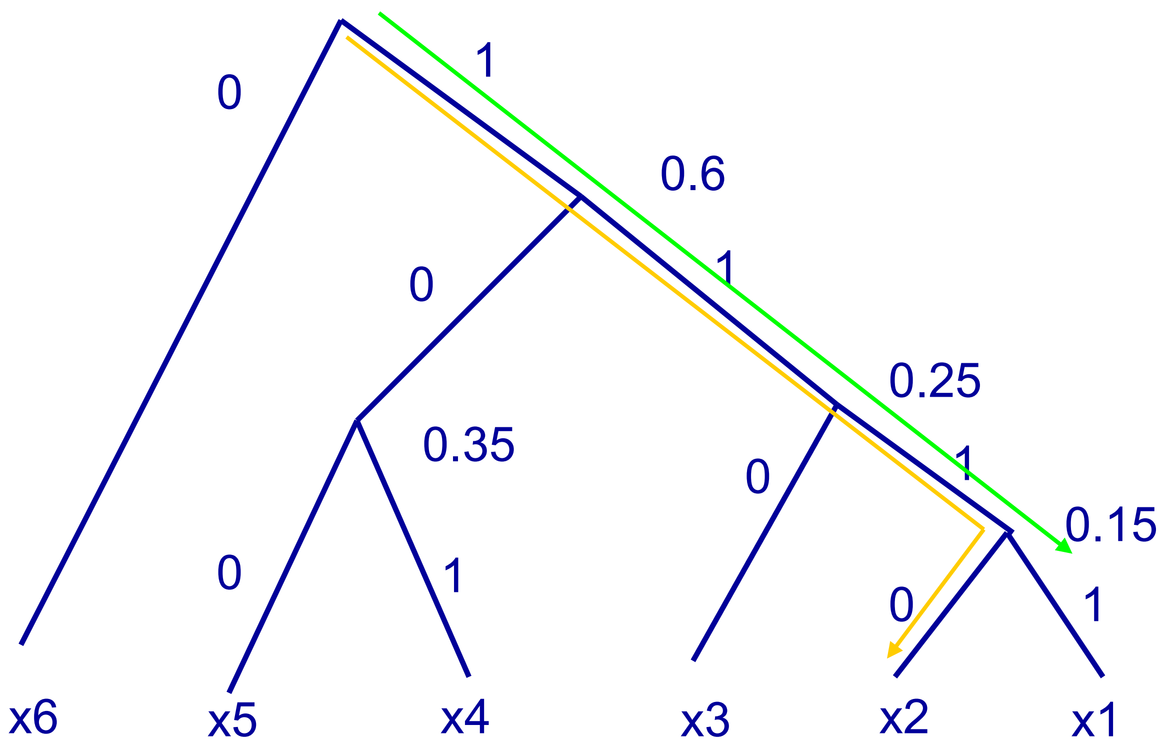
Recursivity: An optimal prefix tree for K symbols can be obtained by constructing an optimal prefix tree for the K-1 symbols

$$\{(x_{12},p_{12}),(x_2,p_2),\ldots,(x_K,p_K)\}$$

and by dividing the leafs of p_{12} in two children corresponding to x_1 and x_2

Huffman code

- Example
  - {p1=0.05, p2=0.1, p3=0.1, p4=0.15, p5=0.2, p6=0.4}



x1	1111
x2	1110
x3	110
x4	101
x5	100
x6	0

Arithmetic coding

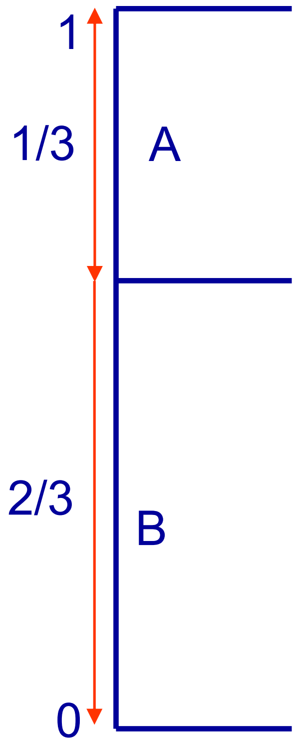
- The symbols are on the number line in the probability interval 0 to 1 in a sequence that is known to both encoder and decoder
- Each symbol is assigned a sub-interval equal to its probability
- Goal: create a codeword that is a *binary fraction* pointing to the interval for the symbol being encoded
- Coding additional symbols is a matter of subdividing the probability interval into smaller and smaller sub-intervals, always in proportion to the probability of the particular symbol sequence

Arithmetic coding

- Example

  p(A)=1/3

  p(B)=2/3

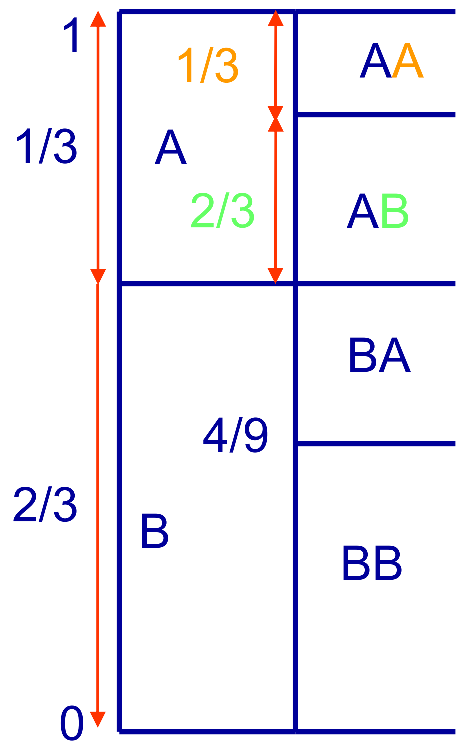


Arithmetic coding

- Example

  p(AA)=1/3*1/3=1/9 p(BA)==1/3*2/3=2/9

  p(AB)=2/9 p(BB)=4/9



Arithmetic coding

- After encoding many symbols
  - the final interval *width* P is the *product* of the probabilities of all symbols coded;
  - the interval *precision*, the number of bits required to express an interval of that size, is given approximately by $-log_2(P)$.

  Therefore, since

$$P=p_1*p_2*....*p_N$$

the number of bits of precision is approximately

$$-\ log_2(P)= -(log_2(p_1)+log_2(p_2)+....+\ log_2(p_N))$$

thus *the codestream length will be very nearly equal to the information for the individual symbol probabilities*, and the average number of bits/symbol will be very close to the bound computed from the entropy.

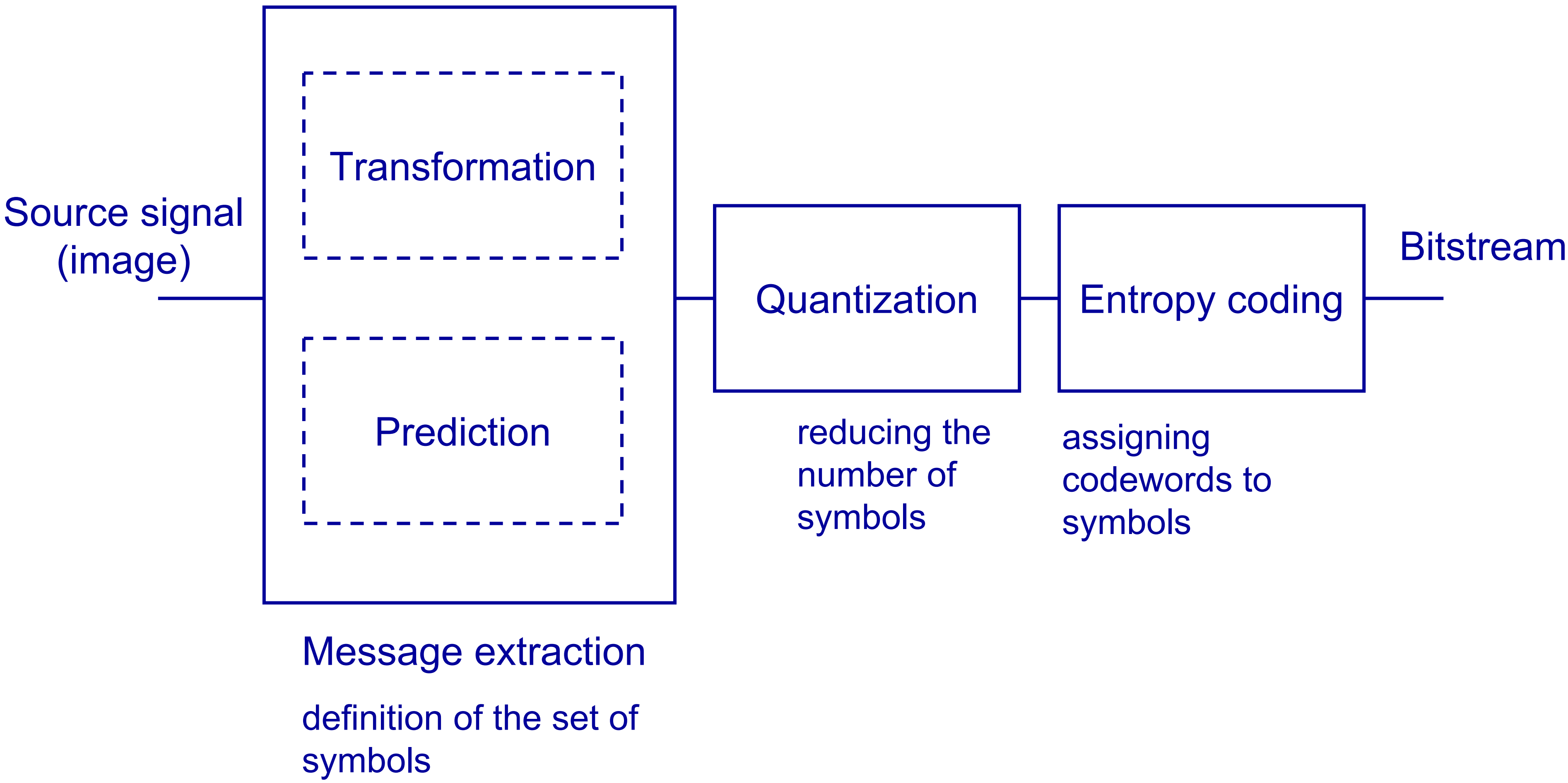
- *Adaptive* arithmetic coding
  - The probability tables for the different symbols can be made adaptive to the source statistics and updated during encoding

