Compression and Coding

Theory and Applications
Part 1: Fundamentals
What is the problem?

Transformation

Transmitter (Encoder)

Channel

Ordering (significance)

Receiver (Decoder)
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 a 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 homogeneity 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

⇒ 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
    \[ R < C \] 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:
    \[ R \geq R(D) \] for fidelity given by $D$

$R(D)$: rate distortion function
Qualitative $R(D)$ curves

- $R(D)$ curves are monotonically non-increasing
  - Noteworthy points
    - $R(0)$: rate needed for exact reproduction of the source $\iff$ entropy of the source
    - $R_{opt}$, $D_{opt}$: minimum rate for a given distortion / minimum distortion at a given rate

![Diagram](image)
Entropy Coding

Fundamentals
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 $s$.
  - $p(s)=1 \rightarrow$ There is no uncertainty on the expectation on value taken by the RV $\rightarrow$ 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 $\rightarrow I(s)=0$
  - $p(s)<<(\text{very small}) \rightarrow$ the value $s$ is highly improbable $\rightarrow$ it corresponds to a rare event $\rightarrow$ 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 $I(s) \rightarrow \text{infinity}$
  - Summary: symbols that are certain convey no information, while very improbable symbols are highly informative
• 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

\[ I_k = -\log_2 p_k = -\log_2 p(x_k) \]
\[ p(x_k) = 1 \rightarrow I_k = 0 \]
\[ p(x_k) \ll 1 \rightarrow I_k \rightarrow \infty \]
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

  \[ \text{Same probability} \leftrightarrow \text{Maximum uncertainty} \]
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 (maximum uncertainty)
  $$H(X) = \sum_{k=1}^{K} p_k I_k = \sum_{k=1}^{K} \frac{1}{K} \log_2 K = K \frac{1}{K} \log_2 K = \log_2 K$$
  - It can be shown that this corresponds to the upper bound
    $$0 \leq H(X) \leq \log_2 K$$
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 \( \log_2 K \) 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
  \{w_1=0, w_2=10, w_3=110, w_4=101\}
  → 1010 can be read as both w_2w_2 and w_4w_1: ambiguous!
  → 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
  – \( \{p_1=0.05, p_2=0.1, p_3=0.1, p_4=0.15, p_5=0.2, p_6=0.4\} \)
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
  \[ p(A) = \frac{1}{3} \]
  \[ p(B) = \frac{2}{3} \]
Arithmetic coding

- Example
  \[ p(\text{AA}) = \frac{1}{3} \times \frac{1}{3} = \frac{1}{9} \]
  \[ p(\text{BA}) = \frac{1}{3} \times \frac{2}{3} = \frac{2}{9} \]
  \[ p(\text{AB}) = \frac{2}{9} \]
  \[ p(\text{BB}) = \frac{4}{9} \]
Arithmetic coding

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

Therefore, since

\[
P = p_1 \cdot p_2 \cdot \ldots \cdot p_N
\]

the number of bits of precision is approximately

\[
- \log_2(P) = -(\log_2(p_1) + \log_2(p_2) + \ldots + \log_2(p_N))
\]

thus \textit{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.

- \textit{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

Source signal (image)

Transformation

Prediction

Message extraction
definition of the set of symbols

Quantization

Entropy coding

reducing the number of symbols

assigning codewords to symbols

Bitstream
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

- Still images (JPEG lossless)
- Image sequences: motion compensation (MPEG4)
Intra-frame *linear* prediction

\[
\begin{array}{ccc}
12 & 34 & 27 & 42 \\
21 & 3 & 44 & 1 \\
12 & 34 & 27 & 42 \\
\end{array}
\]

\[
\begin{array}{ccc}
A & B & C \\
\hline
D & X & \\
\end{array}
\]

\[
X_{\text{est}} = aA + bB + cC + dD \\
E = X - X_{\text{est}}
\]

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

previous frame

current frame

next frame
Transform based coding

- Given the source signal, it can be convenient to project the data to a different domain to improve compression ⇒ 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
  \[ Y = \sum_mA[m]g_m \]
  \[ A[m] = \langle Y, g_m \rangle \]
- The coefficients $A[m]$ are quantized and then encoded
  \[ A_Q[m] = Q\{A[m]\} \]

Reconstructed signal (after entropy decoding)

\[ Y_{\text{dec}} = \sum_mA_Q[m]g_m \]

- 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 $\Rightarrow$ quantization

\[
f[n] \text{ in } L^2(\mathbb{Z}) \xrightarrow{\text{Quantizer}} f_q[n] \text{ in } L^2(\mathbb{Z})
\]

Discrete function $f_q[n]$ is obtained through the quantization process from the continuous function $f(t)$.

