Lecture 6
Still Image Coding

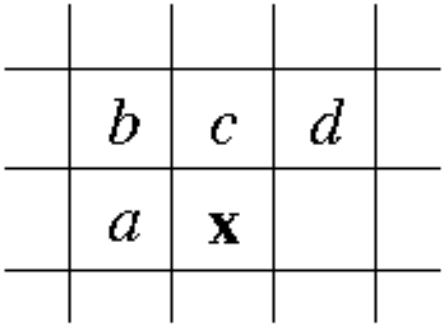
- Lossless Image Compression Methods
  - Predictive coding
  - RL coding
  - Ziv-Lempel coding
- Lossless Image Compression Standards
  - Fax Standards: ITU-T G3, G4 and ISO JBIG
  - ISO JPEG LS
- Lossy Image Compression Standards
  - JPEG
  - JPEG 2000

Lossless Predictive coding

- The new information in a pixel is defined as the difference between the actual value and a prediction of that pixel based on a set of its previously coded neighbors.
- Examples of predictors:

$$\hat{x} = \text{int}\left(\frac{a+c}{2}\right)$$

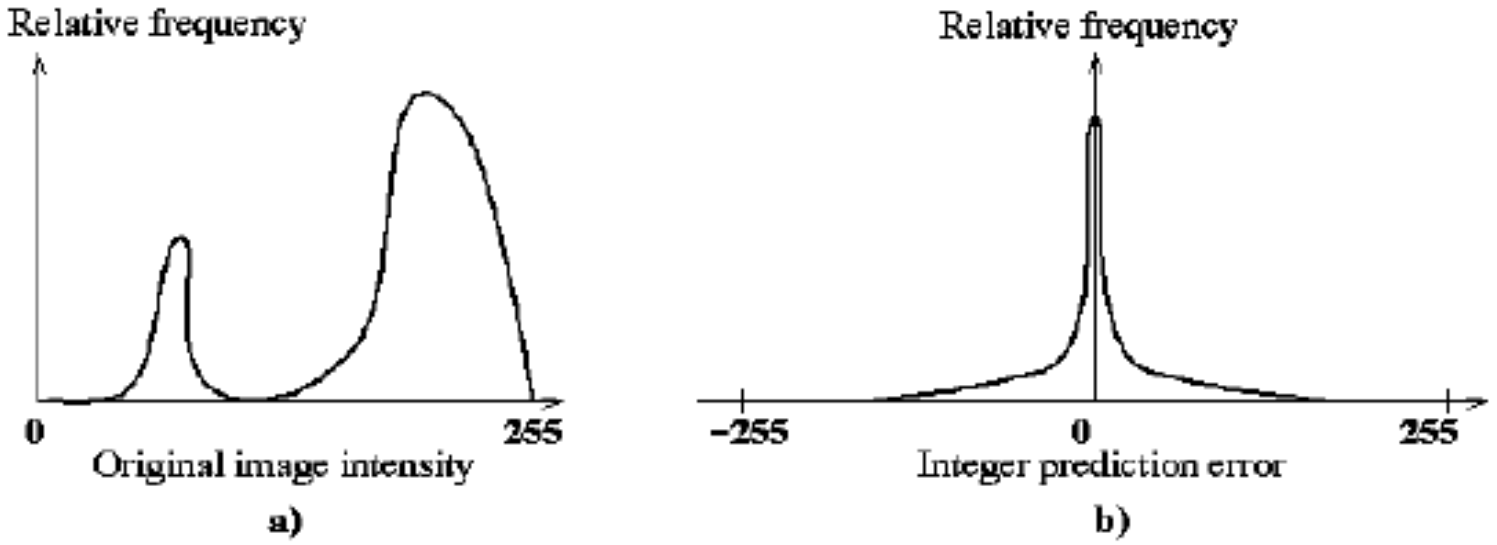
$$\hat{x} = \text{int}\left(\frac{a+b+c+d}{4}\right)$$



The prediction is rounded to the nearest integer to ensure an integer prediction error.

- The predictive transform removes spatial redundancy.

Prediction error



- If the input image has the dynamic range (0,255), then the prediction error image has a dynamic range of (-255,255).
- However, the prediction error image has much smaller entropy compared with the original image, as the histogram of the prediction error is highly peaked about 0.

Run Length coding (RLC)

- 1-D RLC: Encode each contiguous sequence of 0's or 1's in a left to right scan of a row by its length.
  - specify the first run of each row
  - assume that each row begins with a white run, whose run length may be zero.
- More compression is achieved by VLC of the run lengths.
- 2-D RLC: Exploits 2-D correlation patterns.
- RLC is the standard approach in FAX coding.

Example

- Encode

0 0 0 0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0

- RLC

4 0 1 2 0 0 9 2

Bit-Plane/Run-Length Coding

- *Bit-Plane Decomposition*: Decompose a multilevel (monochrome or color) image into a series of binary images.
- The levels of an *m*-bit gray-scale image can be represented as

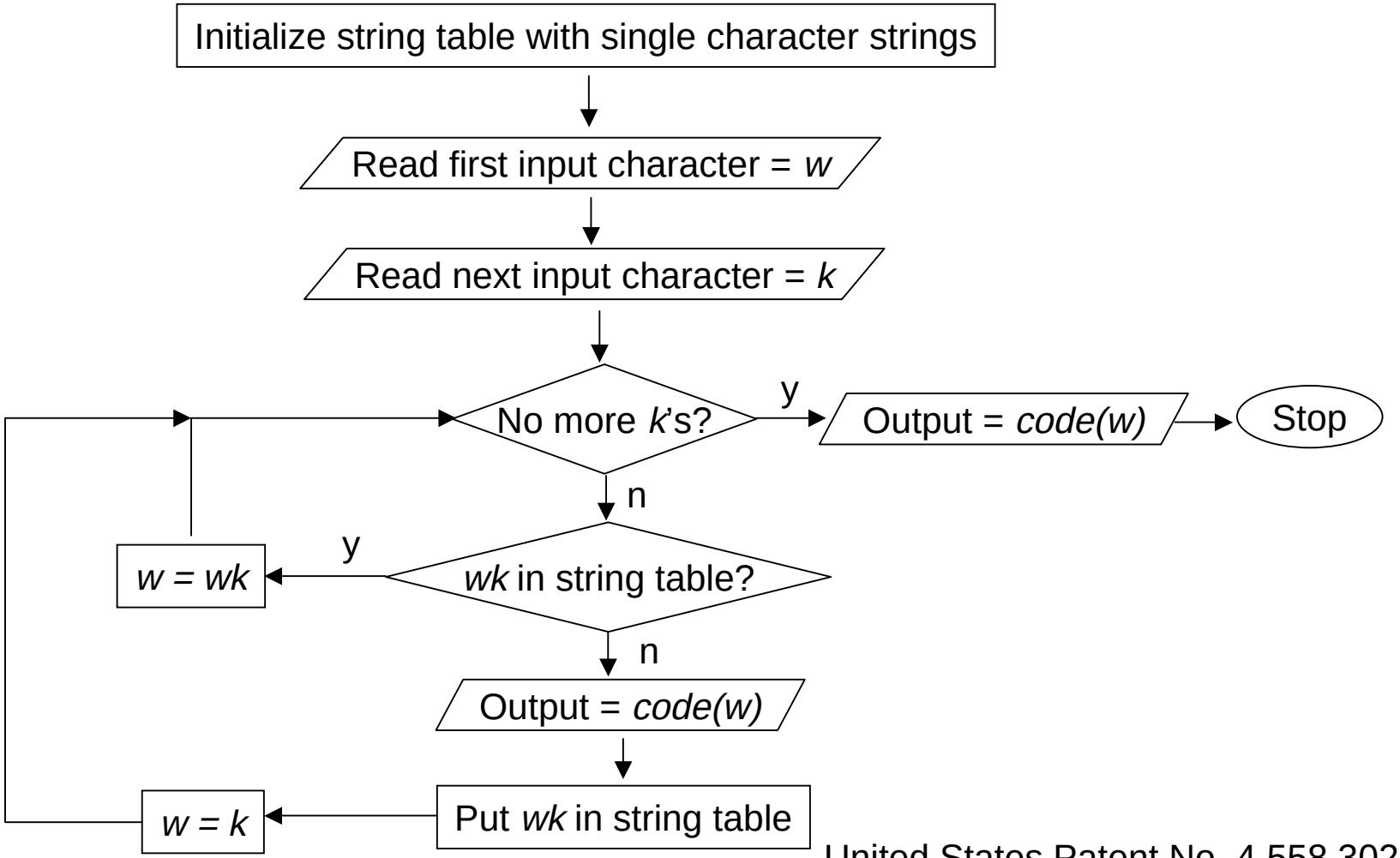
$$a_{m-1}\, 2^{m-1} + a_{m-2}\, 2^{m-2} + \ldots + a_1\, 2 + a_0\, 2^0$$

- where a_i , i=0,…, m-1 are either 0 or 1.

Ziv-Lempel coding

- The input sequence is parsed into nonoverlapping blocks of variable size to construct a dictionary of blocks seen thus far. The dictionary is initialized with the binary symbols 0 and 1.
- The length of the next block to be parsed, *n*, is set equal to that of the ongest word that is already in the dictionary.
  - If the next parsed block, say *w*, is already in the dictionary, the encoder sends to the channel a fixed-length code for the index of w. Before continuing parsing, *w* is concatenated with the next input symbol and added to the dictionary.
  - If *w* is not in the dictionary, the encoder sends a fixed-length code for the first *n-1* bits of *w* (which must be in the dictionary) and adds *w* to the dictionary.
- This procedure is repeated until the entire input sequence is coded.

LZW Algorithm



United States Patent No. 4,558,302,
Patented by Unisys Corp.

Example (1/2)

Input	*Dictionary*	*Index*	*Output*
	0	0	
	1	1	
0	01	2	0
1	11	3	1
1	10	4	1
0	00	5	0
0			
1	011	6	2
1			
0	100	7	4
0			
1	010	8	2
0			
1			
1	0110	9	6
0	.	.	

Example (2/2)

- Assume a dictionary of 16 phrases (4 bit index).

Data	a	b	b	a	a	b	b	a	a	b	a	b	b	a	a	a	a	b	a	a	b	b	a	b	a
Encryption	0	1	1	0	2		4		2		6			5		5		7			3		8		

- This case does not result in compression.
  - *Source*: 25 characters in a 2-character alphabet require a total of 25 bits.
  - *Output*: 13 pointers of 4 bits require a total of 52 bits.
- This is because the length of the input data in this example is too short.
- In practice, the Lempel-Ziv algorithm works well only when the input data is sufficiently large and there is sufficient redundancy in the data.

Dictionary			
Index	**Entry**	**Index**	**Entry**
0	a	8	aba
1	b	9	abba
2	ab	10	aaa
3	bb	11	aab
4	ba	12	baab
5	aa	13	bba
6	abb	14	
7	baa	15	

Still image compression standards

ITU-T G3/G4	Binary images (non-adaptive) - for fax transmission of documents (1980)
ISO JBIG	Binary and bit-plane encoding - to handle progressive transmission and halftones (1994)
ISO JPEG	Still frame gray scale and color image - Block based DCT (1993)
ISO JPEG-LS	New Lossless Coding Standard - nonlinear prediction, context-based Rice-Goulomb coding (1997)
ISO JPEG 2000	New Wavelet Coding Standard

Bilevel Image Compression Standards

- ITU-T Group 3: Non-adaptive, 1-D or 2-D RLC. Typical compression ratio: 15:1
- ITU-T Group 4: Only 2-D RLC is allowed. Same Huffman tables. 30% better performance
- ISO JBIG: Adaptive arithmetic coding, adaptive templates for halftones

1-D RLC

- Each line of the image is encoded as a series of VLC that represent the run-lengths of alternating white and black runs in a left-to-right scan.
- Since the statistics of white and black runs are different, we have different codebooks to represent the white and black run-lengths.
- The symbol probabilities were obtained from a number of test documents both typed and handwritten in several languages.
- ***Run***

1-D RLC

Run Length	*White Codeword*	*Black Codeword*
0	00110101	0000110111
⋮	⋮	⋮
63	00110100	000001100111

Run Length	*White Codeword*	*Black Codeword*
64	11011	0000001111
128	10010	000011001000
⋮	⋮	⋮
1728	010011011	0000000100101

Each line begins with a white run, whose length may be zero, and ends with an EOL codeword.

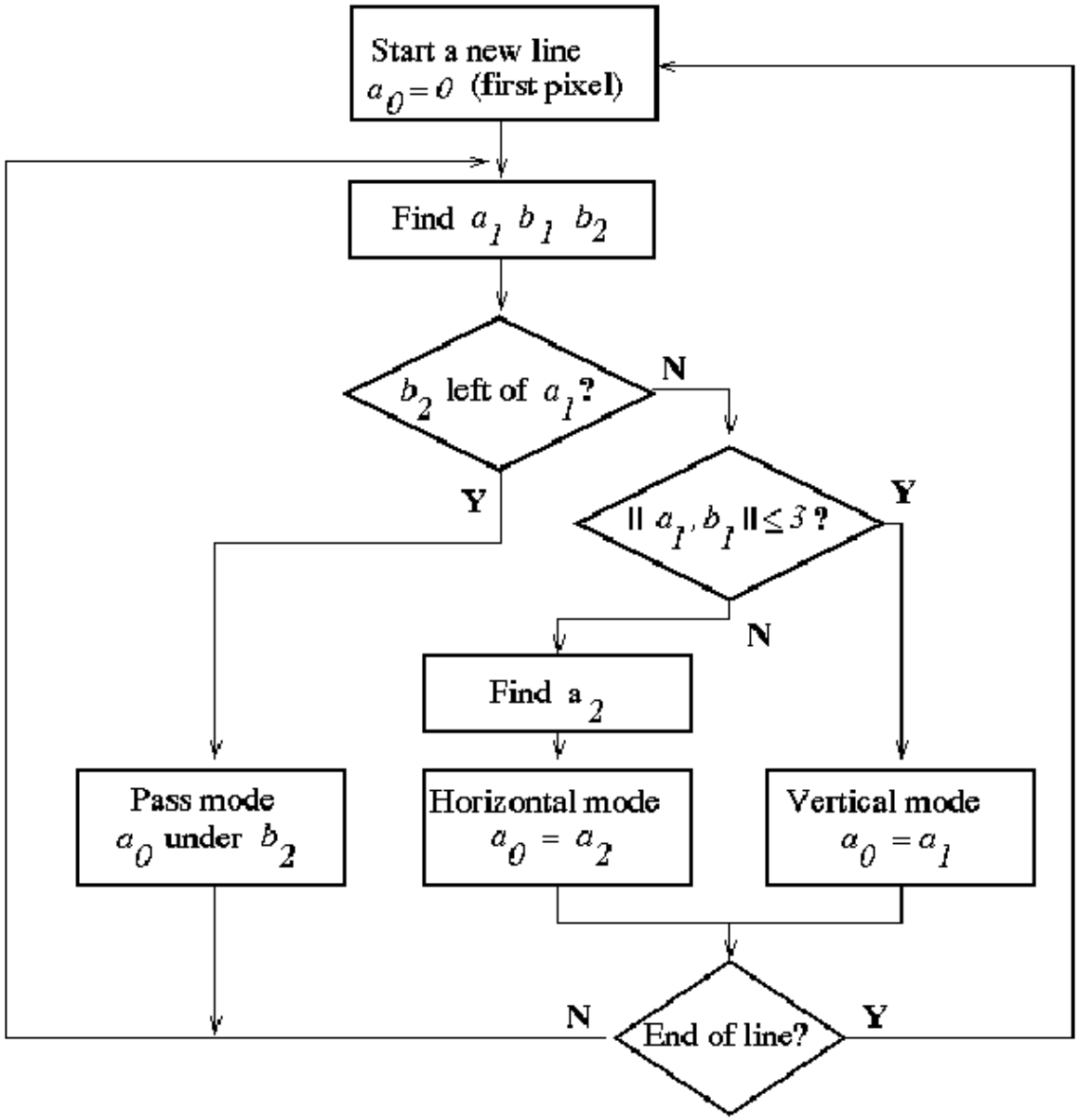
2-D RLC

- Two-dimensional RLC is based on the principle of relative address coding.
  - The position of each black-to-white or white-to-black transition is coded with respect to the position of a *reference element* a_0 that is on the current coding line.
  - The first reference element is set as an imaginary white element to the left of the first pixel of each new coding line.
  - Given a_0 , find four *changing elements*, a_1 , a_2 , b_1 , and b_2 .
  - Next, select one of the three modes: *pass mode*, *vertical mode*, or *horizontal mode*, and establish a new *reference element* for the next coding cycle.