Arithmetic coding

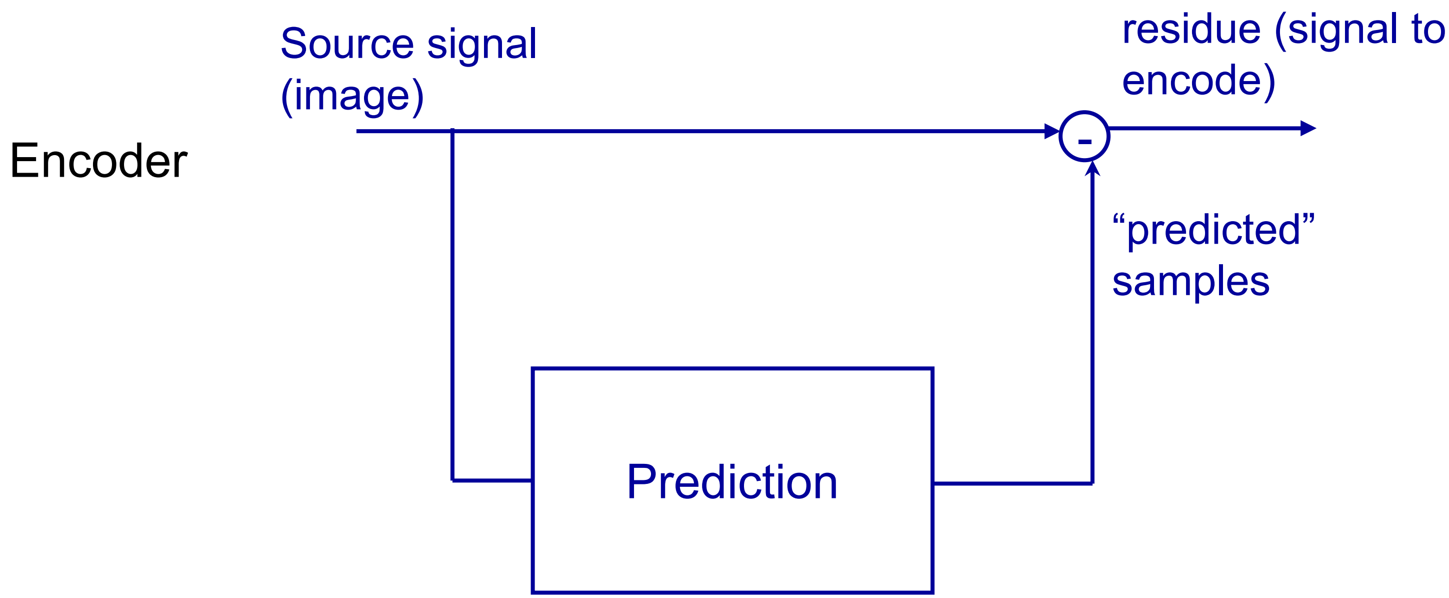
- Features
  - Does not require integer length codes
  - Encodes sequences of symbols
  - Each sequence is represented as an interval included in [0,1]
  - The longer the sequence, the smaller the interval and the larger the number of bits needed to specify the interval
  - The average bit rate asymptotically tends to the entropy lower bound when the sequence length increases

  - On average, performs better than Huffman coding
  - Moderate complexity
  - Used in JPEG2000

Coding systems



Prediction based coding

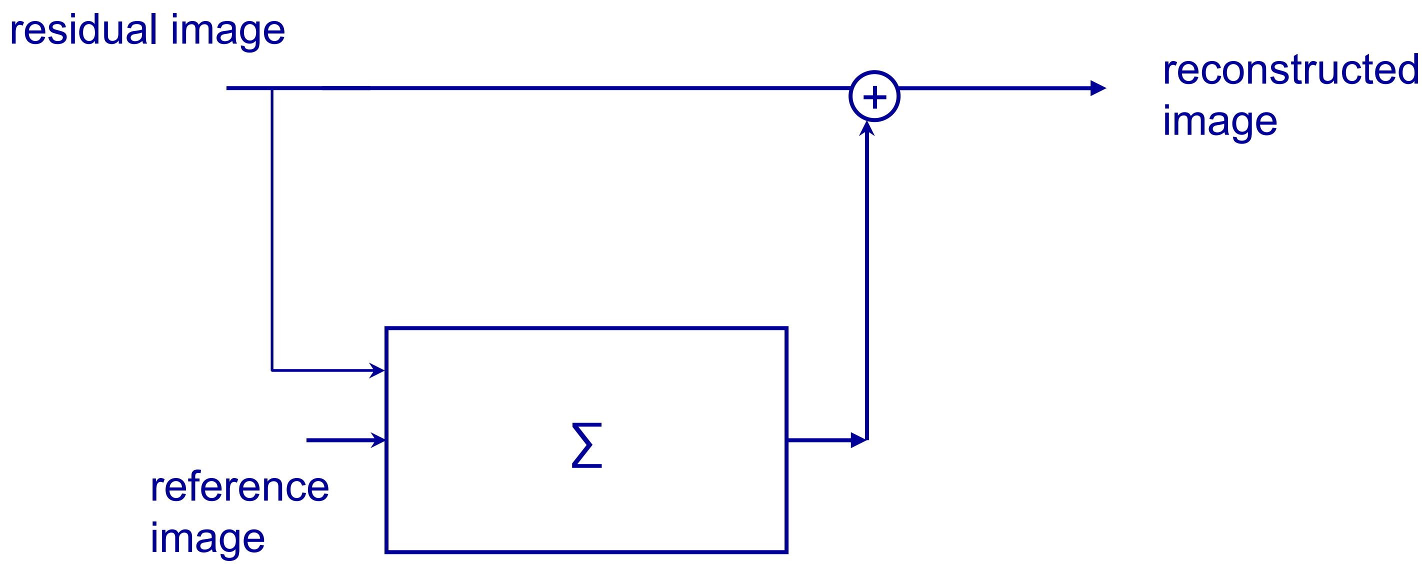


The value of the samples are estimated according to a predefined rule and the resulting values are **subtracted** from the corresponding ones in the original image to obtain the **residual** (or error) image. This last one is then quantized and entropy coded.

- Still images → spatial (intra-frame) prediction
- Image sequences → temporal (inter-frame) prediction

Prediction based coding

Decoder



- Still images (JPEG lossless)
- Image sequences : motion compensation (MPEG4)

Intra-frame *linear* prediction

12	34	27	42
21	3	44	1
12	34	27	42

A	B	C
D	X	

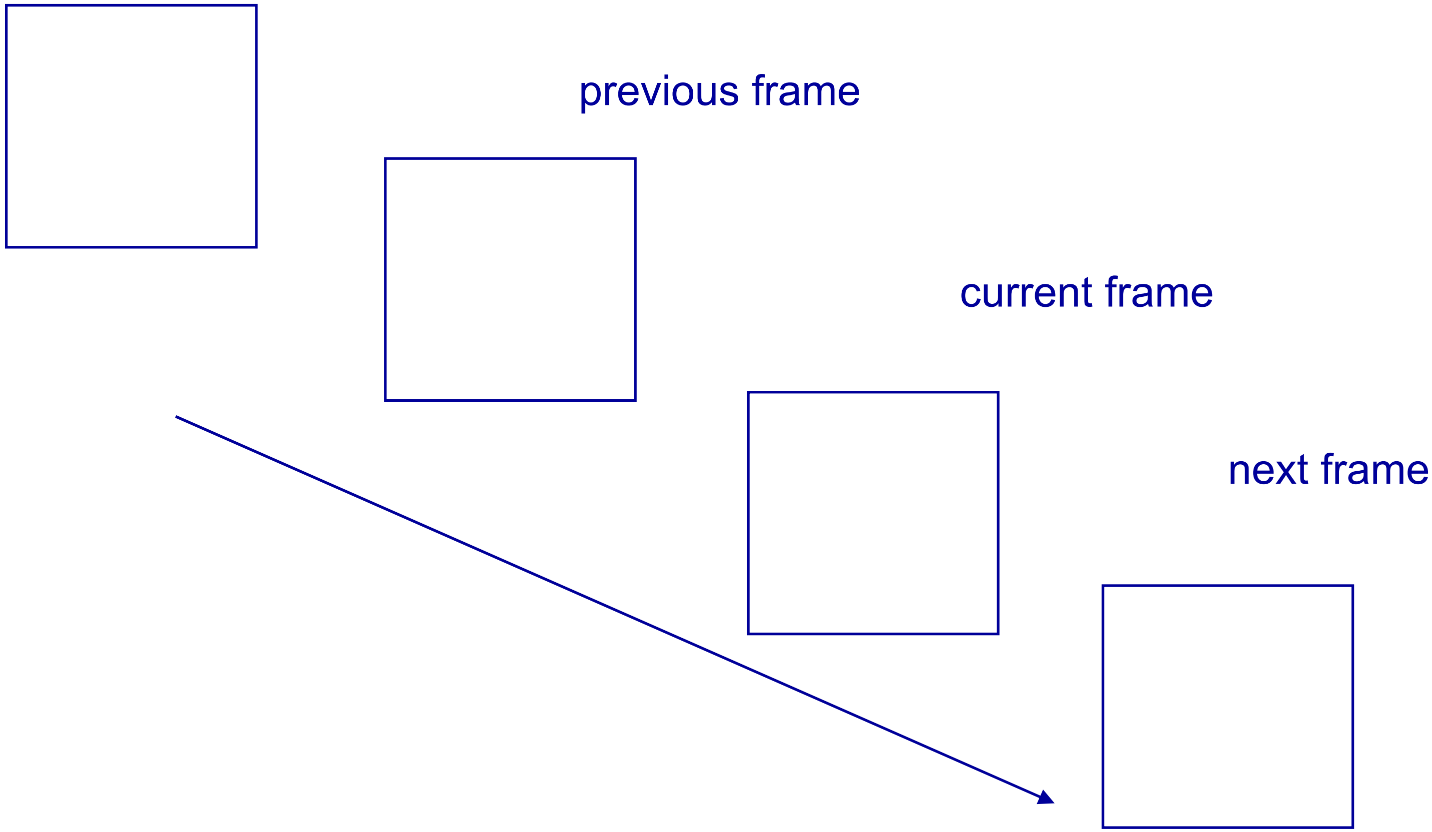
$X_{est} = aA+bB+cC+dD$

$E = X - X_{est}$

symbol to predict or estimate

The error image is quantized and entropy encoded. At the receiver, it is decoded and used to recover the original image.

Inter-frame prediction



Transform based coding

- Given the source signal, il can be convenient to project the data to a different domain to improve compression $\Rightarrow$ transformation
  - Discrete Cosine Transform (DCT), used in JPEG
  - Discrete Wavelet Transform (DWT), used in JPEG2000
- The transformed coefficients are then to be quantized for mapping to a finite set of symbols
- Such symbols can also be mapped to another set of symbols to further improve compression performance
  - Embedded Zerotree Wavelet based coding (EZW)
  - Layered Zero Coding (LZC)
  - Multidimensional LZC (for volumetric data, after a 3D DWT)

Transform based coding

- Consider the signal as a r.v. of N samples: Y[n]
- Project it to an (orthonormal) basis

$$Y=\sum_m A[m] g_m$$

$$A[m]= \langle Y, g_m \rangle$$

- The coefficients A[m] are quantized and then encoded

$$A_Q[m]=Q\{A[m]\}$$

Reconstructed signal (after entropy decoding)

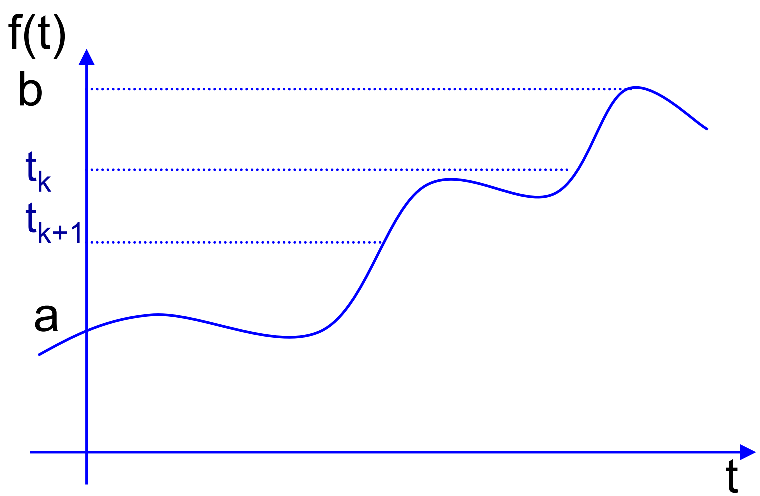
$$Y_{dec}=\sum_m A_Q[m] g_m$$

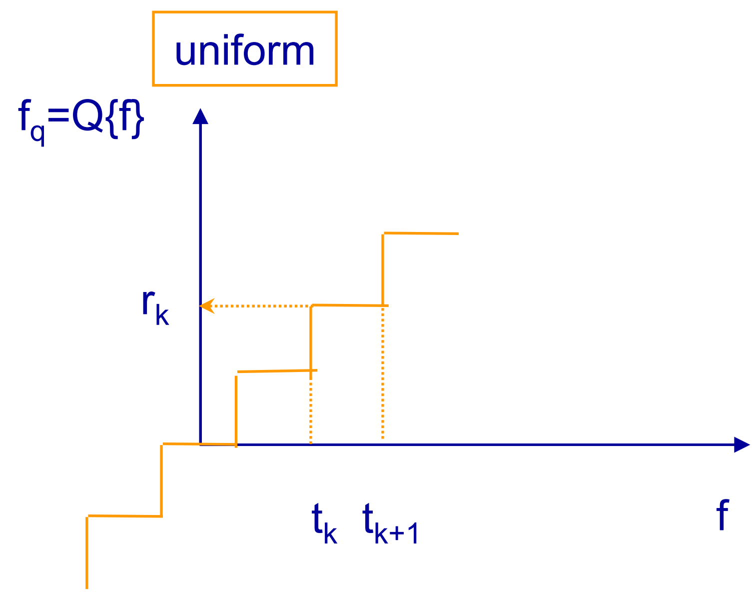
  – With quantization, the decoded signal is an approximation of the original signal and the degree of distortion depends on the strength of the quantization

