Lossless Image Compression

C.M. Liu
Perceptual Signal Processing Lab
College of Computer Science
National Chiao-Tung University

http://www.csie.nctu.edu.tw/~cmliu/Courses/Compression/

Office: EC538
(03)5731877
cmliu@cs.nctu.edu.tw

Lossless JPEG (1992)

- ITU Recommendation T.81 (09/92)
- Compression based on 8 predictive modes ("selection values):

  0 P(x) = x (no prediction)

  1 P(x) = W

  2 P(x) = N

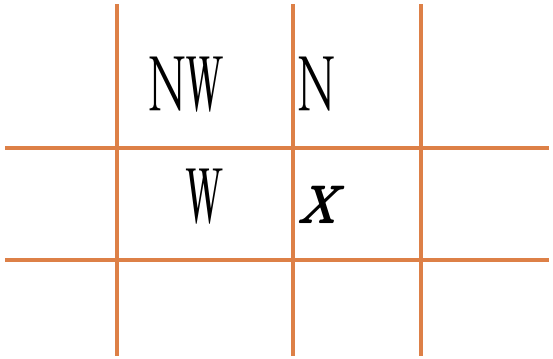
  3 P(x) = NW

  4 P(x) = W + N - NW

  5 P(x) = W + ⌊(N-NW)/2⌋

  6 P(x) = N + ⌊(W-NW)/2⌋

  7 P(x) = ⌊(W + N)/2⌋



- Sequence is then entropy-coded (Huffman/AC)

Lossless JPEG (2)

- Value 0
  - used for differential coding only in hierarchical mode
- Values 1, 2, 3
  - One-dimensional predictors
- Values 4, 5, 6, 7
  - Two-dimensional
- Value 1 (W)
  - Used in the first line of samples
  - At the beginning of each restart
  - Selected predictor used for the other lines
- Value 2 (N)
  - Used at the start of each line, except first
- Default predictor value: 2^{P-1}
  - At the start of first line
  - Beginning of each restart

Lossless JPEG Performance

- JPEG prediction mode comparisons

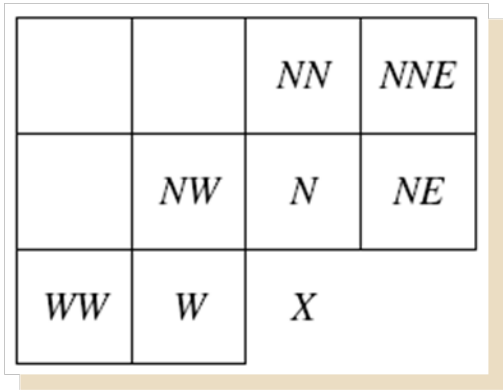
Image	JPEG 0	JPEG 1	JPEG 2	JPEG 3	JPEG 4	JPEG 5	JPEG 6	JPEG 7
Sena	53,431	37,220	31,559	38,261	31,055	**29,742**	33,063	32,179
Sensin	58,306	41,298	37,126	43,445	**32,429**	33,463	35,965	36,428
Earth	38,248	32,295	**32,137**	34,089	33,570	33,057	33,072	32,672
Omaha	56,061	**48,818**	51,283	53,909	53,771	53,520	52,542	52,189

- JPEG vs. GIF vs. PNG

Image	Best JPEG	GIF	PNG
Sena	31,055	51,085	31,577
Sensin	32,429	60,649	34,488
Earth	32,137	34,276	26,995
Omaha	48,818	61,341	50,185

Context-Adaptive Lossless Image Compression [Wu 95/96]

- Two modes: gray-scale & bi-level
  - We are skipping the lossy scheme for now
- Basic ideas
  - find the best context from the info available to encoder/decoder
  - estimate the presence/lack of horizontal/vertical features



$$d_h = |W - WW| + |N - NW| + |NE - N|$$
$$d_v = |W - NW| + |N - NN| + |NE - NNE|.$$

CALIC: Initial Prediction


```
if d_h-d_v > 80              // sharp horizontal edge
   X* = N
else if d_v-d_h > 80         // sharp vertical edge
   X* = W
else {                       // assume smoothness first
   X* = (N+W)/2 +(NE-NW)/4
   if d_h-d_v > 32           // horizontal edge
        X* = (X*+N)/2
   else if d_v-d_h > 32      // vertical edge
        X* = (X*+W)/2
   else if d_h-d_v > 8       // weak horizontal edge
        X* =(3X*+N)/4
   else if d_v-d_h > 8       // weak vertical edge
        X* = (3X*+W)/4
}
```

CALIC: Prediction Refinement

- Generate texture descriptor
  - Y = [N, W, NW, NE, NN, WW, 2N −NN, 2W −WW]
  - For each Y[k], k=0..7
    - If Y[K] < X*
      - Y[k] = 1
    - Else
      - Y[k] = 0
  - Due to dependencies, only 144 possible decriptors
- $\delta = d_h + d_v + 2|N-N^*|$
  - Split δ into four intervals & encode with 2 bits
- Total possible contexts: 144x4 = 576
- Once value is predicted, difference/residual is coded