The diagram illustrates a uniform quantization scheme where $f_q = Q\{f\}$.

The quantization levels are represented by $t_k$ and $t_{k+1}$, with quantization errors $r_k$.
Scalar quantization

- A scalar quantizer $Q$ approximates $X$ by $X^\sim = Q(X)$, 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 $\{(y_{k-1}, y_k)\}_{1 \leq k \leq K}$ of variable length, with $y_0 = a$ and $y_K = b$.
- A scalar quantizer approximates all $x \in (y_{k-1}, y_k]$ by $x_k$:
  \[ \forall x \in (y_{k-1}, y_k], \quad Q(x) = x_k \]
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 1 and \(x_k=k\) for any \(k \in \mathbb{Z}\).

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

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

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

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

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

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

\[ y_k - y_{k-1} = \Delta \quad \text{for} \quad 1 \leq k \leq K. \]

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

\[ d = \frac{\Delta^2}{12} \sum_{k=1}^{K} p_k = \frac{\Delta^2}{12}. \]  \hspace{1cm} (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 $\delta_k$
  - $p(f)$ 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 → 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 $\Rightarrow$ quantization

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

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

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

• 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

\[ f_{q,k} = \frac{t_k + t_{k-1}}{2} \]

  – \( D(R) \) is minimal for uniform scalar quantization and given by

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

\( \Delta \) being the quantization step size and \( \sigma \) 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: $T \rightarrow T/2$
  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 non-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:
    
    $$b_m(p,q) = 0 \quad \text{if} \quad |a_m(p,q)| \leq T$$
    $$b_m(p,q) = 1 \quad \text{if} \quad a_m(p,q) > T$$
    $$b_m(p,q) = 2 \quad \text{if} \quad a_m(p,q) < -T$$
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-length 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.,
  
  \[
  \begin{align*}
  &56\ 56\ 56\ 82\ 82\ 82\ 83\ 80 \\
  &56\ 56\ 56\ 56\ 80\ 80\ 80 \\
  \end{align*}
  \]
  
  – 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

```
0 0 0 0 0 0 0 0 0
0 0 1 1 2 3 3 3 3
0 1 1 3 3 3 4 4 4
0 1 3 3 5 5 4 4 4
0 2 3 3 5 5 5 4 4
0 0 2 3 3 4 6 6 6
0 0 0 2 2 3 4 4 4
0 0 0 0 0 0 0 0 0
```

<table>
<thead>
<tr>
<th>gray level</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>#pixels</td>
<td>26</td>
<td>5</td>
<td>5</td>
<td>13</td>
<td>8</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
Run-length coding

<table>
<thead>
<tr>
<th>row #</th>
<th>run-length code (gl,rl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0,8)</td>
</tr>
<tr>
<td>1</td>
<td>(0,2), (1,2), (2,1), (3,3)</td>
</tr>
<tr>
<td>2</td>
<td>(0,1), (1,2), (3,3), (4,2)</td>
</tr>
<tr>
<td>3</td>
<td>(0,1), (1,1), (3,2), (5,2), (4,2)</td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td>7</td>
<td>(0,8)</td>
</tr>
</tbody>
</table>
Run-length coding

<table>
<thead>
<tr>
<th>row #</th>
<th>binary code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000 111</td>
</tr>
<tr>
<td>1</td>
<td>000 001 001 001 010 000 011 010</td>
</tr>
<tr>
<td>2</td>
<td>000 000 001 001 011 010 100 001</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>000 111</td>
</tr>
</tbody>
</table>

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 $\Delta$ 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

