The Set Partitioning in Hierarchical Trees (SPIHT) Algorithm

by

William A. Pearlman

Rensselaer Polytechnic Institute

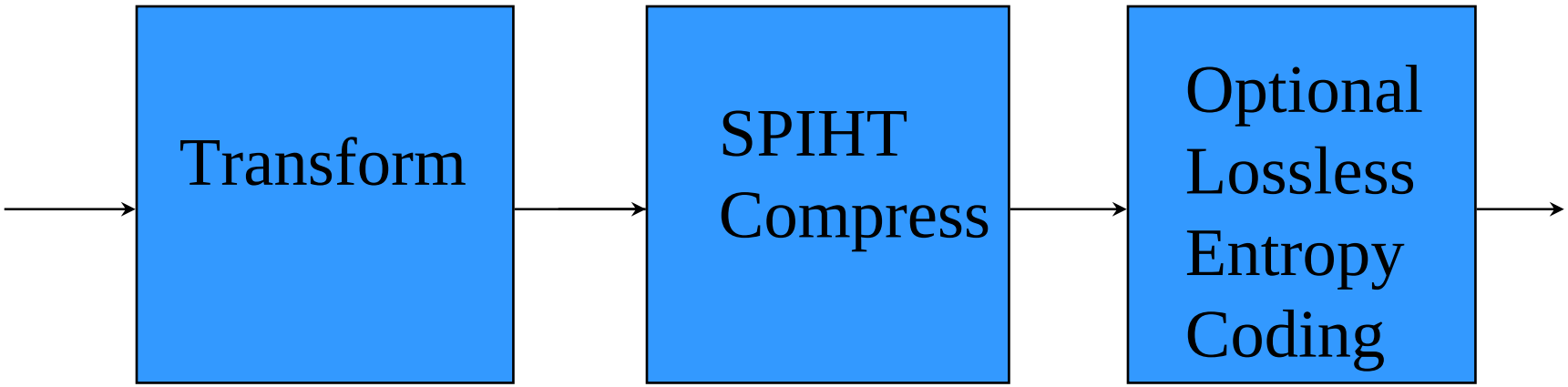
Presented to JPEG2000 Working Group

1 November 1997

The SPIHT Algorithm

- Inherent Characteristics-
  - efficient
  - completely embedded
  - precise rate control
  - idempotent
  - simple and fast
  - self-adaptive: no training required
- Supports
  - images of 8, 16, or larger bit depth
  - images of unrestricted dimensions
  - progressive lossy to lossless compression
  - multiresolution encoding or decoding
  - modularity

Modularity



Transform: wavelet, wavelet packet (subband),
S+P, lifting, DCT, LOT

Entropy coding: none, Huffman, arithmetic

Goal of SPIHT

- Sort transform coefficients by msb -
  - progressive selection of coefficients such that $|c_{i,j}| \geq 2^n, n = n_0, n_0 - 1, n_0 - 2, \ldots$
- Use transform characteristics to identify efficiently groups with same msb
- Send remaining bits by order of importance
  - first those identifying msb
  - then those of same bit plane with larger msb's
- Binary results of msb tests sent to decoder
  - enables decoder to duplicate encoder's execution path

Sorting by Magnitude and Bit-Plane Transmission

Transmission of magnitude-sorted coefficients

	sign	s	s	s	s	s	s	s	s	s	s	s	s	s
msb	5	1	1	0	0	0	0	0	0	0	0	0	0	0
	4	→	→	1	1	0	0	0	0	0	0	0	0	0
	3	→	→	→	→	1	1	1	1	0	0	0	0	0
	2	→	→	→	→	→	→	→	→	1	1	1	1	1
	1	→	→	→	→	→	→	→	→	→	→	→	→	→
lsb	0	→	→	→	→	→	→	→	→	→	→	→	→	→