CALIC: Coding Residuals

$$
\begin{aligned}
0 \le \delta < q_1 &\Rightarrow \text{Context } 1 \\
q_1 \le \delta < q_2 &\Rightarrow \text{Context } 2 \\
q_2 \le \delta < q_3 &\Rightarrow \text{Context } 3 \\
q_3 \le \delta < q_4 &\Rightarrow \text{Context } 4 \\
q_4 \le \delta < q_5 &\Rightarrow \text{Context } 5 \\
q_5 \le \delta < q_6 &\Rightarrow \text{Context } 6 \\
q_6 \le \delta < q_7 &\Rightarrow \text{Context } 7 \\
q_7 \le \delta < q_8 &\Rightarrow \text{Context } 8
\end{aligned}
$$

- q_1-q_8 can be user-defined

CALIC: Residual Remapping

- Assume pixels are in the [0, M-1] range
  - Possible residuals are in the [-(M-1), M-1] range
  - However, the number of discrete values is still M
- Generally,
  - $X-X^* \in [-X^*, M-1-X^*]$
  - Re-map to [0,M-1]:
    - $X^* \leq (M-1)/2$

$$
\begin{aligned}
0 &\rightarrow 0 \\
1 &\rightarrow 1 \\
-1 &\rightarrow 2 \\
2 &\rightarrow 3 \\
&\vdots \quad \vdots \\
-\hat{X} &\rightarrow 2\hat{X} \\
\hat{X}+1 &\rightarrow 2\hat{X}+1 \\
\hat{X}+2 &\rightarrow 2\hat{X}+2 \\
&\vdots \quad \vdots \\
M-1-\hat{X} &\rightarrow M-1
\end{aligned}
$$

Recursive Indexing

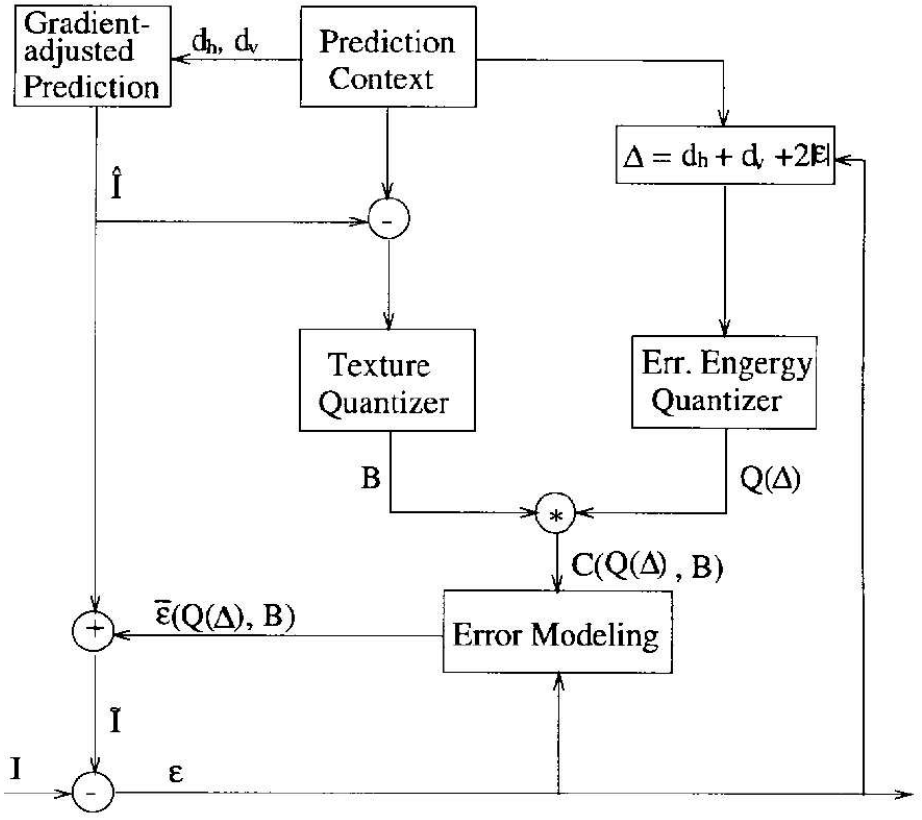
- A technique for representing large number ranges with a small codeword set
- Assume codewords are in the 0..n-1 range
- Coding rules for encoding X

```
while( X > n-2) {
    send n-1
    X = X - (n-1)
}
send X
```

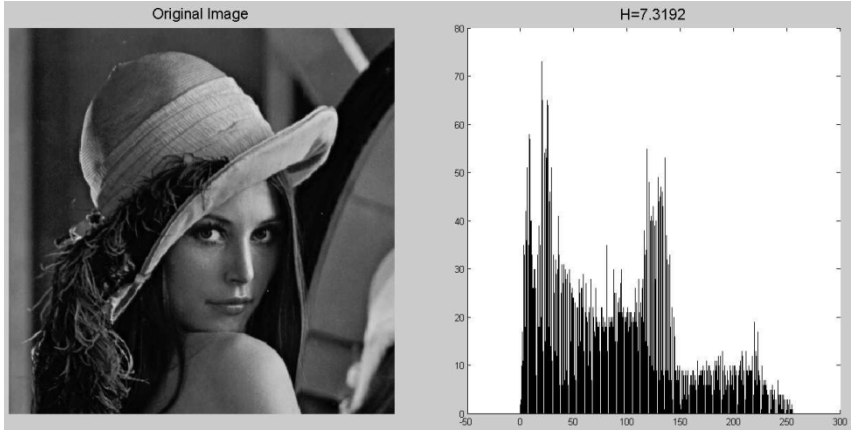
- Follow with entropy coding
  - Optimal for geometrically distributed sources