<table>
<thead>
<tr>
<th>T=2</th>
<th>significance map</th>
</tr>
</thead>
<tbody>
<tr>
<td>1340</td>
<td>011011100011</td>
</tr>
<tr>
<td>5278</td>
<td>011011100011</td>
</tr>
<tr>
<td>4511</td>
<td>011011100011</td>
</tr>
<tr>
<td>0243</td>
<td>011011100011</td>
</tr>
</tbody>
</table>

Realizations of the RV $Z$: $12111442$  
Realizations of the RV $I$: $011010111110000011$  
Sequence of symbols: $011010111110000011$  
Entropy coding: $011010111110000011$
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 non 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

- **Initialization**
- **Encoding the Significance map**
- **Quantization refinement**
- **Precision refinement**

$n = \lceil \sup_{m} \log_2 |a[m]| \rceil$

- **Layered Zero Coding (LZC)**
  - Exploitation of residual correlation among subband coefficients
- **Zerotree Coding (EZW)**
  - Update the value of the significant coefficients
  - Decrease the quantization bin: $n \rightarrow n-1$
Embedded Zero-Tree 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
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 predefined threshold $T$

$$T_0 = 2^\left\lfloor \log_2 x_{\text{max}} \right\rfloor$$
Significance map:

\[ b_{j}^{k}[p,q] = \begin{cases} 
1 & \text{if } 2^{n} \leq d_{j}^{k}[p,q] < 2^{n+1} \\
-1 & \text{if } -2^{n+1} < d_{j}^{k}[p,q] \leq -2^{n} \\
0 & \text{otherwise} 
\end{cases} \]

- Encoding the SM
  - *inter-band dependencies* ⇒ *quad-trees*
  - *Primary pass* ⇒ ZTR, IZ, POS, NEG
    - ZTR: \( \frac{1}{3}(4^{j}-1) \) symbols
- Quantization refinement
  - *Secondary pass* ⇒ HIGH, LOW

→ trees of zeros ↔ *zerotrees*
Significant Map Coding Using Zerotree

• Scan order:

  From lower subband to higher subband
EZW algorithm

- Initialization (set $T_0$)
  - Dominant pass
    - 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
  - Subordinate pass
    - 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)
- Quantization refinement ($T_0 = T_0/2$)
- no
  - $T_0 = 1$
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.
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 $t_0$ is decided as:

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

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**

1. **START**
2. \(|c(x,y,z)| \geq Q_i\) → yes\(\Rightarrow s(x,y,z) = \text{POS}\)
3. \(|c(x,y,z)| \geq Q_i\) → no\(\Rightarrow\) any child?
4. any child? → yes\(\Rightarrow\) all children are insignificant?
5. all children are insignificant? → yes\(\Rightarrow Prob(IZ) > Prob(ZTR)\)
6. Prob (IZ) > Prob(ZTR) → yes\(\Rightarrow s(x,y,z) = IZ\)
7. Prob (IZ) > Prob(ZTR) → no\(\Rightarrow s(x,y,z) = ZTR\)
8. any child? → no\(\Rightarrow c(x,y,z) > 0\)
9. \(c(x,y,z) > 0\) → yes\(\Rightarrow s(x,y,z) = IZ\)
10. \(c(x,y,z) > 0\) → no\(\Rightarrow s(x,y,z) = ZTR\)
11. encode(primprob, s(x,y,z))
12. end
EZW : Subordinate Pass

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

$$T_n, 1.5 \times T_n, 2 \times T_n = T_{n+1}$$
Example

**Dominant pass**

\[ T_0 = 32 \]
\[ x = 63 \]
\[ \text{uncertainty interval} = [32, 64] \]

reconstruction value at the end of the dominant pass

\[ 63 > 48 \rightarrow \text{HIGH (symbol=1)} \]

**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 \texttt{P}, \texttt{N} or \texttt{Z} then
    {
        code_next_scan_coefficient();
        put_coded_coefficient_in_fifo();
        if coefficient was coded as \texttt{P} or \texttt{N} then
        {
            add abs(coefficient) to subordinate list;
            set coefficient position to zero;
        }
    }
}
/ * * 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)

$T_0 = 32$

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

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.
After this two step, we finish one iteration.

