An Overview of JPEG2000

Outlines

- Standard Bodies:
  - ISO: SC29WG01 (JPEG, JBIG)
- JPEG2000:
  - The Scope and Goal of JPEG2000
  - JPEG2000 Schedule
- Embedded Coding:
  - **Embedded Block Coding with Optimized Truncation(EBCOT)**
- Region of Interest(ROI)
  - The Feature and Issue of ROI
  - Derive the ROI mask
  - Two Implementations of ROI : Sequence based mod and Scaling based mode
  - Multiple ROIs

The Scope and Goal of JPEG2000

- A new and unified coding scheme for all kinds of still images:
  - Color, gray-level, and bi-level images
  - Natural images, scientific, medical, aerial, remote sensing imagery, text, and rendered graphics
- Low bit-rate compression performance
- Lossless and lossy compression
- Large images
- Single decompression architecture
- Computer generated imagery
- Compound document
- Transmission in noisy environments

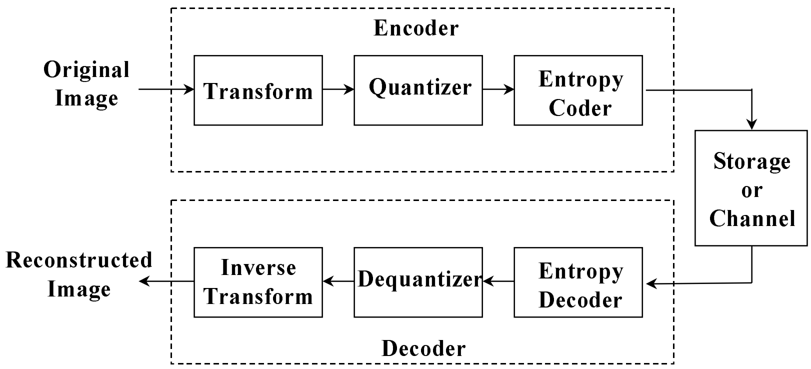
JPEG 2000 Features

- Not only better efficiency, but also more functionality
  - Superior low bit-rate performance
  - Lossless and lossy compression
  - Progressive transmission by quality, visual and resolution
  - Fixed-rate, fixed-size, limited workspace memory
  - Random codestream access and processing
  - Robustness to bit errors

JPEG2000 Schedule

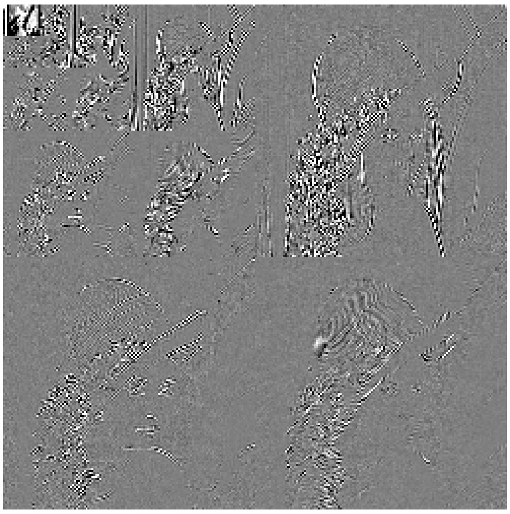
- Call for Contributions: 97 / 3
- Algorithm Evalution: 97 / 11
- Working Draft: 99 / 3 -- 99 / 7 (original:98 / 7 -- 98 / 11)
- Committee Draft: 99 / 7 -- 00 / 11? (orig: 98 / 11 -- 00 / 3)
- Draft Int’l Standard: 00 / 11 -- 01 / 3? (orig: 00 / 3 -- 00 / 7)
- International Standard: 01 / 3 -- 01 / 11? (orig: 00 / 7 -- 01 / 3)

