Embedded Image Coding Using Zerotrees of Wavelet Coefficients

J. M. Shapiro, *IEEE Trans. Signal Processing*, December 1993

Abstract

The embedded zerotree wavelet algorithm (EZW) is a simple, yet remarkably effective, image compression algorithm, having the property that the bits in the bit stream are generated in order of importance, yielding a fully embedded code. The embedded code represents a sequence of binary decisions that distinguish an image from the "null" image. Using an embedded coding algorithm, an encoder can terminate the encoding at any point thereby allowing a target rate or target distortion metric to be met exactly. Also, given a bit stream, the decoder can cease decoding at any point in the bit stream and still produce exactly the same image that would have been encoded at the bit rate corresponding to the truncated bit stream. In addition to producing a fully embedded bit stream, EZW consistently produces compression results that are competitive with virtually all known compression algorithms on standard test images. Yet this performance is achieved with a technique that requires absolutely no training, no pre-stored tables or codebooks, and requires no prior knowledge of the image source.

The EZW algorithm is based on four key concepts: 1) a discrete wavelet transform or hierarchical subband decomposition, 2) prediction of the absence of significant information across scales by exploiting the self-similarity inherent in images, 3) entropy-coded successive-approximation quantization, and 4) universal lossless data compression which is achieved via adaptive arithmetic coding.

Why Wavelets?

- Traditional DCT & subband coding: trends "obscure" anomalies that carry info
  - E.g., edges get spread, yielding many non-zero coefficients to be coded
- Wavelets are better at localizing edges and other anomalies
  - Yields a few non-zero coefficients & many zero coefficients
  - Difficulty: telling the decoder "where" the few non-zero's are!!!
- Significance map (SM): binary array indicating location of zero/non-zero coefficients
  - Typically requires a large fraction of bit budget to specify the SM
  - Wavelets provide a structure (zerotrees) to the SM that yields efficient coding

Zerotree Coding

- Every wavelet coefficient at a given scale can be related to a set of coefficients at the next finer scale of similar orientation
- Zerotree root (ZTR) is a low scale "zero-valued" coefficient for which all the related higher-scale coefficients are also "zero-valued"
- Specifying a ZTR allows the decoder to "track down" and zero out all the related higher-scale coefficients
- See figures in the printed out Web Page attached at the end of this handout

EZW Algorithm

Uses successive approximation quantization together with zerotree coding to provide embedded bit stream for image.

Sequence of Decreasing Thresholds: $T_o, T_1, \ldots, T_{N-1}$

with $T_i = T_{i-1}/2$ and $|\text{coefficients}| < 2\,T_o$

Maintain Two Separate Lists:

- Dominant List
  - *coordinates* of those coefficients not yet found to be significant
- Subordinate List
  - *magnitudes* of those coefficients found to be significant

For each threshold, perform two passes: Dominant Pass followed by Subordinate Pass

Dominant Pass (Significance Pass)

- Coefficients w/ coordinates on the Dominant List are compared to T_i to determine significance and, if significant, their sign
- The resulting significance map is zero-tree coded and sent
  - Code using four symbols:
    - zerotree root
    - isolated zero
    - positive significant
    - negative significant
  - Entropy code using adaptive AC, and send
  - For each coefficient coded as significant (pos. or neg.)
    - put its magnitude on the Subordinate List
    - remove it from the Dominant List

Subordinate Pass (Refinement Pass)

- Provide one more bit on the magnitudes on the Subordinate List as follows
  - Halve the quantizer cells
  - If magnitude is in upper half of old cell, provide "1"
  - If magnitude is in lower half of old cell, provide "0"
- Entropy code sequence of 1's and 0's using adaptive AC, and send

Stop when bit budget is exhausted. Encoded stream has embedded in it all lower-rate encoded versions. Thus, encoding/decoding can be terminated prior to reaching the full-rate version.

EZW Encoder Pseudocode

Note: stop at any point where bit budget is exceeded

Initialize

$T_0 = 2^{\lfloor \log 2(\text{max coeff}) \rfloor}$

k=0

Dominant List = All Coefficients

Subordinate List = Empty

Significance Pass

For each entry $w(m)$ in Dominant List (note: scan using any appropriate order)

    If $|w(m)| \geq T_k$ [i.e. $w(m)$ is significant]

        If $w(m)$ is positive

            Output symbol *sp*

        Else [i.e., $w(m)$ is negative]

            Output symbol *sn*

        Endif on sign

        Put $w(m)$ on the Subordinate List

        Remove $w(m)$ from the Dominant List

    Else [i.e., $|w(m)| < T_k$; insignificant]

        Case #1: $w(m)$ is a non-root part of a zerotree

            Don't Code – it is "predictably insignificant"

        Case #2: $w(m)$ is a zerotree root

            Output symbol *zr*

        Case #3: $w(m)$ is an isolated zero

    Endif on significance

    Entropy code symbols using adaptive AC (Optional, but recommended)

    Send bits

End loop through Dominant List

Refinement Pass

For each entry $w(m)$ in Subordinate List

    If $w(m) \in$ Bottom Half of $[T_k, 2T_k]$

        Output L ("L" for "low")

    Else [i.e., $w(m) \in$ Top Half of $[T_k, 2T_k]$]

        Output H ("H" for "high")

    Endif on "bottom/top"

    Entropy code H's and L's using adaptive AC (Optional, but recommended)

    Send bits

End loop through Subordinate List

Update

$T_{k+1} = T_k/2$

k=k+1

Go to Significance Pass