ITU-T 2-D codes

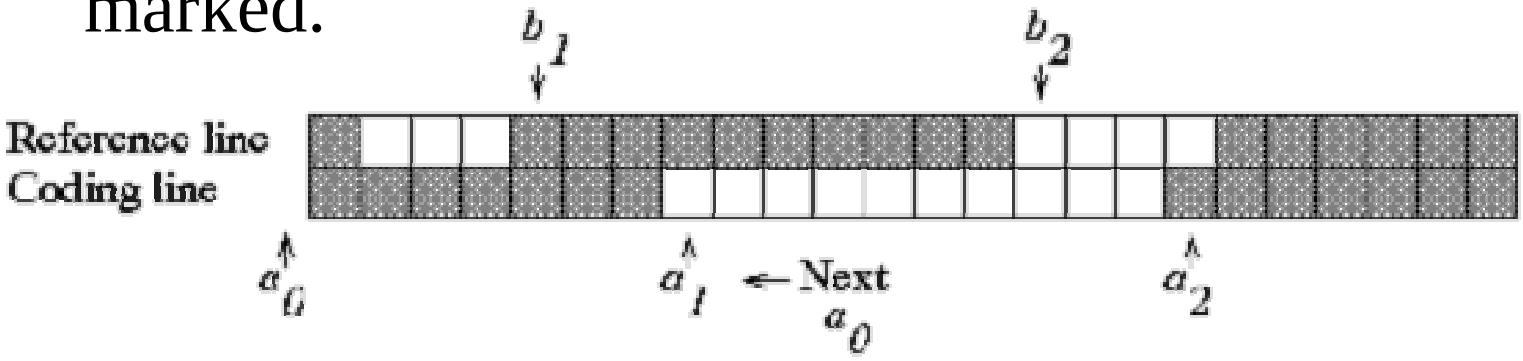
Mode	*Codeword*
Pass	0001
Horizontal	001+M(a0,a1)+M(a1,a2)
Vertical	
a1 below b1	1
a1 one to the right of b1	011
a1 two to the right of b1	000011
a1 three to the right of b1	0000011
a1 one to the left of b1	010
a1 two to the left of b1	000010
a1 three to the left of b1	0000010
Extension	0000001xxx

Algorithm


Start a new line
$a_0 = 0$ (first pixel)
Find a_1 b_1 b_2
b_2 left of a_1?
N
Y
$|| a_1, b_1 || \leq 3$?
Y
N
Find a_2
Pass mode
a_0 under b_2
Horizontal mode
$a_0 = a_2$
Vertical mode
$a_0 = a_1$
N
End of line?
Y

Example

- Suppose we have two lines of a binary image, and the initial positions of the parameters a_0 , a_1 , a_2 , b_1, and b_2 are as marked.
- Since, b_2 is to the right of a_1, and the distance between a_1 and b_1 is equal to 3, we are in the vertical mode, and the next value of $a_0 = a_1$ as marked.



JBIG standard

- Why a new standard?
  - To develop adaptive algorithms that outperform existing standards.
  - To accomodate progressive image transmission and digital halftones.
- JBIG gets another 30% better compression from G4 on text images by using adaptive binary arithmetic coding, and provides about 8:1 compression on digital halftones (using adaptive templates) which may expand with G3/G4.
- JBIG syntax also allows coding of images with more than one bit/pixel using intra-bit-plane encoding.

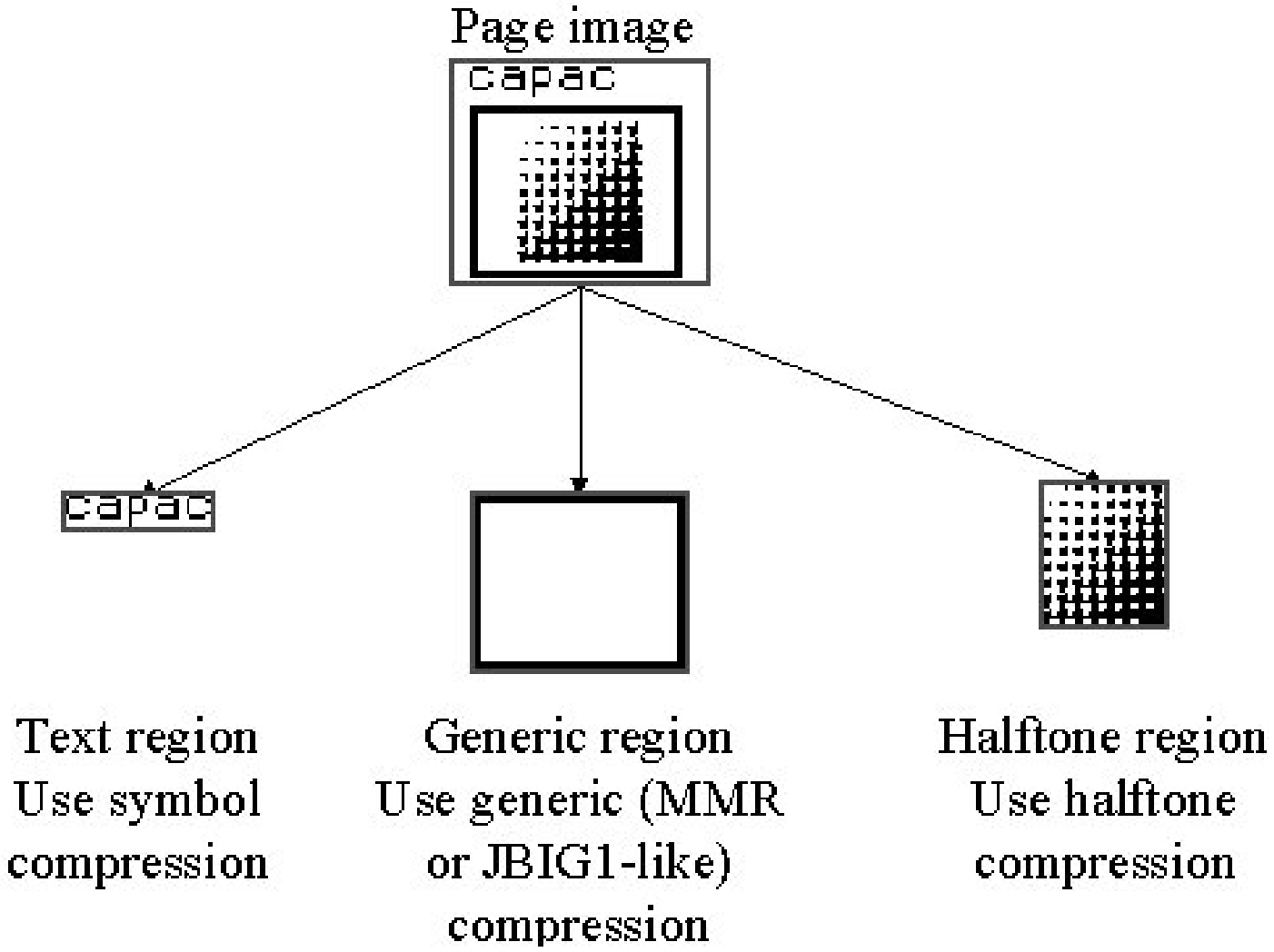
JBIG-2

- Bilevel image compression standard
- Designed for compact lossy and lossless coding
- Uses segmentation and type-specific encoding
- Designed to be embedded in other file formats

Region segmentation

- A page in JBIG2 file is broken up into various regions
  - Regions may overlap
  - Regions use different coding methods
  - Typically one text region, one generic region and one halftone region
  - Regions can be lossy
  - Refinement of regions is possible

Segmentation example



JPEG Standard (ISO-10918)

- JPEG describes a family of image compression techniques fro continuous tone still images
- JPEG is not a complete arhitecture for image exchange

JPEG (1/2)

- Joint Photographic Experts Group
- *Joint*: ISO, ITU-T (CCITT), IEC
- *Photographic*: continuous-tone still images
- Started in 1986 - An international standard in 1993
- JPEG2000 standardization is in progress

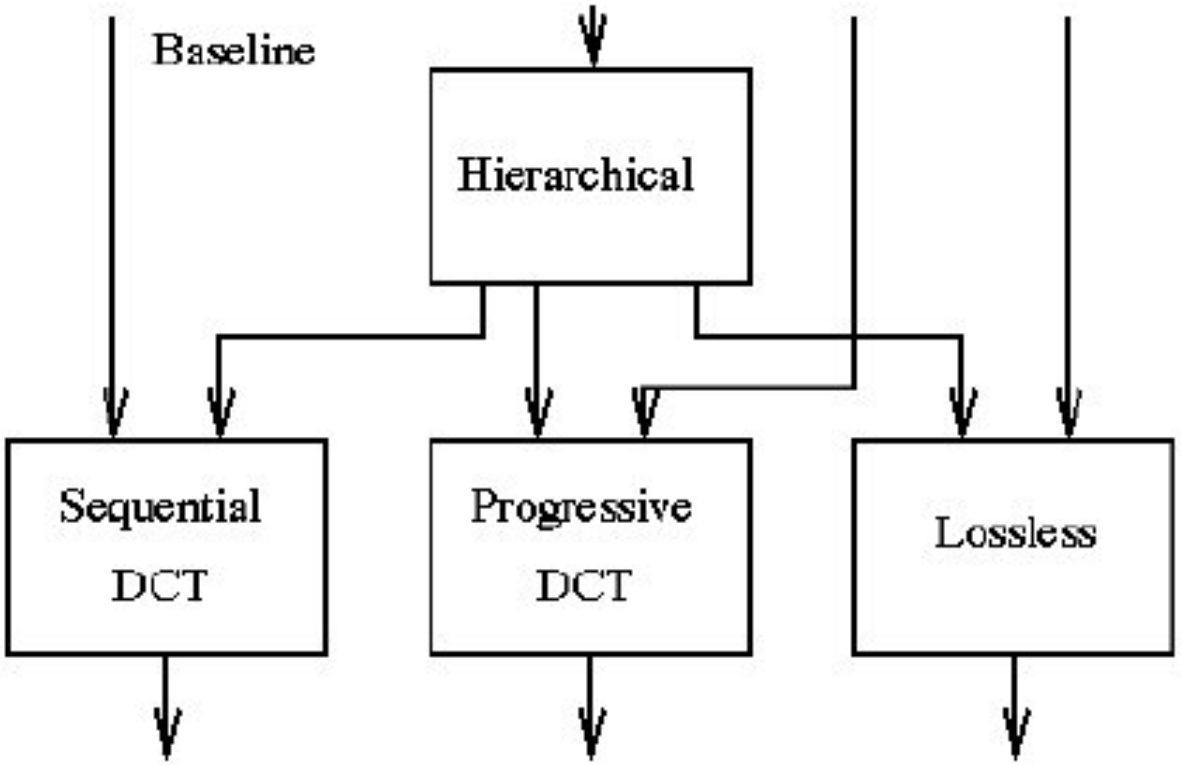
JPEG (2/2)

- JPEG describes a family of image compression techn. for continuous tone (grayscale or color) still images.
- JPEG is not a complete architecture for image exchange. No particular file format, spatial resolution or color space model is specified.
- Examples of image file formats that allow JPEG compressed images:
  - Tag Image File Format (TIFF TM) Revision 6.0 supports grayscale, RGB, CMYK, and YUV color spaces for JPEG baseline and Huffman lossless coding modes.
  - JPEG File Interchange Format (JFIF) is a minimal file format which enables JPEG bitstreams to be exchanged. It uses a standard color space (ITU-R 601-1).