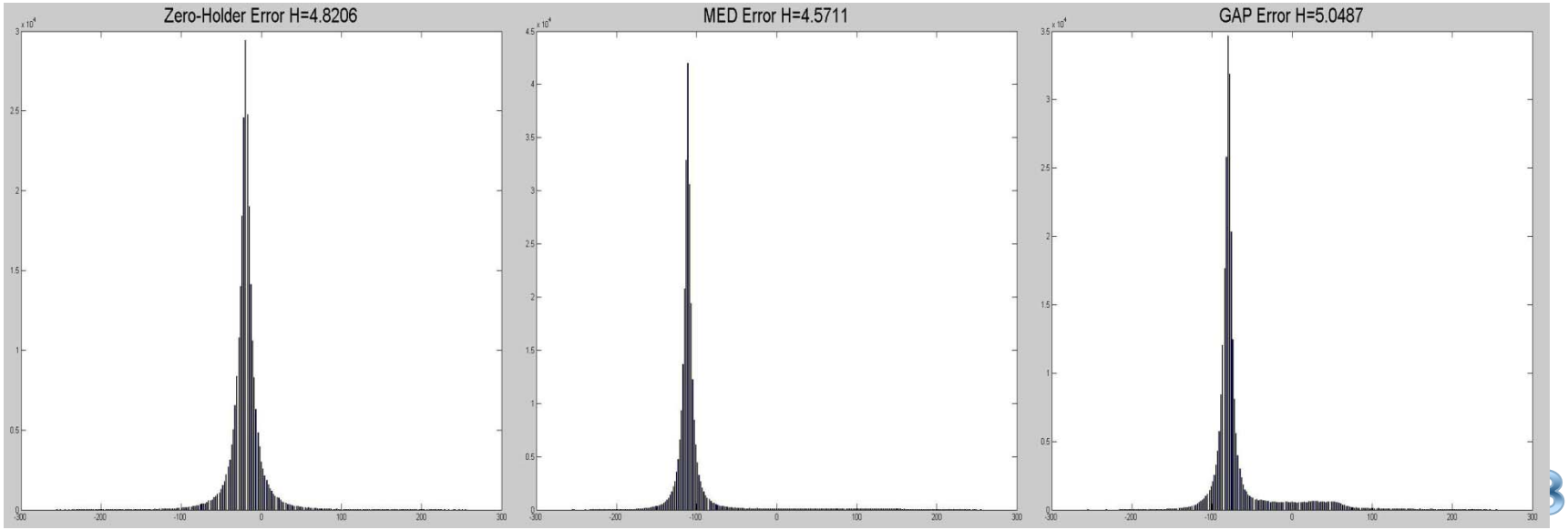
CALIC Summary

1. Find initial prediction $\ddot{X}$.
2. Compute prediction context.
3. Refine prediction by removing the estimate of the bias in that context.
4. Update bias estimate.
5. Obtain the residual and remap it so the residual values lie between 0 and $M-1$, where M is the size of the initial alphabet.
6. Find the coding context k.
7. Code the residual using the coding context.