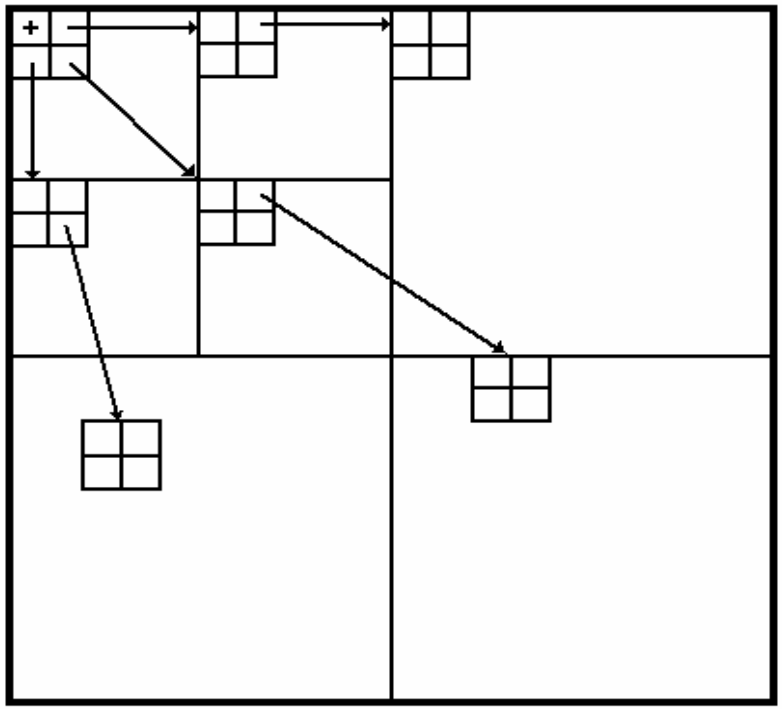
Set Partitioning Sorting

- Significance Test

$$S_n(U) = \begin{cases} 1, & \max_{(i,j)\in U} |c_{i,j}| \geq 2^n \\ 0, & \text{otherwise} \end{cases}$$

- Basic Algorithm
  - select partitions of pixels U_m
  - for each $n = n_0, n_0 - 1, n_0 - 2, \ldots$
    - if $S_n(U_m) = 0$ (the set is insignificant) then disregard pixels in U_m
    - if $S_n(U_m) = 1$ (the set is significant) then use recursive algorithm to partition U_m
  - test sets until all significant coefficients found

Set Partitions in Hierarchical Trees: Wavelet Transform Example



- $\mathcal{O}(i,j)$: offspring of node (i,j)
- $\mathcal{D}(i,j)$: all descendants of node (i,j)
- $\mathcal{L}(i,j) = \mathcal{D}(i,j) - \mathcal{O}(i,j)$
- $\mathcal{H}$: tree roots

Hierarchical Trees

- Structure not part of algorithm
- Various possibilities for structure
  - fixed
  - transform-dependent
  - data-dependent