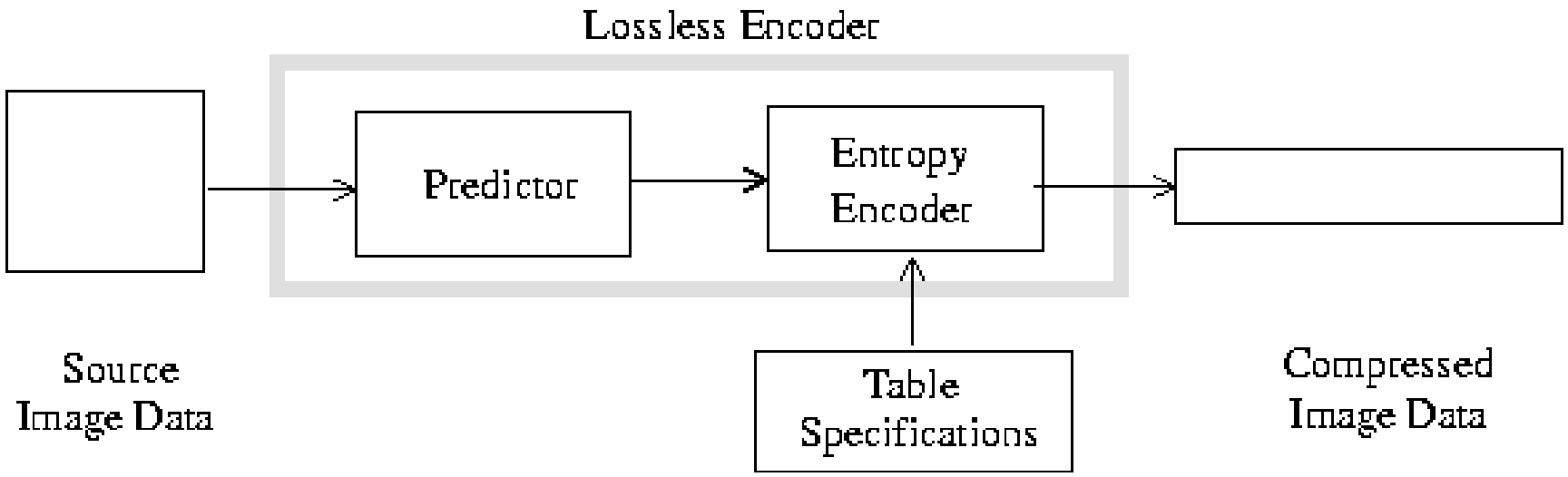
JPEG - Features

- Resolution independence
- Precision
- No absolute bit-rate targets
- Luminance-chrominance seperability
- Extensible

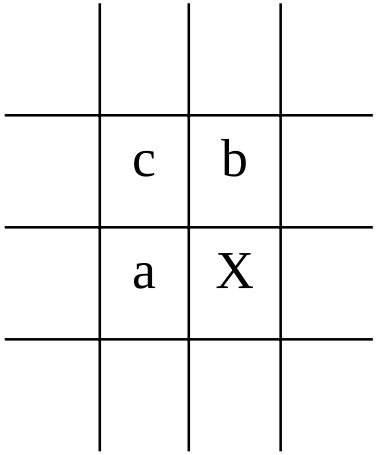
JPEG - Modes of operation



The Lossless JPEG standard (1/2)

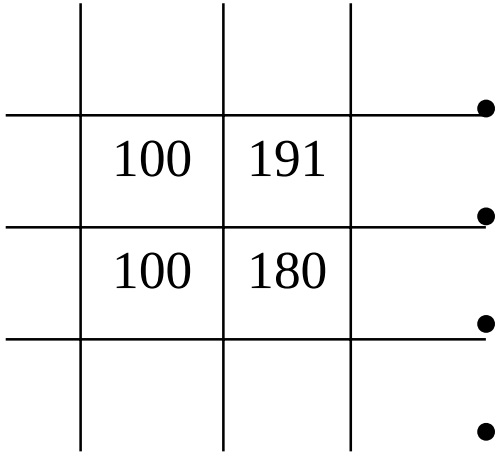


The Lossless JPEG standard (2/2)



- Try to predict the value of pixel X as prediction residual r=y-X
  - y can be one of 0, a, b, c, a+b-c, a+(b-c)/2,b+(a-c)/2
  - The scan header records the choice of y
  - Residual r is computed modulo 2^{16}
  - The number of bits needed for r is called its *category*, the binary value of r is its *magnitude*
  - The category is Huffman encoded

Example



- y=(a+b)/2 = 145
- r=145-180=-35
- Category (r) = 6, Magnitude (r) = 100011
- 1’s complement of cat (r) = 011100
- Rep(35)={6,011100}
- Let Huff. code(6) = 1110
- Code(-35)=1110011100

MSB=0 for numbers < 0

10 bits

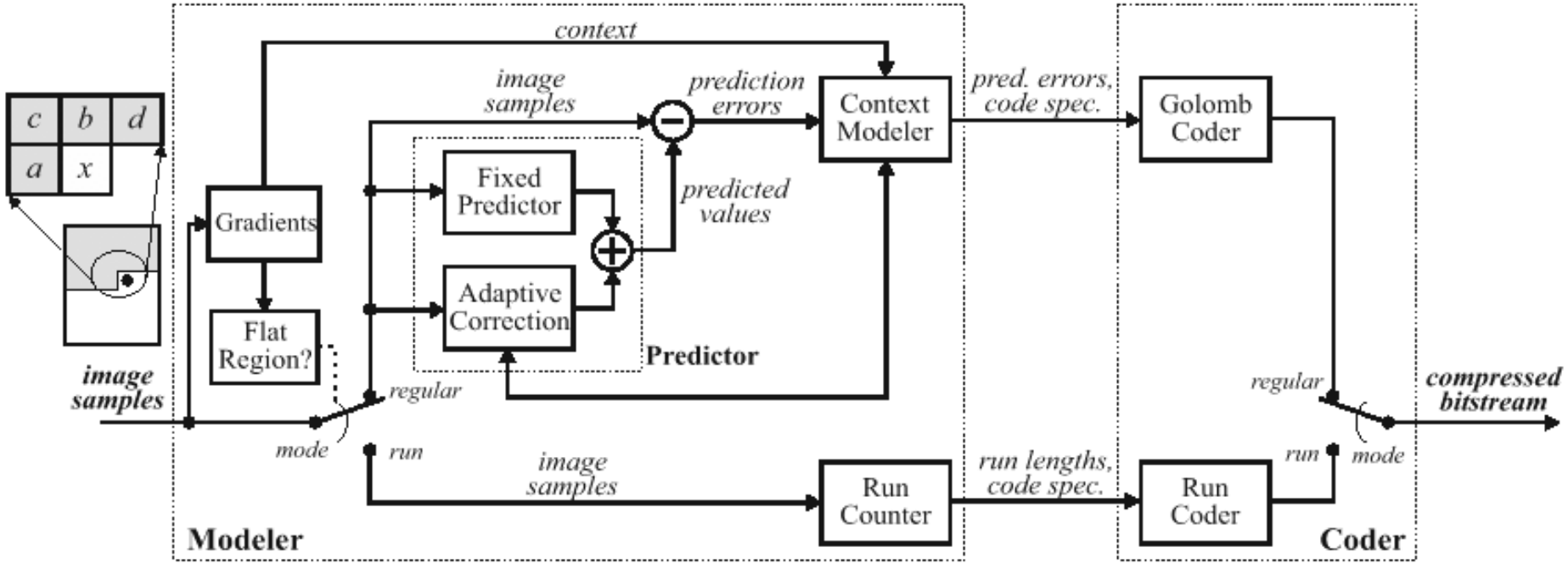
Predictors for lossless coding

selection-value	prediction
0	no prediction
1	A
2	B
3	C
4	A+B-C
5	A+((B-C)/2)
6	B+((A-C)/2)
7	(A+B)/2

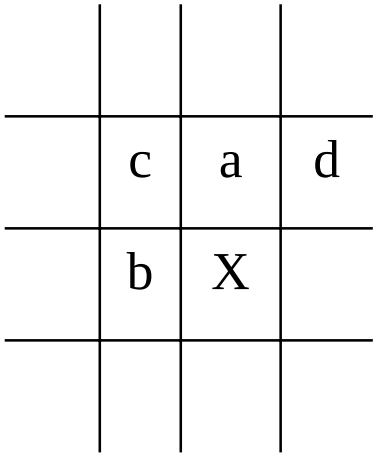
The JPEG-LS standard (1/5)

- Loco project (http://www.hpl.hp.com/loco/)
- Near-lossless encoding
  - decoder output does not differ from the input by no more than a pre-specified value
- JPEG-LS coder
  - Context modeling – encoding of a pixel depends on the previous pixels
  - Run-length encoder – for smooth parts of the image
  - Predictor – like in the lossless JPEG scheme
  - Error Coder – to reconstruct the difference between the prediction and the signal

The JPEG-LS standard (2/5)



The JPEG-LS standard (3/5)



- Context Model
  - two probability models: flat areas and edge-areas
  - compute d1=*d*-*a*, d2=*a*-*c*, d3=*c*-*b*
  - quantize d1, d2, d3 to Q1, Q2, Q3 using thresholds T1, T2, T3. For 8-bit images they are 3, 7, 21
  - any Q can take up to 9 possible values based on the threshold interval it is in

The JPEG-LS standard (4/5)

- Run-mode coder
  - If gradients are close to 0, the encoder gets into a run-mode
  - So long as $|x-b|$, the encoder reads subsequent samples
  - Then it returns the run-length
  - If end-of-line is reached, it encodes the last sample
- At the pixel x the predictor computes prediction error $e=y-x$ where

$$y=\begin{cases}\min(a,b) \text{ if } c \quad \max(a,b)\\ \max(a,b) \text{ if } c \quad \min(a,b)\\ a+b-c\end{cases}$$

  - Remove any prediction bias
- Remap e to $e' = 2e$ for positive e and $-2e-1$ for negative e
- Encode e' with Golomb-Rice encoding

The JPEG-LS standard (4/5)

- The parameter k for Golomb-Rice encoding is obtained by
  - $k = \left\lceil \log_2 \frac{A[i]}{N[i]} \right\rceil$
  - A[i]: accumulated sum of prediction errors
  - N[i]: number of prediction residuals seen in context i
- Removal of prediction bias
  - Idea: the prediction error must follow a 2-sided geometric distribution
  - Computed using A[i] – how?
  - B[i], sum of errors after correction and C[i], the correction itself are also stored

Golomb-Rice Compression

- Take a source having symbols occurring with a geometric probability distribution
  - $P(n)=(1-p_0)\,p_0^n$ n 0, $0<p_0<1$
  - Here is P(n) the probability of run-length of n for any symbol
- Take a positive integer m and determine q, r where n=mq+r.
- G_m code for n: $\overbrace{0\ldots01}^{q}$ + bin(r)

has $\log_2 m$ bits if r >sup($\log_2 m$)

- Optimal if
  - $m = \left\lceil -\frac{\log(1+p_0)}{\log(p_0)} \right\rceil$

JPEG codec

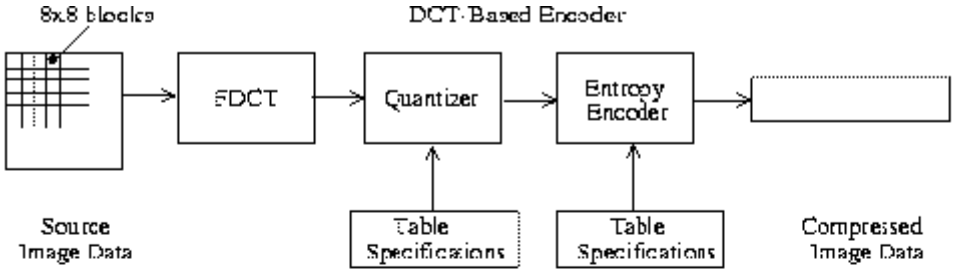


Figure 1. DCT-Based Encoder Processing Steps

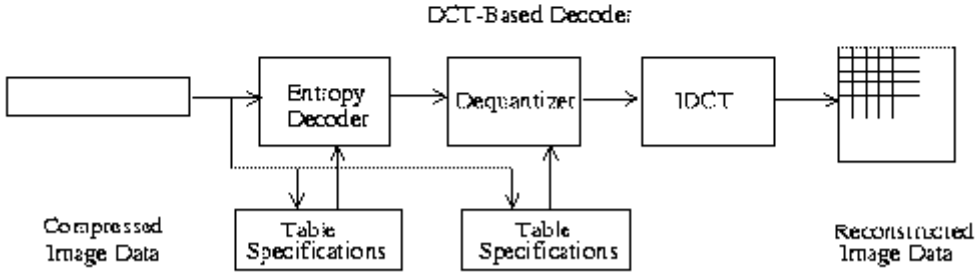


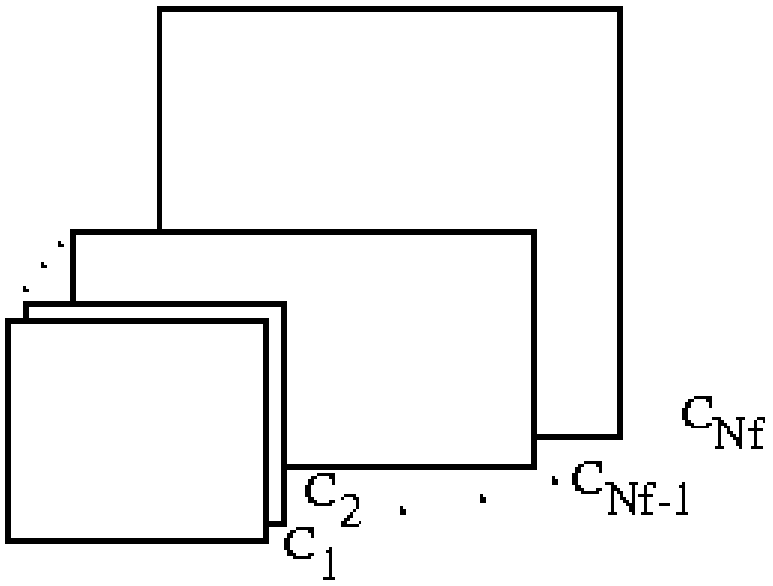
Figure 2. DCT-Based Decoder Processing Steps