Quantization

- A/D conversion ⇨ quantization

$f[n]$ in $L^2(Z)$ → Quantizer → discrete function $f_q[n]$ in $L^2(Z)$





Scalar quantization

- A scalar quantizer Q approximates X by $\tilde{X}=Q(X)$, which takes its values over a finite set.
- The quantization operation can be characterized by the MSE between the original and the quantized signals

$$d = E\{(X-\tilde{X})^2\}.$$

- Suppose that X takes its values in $[a, b]$, which may correspond to the whole real axis. We decompose $[a, b]$ in K intervals $\{(y_{k-1}, y_k]\}_{1 \le k \le K}$ of variable length, with $y_0=a$ and $y_K=b$.
- A scalar quantizer approximates all $x \in (y_{k-1}, y_k]$ by x_k:

$$\forall x \in (y_{k-1}, y_k], \qquad Q(x) = x_k$$

Scalar quantization

- The intervals $(y_{k-1}, y_k]$ are called *quantization bins*.
- Rounding off integers is an example where the quantization bins

$$(y_{k-1}, y_k]=(k-1/2, k+1/2]$$

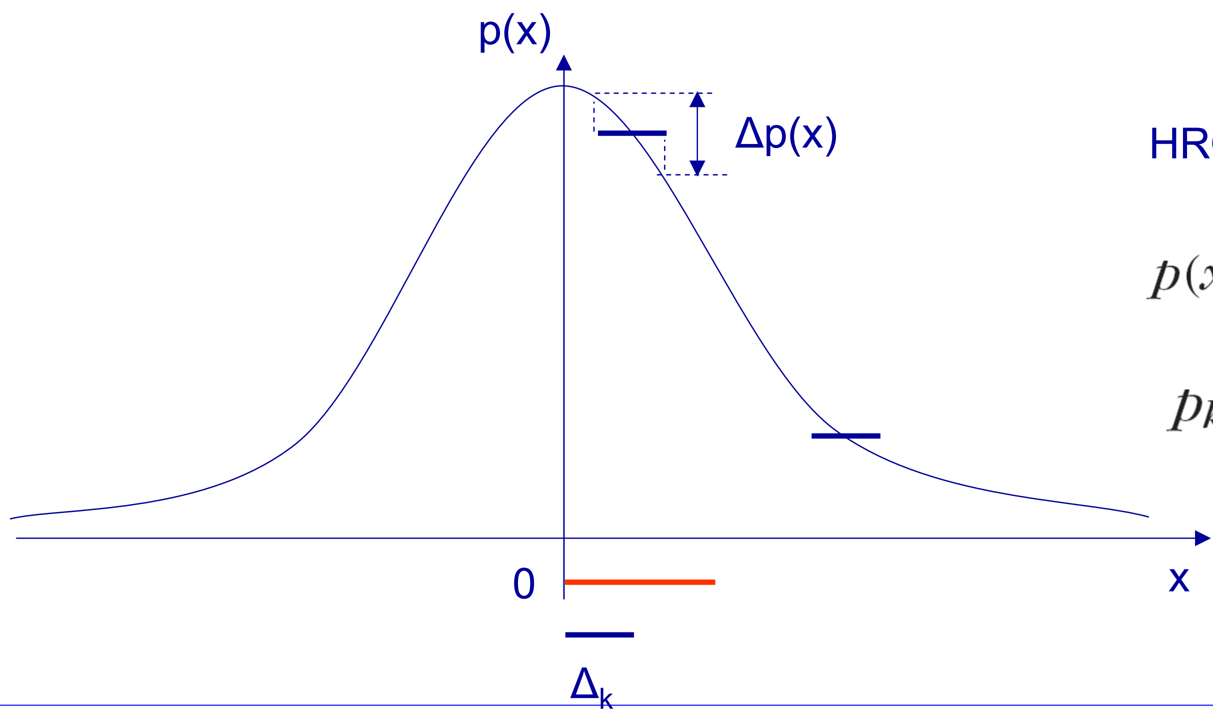
have size 1and $x_k=k$ for any $k\in Z$.

- High resolution quantization
  - Let $p(x)$ be the probability density of the random source X. The mean-square quantization error is

$$d = E\{(X-\tilde{X})^2\} = \int_{-\infty}^{+\infty} \left(x - Q(x)\right)^2 p(x)\, dx.$$

HRQ

– A quantizer is said to have a *high resolution* if *p(x)* is *approximately constant* on each quantization bin. This is the case if the sizes *k* are sufficiently small relative to the rate of variation of *p(x)*, so that one can neglect these variations in each quantization bin.



HRQ: $\Delta p(x) \to 0$

$$p(x) = \frac{p_k}{\Delta_k} \quad \text{for } x \in (y_{k-1}, y_k],$$

$$p_k = \Pr\{X \in (y_{k-1}, y_k]\}.$$

Scalar quantization

- Teorem 10.4 (Mallat): For a high-resolution quantizer, the mean-square error *d* is minimized when $x_k=(y_k+y_{k+1})/2$, which yields

$$d = \frac{1}{12}\sum_{k=1}^{K} p_k \Delta_k{}^2$$

Proof. The quantization error (10.15) can be rewritten as

$$d = \sum_{k=1}^{K} \int_{y_{k-1}}^{y_k} (x - x_k)^2\, p(x)\, dx.$$

Replacing $p(x)$ by its expression (10.16) gives

$$d = \sum_{k=1}^{K} \frac{p_k}{\Delta_k} \int_{y_{k-1}}^{y_k} (x - x_k)^2\, dx. \qquad \textbf{(10.18)}$$

One can verify that each integral is minimum for $x_k = (y_k + y_{k-1})/2$, which yields (10.17). ■

Uniform quantizer

The uniform quantizer is an important special case where all quantization bins have the same size

$$y_k - y_{k-1} = \Delta \quad \text{for} \quad 1 \leqslant k \leqslant K.$$

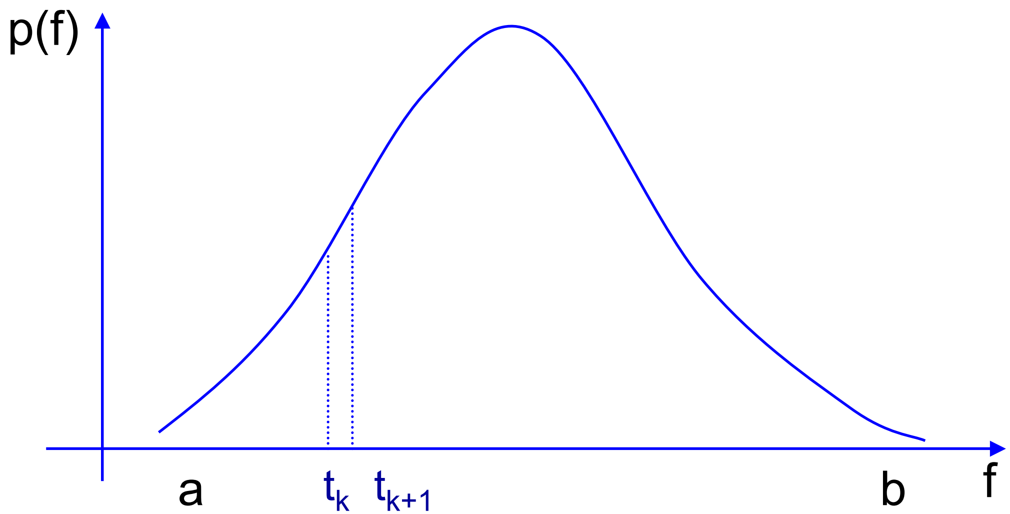
For a high-resolution uniform quantizer, the average quadratic distortion (10.17) becomes

$$d = \frac{\Delta^2}{12} \sum_{k=1}^{K} p_k = \frac{\Delta^2}{12}. \tag{10.19}$$

It is independent of the probability density $p(x)$ of the source.

High resolution quantization

- Definition: A quantizer is said to be high resolution if $p(f)$ is approximately constant on each quantization bin of size δ_k
  - $p(f)$ is the pdf of the random variable f

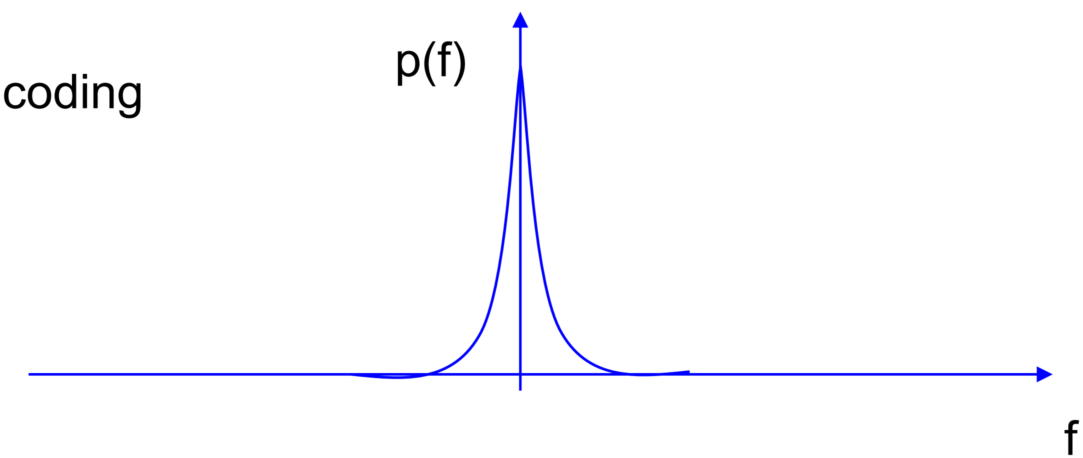


Such an hypothesis is in general NOT true for low bit-rate coding (high compression rates) where the size of the quantization bin is large with respect to the pdf of the quantized variable

Low bit rate coding

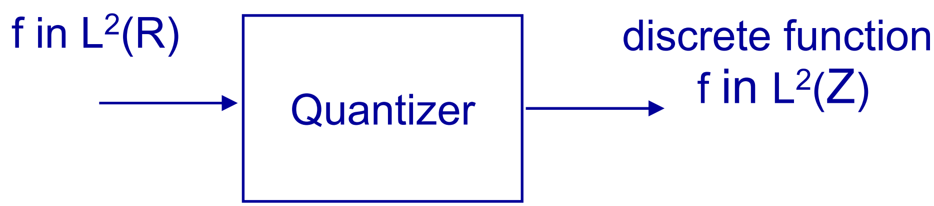
- The quantization step is large $\rightarrow$ many quantized coefficients are set to zero
- The zero-bin interval [-T,T] corresponds to the threshold for significance of the coefficients at the considered precision (level of quantization)
- Efficient coding can be obtained by splitting the encoding phase in two successive steps:
  - Encoding of the positions of the zero and no-zero coefficients (significance map)
  - Encoding of the amplitude of the no-zero (significant) coefficients