A Generic Image Codec



@NTUEE
DSP/IC Lab

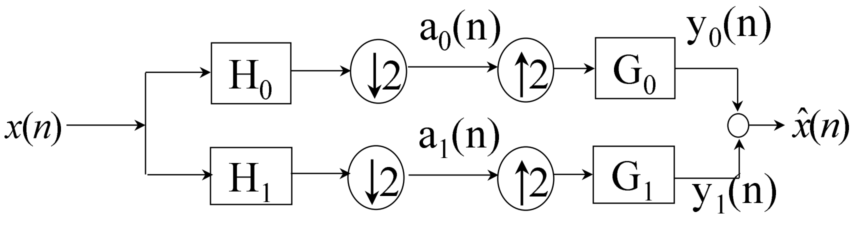
Wavelet Transform - A sample



7

@NTUEE
DSP/IC Lab

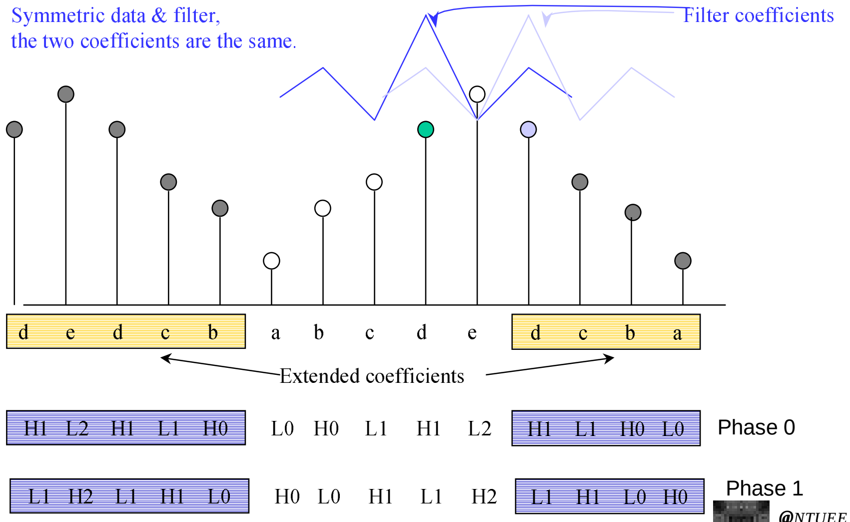
Wavelet Transform - 1D



8

@NTUEE
DSP/IC Lab

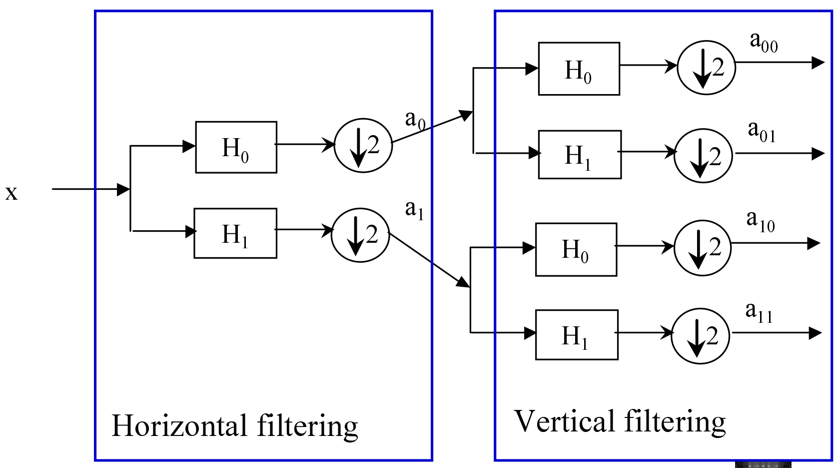
Symmetric Extension for Odd-tap Filters



@NTUEE DSP/IC Lab

9

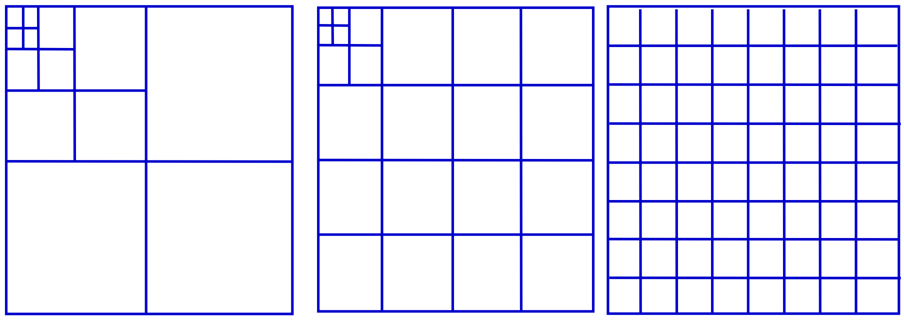
Wavelet Transform - 2D



@NTUEE DSP/IC Lab

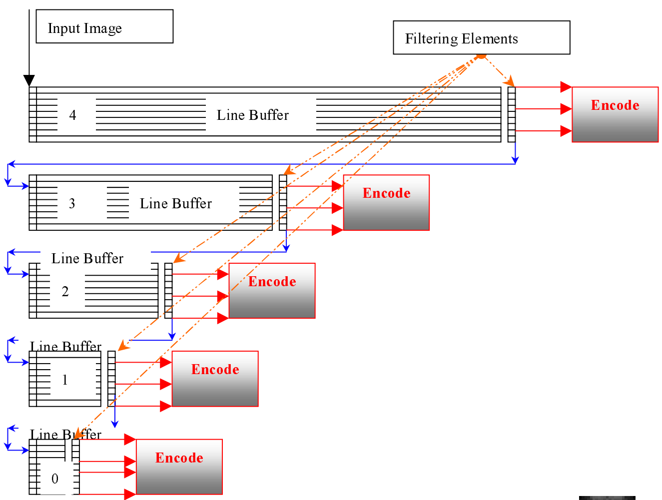
10

Wavelet Packet Structure



mallat spacl packet

Line based Transform



Recommended Wavelet Filter

- Wavelet 9-7 filter for lossy compression
  - Best performance at low bit rate
  - Relatively high implementation complexity, especially for hardware

i	h_0	h_1
0	0.6029490182363579	1.115087052456994
1	0.2668641184428723	-0.5912717631142470
2	-0.07822326652898785	-0.05754352622849957
3	-0.01686411844287495	0.09127176311424948
4	0.02674875741080976	

13

@NTUEE DSP/IC Lab

Recommended Wavelet Filter (cont.)

- CRF 13-7 filter for lossy integer arithmetic mode
  - Integer arithmetic, low implementation complexity

i	h_0	h_1
0	164/256	1
1	80/256	-9/16
2	-31/256	0
3	-16/256	1/16
4	14/256	
5	0	
6	-1/256	

14

@NTUEE DSP/IC Lab

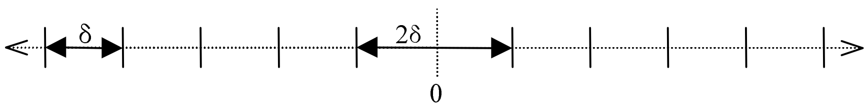
Recommended Wavelet Filter (cont.)

- I 5-3 filter for lossless coding
  - Integer arithmetic, low implementation complexity

i	h_0	h_1
0	6/8	1
1	2/8	-1/2
2	-1/8	0

Quantizer

- Scalar quantizer with a dead-zone which the central dead-zone is twice as large as the step size, δ



$$v[m,n] = \left\lfloor \frac{|s[m,n]|}{\delta} \right\rfloor,$$

Quantized Magnitude

$$\chi[\mathrm{m},\mathrm{n}] = \begin{cases} 0 & +s \\ 1 & -s \end{cases}$$

Sign

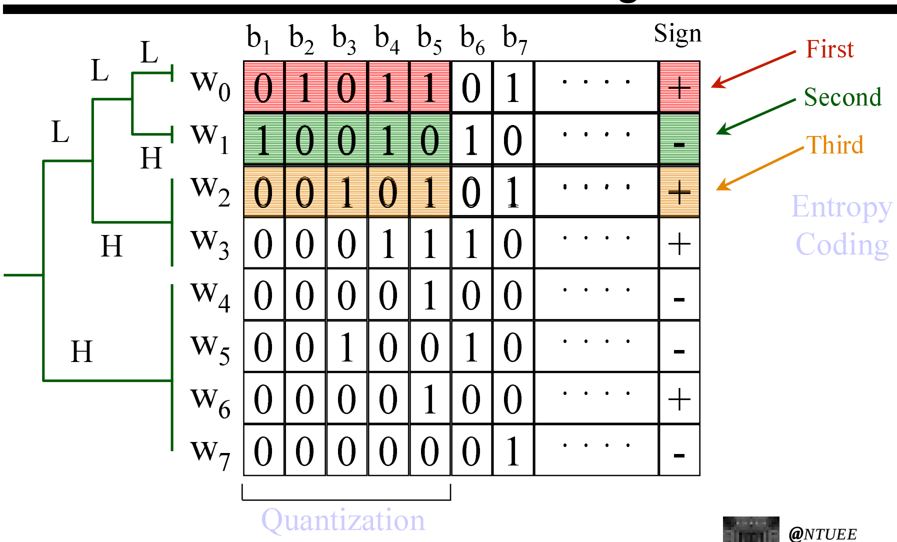
Entropy Coding

- **Embedded Coding:** A major development of recent years
- **Basic Idea:** Bit plane approach
  - The most significant bits are transmitted first.
- **Merits:**
  - The ability to terminate encoding and decoding at any point
  - Ideal for progressive image transmission
  - Easy rate control
- **Drawback:** Need to scan coefficients multiple times
- **Embedding Coding Algorithm:**
  - *Embedded Block Coding with Optimized Truncation(EBCOT)* adopted by VM3A