JPEG - Baseline algorithm

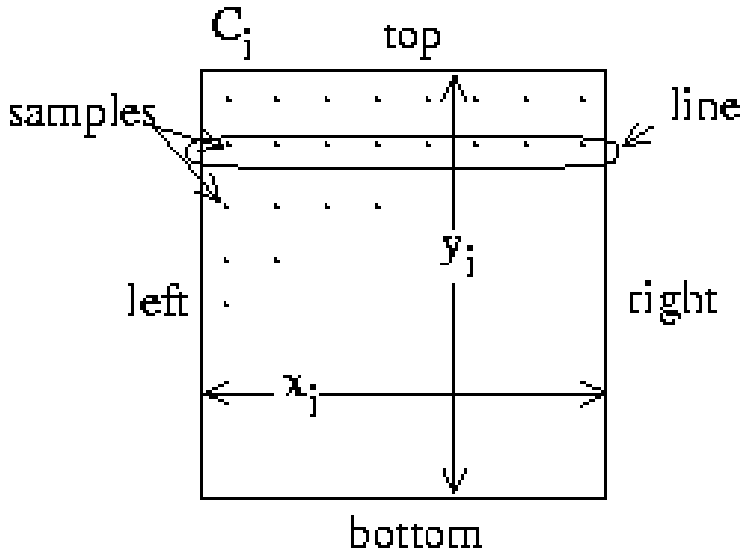
- DCT computation
- Quantization
- VLC

- Input resolution: 8 bits/pixel
- DCT values: 11 bits/sample

JPEG source model



(a) Source image with multiple components



(b) Characteristics of an image component

JPEG - Color images

Y1	Y2	Y3	Y4
Y5	Y6	Y7	Y8
Y9	Y10	Y11	Y12
Y13	Y14	Y15	Y16

Cr1	Cr2
Cr3	Cr4

Cb1	Cb2
Cb3	Cb4

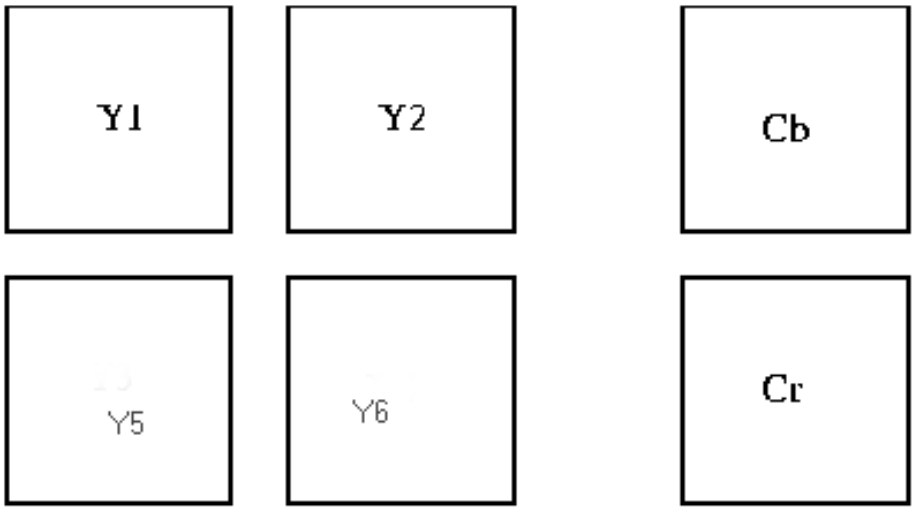
$$Y = 0.3R + 0.6G + 0.1B$$

$$Cr = \frac{B - Y}{2} + 0.5$$

$$Cb = \frac{R - Y}{1.6} + 0.5$$

Minimum coded unit

- Components are arranged as MCUs – Smallest group of interleaved data units

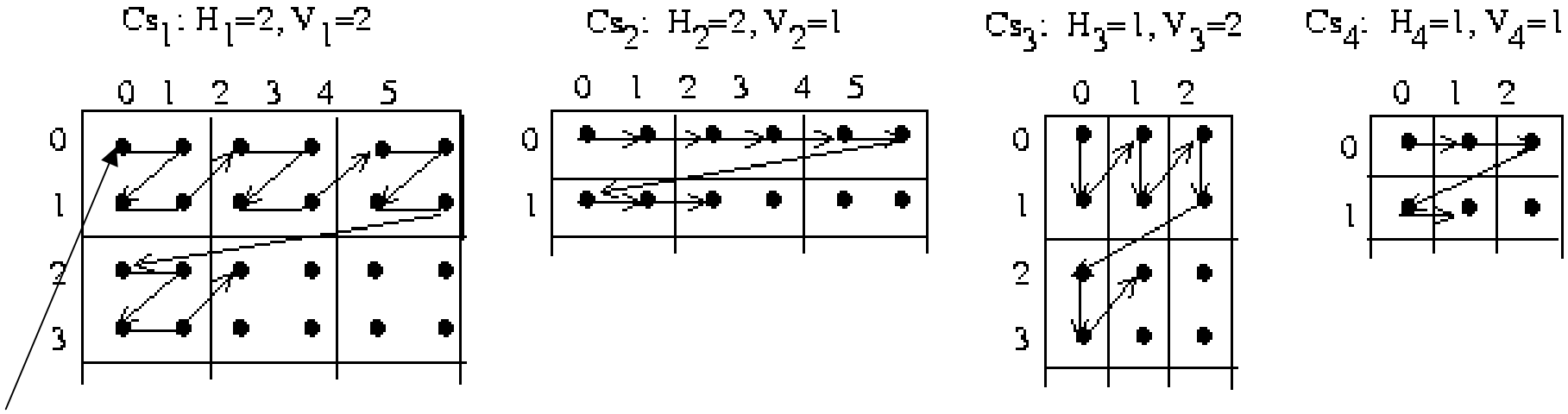


Minimum coded unit for color images

Interleaved vs non-interleaved scan

- Noninterleaved
  - Scan1: Y1,Y2,Y3,…,Y16
  - Scan2: Cr1,CR2,Cr3,Cr4
  - Scan3: Cb1,Cb2,Cb3,Cb4
- Interleaved
  - Y1,Y2,Y5,Y6,Cr1,Cb1,Y3,Y4,Y7,Cr2,Cb2

Generalized interleaved data ordering



Data unit: 8x8 blok of DCT coeff or individual sample

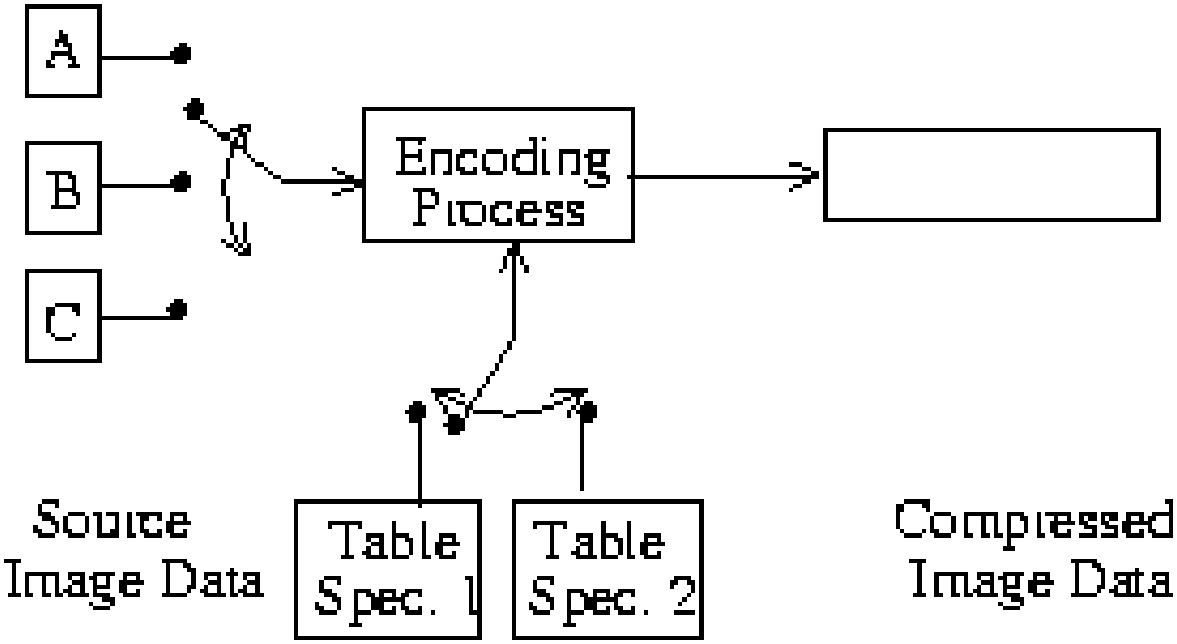
$$MCU_1 = d^1_{00}\ d^1_{01}\ d^1_{10}\ d^1_{11}\ d^2_{00}\ d^2_{01}\ d^3_{00}\ d^3_{10}\ d^4_{00},$$

$$MCU_2 = d^1_{02}\ d^1_{03}\ d^1_{12}\ d^1_{13}\ d^2_{02}\ d^2_{03}\ d^3_{01}\ d^3_{11}\ d^4_{01},$$

$$MCU_3 = d^1_{04}\ d^1_{05}\ d^1_{14}\ d^1_{15}\ d^2_{04}\ d^2_{05}\ d^3_{02}\ d^3_{12}\ d^4_{02},$$

$$MCU_4 = \underbrace{d^1_{20}\ d^1_{21}\ d^1_{30}\ d^1_{31}}_{Cs_1\text{ data units}}\ \underbrace{d^2_{10}\ d^2_{11}}_{Cs_2}\ \underbrace{d^3_{20}\ d^3_{30}}_{Cs_3}\ \underbrace{d^4_{10}}_{Cs_4},$$

Encoding process



Tables

- Quantization table
  - 64 entries
  - must be specified by the application
  - Up to 4 tables
- Huffman tables
  - for (runlegth, size)
  - Up to 4 tables

Quantization tables

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

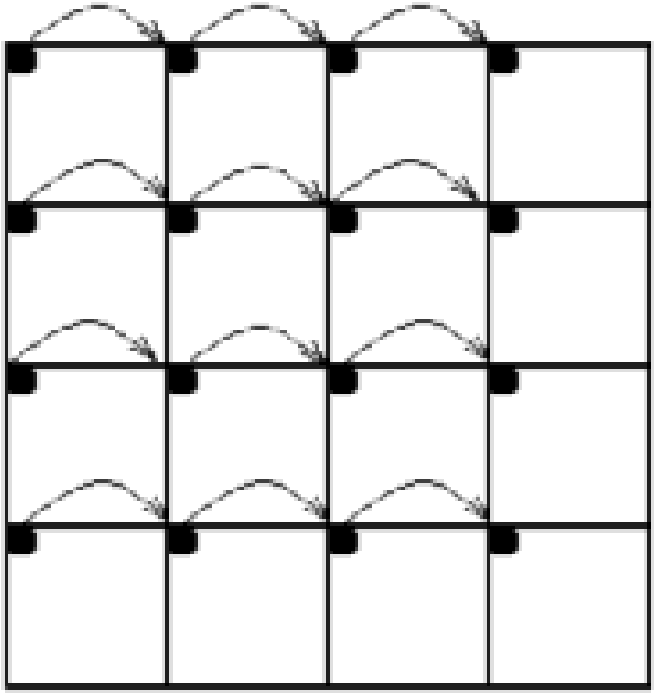
Luminance Quantization Table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Chrominance Quantization Table

Coding of DC coefficients

DPCM coding of DC coefficients

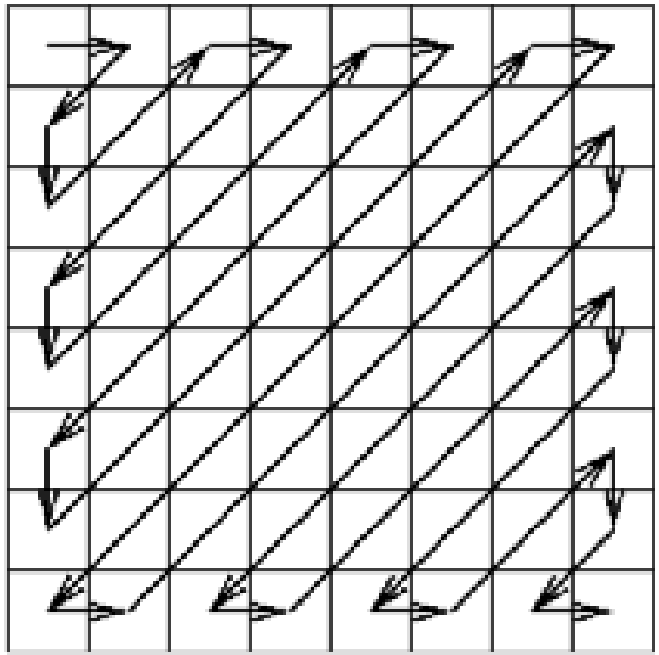


Coding of AC coefficients

Because the location of the retained coefficients vary from block to block, the quantized AC coefficients are zigzag scanned and ordered into (run, level) pairs.

Level = the value of a nonzero coefficient;

Run = the number of zero coefficients preceeding it.



These (run, level) pairs are entropy coded, that is, longer codes for less frequent pairs and vice versa).

Entropy coding (1/2)

- From zig-zag arrangemnt, form the symbol sequence

  (runlength, size),(coefficient_value)

  – runlength: number of zeros

  – coefficient_value: nonzero coefficient

  – size: size of the nonzero coefficient

Entropy coding (2/2)

- Can be Huffman coding or arithmetic coding
- Very few *codec*s implement arithmetic coding
- Arithmetic coding gives 5-10% better results

JPEG baseline – Example (1/3)

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-38	-19	-43	-59	-56
-66	-69	-60	-15	16	-24	-62	-55
-65	-70	-57	-6	26	-22	-58	-59
-61	-67	-60	-24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

Level shifted 8 × 8 original image block (shifted by 128)

-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

The result of the forward DCT

JPEG baseline – Example (2/3)

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

The JPEG recommended quantization matrix

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

The quantized coefficient array

JPEG baseline – Example (3/3)

- 1-D coefficient sequence after zig-zag scanning

-26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 0 -1 2 0 0 0 0 0 -1 –1 EOB

where EOB denotes the end of the block.

- Coding the DC coefficient: Encode the difference between DC coefficients of the current and previous blocks.
- Coding the AC coefficients: Define (RUN,LEVEL) as symbols;

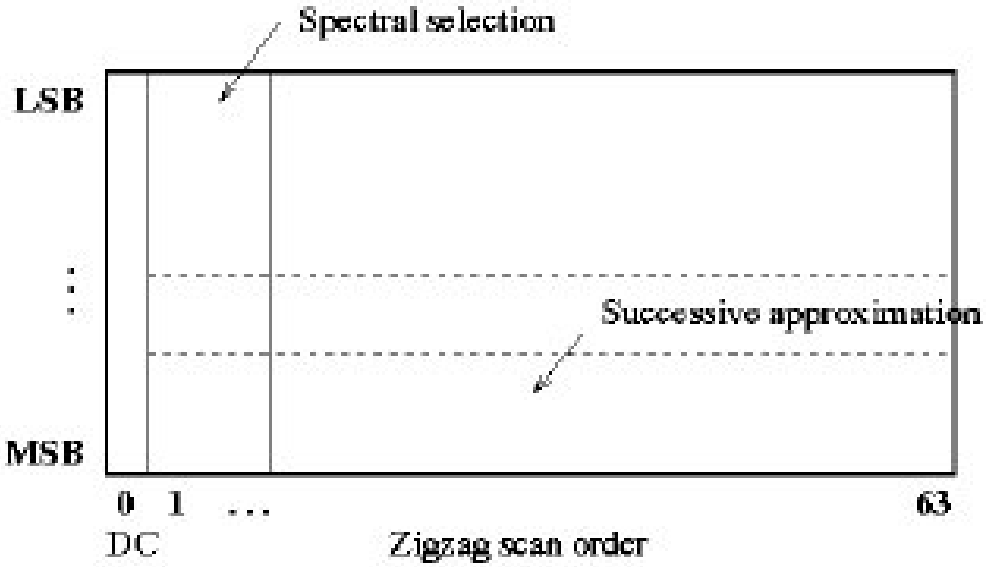
e.g., (0,-3); (0,1); (0,-3); ...