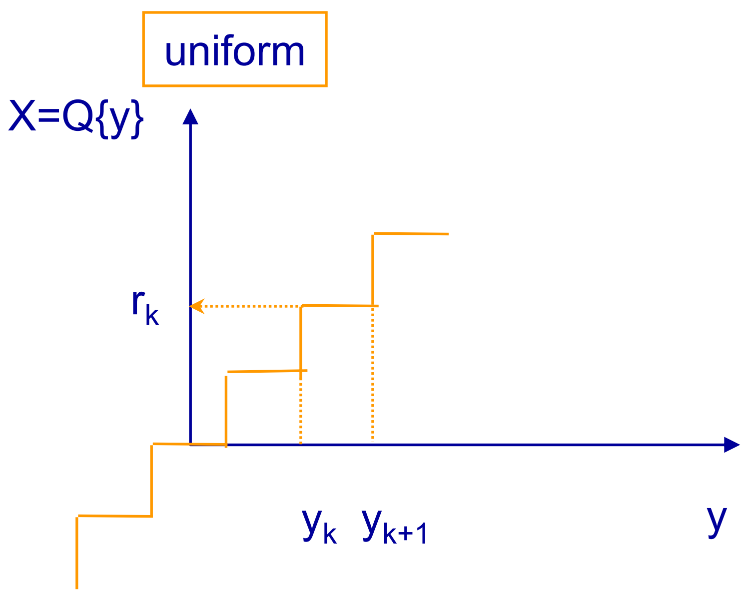
Wavelet-based coding

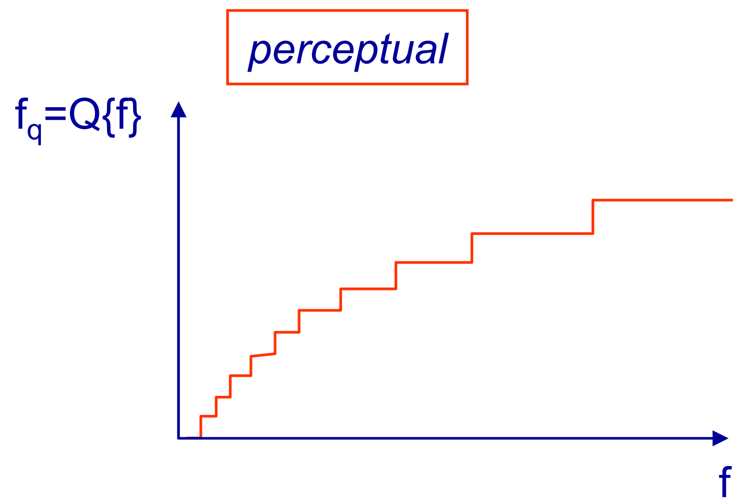


Quantization

- A/D conversion ⇨ quantization







The sensitivity of the eye decreases increasing the background intensity (Weber law)

Quantization revisited

- To analyze the error due to quantization we need a measure for the distortion

$$D=E\{||Y-Y_Q||^2\}= \sum_m E\{||A[n]-A_Q[n]||^2\}$$

- The distortion depends on the resolution of the quantization (the quantization step size), which rules the number of bits needed to represent the quantized coefficients. This gives an intuition of the functional relation between D and R: D=D(R)
- **Design of the quantizer**. Under the assumption of *high resolution quantization*
  - The **RMS** value of the distortion D is **minimized** when the reconstruction level is the average of the bin boundary values

$$\tilde{f}_{q,k} = \frac{t_k + t_{k-1}}{2}$$

  - **D(R)** is **minimal** for ***uniform scalar quantization*** and given by

$$D(R)=\Delta^2/12= \sigma^2 2^{-2R}$$

Δ being the quantization step size and σ the source variance

Quantization

original

5 levels

10 levels

50 levels

Embedded Coding

Part 2

Embedded transform coding

- For rapid transmission or fast image browsing, one should quickly provide a *coarse image version* which is progressively enhanced as more bits are received and decoded
- Guideline: The decomposition coefficients are *sorted* and the *most significant bits* of the *largest coefficients* are sent first
- The *embedding* of the information is obtained by a *Successive Approximation Quantization (SAQ)* strategy
  1. Set an initial threshold T
  2. Scan the coefficients to get the *significance map* (SM(T))
  3. Encode the SM(T) by entropy coding
  4. Encode the *amplitude* of the significant coefficients (at the current precision set by T)
  5. Halve the threshold: T-> T/2
  6. If threshold > 1 go back to point 2

Embedded transform coding

- The subband coefficients are quantized uniformly with step 2^n which is progressively reduced in the following scans
- The largest value for the threshold is chosen to obtain at least one no zero symbol
- The information on the sign of the significant coefficients is enclosed in the significance map
  - Possible choice for the symbols in the significance map:

$$
\begin{array}{lll}
b_m(p,q) = 0 & \text{if} & |a_m(p,q)| \leq T \\
b_m(p,q) = 1 & \text{if} & a_m(p,q) > T \\
b_m(p,q) = 2 & \text{if} & a_m(p,q) < -T
\end{array}
$$

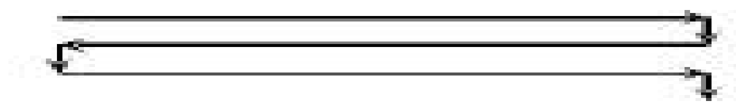
Encoding the significance map

- Significant coefficient
  - Any coefficient $|a_m(p,q)|>T$ which is NOT quantized to zero
- Significance map
  - Binary image whose values $b_m(p,q)$ are defined as follows
  - $b_m(p,q) = 0$ if $|a_m(p,q)| \leq T$
  - $b_m(p,q) = 1$ if $|a_m(p,q)|>T$

- The significance map can then be encoded by
  - Run-lenght coding
    - Store in the random variables Z and I the length of the sequences of zeros and ones and encode such symbols via an entropy coder (Huffman or Arithmetic)
  - More complex algorithms (Zerotrees)
    - **Link** the appearance of zeros across scales to obtain new symbols which summarize the significance of a tree of coefficients at a time, improving the efficiency of the entropy coder

Run-length coding

- Every code word is made up of a pair (g, l) where g is the gray level, and l is the number of pixels with that gray level (length, or “run”).
- E.g.,
    - 56 56 56 82 82 82 83 80
    - 56 56 56 56 56 80 80 80
  - creates the run-length code (56, 3)(82, 3)(83, 1)(80, 4)(56, 5).
- The code is calculated row by row.



- Very efficient coding for binary data.

Run-length coding

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 3 & 3 & 3 \\ 0 & 1 & 1 & 3 & 3 & 3 & 4 & 4 \\ 0 & 1 & 3 & 3 & 5 & 5 & 4 & 4 \\ 0 & 2 & 3 & 3 & 5 & 5 & 5 & 4 \\ 0 & 0 & 2 & 3 & 3 & 4 & 6 & 6 \\ 0 & 0 & 0 & 2 & 2 & 3 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

gray level	0	1	2	3	4	5	6
#pixels	26	5	5	13	8	5	2

Run-length coding

row #	run-length code (gl,rl)
0	(0,8)
1	(0,2), (1,2), (2,1), (3,3)
2	(0,1), (1,2), (3,3), (4,2)
3	(0,1), (1,1), (3,2), (5,2), (4,2)
...	...
7	(0,8)

Run-length coding

row #	binary code
0	000 111
1	000 001 001 001 010 000 011 010
2	000 000 001 001 011 010 100 001
...	...
7	000 111

Compression Achieved

Original image requires 3 bits per pixel (in total - 8x8x3=192 bits).

Compressed image has 29 runs and needs 3+3=6 bits per run (in total - 174 bits or 2.72 bits per pixel).

SM: Encoding the amplitude

- The amplitude of the **significant** coefficients is **uniformly** quantized with step Δ and entropy coded (Huffman or Arithmetic)
  - The coefficients in a given subband (j,k) are random variables for which a pdf can be defined and exploited for entropy coding
- Example

T=2

1	3	4	0
5	2	7	8
4	5	1	1
0	2	4	3

significance map

0	1	1	0
1	0	1	1
1	1	0	0
0	0	1	1

0 1 1 0 1 0 1 1 1 1 0 0 0 0 1 1 sequence of symbols

1 2 1 1 1 4 4 2 $\Rightarrow$ entropy coding

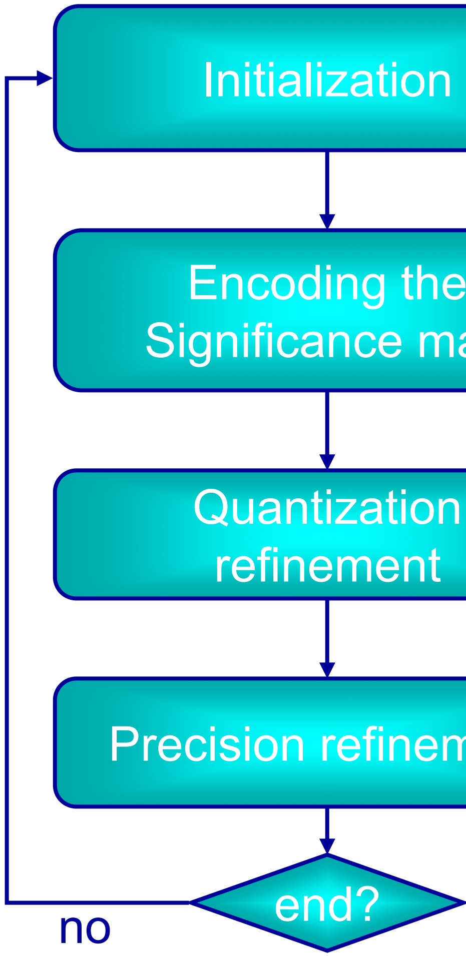
realizations of the rv Z

realizations of the rv I

ETC algorithm

- 1. Initialization
  - Set the initial value of the threshold to the first power of two greater than the largest subband value (magnitude)
- 2. Significance map
  - Store the **significance map** and the **sign** of the no zero coefficients
- 3. Quantization refinement
  - **Update** the values of the coefficients that were already classified as significant during the previous steps
- 4. Precision refinement
  - Halve the threshold value and go back to point 2.