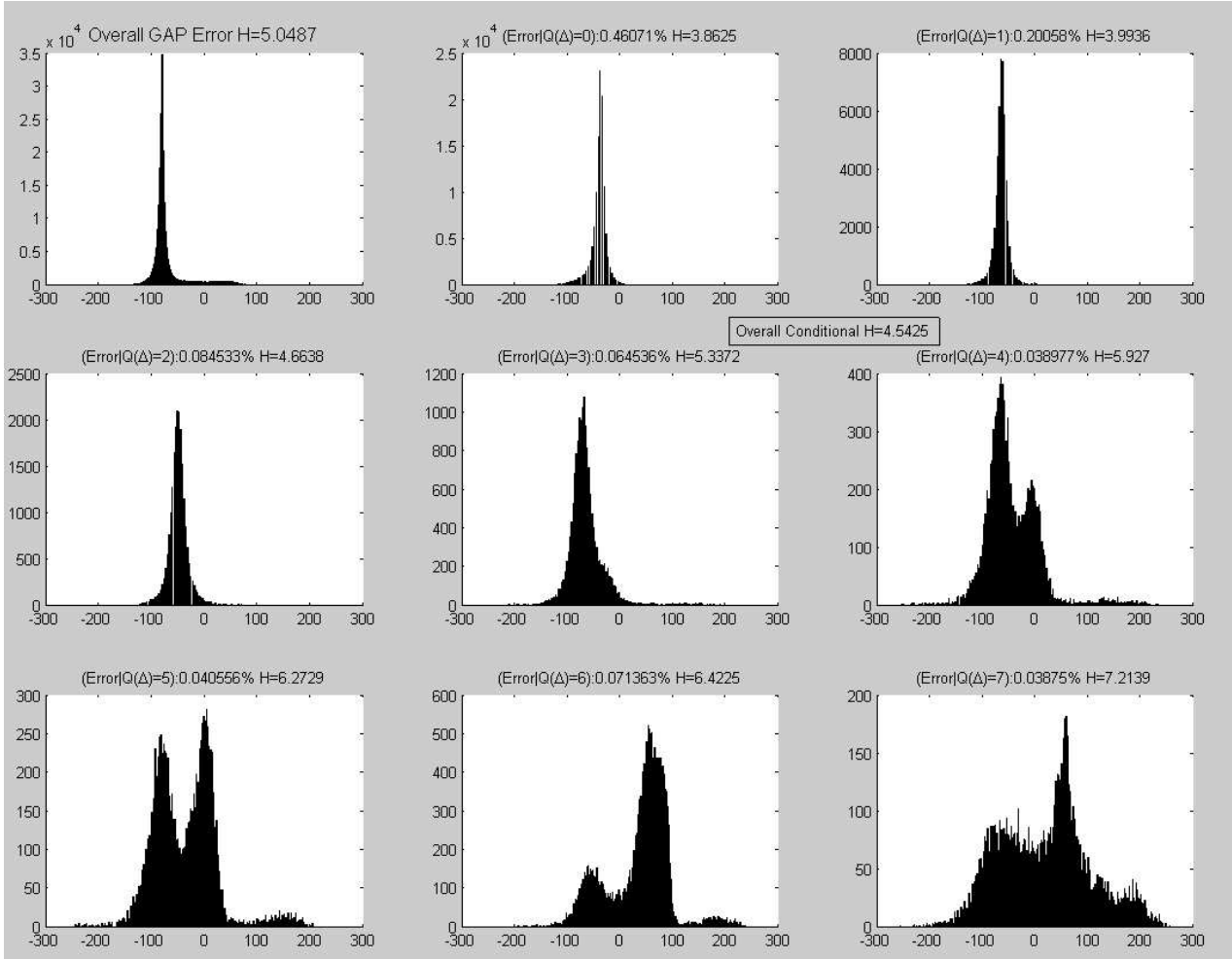

Gradient-adjusted Prediction
d_h, d_v
Prediction Context
$\Delta = d_h + d_v + 2|\varepsilon|$
$\hat{I}$
Texture Quantizer
Err. Engergy Quantizer
B
Q(Δ)
C(Q(Δ) , B)
$\bar{\varepsilon}(Q(\Delta), B)$
Error Modeling
$\tilde{I}$
I
ε