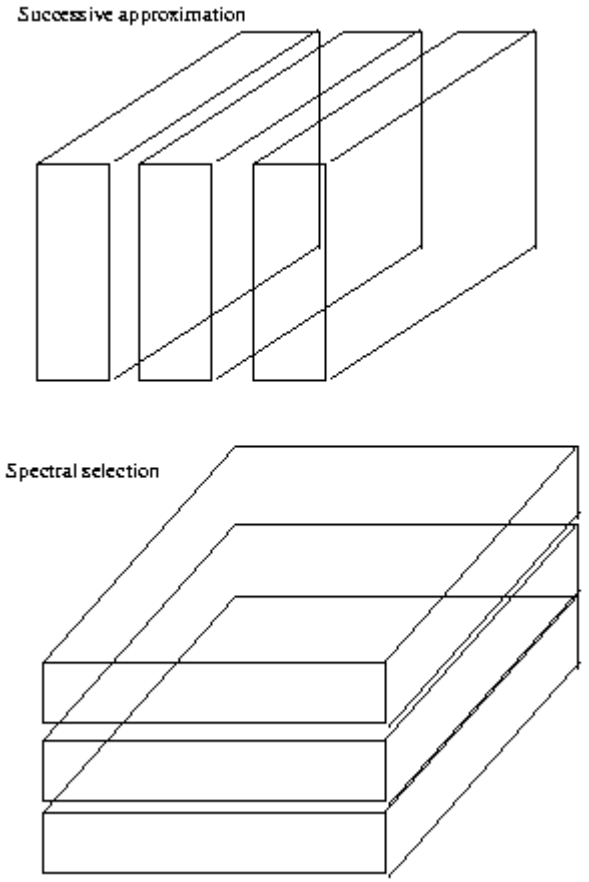
- These symbols are VLC (Huffman or arithmetic) coded.

Progressive encoding (1/2)

- Progressive mode consists of a sequence of ‘scans’ each of which codes a part of the quantized DCT coefficients
- Each image *component* is coded in multiple scans
- Needs an image sized buffer at the output of the quantizer, before entropy encoder
- Spectral selection: send a band of frequencies from each scan
- Successive approximation: send a band of bits

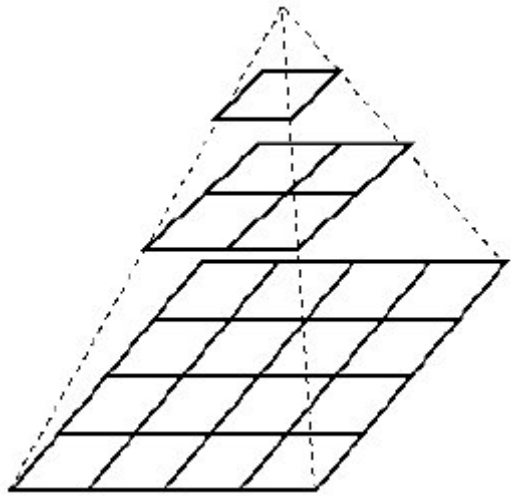
Progressive encoding (2/2)



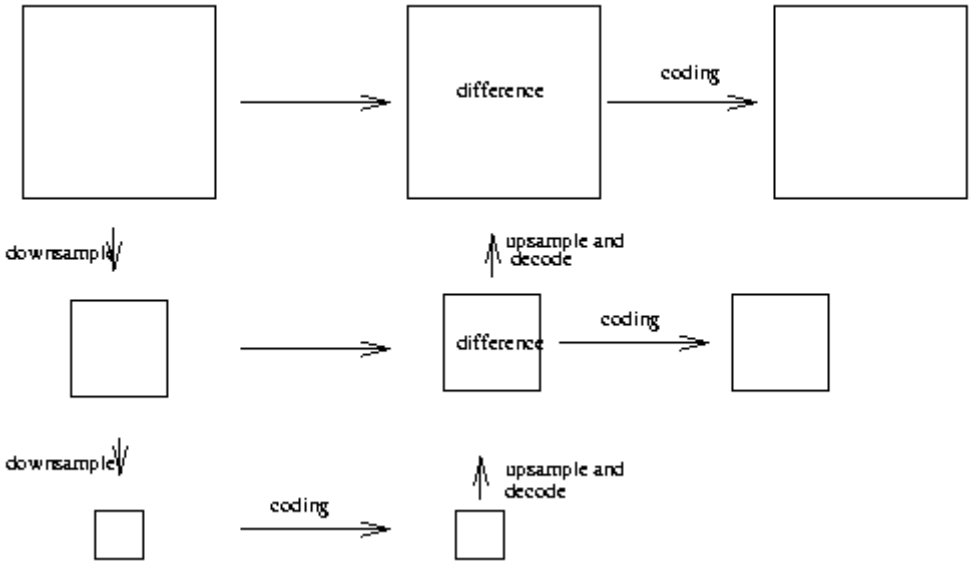


JPEG – Hierarchical (1/2)

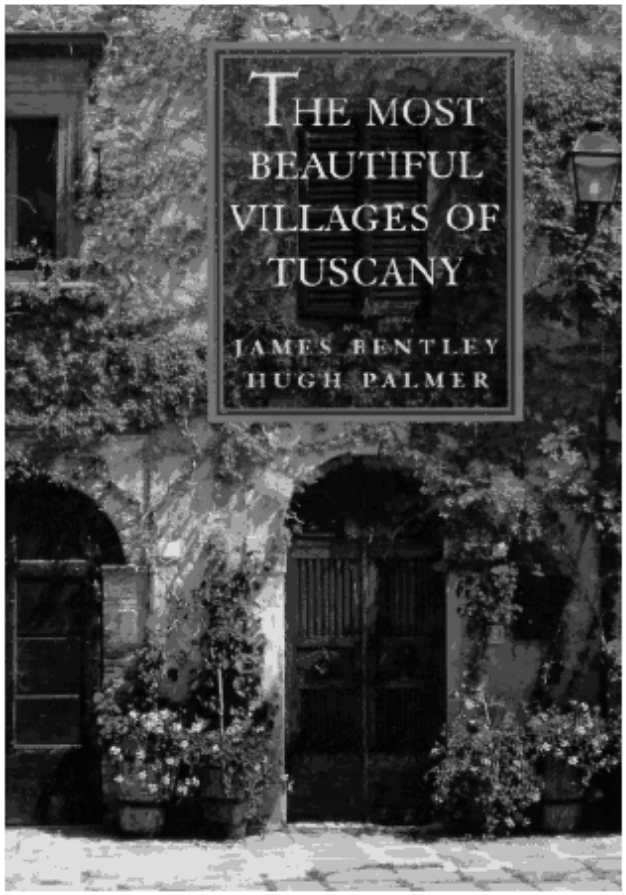
- First stage is coded using progresive or sequantial JPEG modes.
- The output of each hierarchical stage is then upsampled and used as the prediction for the next stage.

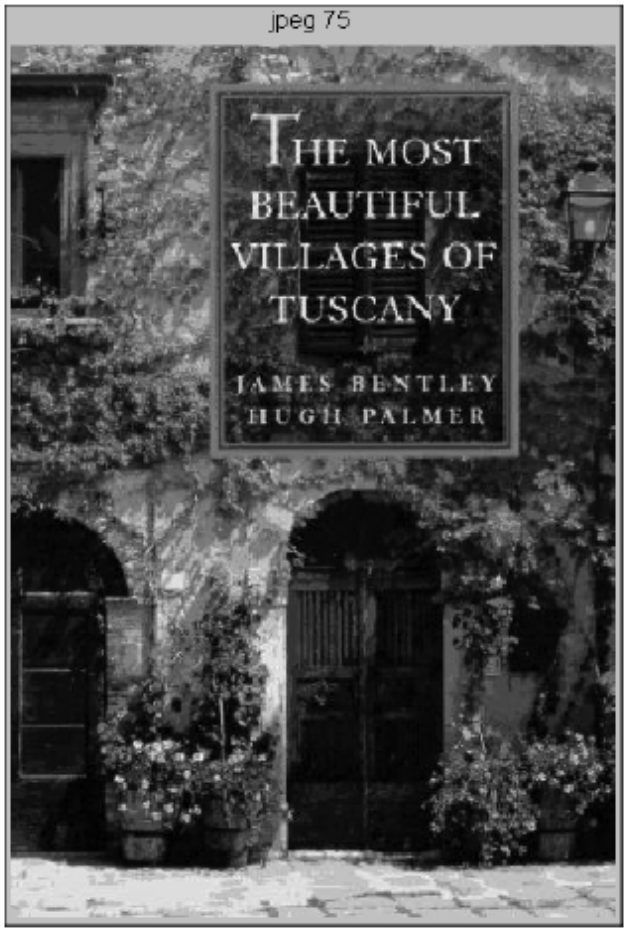


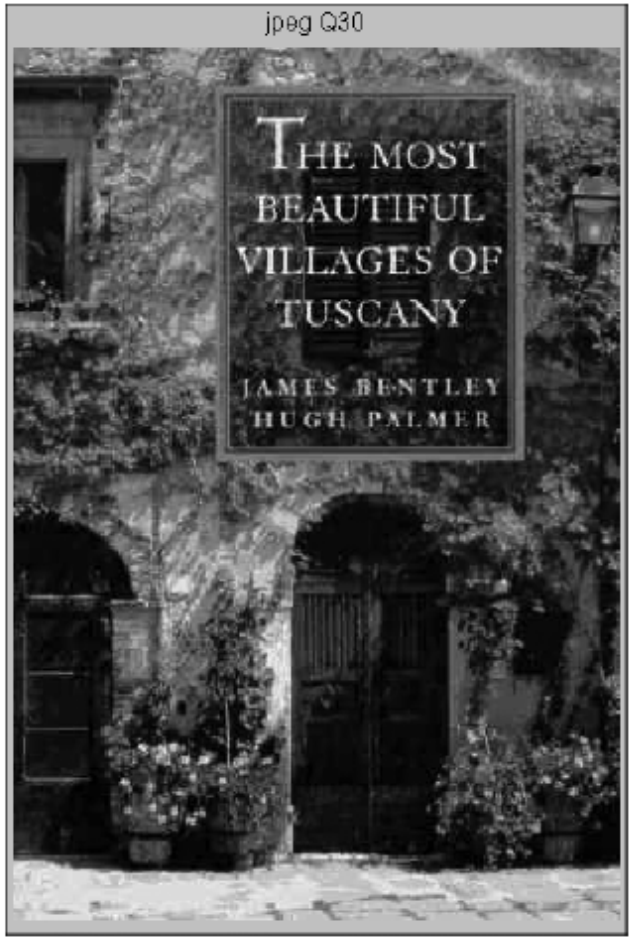
JPEG – Hierarchical (2/2)



JPEG – Example (1/2)







JPEG – Example (2/2)

JPEG - Complexity

- Operations needed for JPEG compression/decompression (millions of operations)

Image size	Color conversion	DCT/ IDCT	Quant.	Huffman coding	Total
Photo	74.2	335.0	31.6	148.9	589.8
VGA	3.7	16.0	1.5	7.4	28.9
CIF	1.2	5.3	0.5	2.5	9.6
QCIF	0.3	1.3	0.1	0.6	2.4

JPEG vs GIF (1/2)

- Typical file size of compressed photographic images with good visual quality

	GIF (8 bits per pixel)		Color JPEG (24 bits per pixel)		Grayscale JPEG (8 bits per pixel)	
	Size[KB]	Ratio	Size[KB]	Ratio	Size[KB]	Ratio
Photo	4000-6000	1:1.3	600-800	1:25	500-700	1:10
VGA	170-250	1:1.3	25-50	1:25	20-40	1:10
CIF	60-90	1:1.3	10-20	1:20	8-15	1:8
QCIF	15-25	1:1.3	3-8	1:15	4-7	1:5

JPEG vs GIF (2/2)

GIF versus JPEG (both ~28k)

JPEG - Pros and Cones

- **Advantages:** Memory efficient, low complexity, compression efficiency
- **Disadvantages:** Poor low bit rate performance, poor error resilience, poor performance on computer generated images, progressive transmission and scalability not efficient

- Superior low bit rate performance:

732 KB image

JPEG

16KB
(45times)

JPEG2000

732 KB image

JPEG

13 KB
(56 times)

JPEG2000

JPEG

original

0.5bpp (16 times)

0.25bpp(32 times)

JPEG2000

JPEG 2000 -Timeline and more information

- Timeline:
  - Committee draft: 12/99
  - International Standard: 12/00-3/01
- More info:
  - www.digitalimaging.org
  - www.jpeg.org

JPEG 2000 requirements (1/2)

- superior low bit-rate performance
- continuous-tone and bi-level compression
- lossy and lossless compression
- progressive transmission by pixel accuracy and resolution
- fixed-rate, fixed-size, limited workspace memory
- random code stream access and processing
- robustness to bit-errors

JPEG 2000 requirements (2/2)

- open architecture
- sequential build-up capability (real time coding)
- backward compatibility with JPEG
- content-based description protective image security
- compatibility with ITU-T recommendations for image interchange
- interface with MPEG-4
- side channel spatial information (transparency)
- object based composition
- object based information embedding

Markets and applications

- Low bandwidth dissemination of imagery
- Medical imagery
- Pre- press imagery
- Internet
- Digital photography
- Photo and art digital libraries
- Facsimile
- Laser print rendering
- Scanner and digital copier memory buffers
- Remote sensing

Timeline of JPEG 2000 standard

Part	Title	CFP	WD	CD	FCD	FDIS	IS
1	JPEG 2000 image coding system core coding system	97/03	99/03	99/12	00/03	00/10	00/12
2	JPEG 2000 image coding system extensions	97/03	00/03	00/08	00/12	01/07	01/10
3	Motion JPEG 2000	99/12	00/07	00/12	01/03	01/07	01/10
4	Conformance testing	99/12	00/07	00/12	01/07	01/11	02/03
5	Reference software	99/12	00/03	00/07	00/12	01/08	01/11
6	Compound image file format	97/03	00/12	01/03	01/11	02/03	02/05

CFP=Call for Proposals, WD=Working Draft, CD=Committee Draft, FCD=Final Committee Draft, FDIS=Final Draft International Standard, IS=International Standard.

JPEG 2000 technology

- Based on wavelets
- uses arithmetic coding
- uses predictive coding

JPEG 2000

- Image Type
  - Image width and height: 1 to $2^{32} - 1$
  - Component depth: 1 to 32 bits
  - Number of components: 1 to 255
  - Each component can have a different depth
  - Each component can have different spans

Resolution on Demand

- JPEG 2000 is based on Wavelet compression. “Features” can be saved in resolution order.