17

@NTUEE DSP/IC Lab

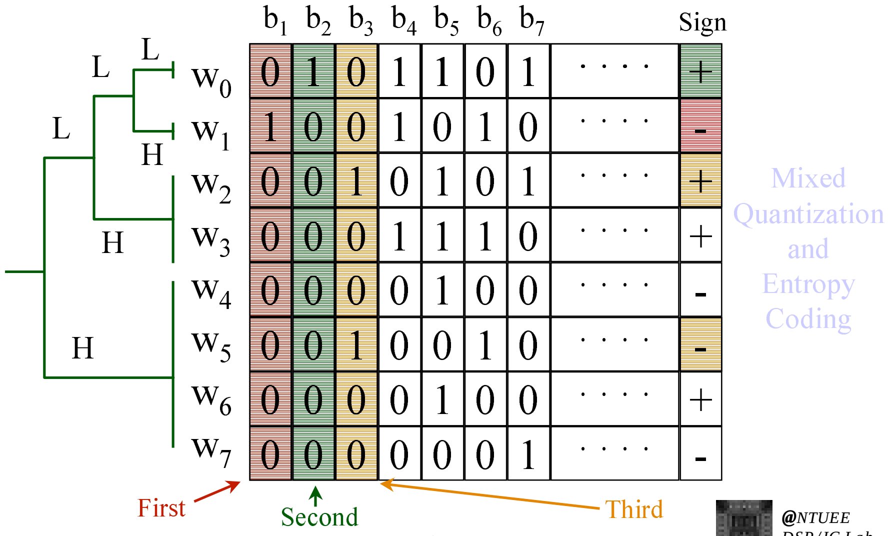
Conventional Coding



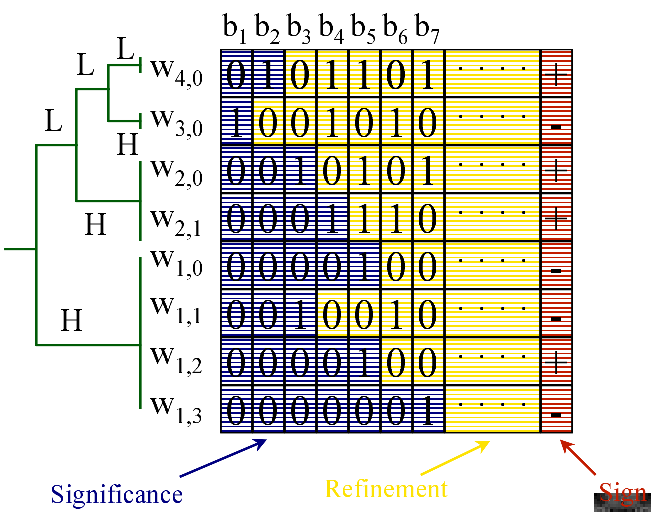
18

@NTUEE DSP/IC Lab

Embedded Coding -- Bit Plane Approach



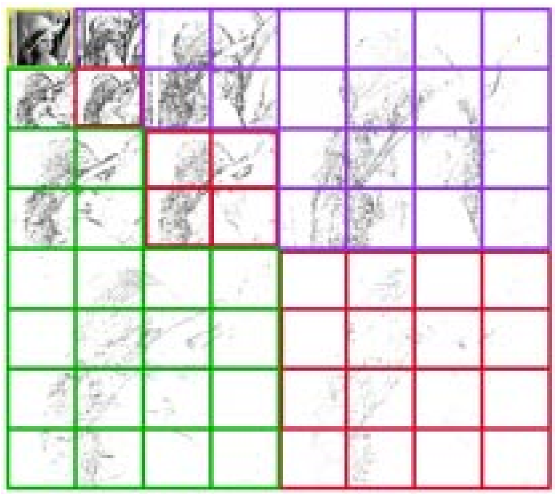
Significance, Sign, Refinement



EBCOT (Embedded Block Coding with Optimized Truncation)

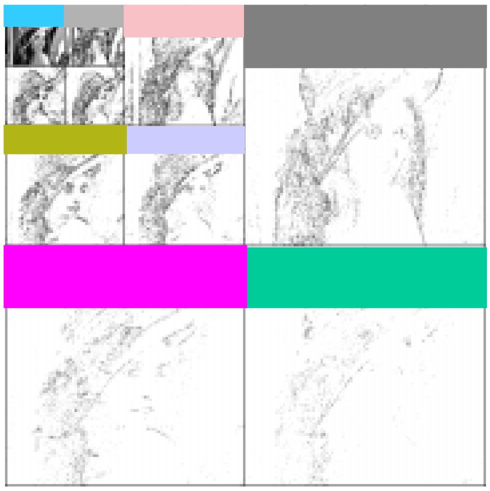
- Proposed by David Taubman, HP, in JPEG Los Angeles meeting, Nov. 1998.
- Key Ideas:
  - Partition subbands into smaller blocks (e.g. 64x64)
  - Form independent, embedded bit-stream for each block
- Key features of EBCOT:
  - Low memory requirement in coding and decoding
  - Easy rate control
  - High compression performance
  - Region of interest (ROI) access
  - Error resilience
  - Simple quantization
  - Modest complexity

Block structure in EBCOT

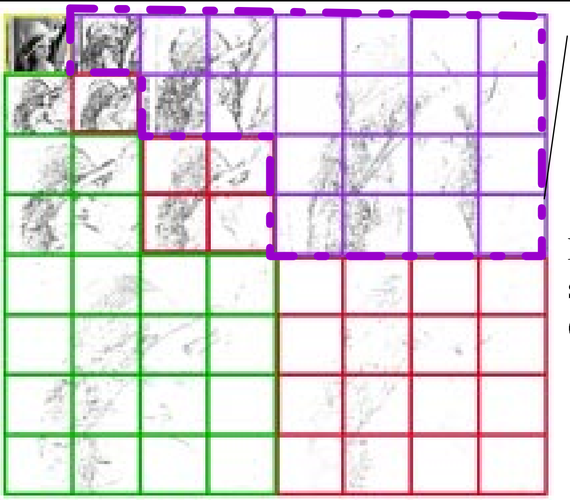


Encode each block separately & record the bitstream of each block. Block size is 64x64.

Line buffering to form blocks



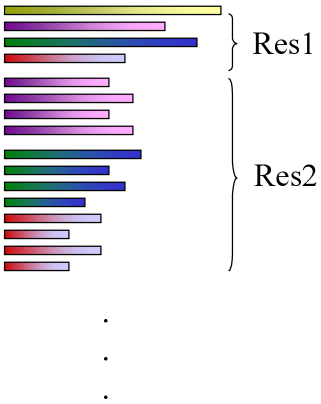
Block Transposition of HL Subbands



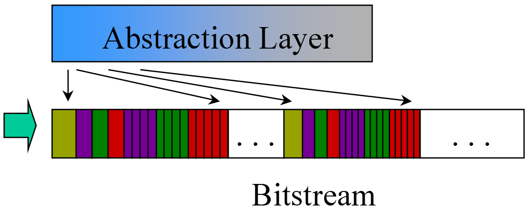
Coefficients in HL band are transposed

Make the edge structure similar to LH subbands (mainly horizontal edges)

EBCOT - Assemble the Bitstream



Encode each block separately & record a bitstream for each block



Reassemble the bitstream, encodes an abstraction layer with the structure of the bitstream:

- The bitstream is ordered with a primary functionality, such as progressive by quality or progressive by resolution
- The bitstream may be parsed to achieve additional functionality, such as progressive by resolution, region of interest access, etc..

25

@NTUEE DSP/IC Lab

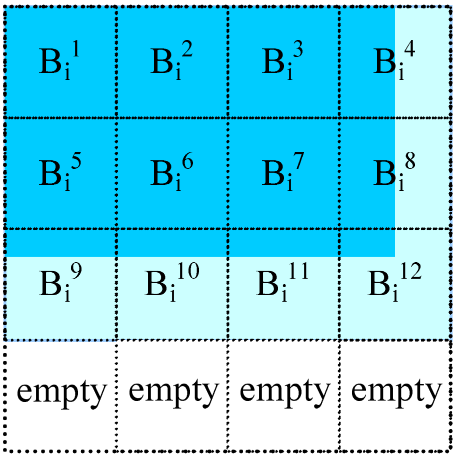
Block Embedded Coding

- Highlight
  - fractional bitplane embedded coding: divide subband bitplane into groups with similar statistical characteristics.
  - Quadtree front end to reduce computational complexity
  - Context arithmetic coding
- Key Steps:
  - When a block is first seen, send the maximum bit plane in the whole block
  - Partition the blocks into sub-blocks 16x16
  - Proceed with bit plane coding
  - A quad-tree code is used up to the sub-block level
  - After reaching the sub-block level, scan the bit planes from more significant to less significant