Embedded Transform Coding



$n=\lfloor \sup_m \log_2 |a[m]| \rfloor$

Exploitation of *residual correlation among subband coefficients*

→ *Layered Zero* Coding (LZC)

→ *Zerotree* Coding (EZW)

Update the value of the significant coefficients

Decrease the quantization bin: $n \rightarrow n-1$

Embedded Zerotree Wavelet (EZW) Coder

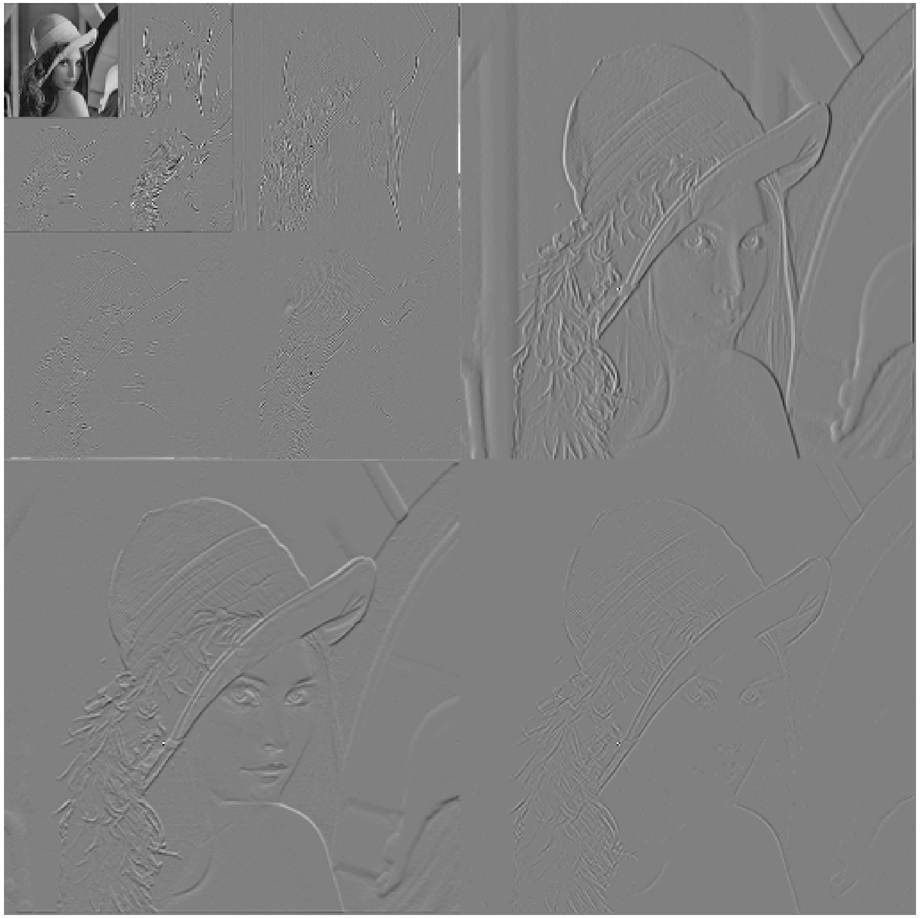
- A quantization and coding strategy
- Incorporates characteristics of wavelet decomposition
- Outperform some generic approach
- Fundamental concept of other wavelet-based coder
- Can be decomposed into two parts:
  - Significant map coding using zerotree
  - Successive approximation quantization

55

EZW – basic concepts

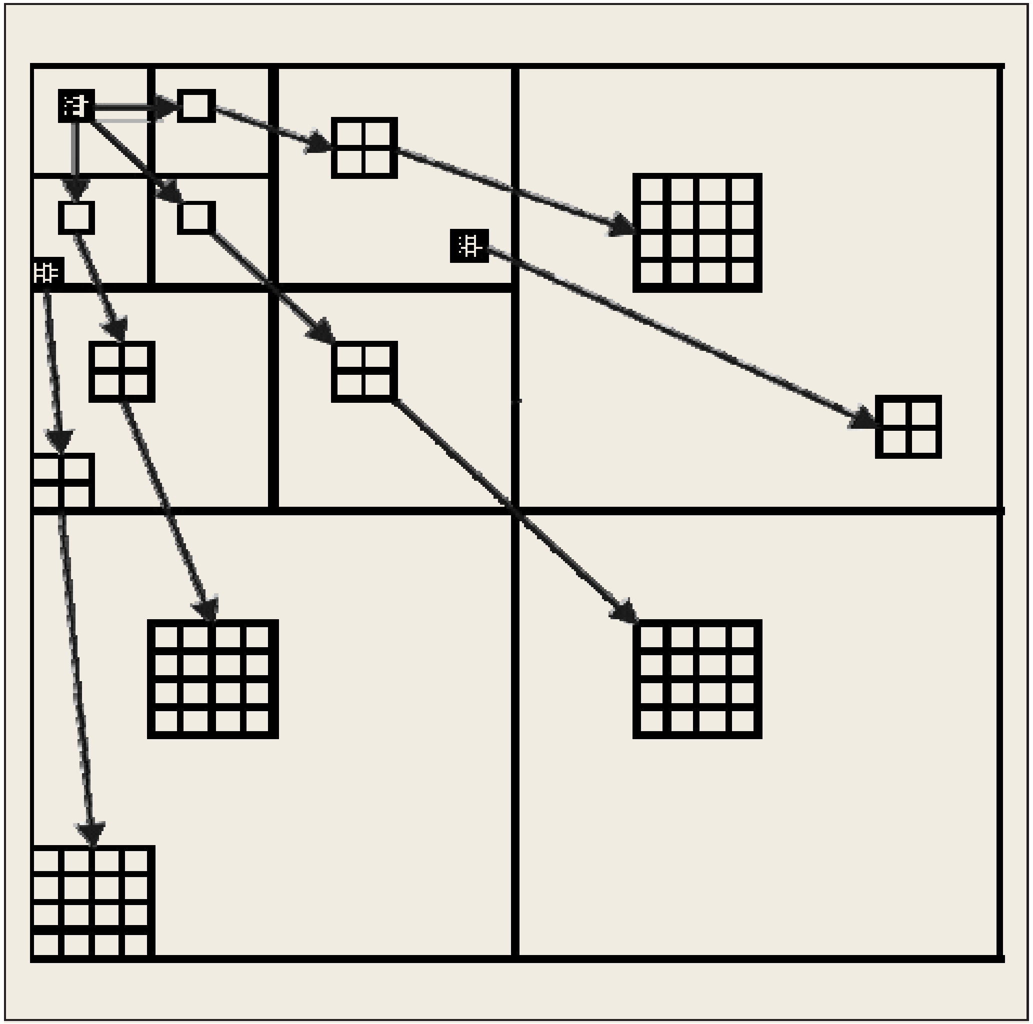
- The definition of the zero-tree:
  - There are coefficients in different subbands that represent the same spatial location in the image and this spatial relation can be depicted by a quad tree except for the root node at top left corner representing the DC coefficient which only has three children nodes.
- Zero-tree Hypothesis
  - If a wavelet coefficient c at a coarse scale is insignificant with respect to a given threshold T, i.e. $|c|<T$ then all wavelet coefficients of the same orientation at finer scales are also likely to be insignificant with respect to T.
- Successive Approximations Quantization (SAQ)
  - A refinement process
  - Multi-pass scanning of coefficient using successive decreasing threshold

Embedded Zerotree Wavelet-based coder



…. look at the notes…

Significant Map Coding Using Zerotree



Four types of Label

1. Positive significant
2. Negative significant
3. Isolated zero
4. Zero tree root

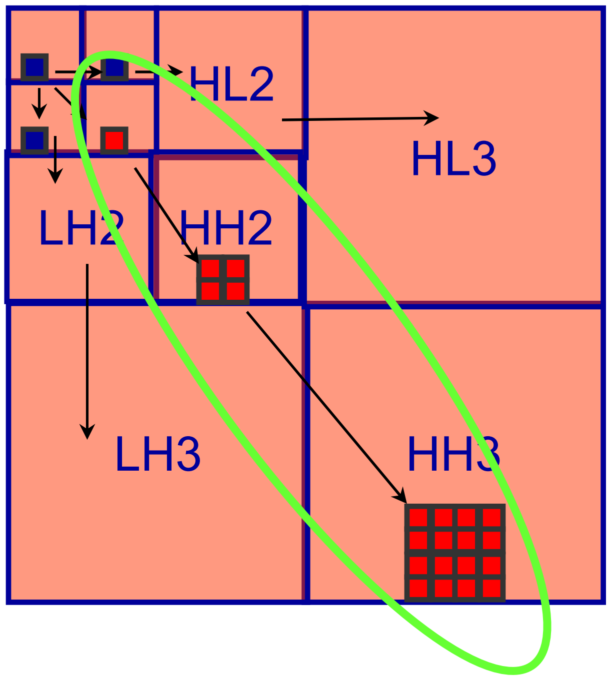
For each coefficient:
Give a label based on predefine threshold T

$$T_0 = 2^{\lfloor \log_2 x_{\max} \rfloor}$$

58

EZW

2D



→ trees of zeros ↔ *zerotrees*

Significance map:

$$b_j^k[p,q] = \begin{cases} 1 & \text{if } 2^n \le d_j^k[p,q] < 2^{n+1} \\ -1 & \text{if } -2^{n+1} < d_j^k[p,q] \le -2^n \\ 0 & \text{otherwise} \end{cases}$$

- Encoding the SM

  inter-band dependencies ⇒quad-trees

  Primary pass ⇒ ZTR, IZ, POS,NEG

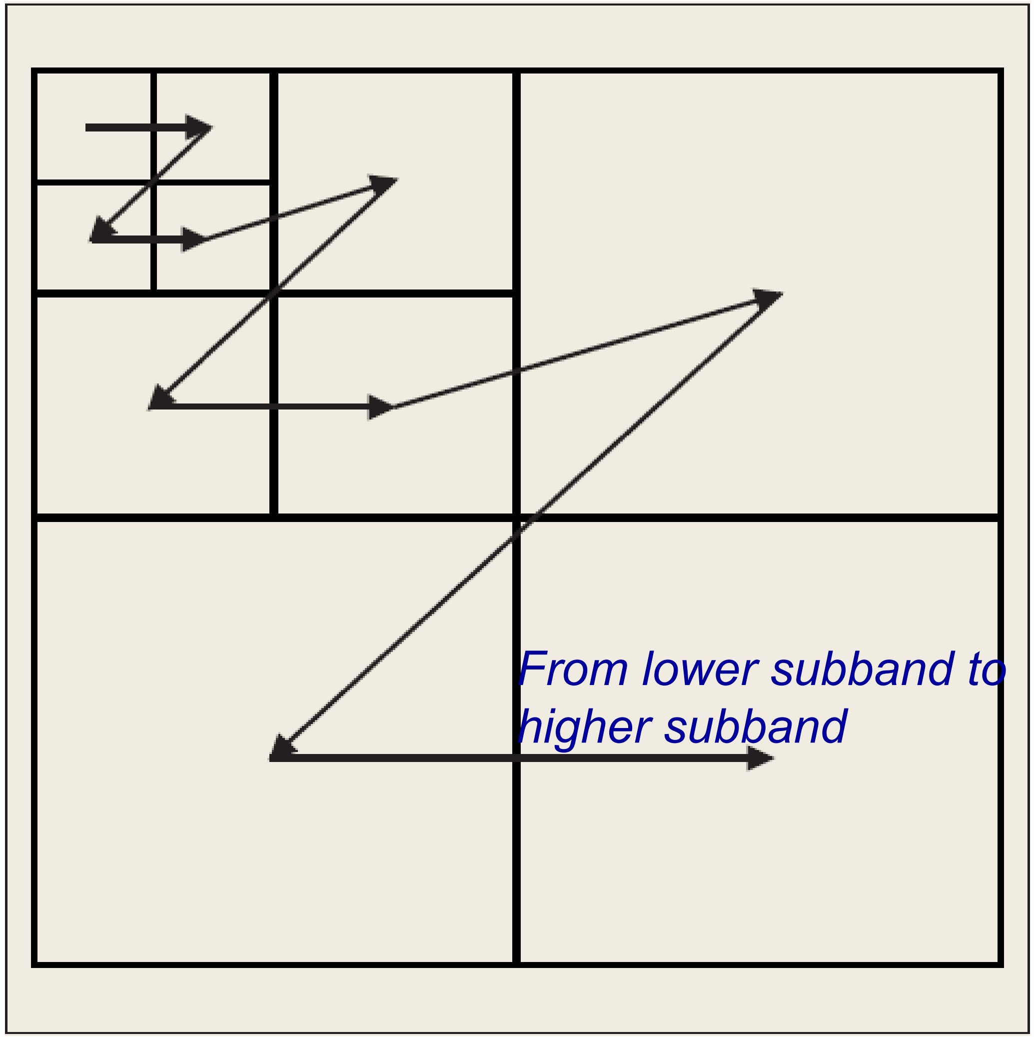
  ZTR:1/3(4^j-1) symbols

- Quantization refinement

  Secondary pass ⇒ HIGH,LOW

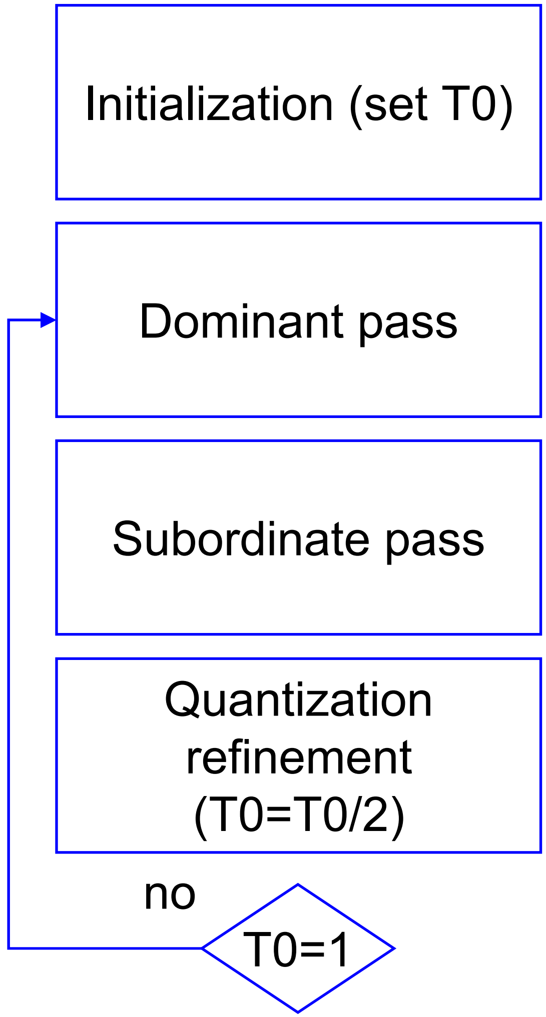
Significant Map Coding Using Zerotree

- Scan order :