\[ T_i = T_i/2 \text{(reduce the threshold)} \]

Repeat until target fidelity or bit-rate is achieve

---

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

<table>
<thead>
<tr>
<th>Subband</th>
<th>Coefficient Value</th>
<th>Symbol</th>
<th>Reconstruction Value</th>
<th>Comment (See Text)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( LL_3 )</td>
<td>53</td>
<td>( ps )</td>
<td>48</td>
<td>1)</td>
</tr>
<tr>
<td>( HL_3 )</td>
<td>-22</td>
<td>( ztr )</td>
<td>0</td>
<td>2)</td>
</tr>
<tr>
<td>( LH_3 )</td>
<td>14</td>
<td>( is )</td>
<td>0</td>
<td>3)</td>
</tr>
<tr>
<td>( HH_3 )</td>
<td>-12</td>
<td>( ztr )</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( LH_2 )</td>
<td>15</td>
<td>( ztr )</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( LH_2 )</td>
<td>-8</td>
<td>( ztr )</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( LH_2 )</td>
<td>34</td>
<td>( ps )</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>( LH_2 )</td>
<td>-2</td>
<td>( ztr )</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( LH_1 )</td>
<td>4</td>
<td>( is )</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( LH_1 )</td>
<td>2</td>
<td>( is )</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( LH_1 )</td>
<td>0</td>
<td>( is )</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( LH_1 )</td>
<td>-2</td>
<td>( is )</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

---

**Table 4. Resulting Output of the Subordinate Pass.**

<table>
<thead>
<tr>
<th>Coefficient Magnitude</th>
<th>Symbol</th>
<th>Reconstruction Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>1</td>
<td>56</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>40</td>
</tr>
</tbody>
</table>

---

\( T_0 = 32 \)
Bitstream

header

\[ Q_{p-1} \quad Q_{p-2} \quad \ldots \quad Q_0 \]

Primary Pass

Secondary Pass

\[ LL,n \quad \ldots \quad LH,n \quad HL,n-1 \quad \ldots \quad HH,n-1 \quad \ldots \quad LH,1 \quad \ldots \quad HH,1 \]

\( p = \text{total number of bitplanes} \)
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 computationally 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 bitrate 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 (roughe), ..., Qn (finer)
  – L quantization layers: L1, L2, ..., Ln

• Guidelines:
  – 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

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

\[
\Delta_{n-1} = 2\Delta_n
\]

- Then, each successive quantization layer doubles the precision with which subband sample values are quantized
Layer Zero Coding (LZC)

- Coefficient to code:
  - $c(x,y,z)$

- $p$ quantizers:
  - $Q_{p-1} > \ldots > Q_i > \ldots > Q_0$
  - $Q_i = 2^i$
  - $p$ = subband bit depth

- Significance state:
  - $s(x,y,z) = \{0,1\}$
  - Coefficient not significant ($s = 0$)
    $\forall j = p-1, \ldots, i, |c(x,y,z)| < Q_j$
  - Coefficient significant ($s = 1$)
    $\exists j = p-1, \ldots, i$ such that $|c(x,y,z)| \geq Q_j$
Bitplan encoding

LZC

the coefficient is still insignificant: use zero coding mode

\[ s(x,y,z) = 0 \]

\[ |c(x,y,z)| \geq Q_i \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 1 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 0 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 1 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 0 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 1 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 0 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 1 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 0 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 1 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 0 \]

\[ c(x,y,z) \geq 0 \]

\[ s(x,y,z) = 1 \]
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

- 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

Neighborhood ⇒ Context
Spatial contextes

- Contextes 2D

Scanning order:
1. coefficient to code
2. neighbor
( and are composed )
Bitstream

header

$Q_{p-1} \quad Q_{p-2} \quad \ldots \quad Q_0$

$LL,n \quad \ldots \quad HH,n \quad LH,n-1 \quad \ldots \quad HH,n-1 \quad \ldots \quad LH,1 \quad \ldots \quad HH,1$

$p=$total number of bitplanes
Coding artifacts at low rates

Original  JPEG  Wavelets
Scalability by quality
Scalability by resolution
Object-based processing