26

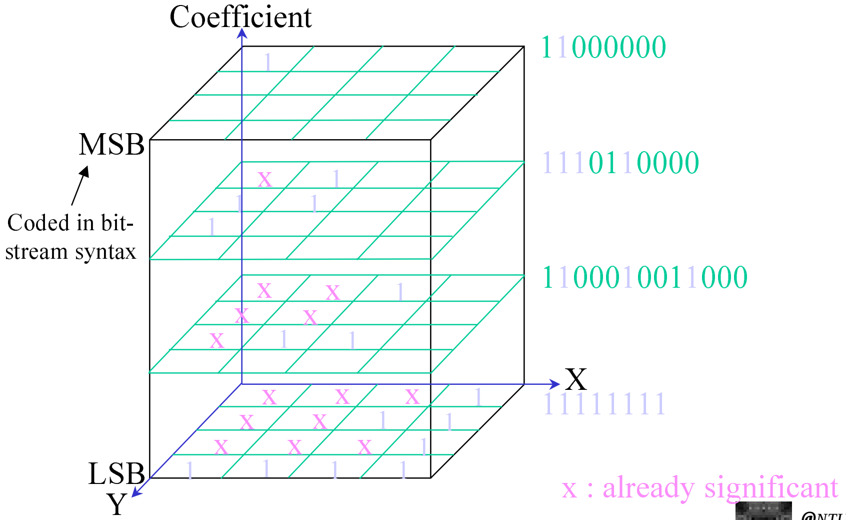
@NTUEE DSP/IC Lab

Subblock Division

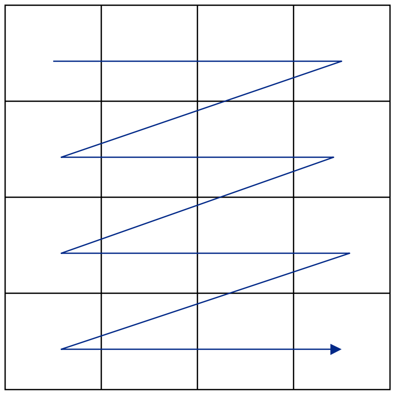


Example:
Block size is 56x36, and subblock size is 16x16.
Only 12 of the 16 subblock are used, and 6 of them are partially used.

Quad-Tree Coding of Subblock Significance



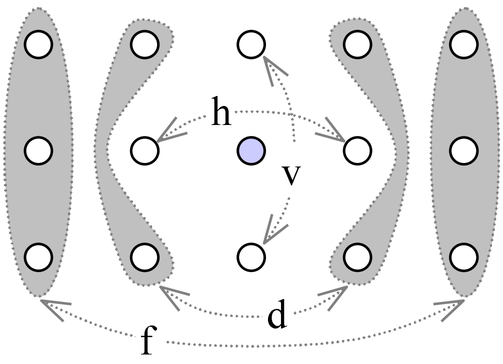
Sub-block Scan Order



Types of Coding Operations

- Zero Coding (ZC)
  - Used to code new significance
  - 10 contexts according to the significance of its neighbors
- Run-Length Coding (RLC)
  - Group 4 insignificant coefficients when they are very probable
  - Reduce the average number of symbols needed to be coded
  - One context for whether all four are insignificant
- Sign Coding (SC)
  - Used to code the sign right after a coefficient is identified significant.
  - 5 contexts based on the sign of four neighbors
- Magnitude Refinement (MR)
  - 3 contexts depending on the significance of its neighbors and whether it is the first time for refinement

Neighborhood in Zero Coding



h: number of significant horizontal neighbors (0-2)

v: number of significant vertical neighbors (0-2)

d: number of significant diagonal neighbors (0-4)

d: number of significant non-immediate neighbors (0-6)

Zero Coding Context

LL, LH and HL subbands					HH subband			
h	v	d	f	context	d	h+v	f	Context
2	x	x	X	9	≥3	x	x	9
1	≥1	x	X	8	2	≥1	x	8
1	0	≥1	X	7	2	0	x	7
1	0	0	X	6	1	≥2	x	6
0	2	x	X	5	1	1	x	5
0	1	x	X	4	1	0	x	4
0	0	≥2	X	3	0	≥2	x	3
0	0	1	X	2	0	1	x	2
0	0	0	≥1	1	0	0	≥1	1
0	0	0	0	0	0	0	0	0

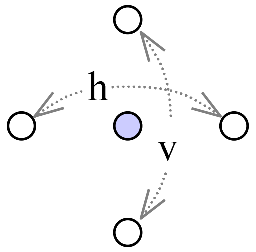
Run-Length Coding (RLC)

- Invoked in normalization if and only if
  - 4 consecutive horizontal symbols are in the insignificant state.
  - All 4 symbols have only zero neighborhoods.
  - All 4 symbols are horizontally adjacent and reside within a sub-block.
  - Aligned on a two-sample boundary.
- Encoding:
  - Encoded a special symbol with a single context. The special symbol identifies if all 4 symbols are insignificant (0: yes, 1:no)
  - if any of symbols is significant, a 0-based index of the 1st significant symbol is sent as a two-bit quantity (arithmetic coding with balanced probability distribution)

33

@NTUEE DSP/IC Lab

Context in Sign Coding



h, v: neighborhood sign status
-1: one or both negative
0: both insignificant or both significant but opposite sign
1: one or both positive

h	v	$\hat{\chi}$	context
1	1	0	4
1	0	0	3
1	-1	0	2
0	1	0	1
0	0	0	0
0	-1	1	1
-1	1	1	2
-1	0	1	3
-1	-1	1	4

Expected Sign

34

@NTUEE DSP/IC Lab

Magnitude Refinement

$\sigma[m,n]$	h+v	Context
1	x	2
0	≥1	1
0	0	0

Any of immediate horizontal and vertical neighbors are significant.

$\sigma[m,n]$: Whether magnitude refinement has been applied
h, v : Defined as in zero coding

Context Statistics

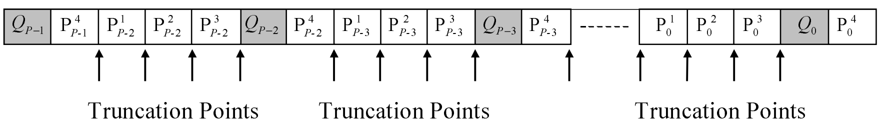
- Each context
  - T: 9 bit count of total number of symbols
  - N: 4 bit count of total number of symbol '1'
  - Both count fields are initialized to 1
  - Whenever either counts overflows, both count fields are normalized with a right shift
  - At end of a bitplane, both count fields of zero coding (10 contexts) are normalized (right shift) so long as: T>32 & N>2
  - Probability can be calculated by a lookup table with 13 bits (16kB)
  - Note:
    - Adapted to the expectation that highly skewed context will be skewed toward the 0 symbol
    - Adaptation is much quicker towards a less skewed distribution