60

EZW algorithm



Assigns symbols POS, NEG, IZ, ZTR to coefficients
Replaces POS and NEG coefs with zeros and adds their values in a secondary list and assigns them a reconstruction value equal to the mid point of the current uncertainty interval

Refines the values assigned to POS and NEG changing the reconstruction value to the mid point of either the upper or the lower subinterval (symbols 1 and 0, respectively)

EZW – the algorithm

- In the dominant_pass
  - All the coefficients are scanned in a special order
  - If the coefficient is a zero tree root, it will be encoded as ZTR. All its descendants don't need to be encoded – they will be reconstructed as zero at this threshold level
  - If the coefficient itself is insignificant but one of its descendants is significant, it is encoded as IZ (isolated zero).
  - If the coefficient is significant then it is encoded as POS (positive) or NEG (negative) depends on its sign.

This encoding of the zero tree produces significant compression because gray level images resulting from natural sources typically result in DWTs with many ZTR symbols. Each ZTR indicates that no more bits are needed for encoding the descendants of the corresponding coefficient

EZW – the algorithm

- At the end of dominant_pass
  - all the coefficients that are in absolute value larger than the current threshold are *extracted and placed without their sign* on the subordinate list and their positions in the image are filled with zeroes. This will prevent them from being coded again.
- In the subordinate_pass
  - All the values in the subordinate list are *refined*. this gives rise to some juggling with uncertainty intervals and it outputs next most significant bit of all the coefficients in the subordinate list.

EZW – the algorithm

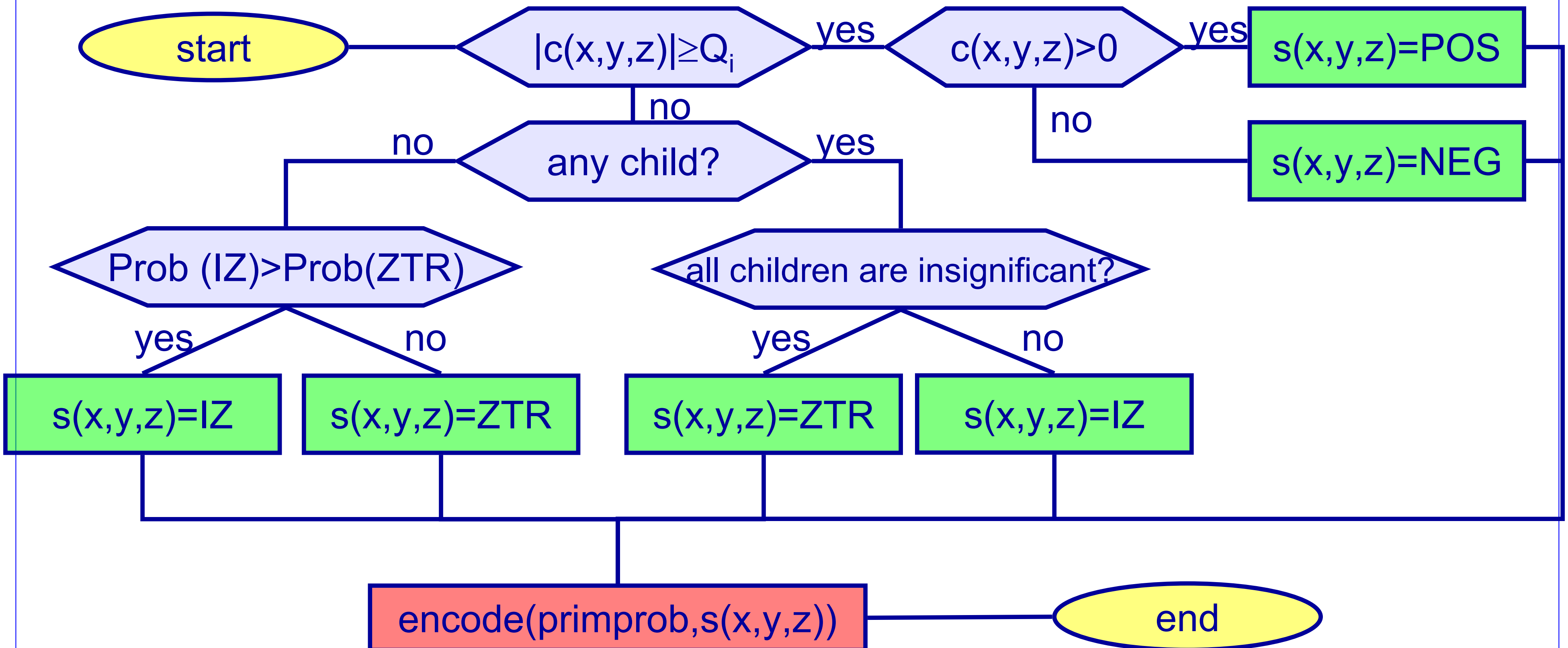
The main loop ends when the threshold reaches a minimum value, which could be specified to control the encoding performance, a "0" minimum value gives the lossless reconstruction of the image

The initial threshold t0 is decided as:

$$t_0 = 2^{\lfloor \log_2(MAX(|\gamma(x,y)|)) \rfloor}$$

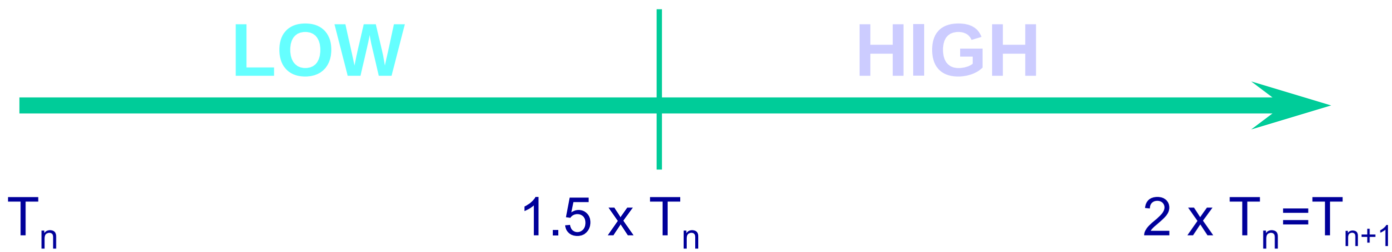
Here MAX() means the maximum coefficient value in the image and $\gamma(x,y)$ denotes the coefficient. With this threshold we enter the main coding loop

EZW : Dominant Pass



EZW : Subordinate Pass

- Concerns significant coefficients
- Refines the value of the significant coefficients by setting the resolution at the current quantization level

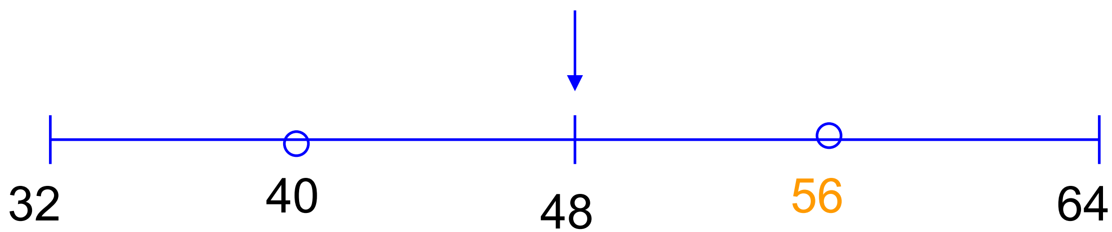


Example

Dominant pass

T0=32
x=63
uncertainty interval=[32,64]

reconstruction value at the end of the dominant pass



Subordinate pass

63>48 → HIGH (symbol=1)

reconstruction value after the subordinate pass

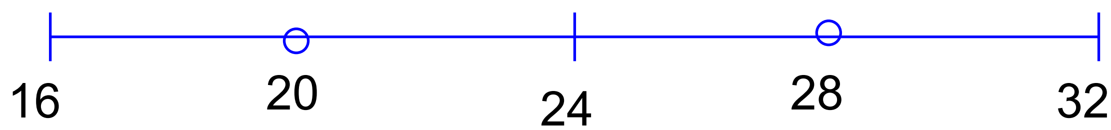
update: at the beginning of the *2° dominant pass* the 63→32 (previous T0 value), so that the **value that goes in the list is 63-32=31**. This is refined as in the next page

Example

Dominant pass

T0=16
x=63 looks like 0 for primary pass (can become a ZTR!)
uncertainty interval=[16,32]

this dominant pass does not concern x!



Subordinate pass

31>24 → HIGH (symbol=1)
set of symbols assigned by the subordinate passes 1,1
reconstruction value after the subordinate pass: 32+28=60

update: value that goes into the list: **63-(32+16)=15**

....

Example

- T0=32
- End of 1° dominant pass: 48
- End of 1° subordinate pass: 56 first value seen by the decoder
- Update: new value in the list to be refined: 63-32=31
- T0=16
- End of 2° dominant pass: -----------
- End of 2° subordinate pass: 2° update: 31-> 28
- New value seen by the decoder: 32+28=60
- Update: new value in the list to be refined: 63-32-16=15
- T0=8
- End of 3° dominant pass: -----------
- End of 3° subordinate pass: 3° update: 15->14
- New value seen by the decoder: 32+16+14=62
- Update: new value in the list to be refined: 63-32-16-8=7 …….
- …….
- Final value seen by the decoder: 32+16+8+4+2+1=63

Algorithm

```
threshold = initial_threshold; do
{
dominant_pass(image);
subordinate_pass(image);
threshold = threshold/2;
}
while (threshold > minimum_threshold);
```

Dominant pass

```
/* * Dominant pass */
initialize_fifo();
while (fifo_not_empty)
{
get_coded_coefficient_from_fifo();
if coefficient was coded as P, N or Z then
{
code_next_scan_coefficient();
put_coded_coefficient_in_fifo();
if coefficient was coded as P or N then
{
add abs(coefficient)  to subordinate list;
set coefficient position to zero; } } }
```

Subordinate pass

```
/* * Subordinate pass */
subordinate_threshold = current_threshold/2;
for all elements on subordinate list do {
if (coefficient > subordinate_threshold) {
output a one;
coefficient = coefficient-subordinate_threshold;
}
else output a zero;
}
```

EZW Example (1/2)

T0 = 32

53	–22	21	–9	–1	8	–7	6
14	–12	13	–11	–1	0	2	–3
15	–8	9	7	2	–3	1	–2
34	–2	–6	10	6	–4	4	–5
–6	5	–1	1	1	3	–1	5
6	1	3	0	–2	2	6	0
4	2	1	–4	–1	0	–1	4
0	–2	7	5	–3	2	–2	3

▲ *17. An example three-level wavelet decomposition used to demonstrate the EZW algorithm.*

*	–22	21	–9	–1	8	–7	6
14	–12	13	–11	–1	0	2	–3
15	–8	9	7	2	–3	1	–2
*	–2	–6	10	6	–4	4	–5
–6	5	–1	1	1	3	–1	5
6	1	3	0	–2	2	6	0
4	2	1	–4	–1	0	–1	4
0	–2	7	5	–3	2	–2	3

▲ *18. The example wavelet transform after the first dominant pass. The symbol * is used to represent symbols found to be significant on a previous pass.*

EZW Example (2/2)

T0 = 32

Table 3. Resulting Output of the First Dominant Pass ($T_0 = 32$).