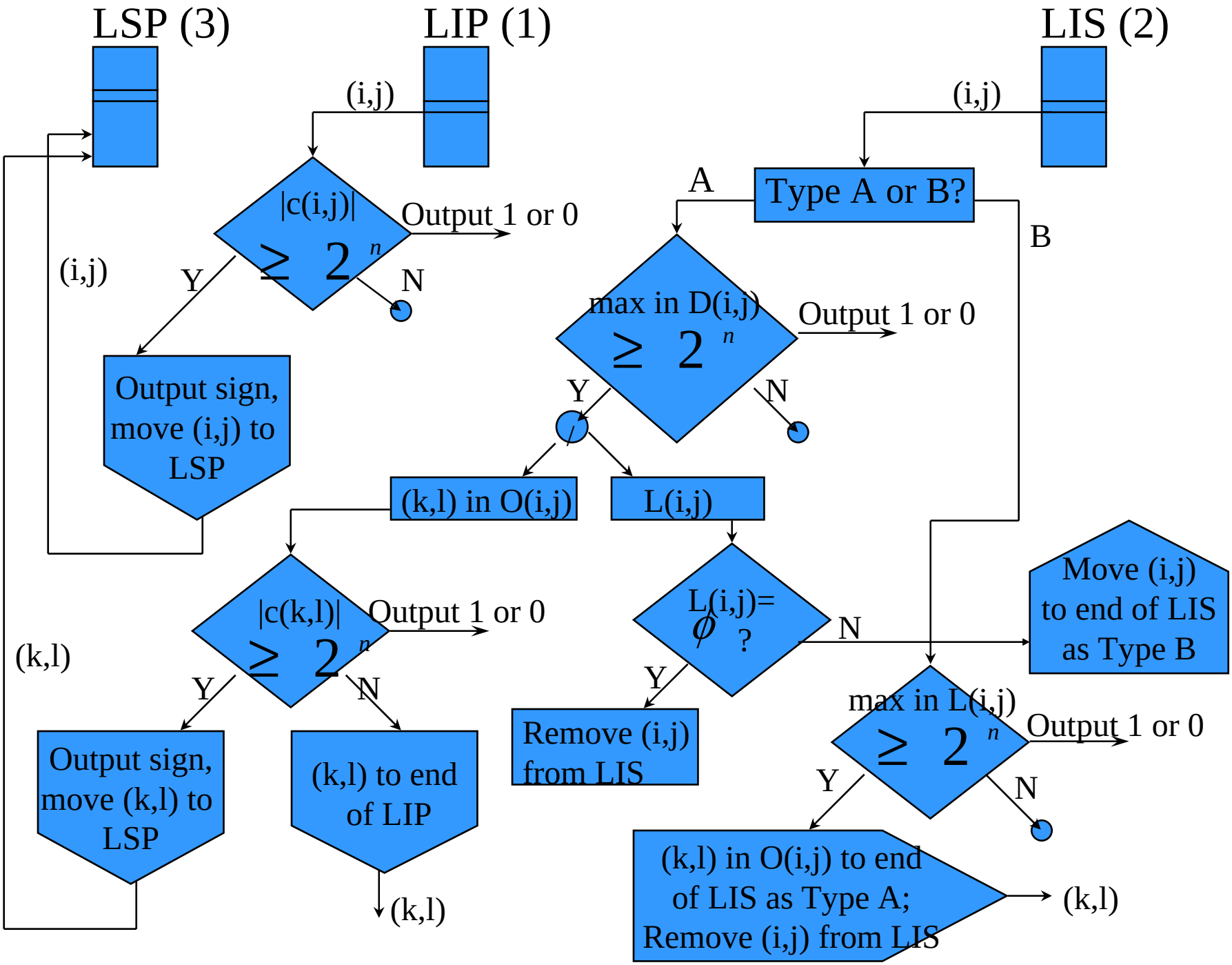
Three Lists

- LIP - list of insignificant pixels
- LIS - list of tree roots (i,j) of insignificant descendant sets $\mathcal{D}(i,j)$ (Type A) or insignificant descendant of offspring sets $\mathcal{L}(i,j) = \mathcal{D}(i,j) - \mathcal{O}(i,j)$ (Type B)
- LSP - list of significant pixels

☮ Lists tested in order LIP, LIS, LSP for efficient embedded coding

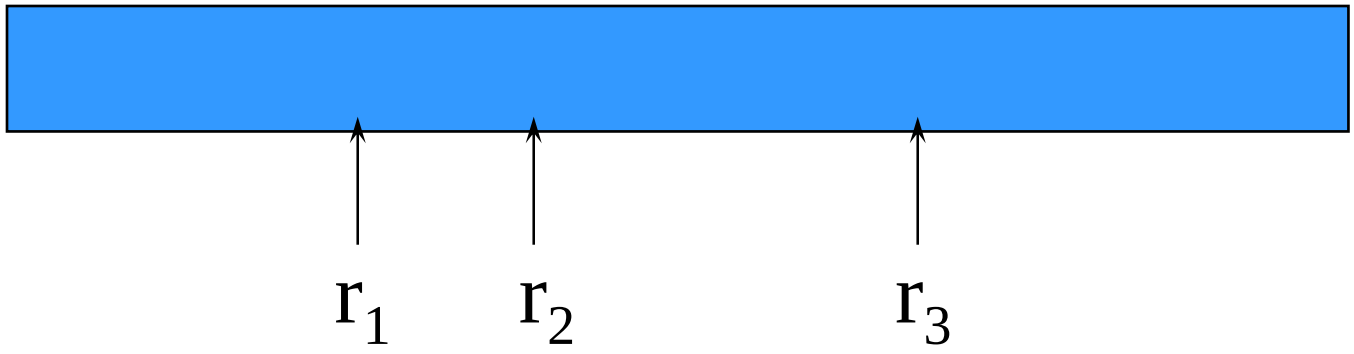
Initialization of Lists

- LIP: co-ordinates of all tree roots-
  - wavelet example: co-ordinates in coarsest scale subband
- LIS: co-ordinates of all tree roots with non-empty desendent trees
  - wavelet example: co-ordinates in coarsest scale subband pointing to descendant trees
- LSP: empty


LSP (3)
LIP (1)
LIS (2)
(i,j)
(i,j)
|c(i,j)| ≥ 2^n
Output 1 or 0
Y
N
(i,j)
Output sign, move (i,j) to LSP
A
Type A or B?
B
max in D(i,j) ≥ 2^n
Output 1 or 0
Y
N
(k,l) in O(i,j)
L(i,j)
|c(k,l)| ≥ 2^n
Output 1 or 0
(k,l)
Y
N
Output sign, move (k,l) to LSP
(k,l) to end of LIP
(k,l)
L(i,j)= ∅ ?
N
Y
Remove (i,j) from LIS
Move (i,j) to end of LIS as Type B
max in L(i,j) ≥ 2^n
Output 1 or 0
Y
N
(k,l) in O(i,j) to end of LIS as Type A; Remove (i,j) from LIS
(k,l)

Refinement Pass

- Output *n*-th bit of all LSP members found significant at thresholds greater than 2^n
- Two bit types in stream: significance test bits and refinement bits
- *Completely embedded code*



Symmetric Decoder

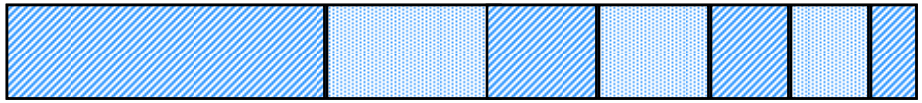
- Replace “output” by “input” only
  - nearly symmetric to encoder (receives largest msb level that encoder finds and outputs).