CALIC-- Examples



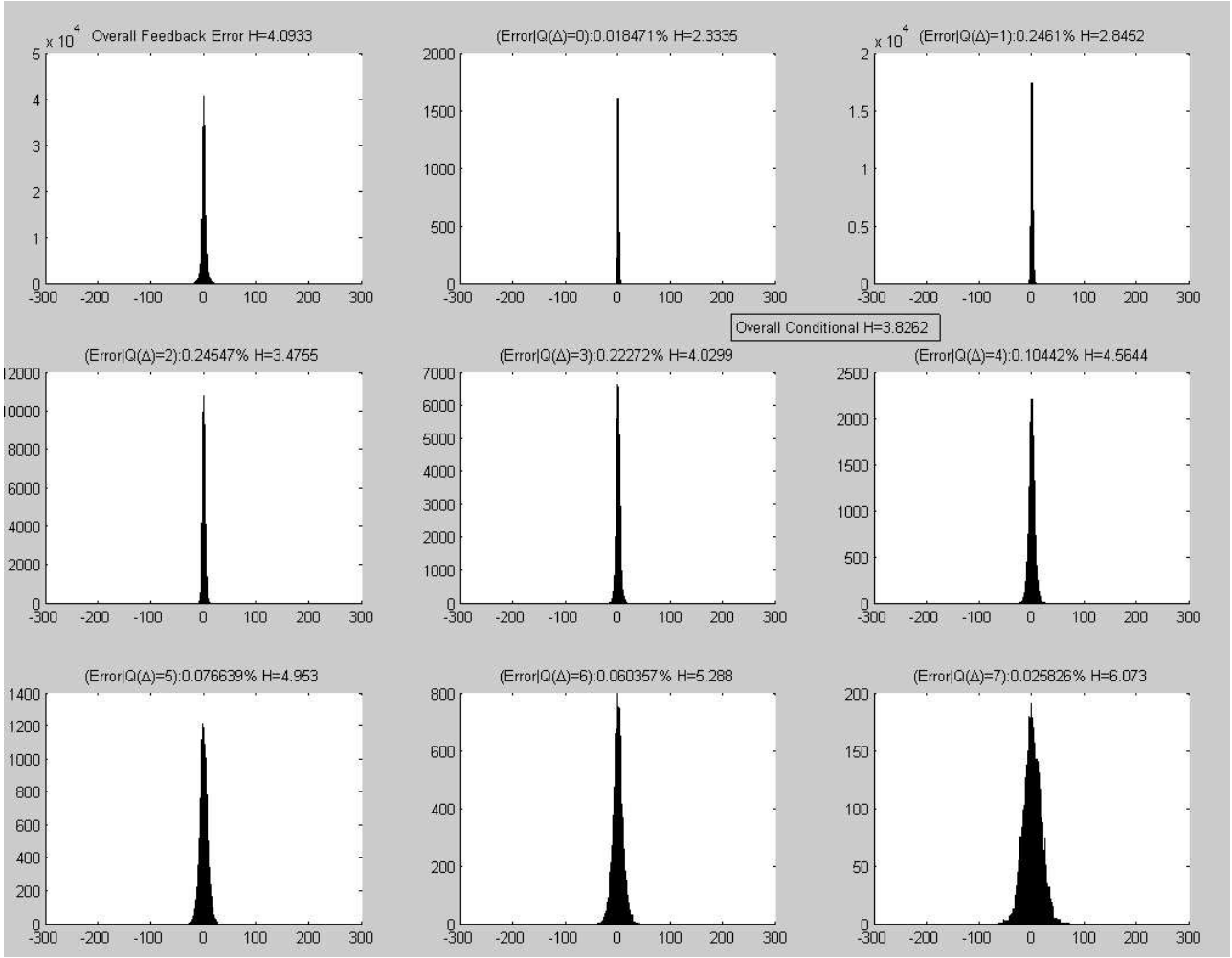


Conditional GAP



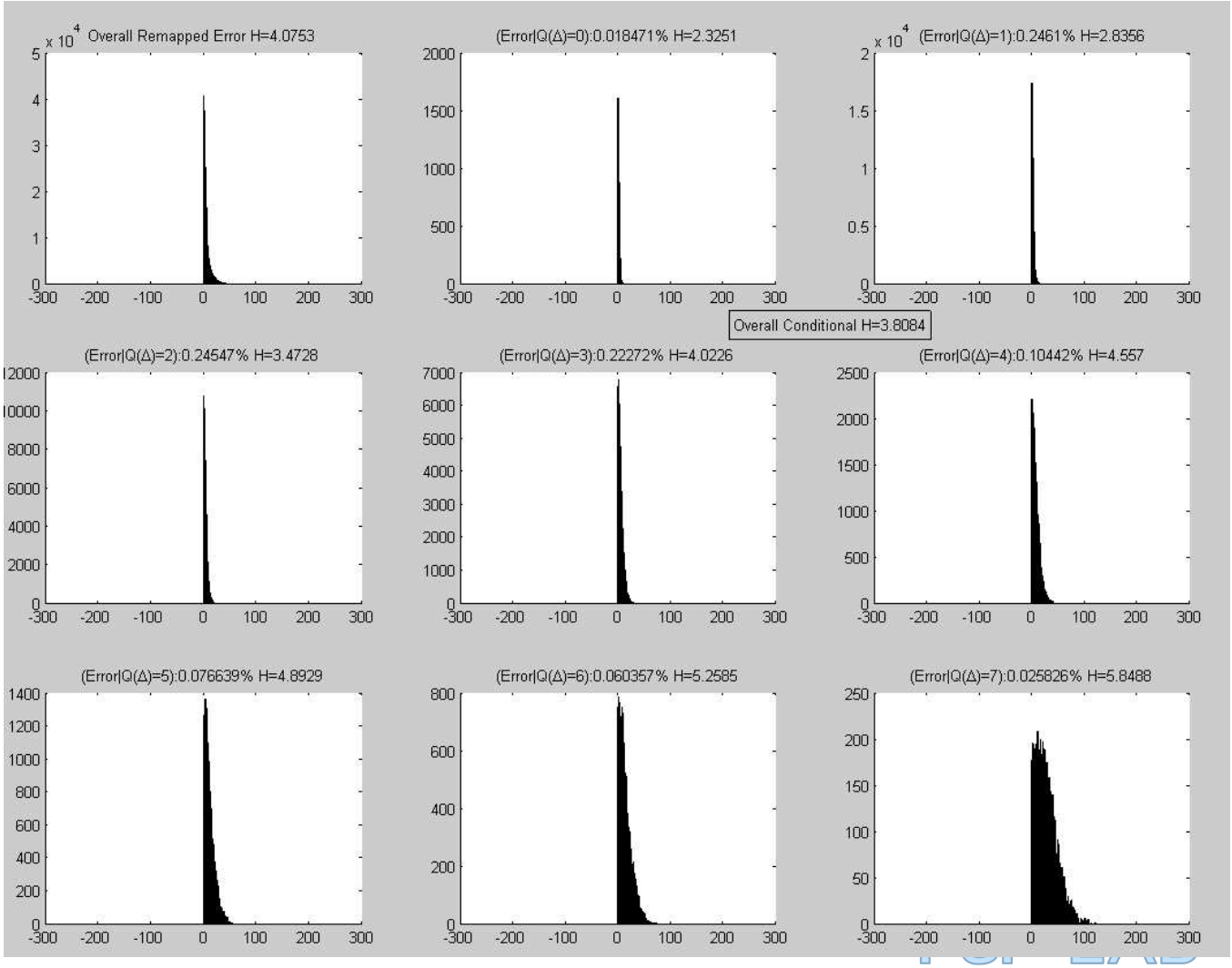
PSP LAB

Feedback Adjusted Error



PSP LAB

Sign Flipping and Remapping



300 Photo Images

Total 300 Images as Bottom Samples		
Codec	**TotalTime(s)**	**Bits/Pixel**
1022x768		
CALIC	127.54489231	3.67644704
LJPEG	40.65375328	4.32413912
BZIP2	87.78945422	4.71980285
GZIP	32.12696433	5.77562299
AC_C	58.30623674	7.55093587
AC_B	233.48027563	5.18200231
LZW	19.50467134	6.19474554

JPEG-LS (lossless)

- Simplified CALIC: LOCO-I
  - Initial prediction:

```
if NW ≥ max(W, N)
X̂ = max(W, N)
else
{
      if NW ≤ min(W, N)
      X̂ = min(W, N)
      else
      X̂ = W + N − NW
}
```

JPEG-LS (2)