Subband	Coefficient Value	Symbol	Recon-struction Value	Comment (See Text)
LL_3	53	*ps*	48	1)
HL_3	–22	*ztr*	0	2)
LH_3	14	*iz*	0	3)
HH_3	–12	*ztr*	0	
LH_2	15	*ztr*	0	
LH_2	–8	*ztr*	0	
LH_2	34	*ps*	48	
LH_2	–2	*ztr*	0	
LH_1	4	*iz*	0	
LH_1	2	*iz*	0	
LH_1	0	*iz*	0	
LH_1	–2	*iz*	0	

Table 4. Resulting Output of the Subordinate Pass.

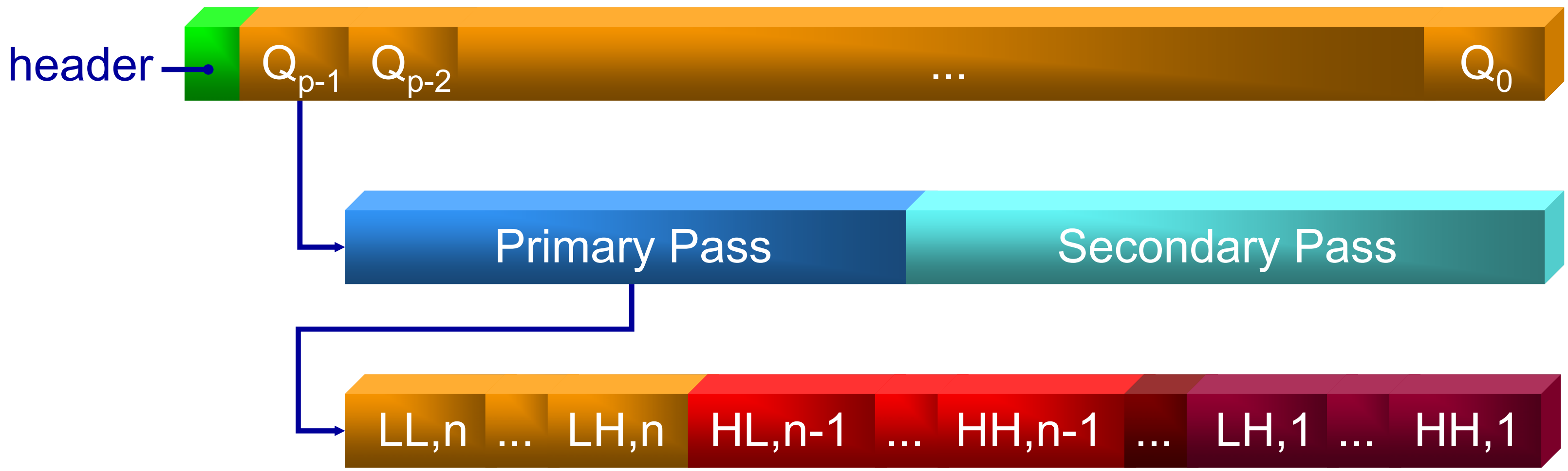
Coefficient Magnitude	Symbol	Reconstruction Magnitude
53	1	56
34	0	40

After this two step, we finish one iteration.

Ti = Ti/2(reduce the threshold)

Repeat until target fidelity or bit-rate is achieve

Bitstream



p=total number of bitplanes

The Limitations of EZW algorithm

- It is not possible to encode sub-images because the entire image must be transformed before the encoding can start.
- EZW algorithm is computational expensive

Layered Zero Coding

- Proposed by Taubman and Zakhor in 1994 [Multirate 3D subband coding of video]
- Idea: multirate coding of subbands
- Advantages
  - Large control over birate granularity
  - Lower computational complexity than EZW
- Basic idea: Progressive quantization and coding of each subband in a sequence of N layers representing progressively finer quantization step sizes
  - N quantizers: Q1 (rougher), …., Qn (finer)
  - L quantization layers: L1, L2,…., Ln
- Gudelines:
  - Each quantizer operates on the subband samples and produces a sequence of symbols. The symbols for quantizer Q1 are encoded into layer L1
  - The information necessary to recover symbols for quantizer Qn, given the symbols for quantizers Qn-1,…,Q1 are already known, is encoded into layer Ln

LZC

- **Thus**, the decoder is able to recover the subband samples as encoded by any quantizer Qn by decoding layers L1,…Ln only.
- **Constraint** for coding gain: the total number of bits required to encode the layers L1,…,Ln be approximately the same as the number of bits required to encode the output of quantizer Qn alone. If this condition is satisfied the multirate property is obtained without sacrificing coding efficiency.
- **How**: exploiting dependencies among quantization layers and/or subbands
  - Statistical dependencies among quantization layers
  - Statistical dependencies among spatially/temporally adjacent subbands
  - Statistical dependencies among hierarchies of subband coefficients
  - Exploiting the presence of large number of zeros in the subbands

LZC

- It can be proved that the coding efficiency condition is met if the set of quantizers satisfy the following condition

$$Q_n(x) = k, \text{ for } x \in I_{n,k}$$

$$P\left(X \in I_{n,Q_n(x[i])} \setminus I_{n-1,Q_{n-1}(x[i])}\right) = 0, \forall x[i], \forall n \geq 2$$

- Interpretation: every quantization interval of Qn is contained in some quantization interval of Qn-1
- Furthermore, arithmetic coding must be used to encode each quantization layer Ln

LZC: design of quantizers

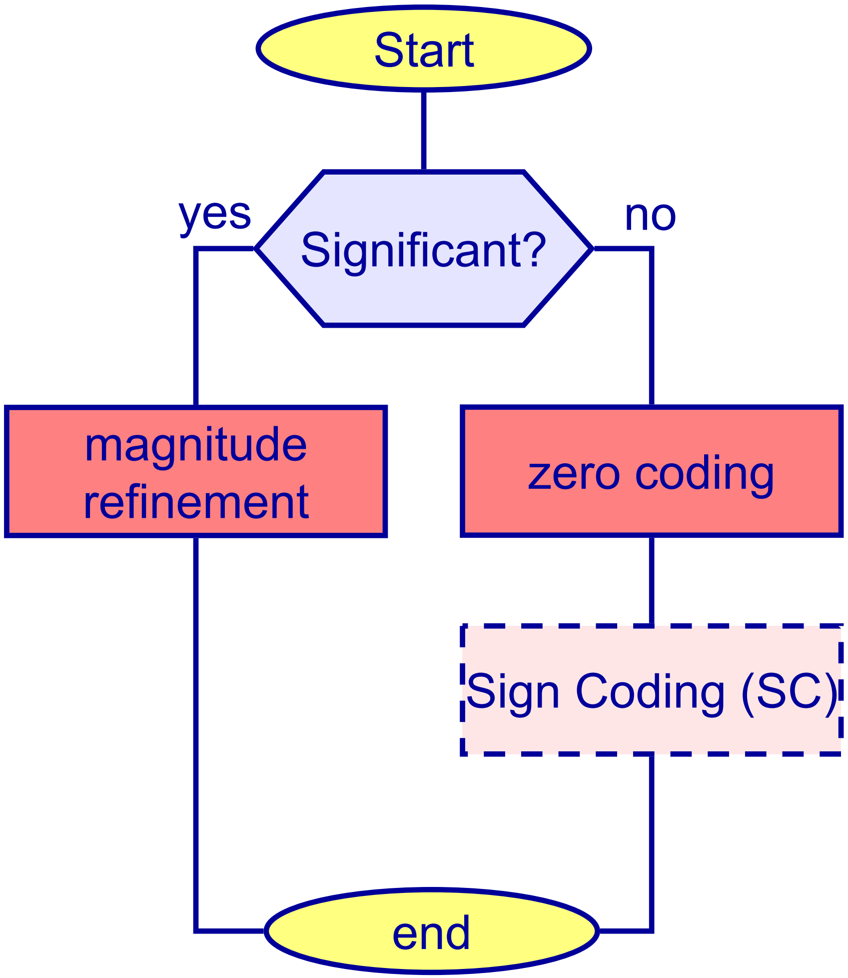
- The set of uniform quantizers with dead-zone having progressively halved step size is chosen

$$I_{n,k} = \begin{cases} \left(-\Theta_n, \Theta_n\right) \text{ if k=0} \\ \left[\Theta_n + \left(k-1\right)\Delta_n, \Theta_n + k\Delta_n\right) \text{ if k>0} \\ \left[-\Theta_n + k\Delta_n, -\Theta_n + (k+1)\Delta_n\right) \text{ if k<0} \end{cases}$$

$$\Delta_{n-1} = 2\Delta_n$$

- Then, each successive quantization layer doubles the precision with which subband sample values are quantized

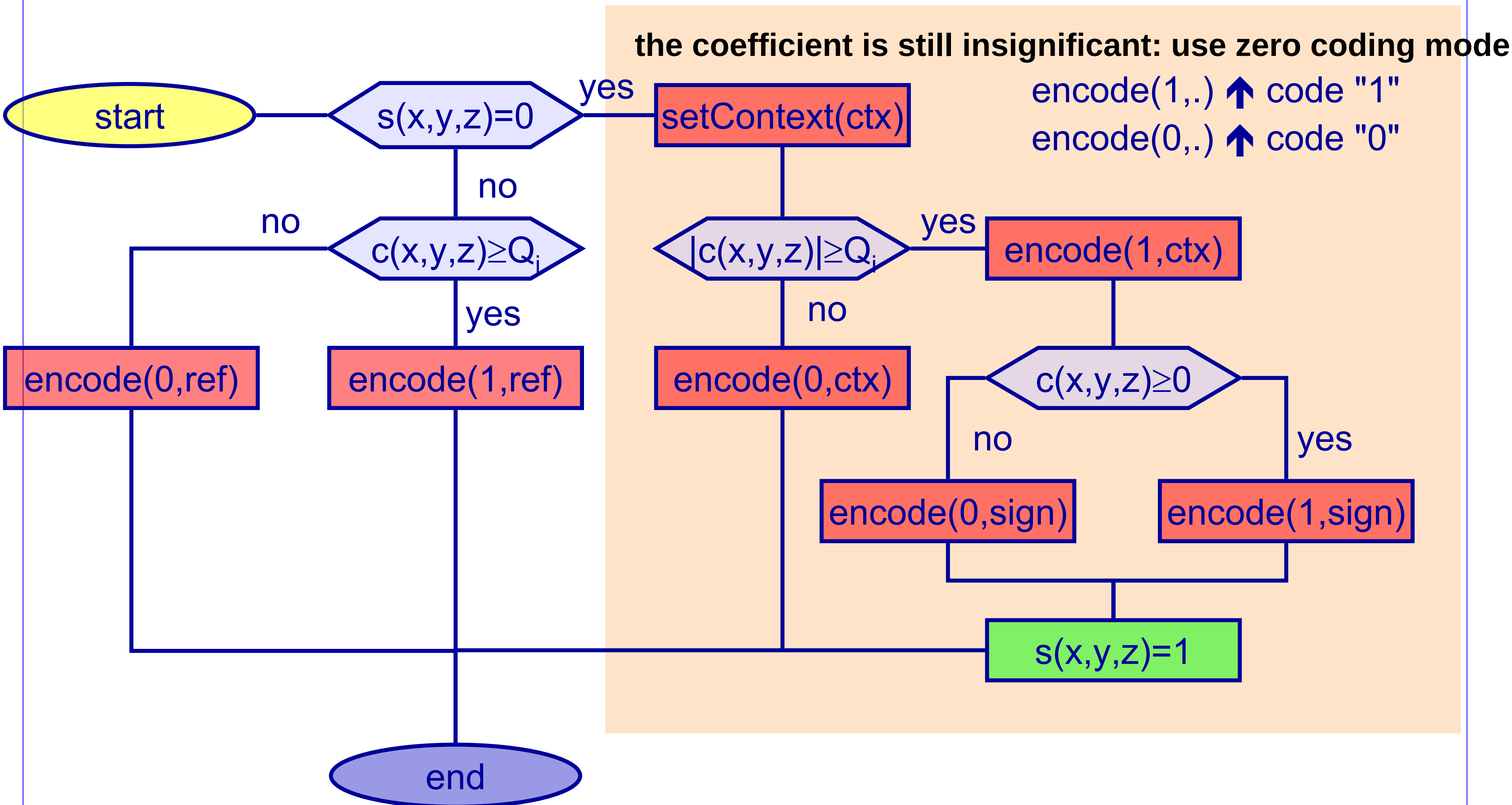
Layer Zero Coding (LZC)