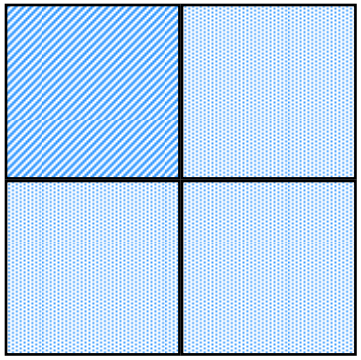
Progressive Lossy to Lossless Coding

- Lossless compression when n=0 pass complete -
  - precision truncation loss for real transform
  - completely reversible for integer transform
- Low-order bits more efficiently coded by switching from bit plane transmission at some high rate
- Efficient lossy reconstruction produced from truncation of file - scalable in fidelity

SPIHT Properties

- Idempotency - lossless re-compression at same bit rate
  - at same bit rate, re-compression builds same ordered lists and transmits same bits
- Multiresolution Scalability





Encoder/decoder tracks resolution of bits automatically

Low Complexity

- No floating point multiplications
- No estimation
- No rate allocation
- Search only for largest msb in transform
  - quick first pass, second pass does coding
- Addressing - increment, decrement, bit shifts
- Efficient without entropy coding
- Most computation for transform

Memory Usage

- Size of transform block (full image for wavelet)
- < 1/4 block size for LIS and LIP
  - rate-dependent
- < block size for LSP
  - rate-dependent

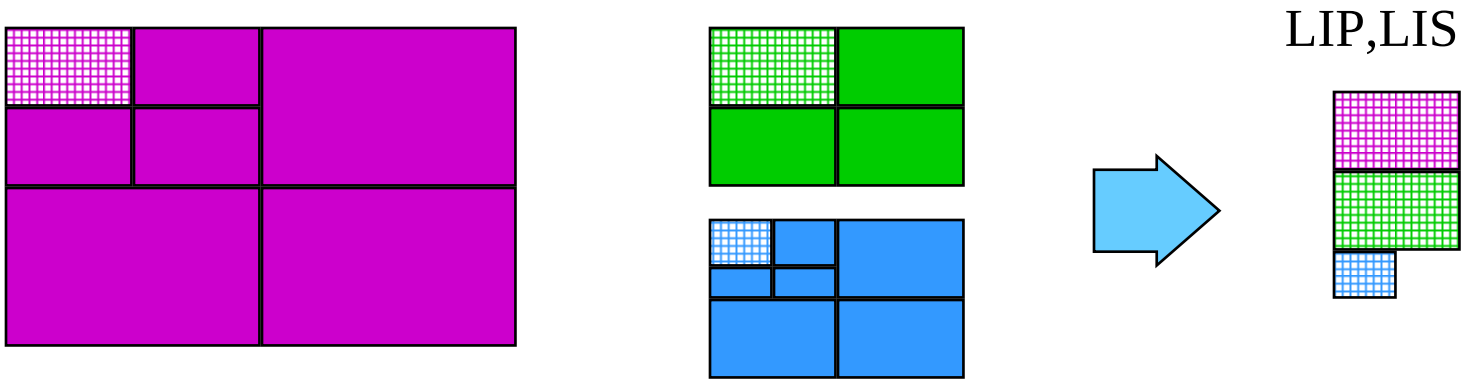
Channel Errors

- Errors in significance test bits:
  - reconstruction possible only up to first such bit error
- Errors in refinement bits:
  - cause graceful degradation
- Propose Sherwood and Zeger parity check +RCPC channel code for transmission:

  duplicated in our lab and corrected all errors at 0.1 error rate

Color and Multi-Spectral Coding

- Color planes coded together
  - LIP, LIS initialized with co-ordinates of root level of transform in all planes
  - No explicit bit allocation
  - Different planes may have different transforms



Conclusion

- SPIHT is a simple and efficient algorithm with many unique and desirable properties

Ref : A. Said and W.A. Pearlman, “A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees,” IEEE Trans. Circuits & Systems for Video Technology, Vol. 6, pp. 243-250, June 1996