- Differences
  - D_1 = NE - N
  - D_2 = N - NW
  - D_3 = NW - W
- Q = (Q_1, Q_2, Q_3)
  - (User-defined) coefficients T_i
- Total contexts:
  - 9x9x9 = 729
- If $Q_1 < 0$ then Q = -Q, SIGN = -1
  - 365, mapped to 0..364
- Correction multiplied by SIGN
- Prediction error mapping

$$
\begin{aligned}
D_i \le -T_3 &\Rightarrow Q_i = -4 \\
-T_3 < D_i \le -T_2 &\Rightarrow Q_i = -3 \\
-T_2 < D_i \le -T_1 &\Rightarrow Q_i = -2 \\
-T_1 < D_i \le 0 &\Rightarrow Q_i = -1 \\
D_i = 0 &\Rightarrow Q_i = 0 \\
0 < D_i \le T_1 &\Rightarrow Q_i = 1 \\
T_1 < D_i \le T_2 &\Rightarrow Q_i = 2 \\
T_2 < D_i \le T_3 &\Rightarrow Q_i = 3 \\
T_3 < D_i &\Rightarrow Q_i = 4
\end{aligned}
$$

$$
\begin{aligned}
r_n < -\frac{M}{2} &\Rightarrow r_n \leftarrow r_n + M \\
r_n > \frac{M}{2} &\Rightarrow r_n \leftarrow r_n - M
\end{aligned}
$$

PSP LAB

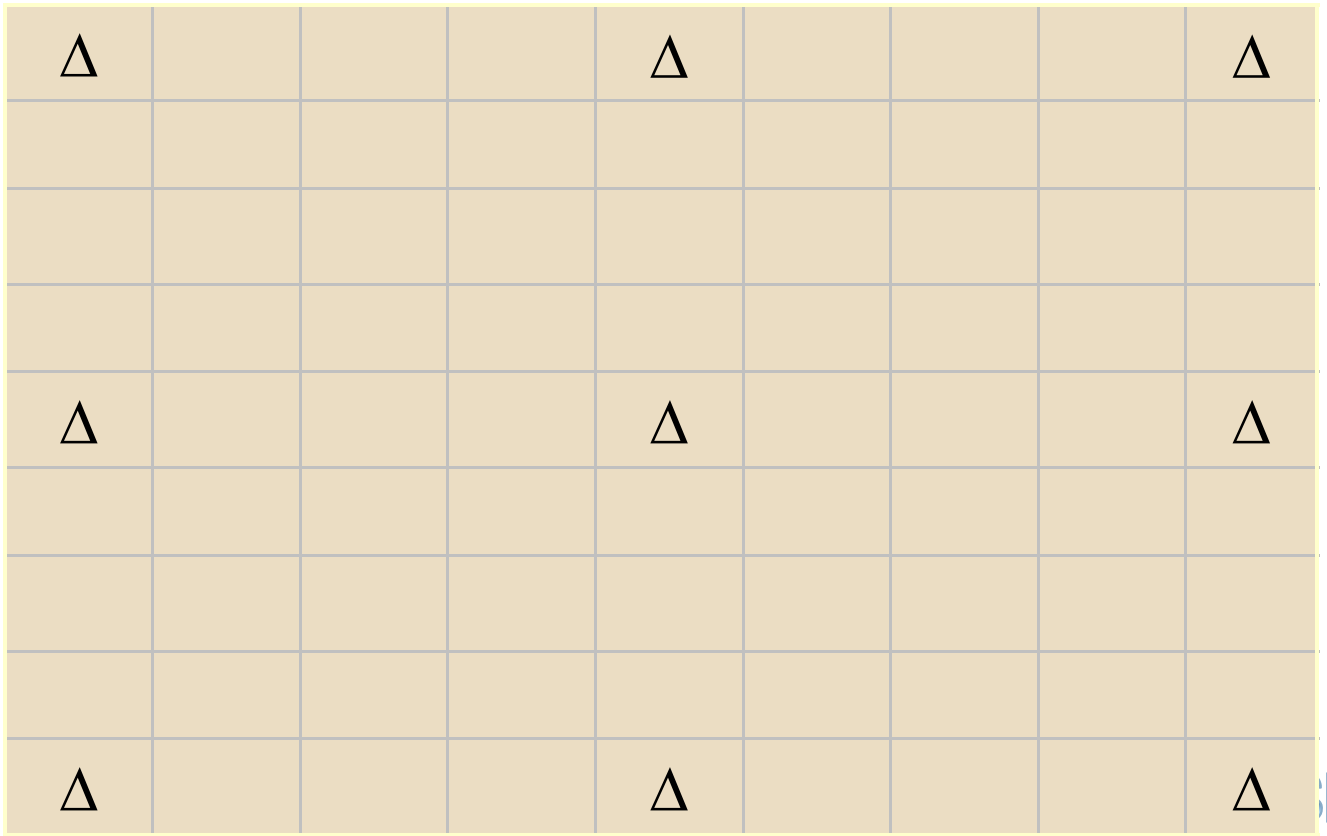
JPEG-LS (3)

- Prediction codes are encoded with an adaptive ***Golomb*** scheme
  - "Flashback" to ***Golomb*** (part of Huffman slide set)
- Performance

Image	Old JPEG	New JPEG	CALIC
Sena	31,055	27,339	26,433
Sensin	32,429	30,344	29,213
Earth	32,137	26,088	25,280
Omaha	48,818	50,765	48,249