Lower Detail Features

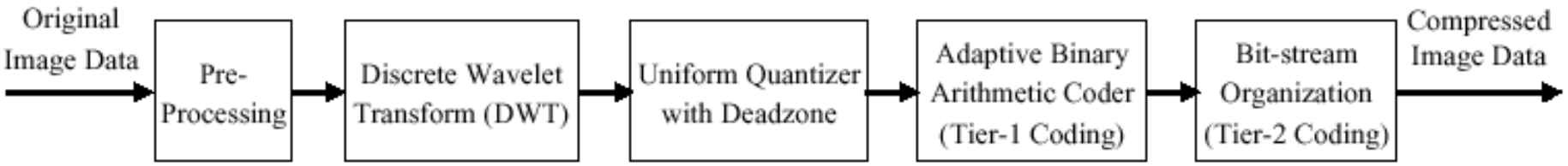
Higher Detail Features

Metadata for Optimized Workflows

- The *two* most important things about an image:
  - the pixels
  - information about the pixels
- The combination of these two elements yields the *power* of digital imaging

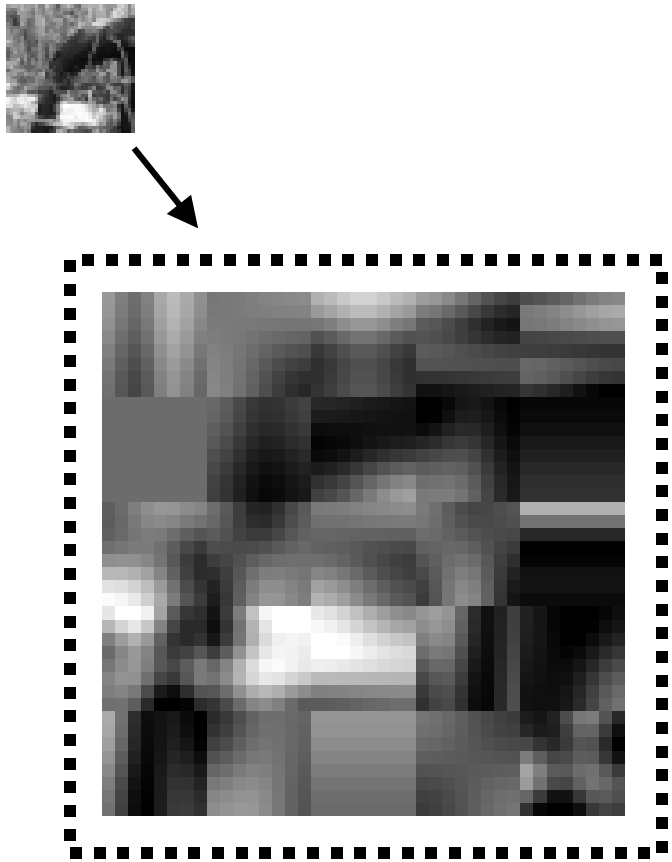
Photographer:	J. Hunt
Date:	08/07/99
Location:	Kern River
Subject:	Golden Trout
Focus Distance:	2 feet
Zoom:	2-to-1

JPEG 2000 – Fundamental building blocks



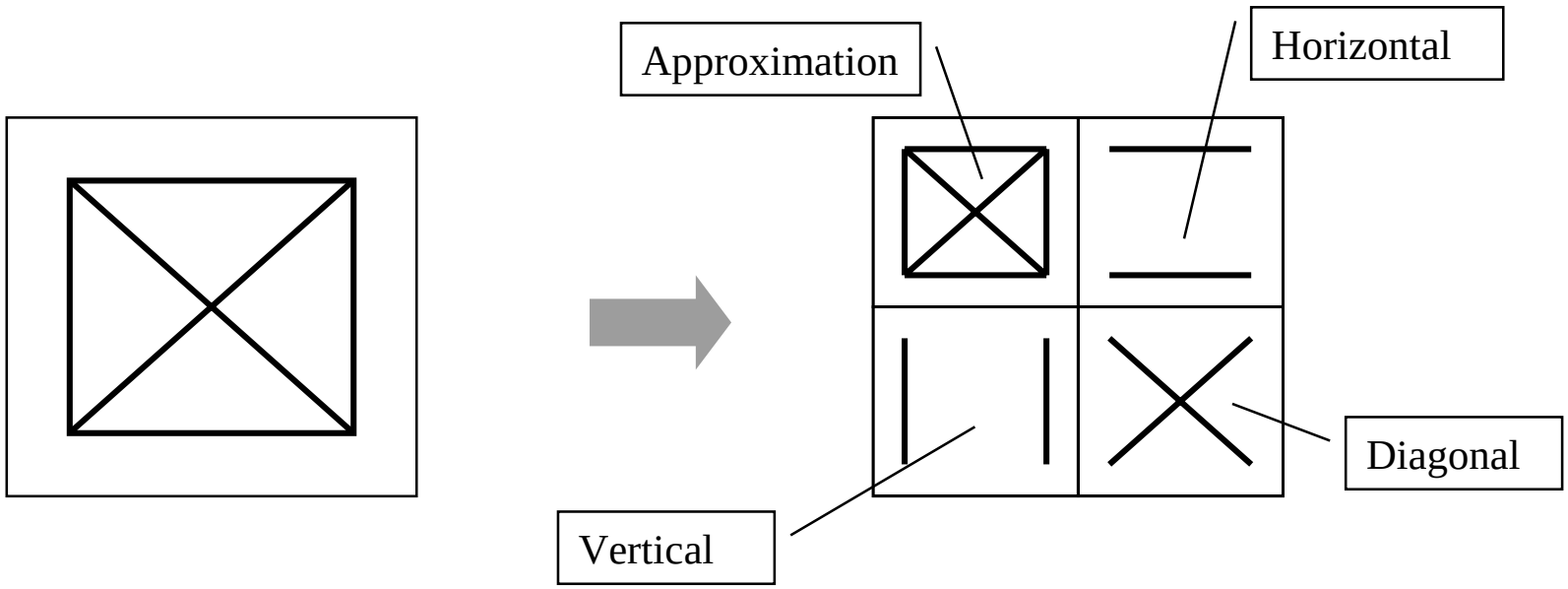
JPEG - Problem

- Breaking the image down into 8 x 8 blocks causes a loss of correlation between the blocks
- Noticable “blocking” effect at high compression rates

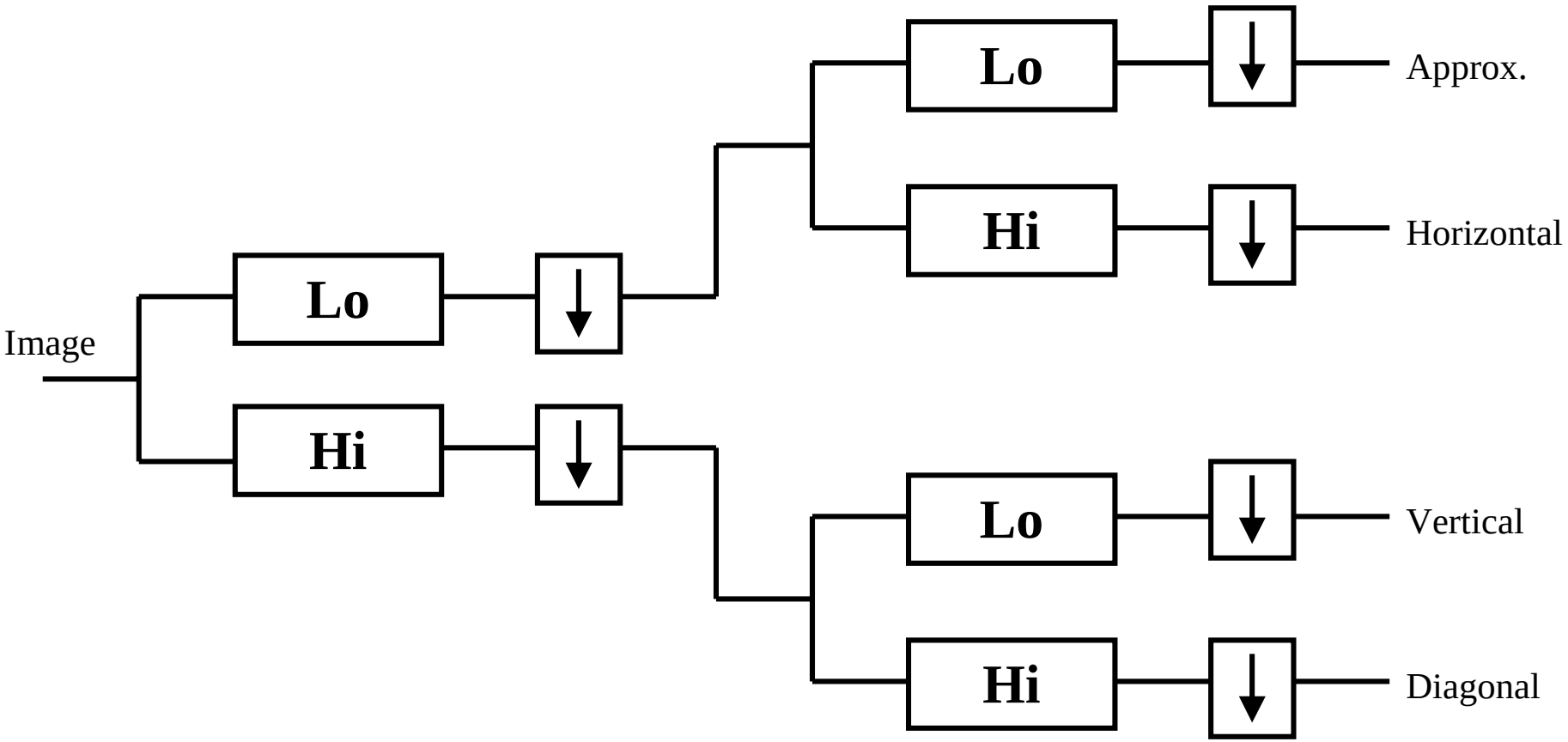


Discrete Wavelet Transform

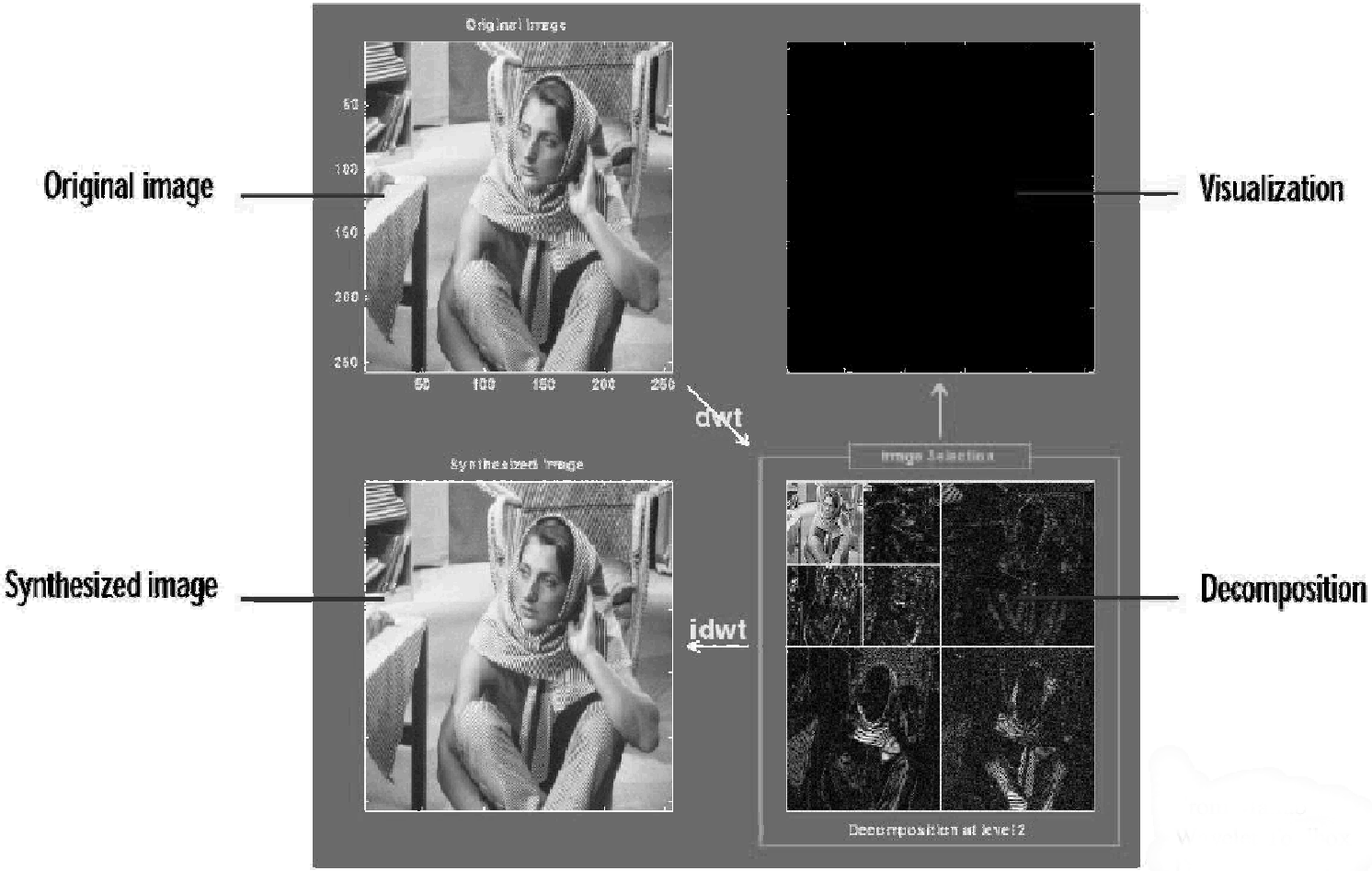
- Decomposes signal into sub-bands distinguishing between high and low frequency components