Fractional Bitplane Block Coding

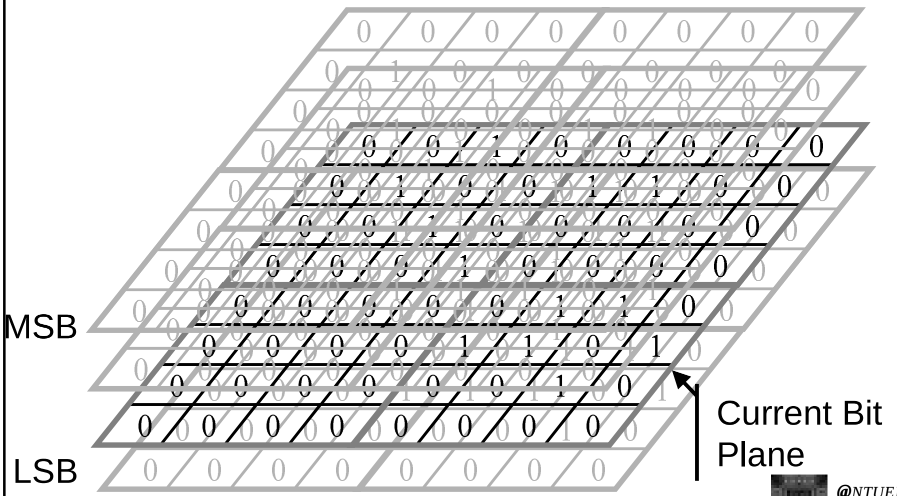
- Within each bitplane, there are 4 passes:
  - Forward significance propagation pass: P^1 (Zero Coding + Sign Coding)
    - Only for subblocks which are significant, and for all samples which are insignificant and have a preferred neighbor (LL, HL, LH: with at least one horizontal significant neighbor, HH: at least one significant diagonal neighbors)
  - Backward significance propagation pass: P^2 (Zero Coding + Sign Coding)
    - Similar to P^1, except in reverse order and consider samples with at least one significant neighbor in the 8 immediate neighbors and not processed in P^1
  - Magnitude refinement pass: P^3 (Magnitude Refinement)
    - Significant coefficients not processed in P^1 and P^2
  - Normalization pass: P^4 (Zero Coding + Run-Length Coding + Sign Coding)
    - All remaining coefficients, except those subblocks identified by quadtree

An Embedded Block Bit-Stream and Its Possible Truncation Points

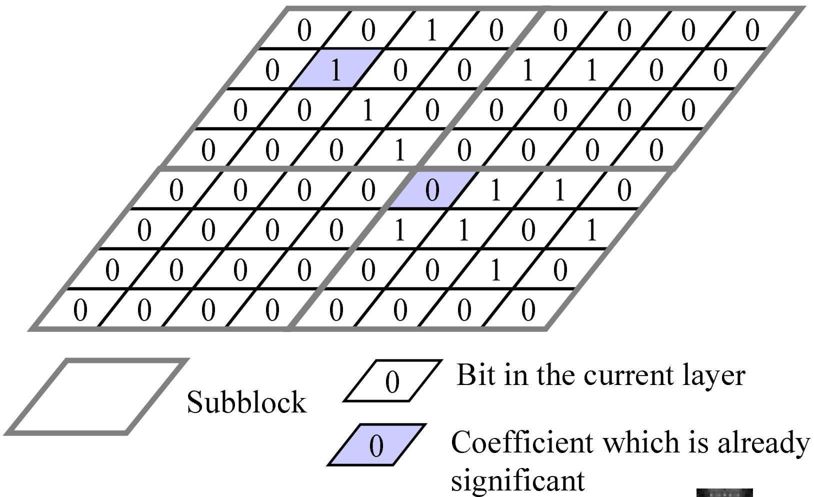
Q_b: quad-tree information for bit-plane b
P_b^i: pass i information for bit-plane b
P-1: Most significant bit for the block



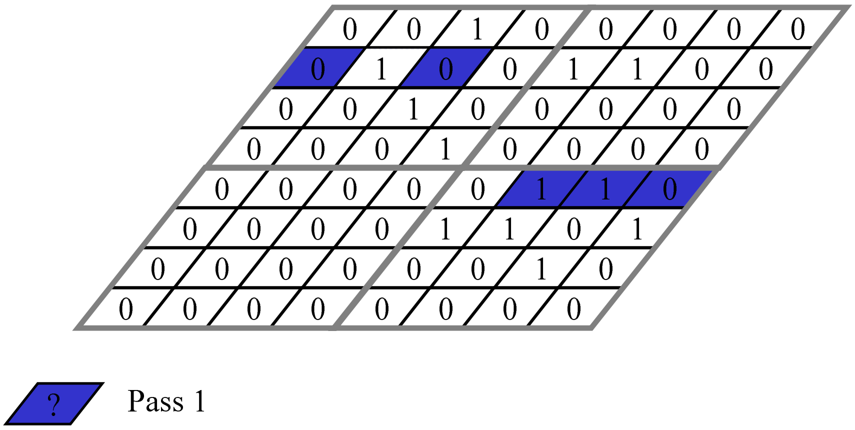
An Example of Bitplane



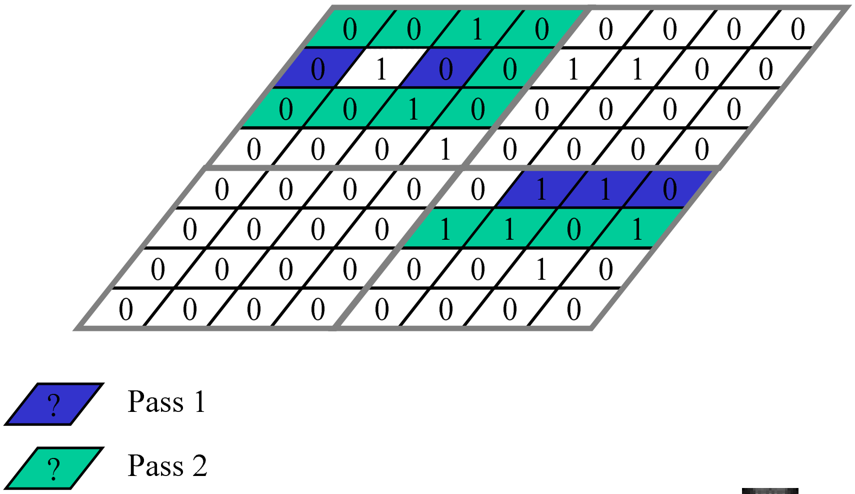
An Example of Bitplane (Continued)



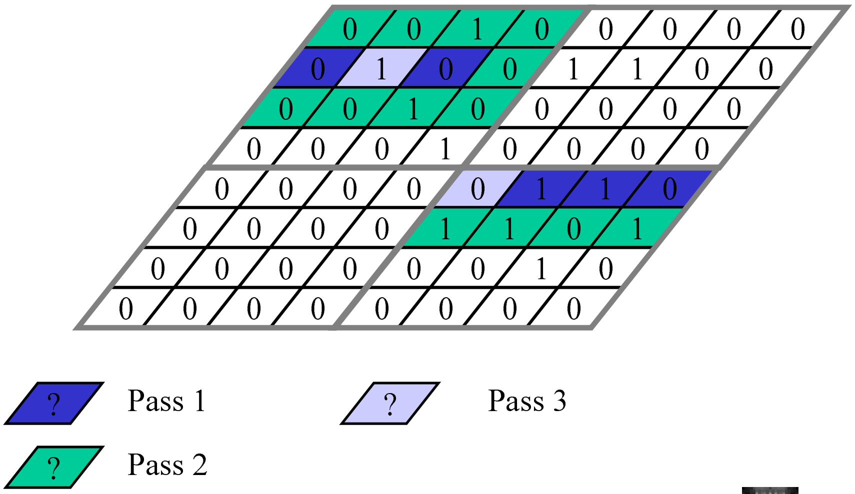
Pass 1: Forward Significance Propagation



