History

- Embedded Block Coding with Optimized Truncation (EBCOT)
  - Taubman – journal paper 2000
  - Algorithm goes back to 1998 or maybe earlier
  - Basis of JPEG 2000
- Embedded
  - Prefixes of the encoded bit stream are legal encodings at lower fidelity, like SPIHT and GTW
- Block coding
  - Entropy coding of blocks of bit planes, not block transform coding like JPEG.

Features at a High Level

- **SNR scalability** (Signal to Noise Ratio)
  - Embedded code - The compressed bit stream can be truncated to yield a smaller compressed image at lower fidelity
  - Layered code – The bit stream can be partitioned into a base layer and enhancement layers. Each enhancement layer improves the fidelity of the image
- **Resolution scalability**
  - The lowest subband can be transmitted first yielding a smaller image at high fidelity.
  - Successive subbands can be transmitted to yield larger and larger images

Block Diagram of Encoder

- **Image** → **Wavelet Transform** → **Partition into Coding Blocks** → **Bit Plane Code** → **Block Decoding**
- Truncate to achieve desired bit rate and maximum fidelity
- Only the most significant bit plane is transmitted

Extreme Case is Normal

- **Image** → **Wavelet Transform** → **Partition into one block** → **Bit Plane Code** → **Truncate to achieve desired bit rate**

Layering

- **Image** → **Wavelet Transform** → **Partition into Coding Blocks** → **Block Decoding**
- Create layers
- Only the most significant bit plane is transmitted
Resolution Ordering

- Assume we are in block k, and c(i,j) is a coefficient in block k.
- Divide c(i,j) into its sign s(i,j) and m(i,j) its magnitude.
- Quantize to v(i,j) = [m(i,j)/q_k + 0.5] where q_k is the quantization step for block k.
- Example: c(i,j) = -10, q_k = 3.
  - s(i,j) = 0
  - v(i,j) = floor(-10/3 + 0.5) = -2

Block Coding

- Sub-block significance coding (like group testing)
  - Some sub-blocks are declared insignificant
  - Significant sub-blocks must be coded
- Contexts are defined based on the previous bit-plane significance.
  - Zero coding (ZC) – 9 contexts
  - Run length coding (RLC) – 1 context
  - Sign coding (SC) – 5 contexts
  - Magnitude refinement coding (MR) – 3 contexts
- Block coded in raster order using arithmetic coding

Bit-Plane Coding of Blocks

- Sub-block significance coding (like group testing)
  - Some sub-blocks are declared insignificant
  - Significant sub-blocks must be coded
- Contexts are defined based on the previous bit-plane significance.
  - Zero coding (ZC) – 9 contexts
  - Run length coding (RLC) – 1 context
  - Sign coding (SC) – 5 contexts
  - Magnitude refinement coding (MR) – 3 contexts
- Block coded in raster order using arithmetic coding

Bit-Plane Coding of Blocks

- Sub-block significance coding (like group testing)
  - Some sub-blocks are declared insignificant
  - Significant sub-blocks must be coded
- Contexts are defined based on the previous bit-plane significance.
  - Zero coding (ZC) – 9 contexts
  - Run length coding (RLC) – 1 context
  - Sign coding (SC) – 5 contexts
  - Magnitude refinement coding (MR) – 3 contexts
- Block coded in raster order using arithmetic coding

Sub-Block Significance Coding

- Quad-tree organized group testing
- Block divided into 16x16 sub-blocks
- Identify in few bits the sub-blocks that are significant

Quad-Tree Subdivision
Quad-Tree Subdivision

Depth-first code: 1 for significant, 0 for insignificant

Quad-Tree Subdivision Coding

Skip symbols that are already known:
1. nodes significant in previous bit plane
2. last child of significant parent of other children is insignificant

ZC – Zero Coding

- LH is transposed so that it can be treated the same as HL. (LH)^T has similar characteristics to HL.
- Each coefficient has its neighbors in the same subband

ZC Contexts

- v = number of vertical neighbors significant in the previous bit-plane
- h = number of horizontal neighbors significant in the previous bit-plane
- d = number of diagonal neighbors significant in the previous bit-plane

Examples
RLC – Run Length Coding

• Looks for runs of 4 that are likely to be insignificant

• If all insignificant then code as a single symbol

• Main purpose – to lighten the load on the arithmetic coder.

SC – Sign Coding

\[
hs = \begin{cases} 
0 & \text{if horizontal neighbors are both insignificant or of opposite sign} \\
1 & \text{if at least one horizontal neighbor is positive} \\
-1 & \text{if at least one horizontal neighbor is negative}
\end{cases}
\]

\[
vs = \begin{cases} 
0 & \text{if vertical neighbors are both insignificant or of opposite sign} \\
1 & \text{if at least one vertical neighbor is positive} \\
-1 & \text{if at least one vertical neighbor is negative}
\end{cases}
\]

MR – Magnitude Refinement

• This is the refinement pass.

• Define \( t = 0 \) if first refinement bit, \( t = 1 \) otherwise.

Bit Allocation

• How do we truncate the encoded blocks to achieve a desired bit rate and get maximum fidelity

Bit Allocation as an Optimization Problem

• Input: Given \( m \) embedded codes and a bit rate target \( R \)

• Output: Find truncation values \( n_k, 1 \leq k \leq m \), such that

\[
D = \sum_k D_k^{n_k}
\]

is minimized and

\[
\sum_k n_k \leq R
\]
Facts about Bit Allocation

- It is an NP-hard problem generally
- There are fast approximate algorithms that work well in practice
  - Lagrange multiplier method
  - Multiple choice knapsack method

Rate-Distortion Curve

- Rate-distortion curve
  - Rate
  - Distortion
  - Encoded block
  - Truncation points

Picture of Bit Allocation

- Block 1
- Block 2
- Block m

Pick one point from each curve so that the sum of the x values is bounded by R and the sum of the y values is minimized.

Good approximate algorithms exist because the curves are almost convex.

Notes on EBCOT

- EBCOT is quite complicated with many features.
- JPEG 2000 based on EBCOT but differs to improve compression and decompression time.
  - EBCOT has
    - Resolution scalability
    - SNR scalability
    - Quantization
    - Bit allocation
    - Arithmetic coding with context and adaptivity
    - Group testing (quad trees)
    - Sign and refinement bit contexts
    - Lots of engineering

Notes on Wavelet Compression

- Wavelets appear to be excellent for image compression
  - No blocking artifacts
  - Wavelet coding techniques abound and are very effective
- Some of the wavelet coding techniques can apply to block transforms.
- Newest generation of image compressor use wavelets, JPEG 2000.