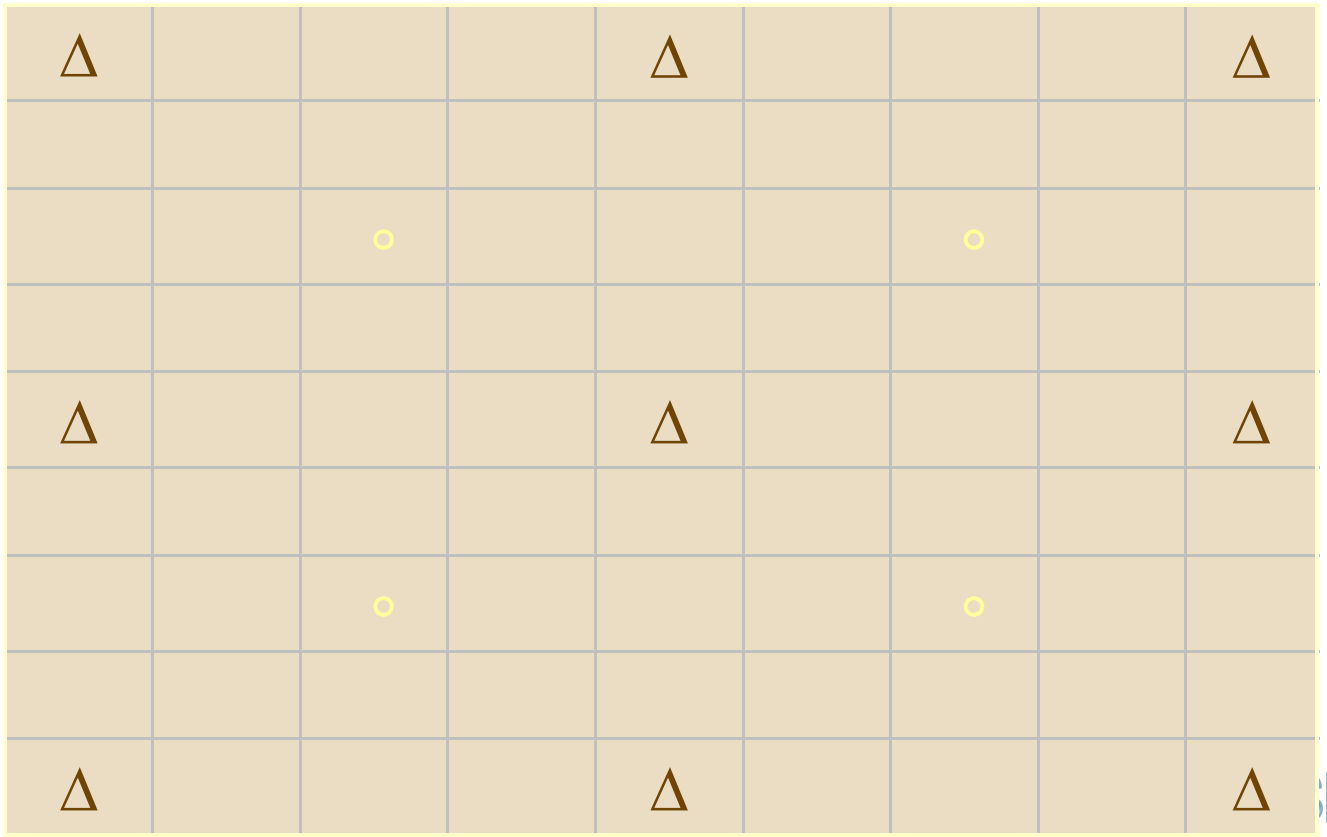
Multi-resolution Coding: HINT

- HINT (Hierarchical INTerpolation)
  - $\Delta \rightarrow \circ \rightarrow X \rightarrow * \rightarrow \bullet$



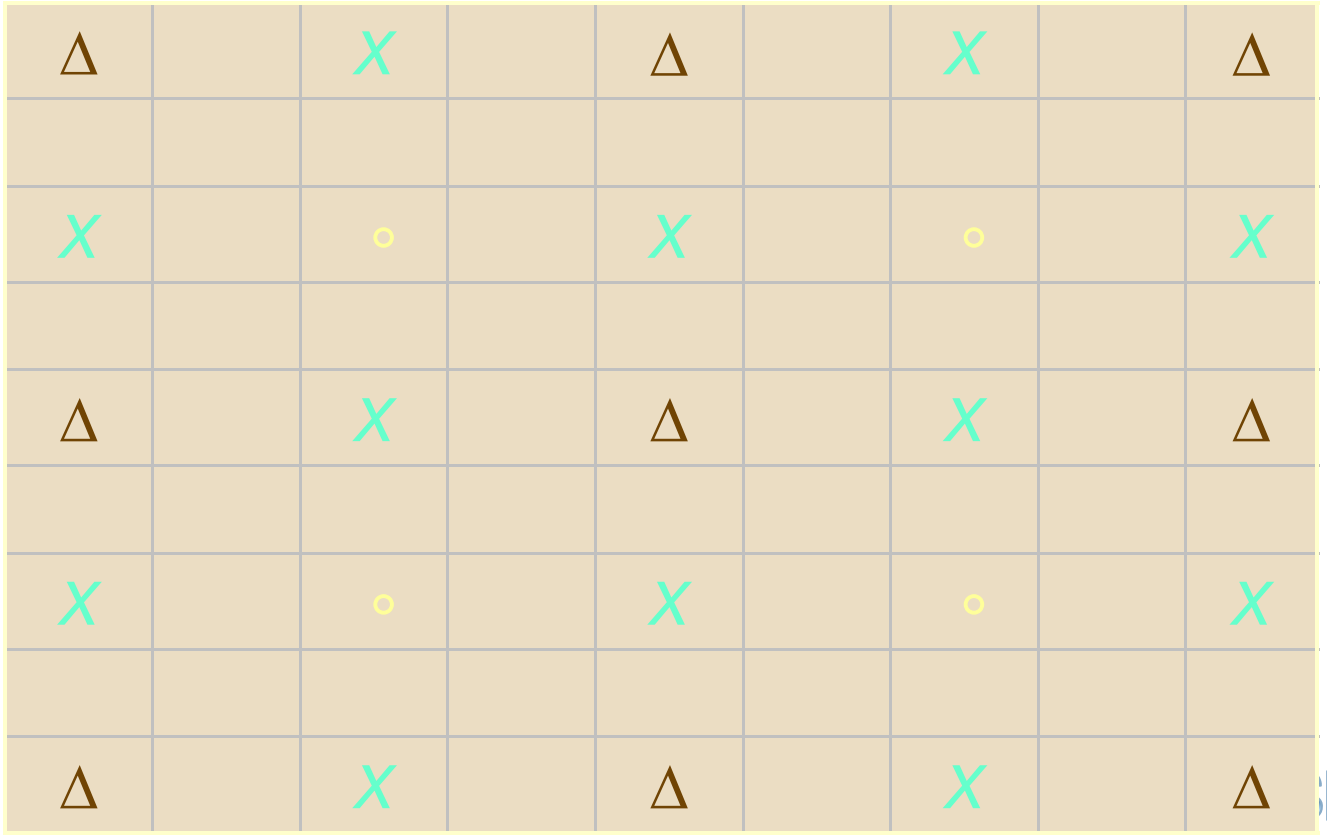
HINT (2)