Pass 2: Backward Significance Propagation



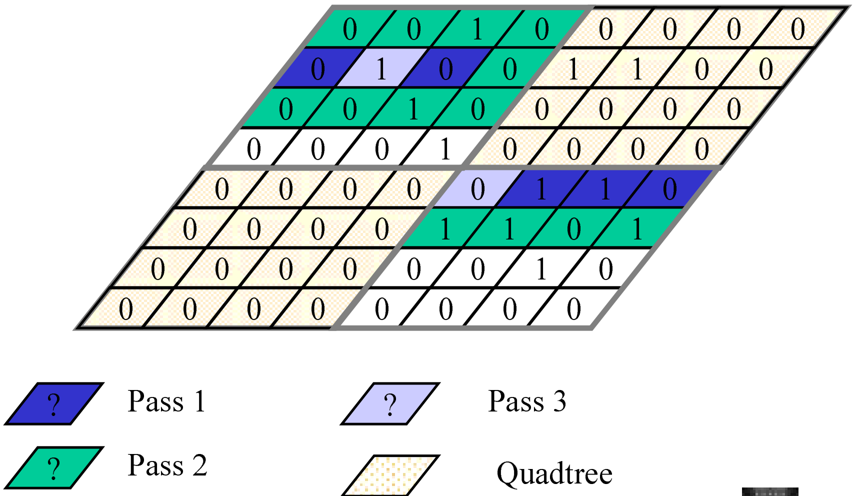
Pass 3: Magnitude Refinement



43

@NTUEE
DSP/IC Lab

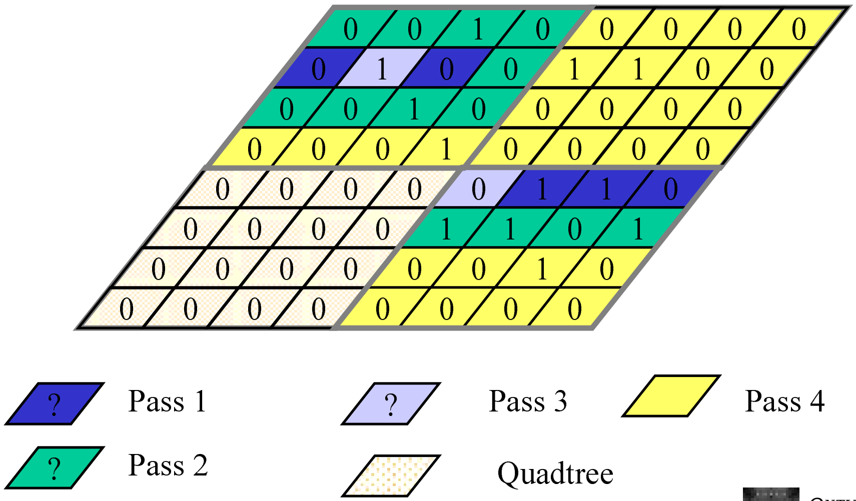
Pass 4’: Quadtree Coding of Subblocks



44

@NTUEE
DSP/IC Lab

Pass 4: Normalization



Post Compression R-D Optimization

- Compress each block independently at a high enough rate
- Store rate distortion information for each coding pass
- After Compressing a number of blocks revisit the bit stream
  - Allocate bits among different blocks
  - Remove all extra bits from blocks in an R-D optimal way
- The rate Distortion Optimization is only done at the encoder
- The decoder does not need to know about R-D information
- Post Rate Distortion Optimization is essential
  - Rate control
  - High compression performance
  - Optimal Decodable Embedded bit stream

Rate-Distortion Optimization

- For each block
  - Embedded encode each block with the fractional bitplane coder
  - The end of each coding pass is a viable truncation point, record the rate R_i^k and distortion D_i^k, where i indexes the block, k indexes truncation point
  - Select a set of monotonically decreasing truncation points:

    1) Set N_i = {the set of all truncation points}, set $S_i^0 = \infty$

    2) Set p = 0,

    3) For k=1,2,3,4,...

    If k belongs to N_i , Set $\Delta R_i^k = R_i^k - R_i^p$ and $\Delta D_i^k = D_i^p - R_i^k$

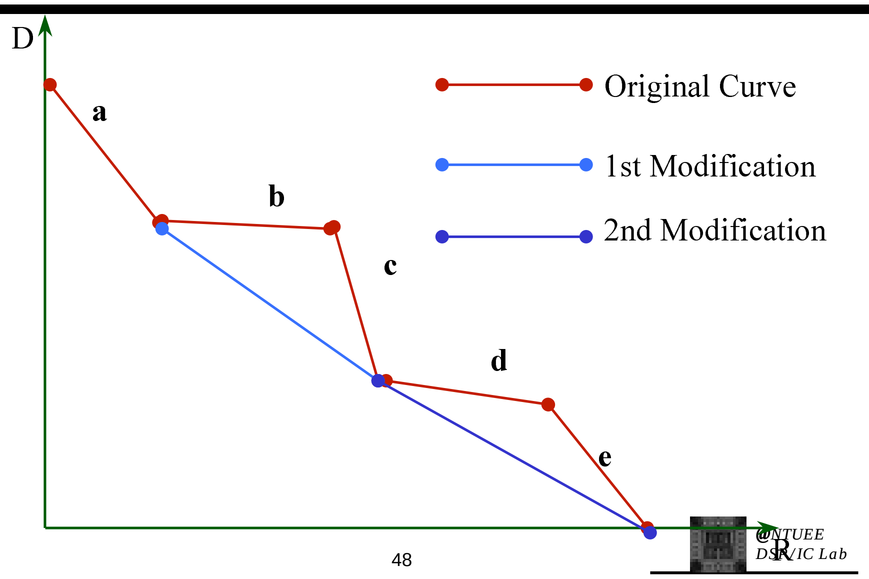
    Set $S_i^k = \Delta D_i^k / \Delta R_i^k$, if p≠0 and $S_i^k > S_i^p$ then remove p from N_i and go to step (2)