- coefficient to code :
  - c(x,y,z)
- p quantifizers :
  - Qp-1 >...> Qi >…> Q0
  - Qi = 2^i
  - p = subband bit depth
- significance state :
  - s(x,y,z) = {0,1}
  - coefficient not significant (s = 0)
    $\forall$ j = p-1,…, i, |c(x,y,z)| < Qj
  - coefficient significant (s = 1)
    $\exists$ j = p-1,…, i such that |c(x,y,z)| $\geq$ Qj

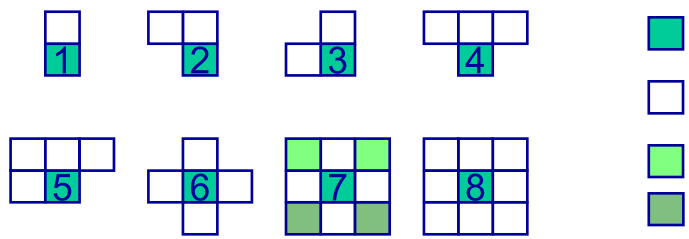
Bitplan encoding

LZC

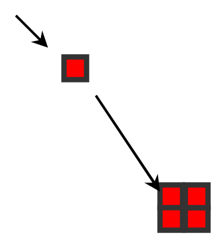


Layered-Zero Coding

- Exploits both intra-band and inter-band residual correlations
  - Intra-band dependencies are modeled by introducing conditional probabilities in entropy coding (***context-adaptive*** arithmetic coding). The probability of a symbol is conditioned to the significance state of its neighbors;

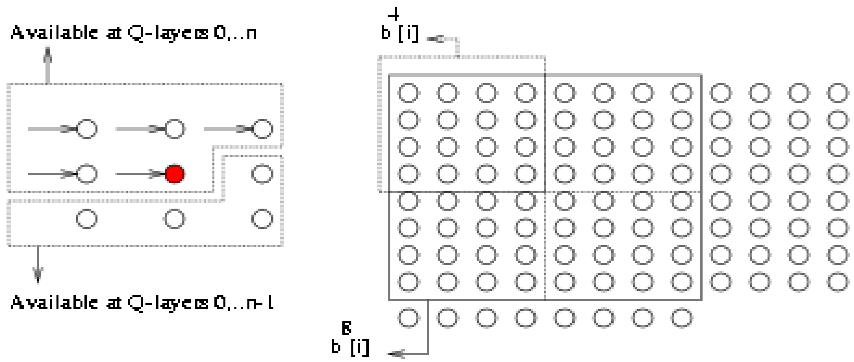


  - Inter-band dependencies are modeled similarly: the probability of a symbol is conditioned to the significance state of it ancestor

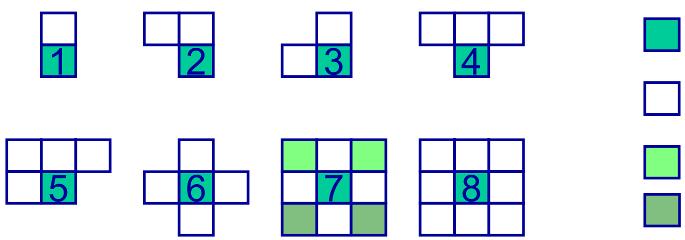


  - *… look at the notes…*

Layered Zero Coding



(a) (b)



Neighborhood ⇒ *Context*

- Encoding the SM ⇒ *Zero Coding*

  a-priori information ⇒ spatial or other kinds of dependencies among coefficients

  Conditioning terms: $\kappa(k,l,j)$

  • spatial (intra-band)
  • inter-band

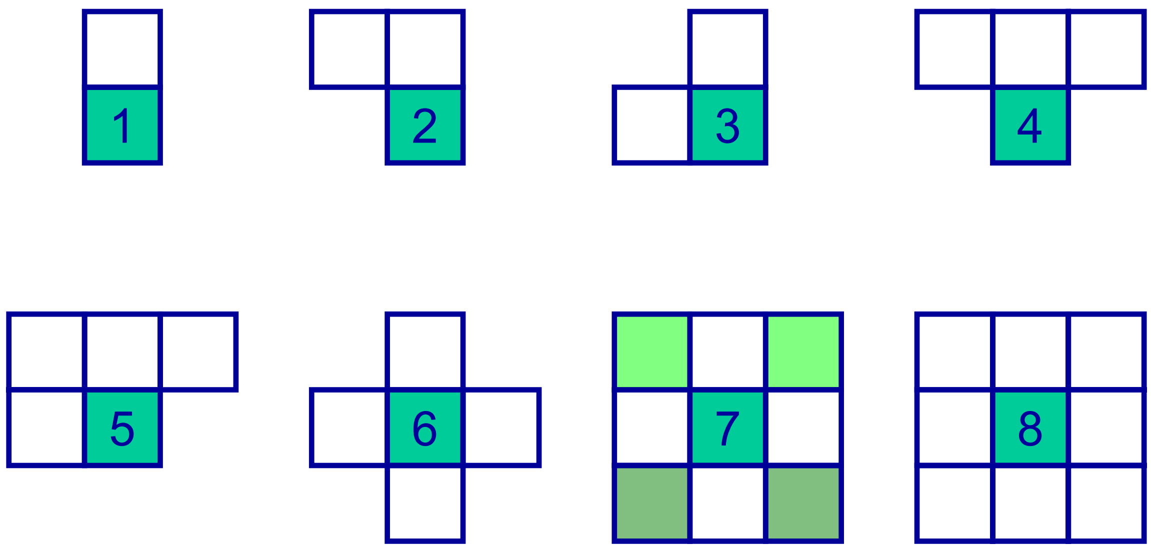
  ⇒ *context-adaptive* arithmetic coding

- Quantization refinement

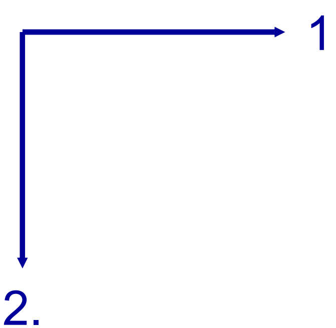
  ⇒ magnitude refinement

Spatial contextes

- Contextes 2D



Scanning order :



coefficient to code

neighbor

(and are composed)

Bitstream

p=total number of bitplanes

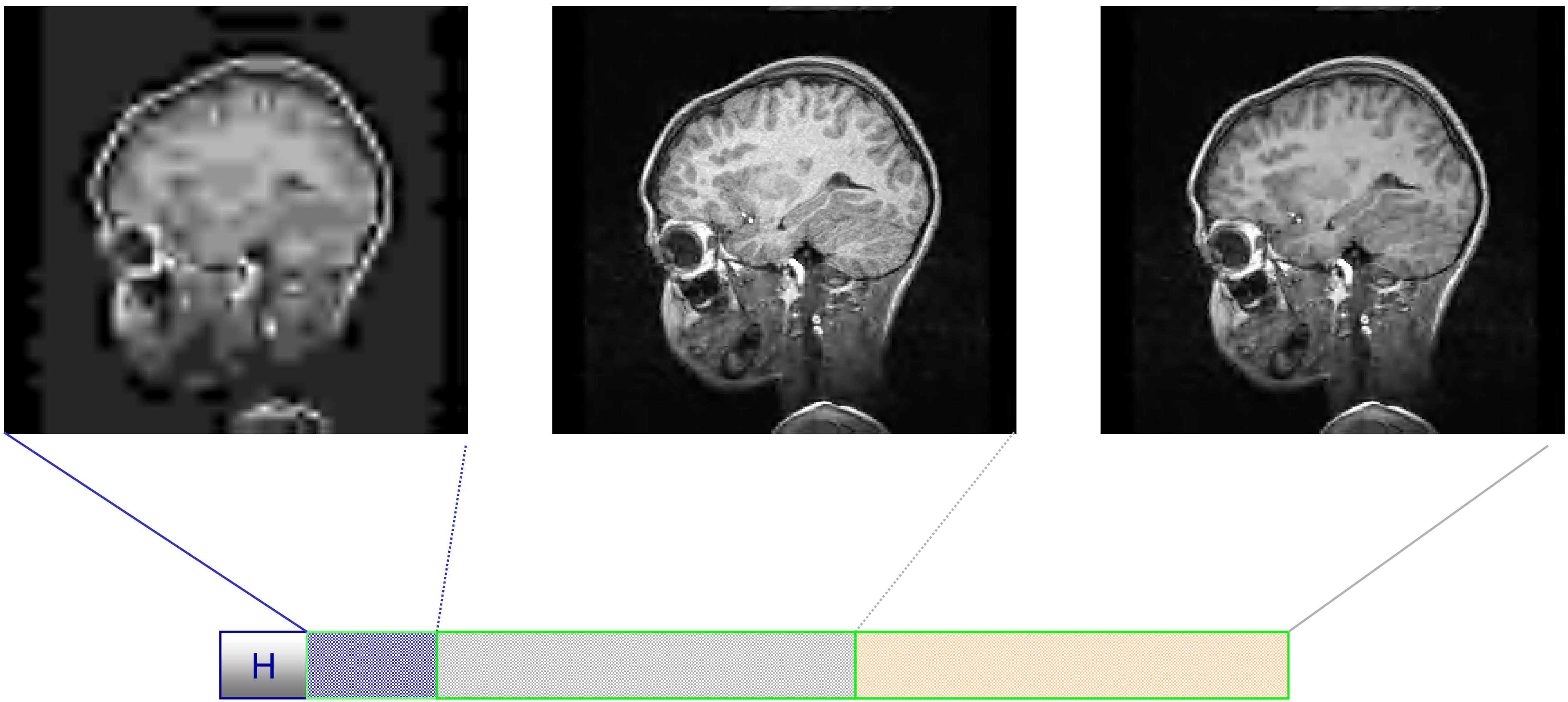
Coding artifacts at low rates

Original

JPEG

Wavelets

Scalability by quality



Scalability by resolution

Object-based processing