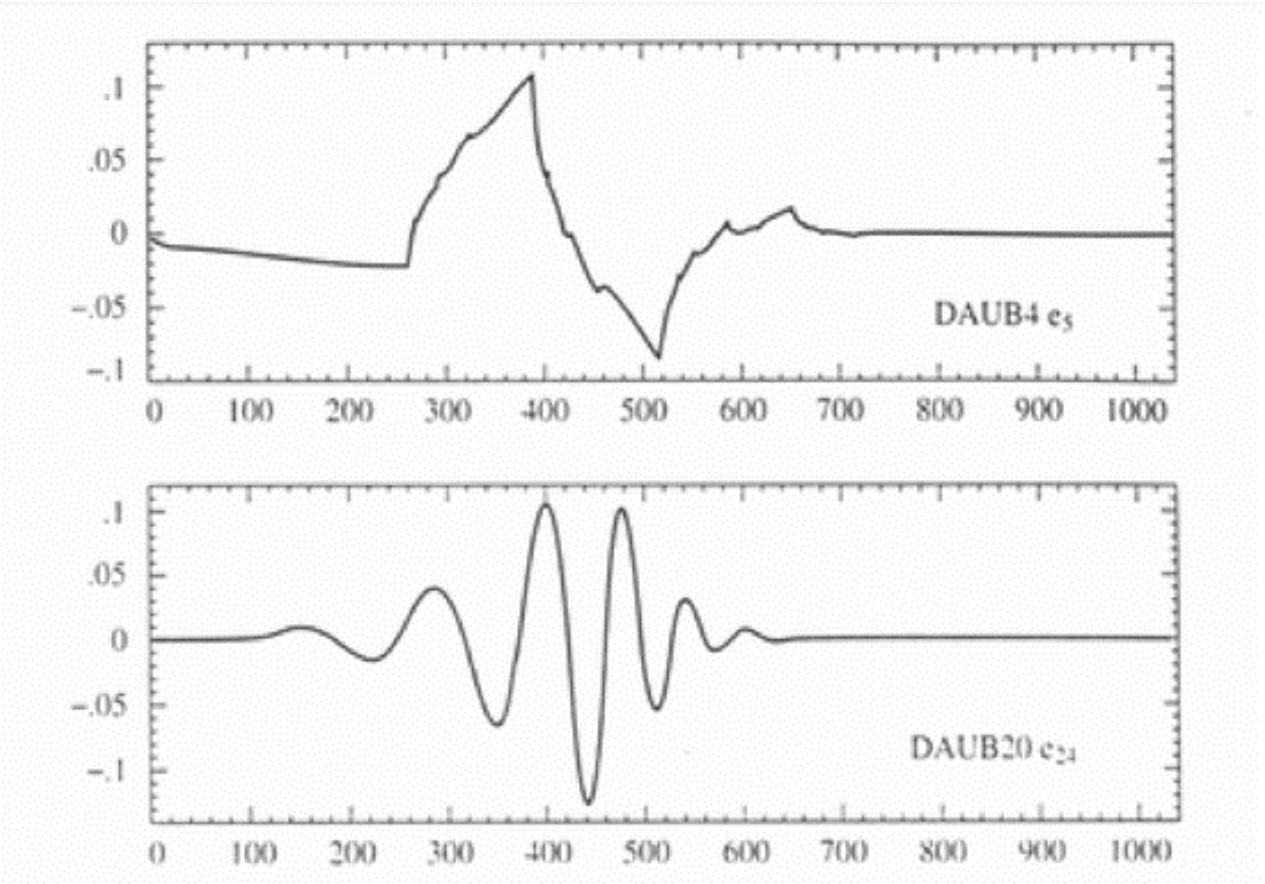
DWT - Mechanism



2-D wavelet decomposition



Mother Wavelets



Time vs.

Magnitude

Wavelets - Why so good?

- Wavelets are time limited and make good basis functions for both high frequency and low frequency signals
- Allow multiresolution analysis due to scaling (dialation) of basis functions
- Fewer basis functions are needed to represent a function over DCT

Quantization - EZW

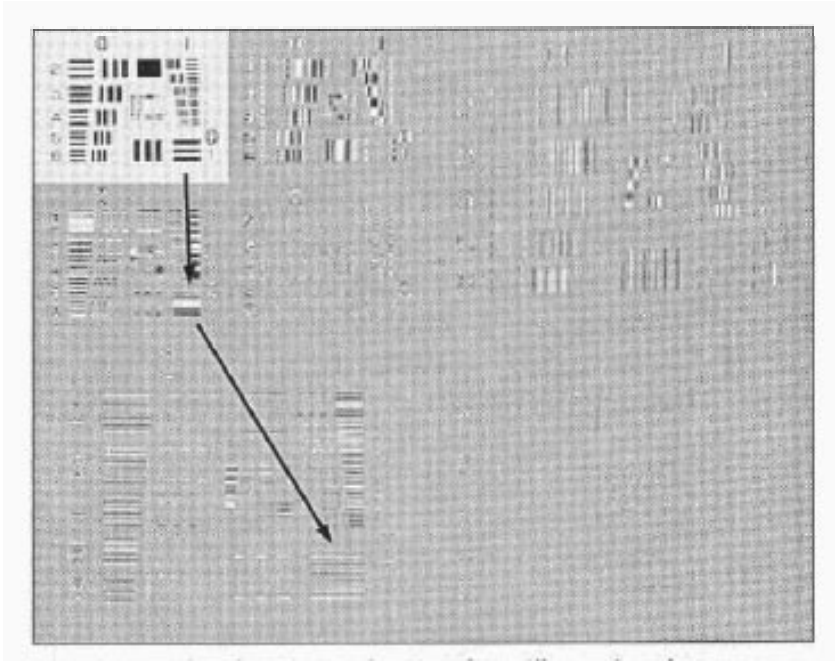
- Can control the level of image quality by setting the number of bits per pixel
- Many variations of the algorithm exists, each providing specific benefits depending upon the application

[**] Sharipo, J. M. “Embedded Image Coding Usig Zerotrees of Wavelet Coefficients,” IEEE Transactions in Signal Processing, Vol. 41, No. 12, 1993, pp.3445-62.

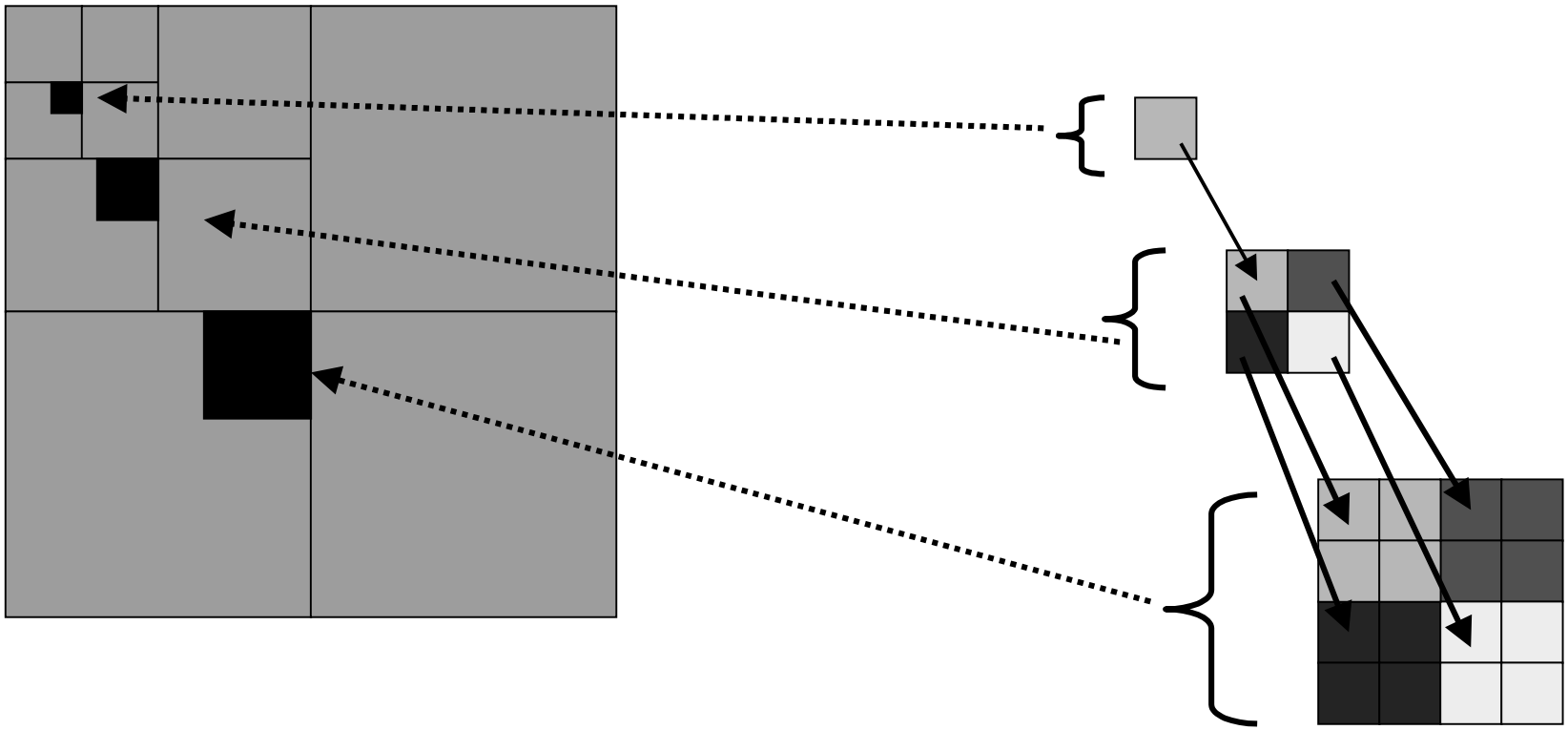
Embedded zero tree wavelet (EZW) coding

"Modern" lossy wavelet coding exploits multi-resolution and self-similar nature of wavelet decomposition

- Energy is compacted into a small number of coeff.
- Significant coeff. tend to cluster at the same spatial location in each freq. subband



EZW



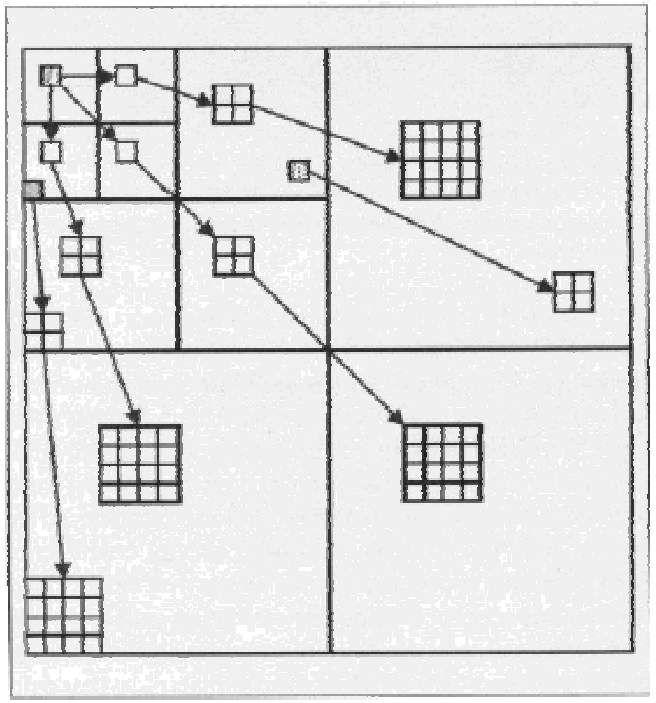
Key concepts

Significance map coding via zero-Tree

- Encode only high energy coefficients
  - *Need to send location info.*
    ➔ *large overhead*
- Encode "insignificance map" using zero-trees

Successive approximation quantization

- Send most-significant-bits first and gradually refine coeff. value
- "Embedded" nature of coded bit-stream
  - *get higher fidelity image by adding extra refining bits*



▲ 15. Example trees that can be defined on the wavelet transform. The roots of the three trees, indicated by shading, originate in the LL_3, LH_3, and HL_3 subbands.

Example (1/3)

Initial threshold ~ $2 \wedge floor(\log_2 x_{max})$

- Put all coeff. in dominant list

Dominant Pass ("zig-zag" across bands)

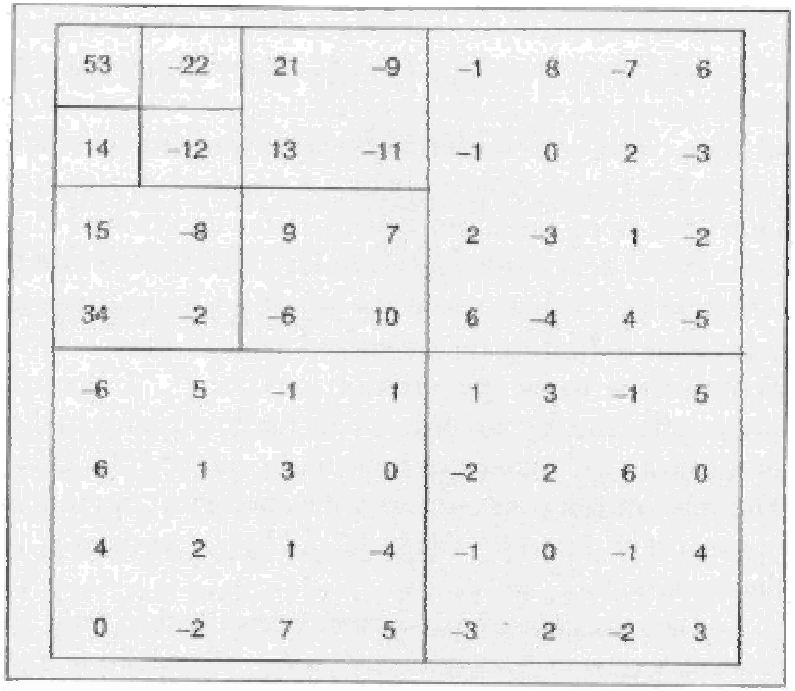
- Assign symbol to each coeff. & entropy encode symbols
  - *ps – positive significance*
  - *ns – negative significance*
  - *iz – isolated zero*
  - *ztr – zero-tree root*
- Significant coeff.
  - *move. to subordinate list*
  - *put zero in dominant list*

Subordinate Pass

- Output one bit for subordinate list
  - *According to position in up/low of quantization interval*

Repeat with half threshold

- Until bit budget achieved



53	-22	21	-9	-1	8	-7	6
14	-12	13	-11	-1	0	2	-3
15	-8	9	7	2	-3	1	-2
34	-2	-6	10	6	-4	4	-5
-6	5	-1	1	1	3	-1	5
6	1	3	0	-2	2	6	0
4	2	1	-4	-1	0	-1	4
0	-2	7	5	-3	2	-2	3

Example (2/3)

*	–22	21	–9	–1	8	–7	6
14	–12	13	–11	–1	0	2	–3
15	–8	9	7	2	–3	1	–2
*	–2	–6	10	6	–4	4	–5
–6	5	–1	1	1	3	–1	5
6	1	3	0	–2	2	6	0
4	2	1	–4	–1	0	–1	4
0	–2	7	5	–3	2	–2	3

▲ *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.*

Table 4. Resulting Output of the Subordinate Pass.

Coefficient Magnitude	Symbol	Reconstruction Magnitude
53	1	56
34	0	40

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

Subband	Coefficient Value	Symbol	Reconstruction Value	Comment (See Text)
LL_3	53	*ps*	48	1)
HL_3	–22	*ztr*	0	2)
LH_3	14	*iz*	0	3)
HH_3	–12	*ztr*	0	
LH_2	15	*ztr*	0	
LH_2	–8	*ztr*	0	
LH_2	34	*ps*	48	
LH_2	–2	*ztr*	0	
LH_1	4	*iz*	0	
LH_1	2	*iz*	0	
LH_1	0	*iz*	0	
LH_1	–2	*iz*	0	

After 1st pass

Example (3/3)

After 2nd pass

Table 5. Resulting Output of the Second Dominant Pass (T_0 =16).