    Otherwise, set p = k

47

@NTUEE DSP/IC Lab

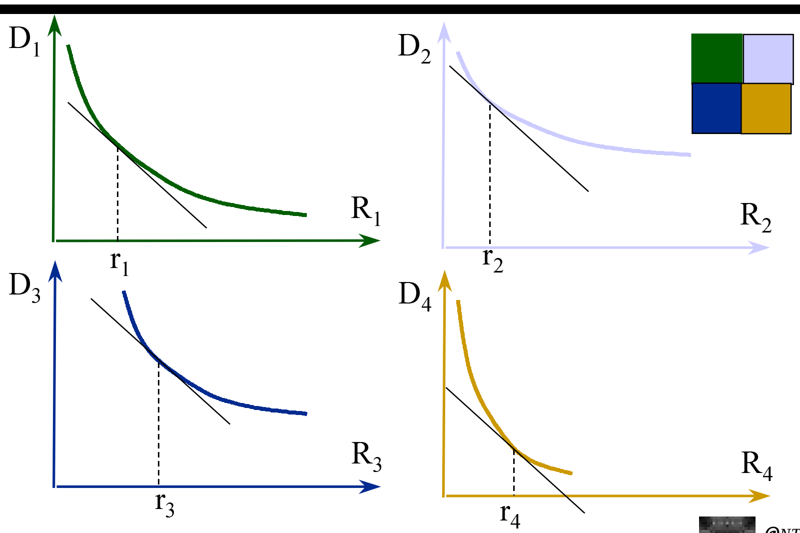
Monotonically Decreasing R-D Slope



48

@NTUEE DSP/IC Lab

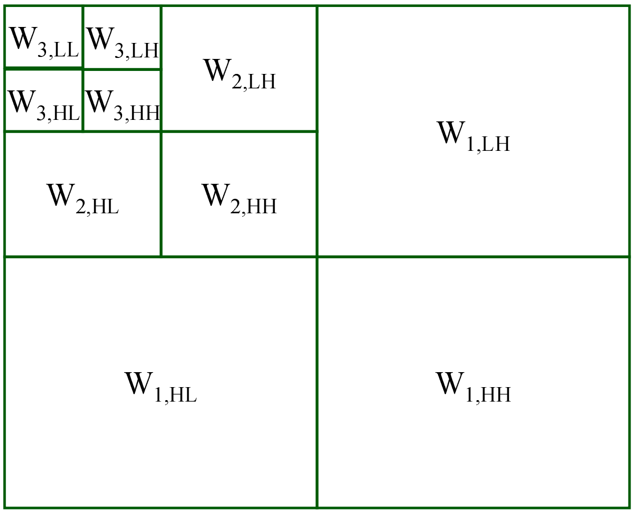
Global R-D Optimization



Visual Weighting

- Fixed visual weighting
  - Contrast Sensitivity Function (CSF) weights
  - Implementation
    - Multiply/ divide the CSF weights
    - Adjust quantization step size according to the weights
- Visual progressive coding
  - Enables the **change** of viewing conditions (visual weighting factor) during the embedding process
  - Improve the **subjectiv**e image quality over the **entire** bit rate range
  - Implementation - use the weighting factor to change the **order** of coding rather than the **value** of coding

CSF Visual Weighting

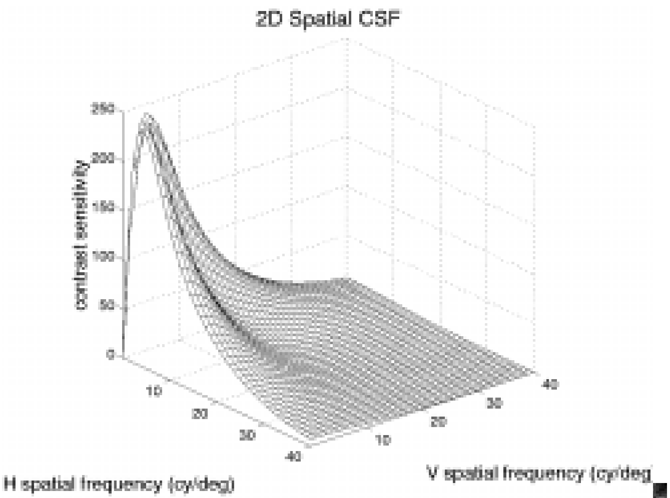


Encoder: Divide

Decoder: Multiple

@NTUEE DSP/IC Lab

2D CSF Function



@NTUEE DSP/IC Lab

VIP - Change of Weighting

Rate	**Visual Weight**	
$r^{(0)}$ -- $r^{(1)}$	$w_0^{(0)}, w_1^{(0)}, \cdots, w_n^{(0)}$	→ 
$r^{(1)}$ -- $r^{(2)}$	$w_0^{(1)}, w_1^{(1)}, \cdots, w_n^{(1)}$	→ 
⋮	⋮	⋮
$r^{(m)}$ --	$w_0^{(m)}, w_1^{(m)}, \cdots, w_n^{(m)}$	→ 

Image Viewing at Low Bitrate

Image Viewing at High Bitrate

Visual Weighting
Traditional vs Visual Progressive

- Traditional:
  - Multiplying/ dividing the coefficient by the weighting factor, quantize if necessary

    $$x_i' = x_i \cdot w_i \quad \text{or} \quad x_i' = Q\left[\frac{x_i \cdot w_i}{q_i}\right]$$
  - The binary representation of the coefficient changes with the weighting factor
  - Change of the weighting factor cause difficulties in the embedded coding
- Visual progressive coding (VIP)
  - Use the weighting factor to change to the **coding order**, rather than the **coding content**

Traditional Weighting

X_1: 13	0 1 1 0 1	w_1=0.333 →	0 0 1 0 0
X_2: 19	1 0 0 1 1	w_2=0.5 →	0 1 0 1 0
X_3: 15	0 1 1 1 1	w_3=1 →	0 1 1 1 1
X_4: 08	0 1 0 0 0	w_4=0.7 →	0 0 1 1 0

Embedded Coded

@NTUEE DSP/IC Lab

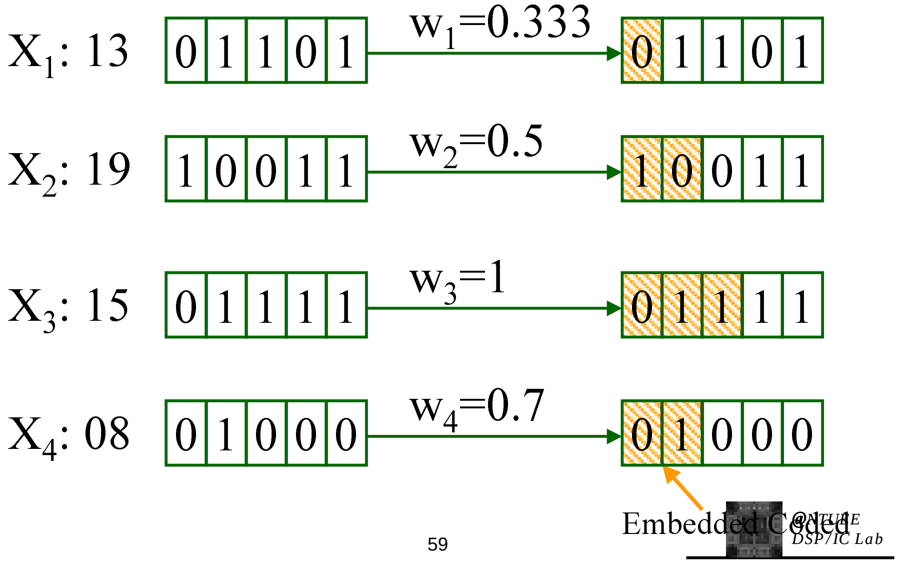
Traditional: Weighting Changes

X_1: 13	0 1 1 0 1	w_1=0.5 →	0 0 1 1 0
X_2: 19	1 0 0 1 1	w_2=0.333 →	0 0 1 1 0
X_3: 15	0 1 1 1 1	w_3=0.5 →	0 1 0 0 0
X_4: 08	0 1 0 0 0	w_4=1.0 →	0 1 0 0 0

The binary representation has changed, which part has been coded, and what to do next?

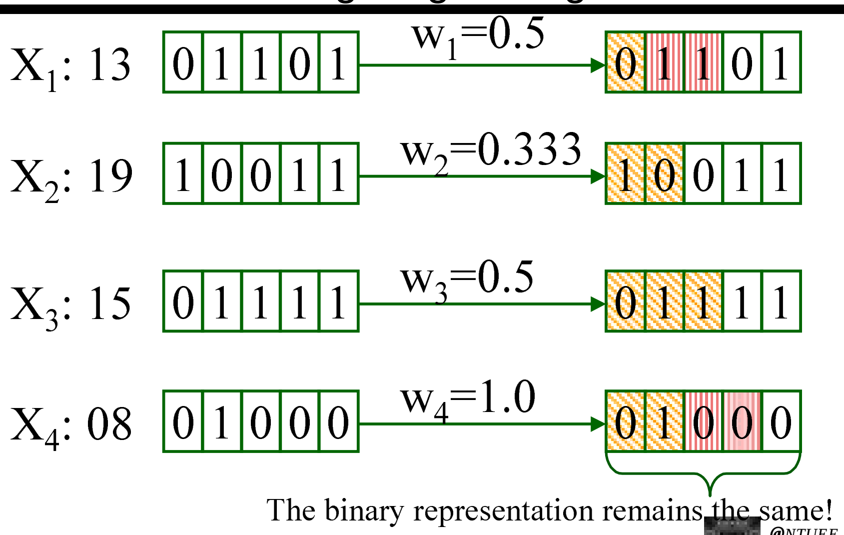
@NTUEE DSP/IC Lab

VIP Weighting



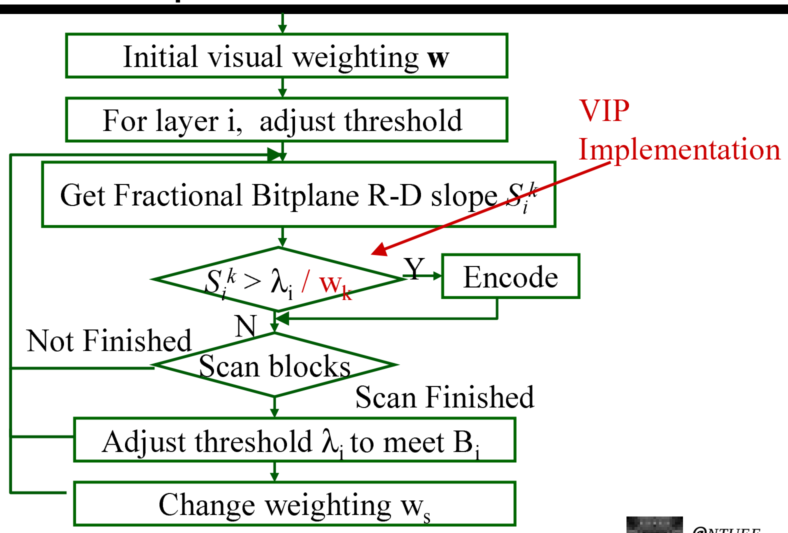
@NTUEE DSP/IC Lab

VIP: Weighting Changes



@NTUEE DSP/IC Lab

VIP Implementation - VM3a



61

@NTUEE DSP/IC Lab

Region of Interest Coding (ROI)

- Allows certain parts of image to be coded with better quality
- Idea:
  - Calculate wavelet transform of whole image/time
  - calculate ROI mask, i.e., set of needed coefficients
- The Feature and Issue of ROI
- Encoding in done in two ways
  - Sequence based mode
  - Scaling based mode

62

@NTUEE DSP/IC Lab

The Feature and Issue of ROI

- Features
  - Progressive mode
  - Reconstruction of RoI is faster than the rest
  - ROIs need to be coded losslessly and the rest of the image only visually losslessly.
  - Savings in storage space and transmission speed
  - Application : Internet, Interactive browser
- Issues
  - No difference coding
  - Visually annoying edges do not occur around the RoI
  - ROI is specified in the beginning, or in the middle of encoding
  - Speed is flexible by the user
  - Independent of the quantization or entropy coding scheme

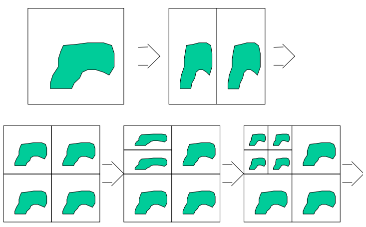
Derive the ROI mask

- Derive the mask

$$M(x,y)=\begin{cases}1 & \text{The wavelet coefficient } (x,y) \text{ is needed}\\ 0 & \text{Accuracy on } (x,y) \text{ can be sacrificed without affecting ROI}\end{cases}$$

Trace backward to see if the coefficients corresponding to the mask

Calculate the lossless mask



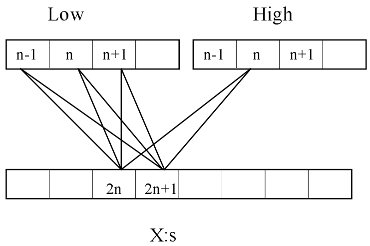
Derive the Corresponding Coefficients

- Example: S+P transform
  - the inverse transform

$$X(2n+1) = L(n) + \frac{H(n)}{2} + \left(\frac{L(n-1) - L(n+1)}{128}\right)$$

$$X(2n) = L(n) - \frac{H(n)}{2} - \left(\frac{L(n-1) - L(n+1)}{128}\right)$$

- Corresponding coefficients



Two Implementations of ROI

- Sequence based code
  - ROI coefficients are coded as independent sequences
  - Allows random access to ROI without fully decoding
  - Can specify exact quality/bitrate for ROI and the BG
- Scaling based mode:
  - Scale ROI mask coefficients up (decoder scales down)
  - During encoding, the ROI mask coefficients are found significant at early stages of the coding
  - ROI always coded with better quality than BG
  - Can't specify rate for BG and ROI

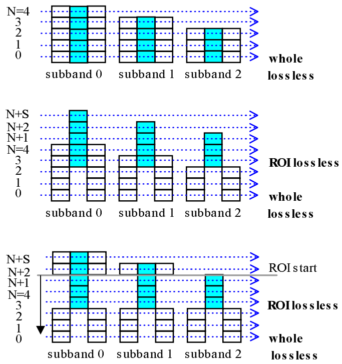
Scaling Based Mode

- ROI can be specified in the beginning or in the middle of encoding

No ROI coding

ROI coding from beginning

ROI coding from middle



@NTUEE DSP/IC Lab

Multiple ROIs

- Mask coefficient
  - If the i^{th} bit is on then it means that the corresponding wavelet coefficient belongs to the i^{th} ROI
  - If a wavelet coefficient belongs to ROI 0 and 3, its corresponding ROI mask coefficient has the binary value `00001001`
- Overlapping ROIs
  - ROIs may overlap in the wavelet domain, even if they do not overlap in the image domain

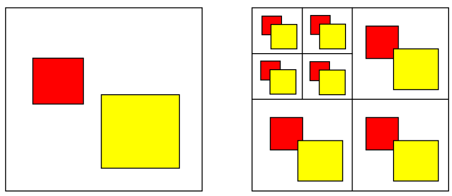


Image domain

Wavelet domain

@NTUEE DSP/IC Lab

Encoding of Multiple ROIs

- Sequence Based Mode
  - The wavelet coefficients that belong to more than one ROI are repeated in all the sequences to which their ROIs correspond.
- Scaling Based Mode
  - If WT coefficients associated with several ROIs, thy are assumed to belong to the ROI whose scaling value is the highest

Conclusion

- JPEG 2000 will offer an still image compression scheme with not only better efficiency, but also many useful functionalities
- It is still an on-going, open standard
- Try to keep the core part of the standard from slipping too long: Three-parts are proposed:
  - Part 1: Core that is required of every decoder
  - Part 2: Value-added technologies (one meeting behind the Part 1)
  - Part 3: Advanced technology extensions (may not be realized)