Subband	Coefficient Value	Symbol	Reconstruction Value	Comment (See Text)
HL_3	–22	*ns*	–24	
LH_3	14	*ztr*	0	4)
HH_3	–12	*ztr*	0	
HL_2	21	*ps*	24	
HL_2	–9	*ztr*	0	
HL_2	13	*ztr*	0	
HL_2	–11	*ztr*	0	
HL_1	–1	*iz*	0	
HL_1	8	*iz*	0	
HL_1	–1	*iz*	0	
HL_1	0	*iz*	0	

Beyond EZW

- EZW has poor error resilience, difficult for selective spatial decoding
- Alternatives:
  - SPIHT : Set Partitioning in Hierarchical Trees
  - EBCOT : Embedded block coding with optimized truncation: Used in JPEG2000

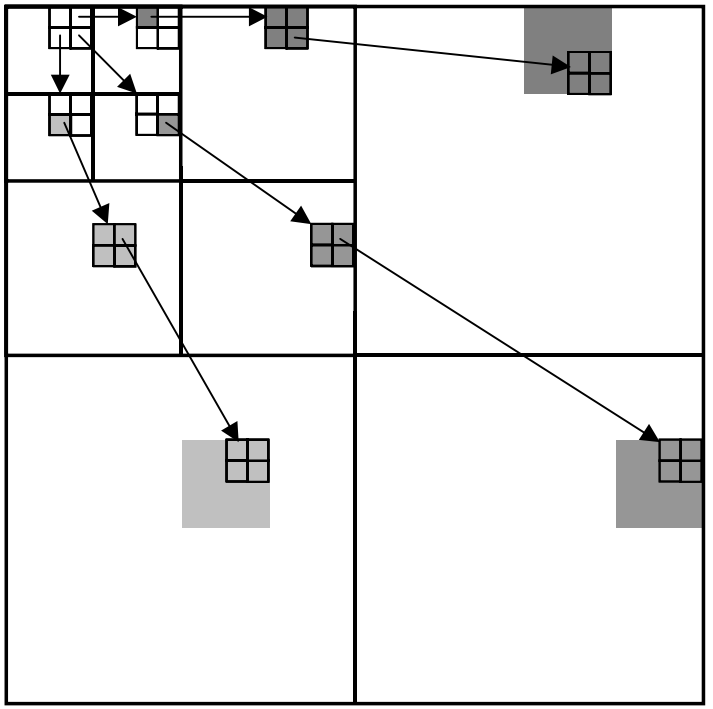
SPIHT Coding (1/2)

- Encoder transmits information about the wavelet of an image
- Partial information on the wavelet is recovered to produce a valid image
- Partial ordering of coefficient values
- Transmits data in bit plane order
- Exploits redundancies between scales

SPIHT Coding (2/2)

- Divides wavelet into *Spatial Orientation Trees*

- Nodes stored in 3 lists
  - *List of Insignificant Pixels*
  - *List of Insignificant Sets*
  - *List of Significant Pixels*



EBCOT

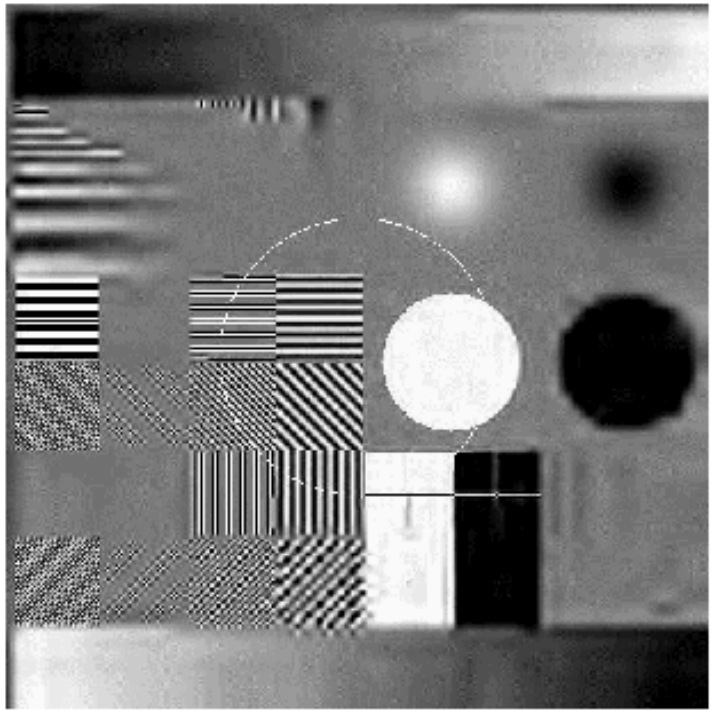
- Each subband is partitioned into a set of blocks (each so-called code-block)
- All blocks within a subband have the same size (possible exception for the blocks at the image boundaries)
- Blocks are encoded independently
- Post-processing operation determines the extent to which each block’s bitstream should be truncated
- Final bitstream is composed of a collection of “layers”

ROI coding

- Allows certain parts of image to be coded with better quality
- BASIC IDEA:
  - Calculate wavelet transform of whole image/time
  - calculate ROI mask == set of coefficients that are needed for ROI
  - Encoding is done in two ways
  - Encoding is progressive by accuracy and resolution

NOTE: ROI mask need NOT be transmitted to decoder (location and shape of ROI needs however)

Example



Sequence coded (BG 0.0625 bpp, ROI 0.9 bpp, 13x7)

Reversible color coding

- Useful for lossless color coding
- Approximation to Y,Cr, Cb

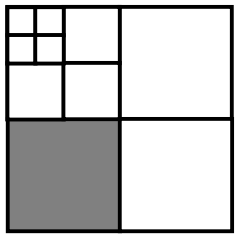
$$Yr = \left\lfloor \frac{R + 2 * G + B}{4} \right\rfloor$$

$$Ur = R - G + offset$$

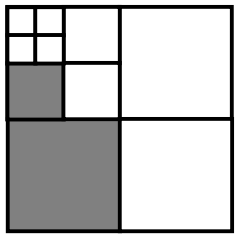
$$Vr = B - G + offset$$

$$offset = 2^{depth} - 1$$

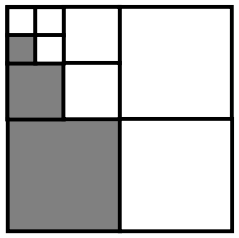
Note: All components must have identical subsampling parameters and same depth.



MSE= 4.1331



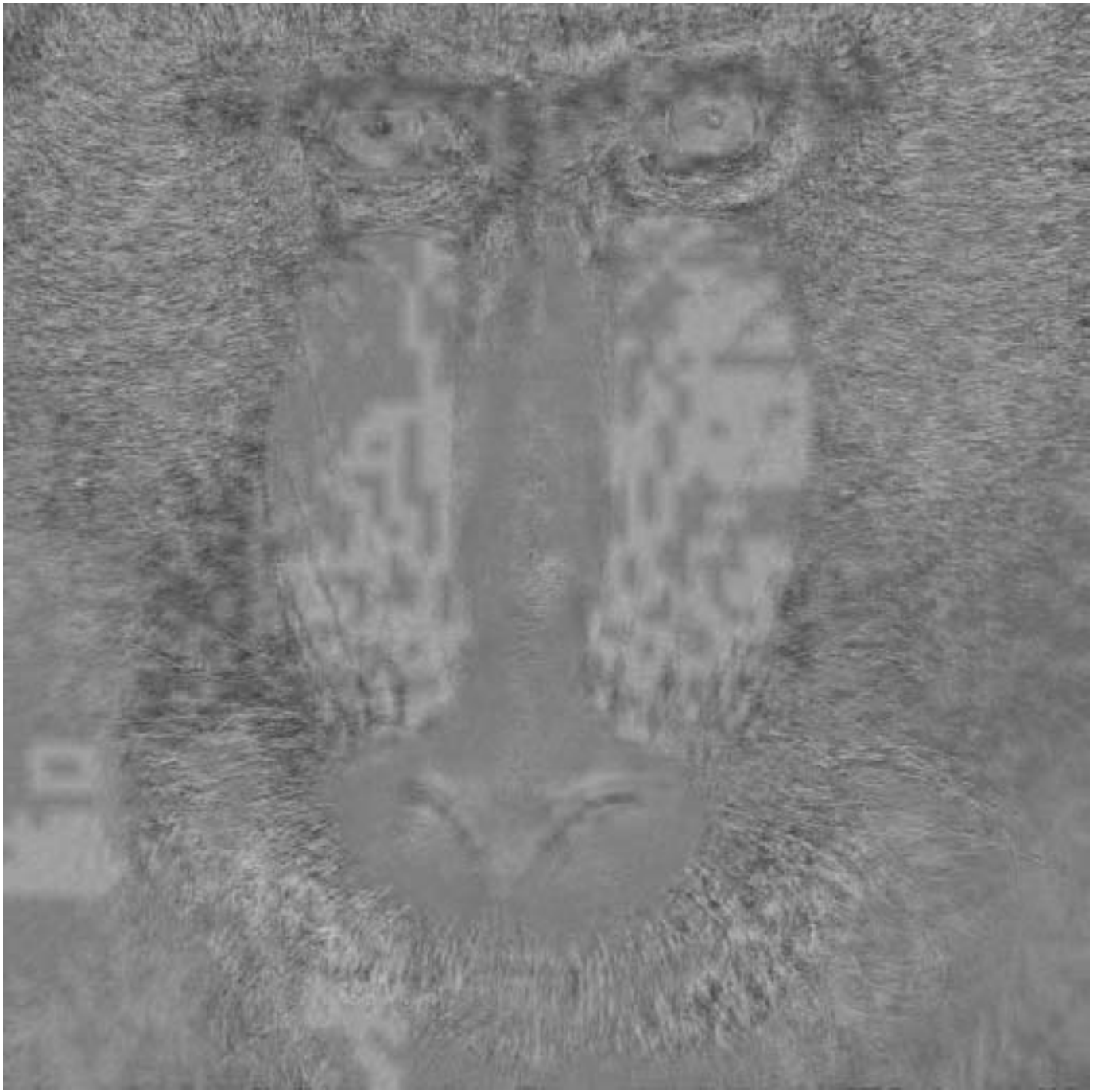
MSE= 14.6747



MSE= 33.2777

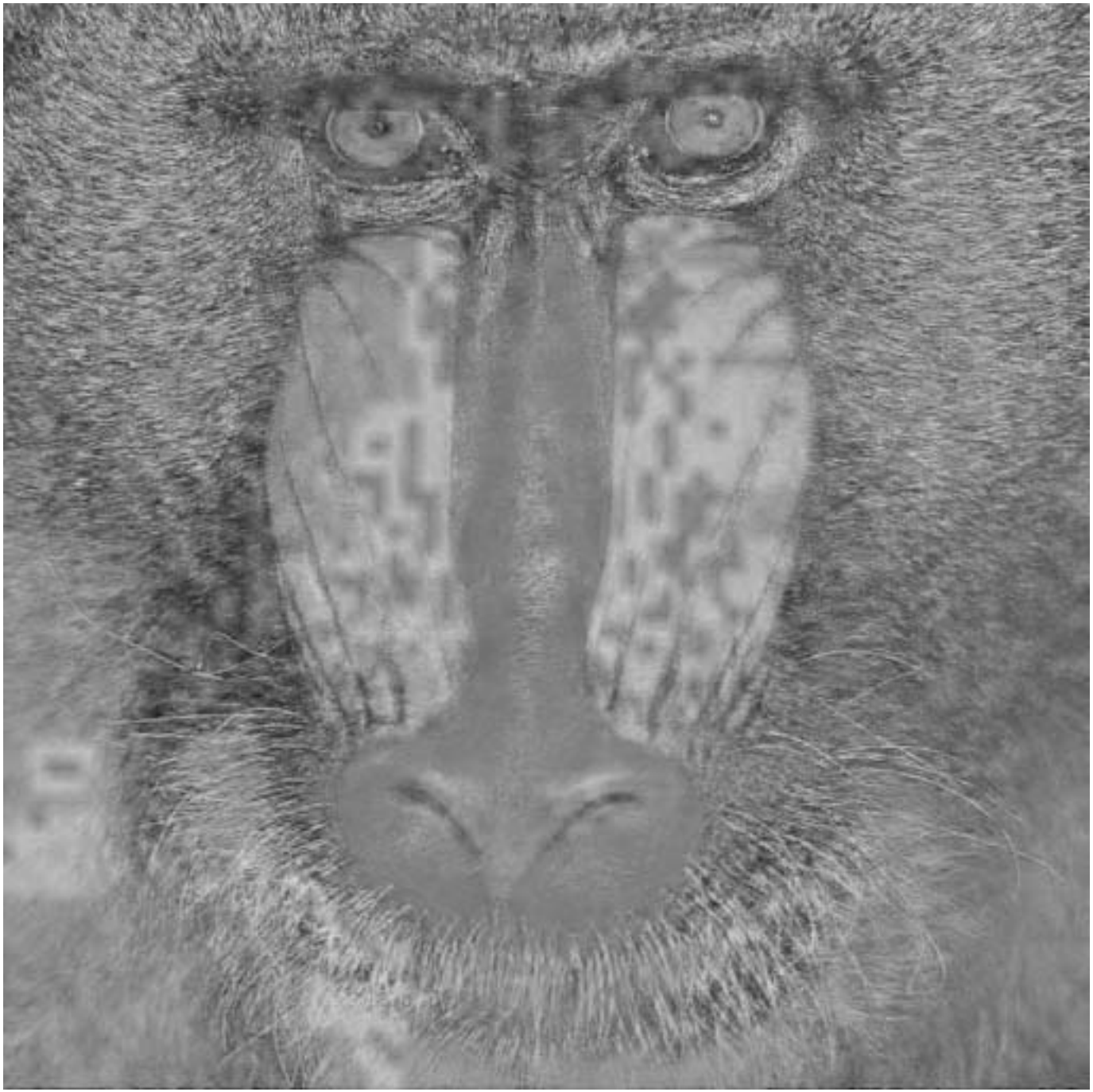
Only pass1

MSE= 992



Only pass2

MSE= 520



Only pass3

MSE= 701

Comparison based on functionality

	JPEG 2000	JPEG-LS	JPEG	MPEG-4 VTC
lossless compression performance	+++	++++	+	-
lossy compression performance	+++++	+	+++	++++
progressive bitstreams	++++	-	+	++
Region of Interest (ROI) coding	+++	-	-	+
arbitrary shaped objects	-	-	-	++
random access	++	-	-	-
low complexity	++	+++++	+++++	+
error resilience	+++	+	+	+++
non-iterative rate control	+++	-	-	+
genericity	+++	+++	++	++