- HINT (Hierarchical INTerpolation)
  - Δ → ○ → X → ✻ → ●



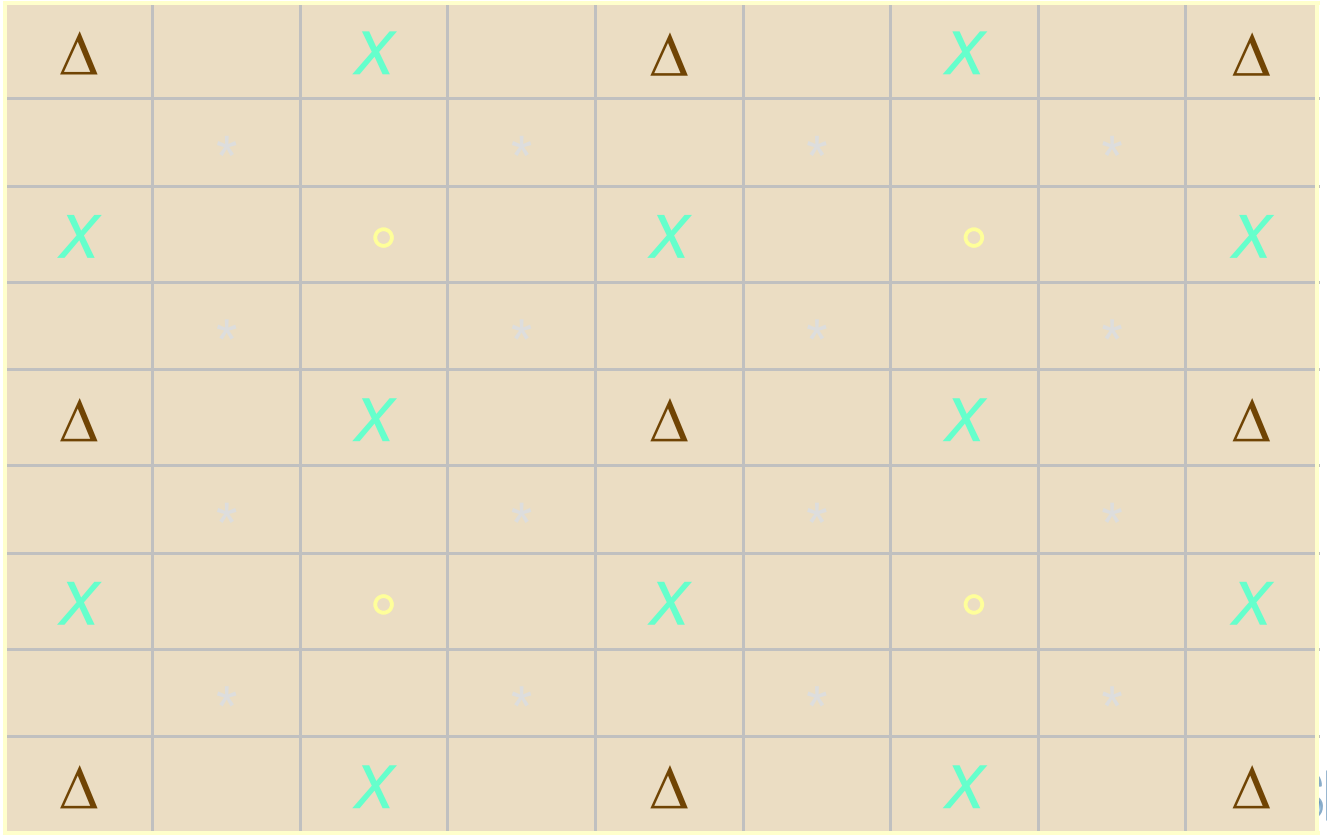
HINT (3)

- HINT (Hierarchical INTerpolation)
  - Δ → ◦ → X → ∗ → ●



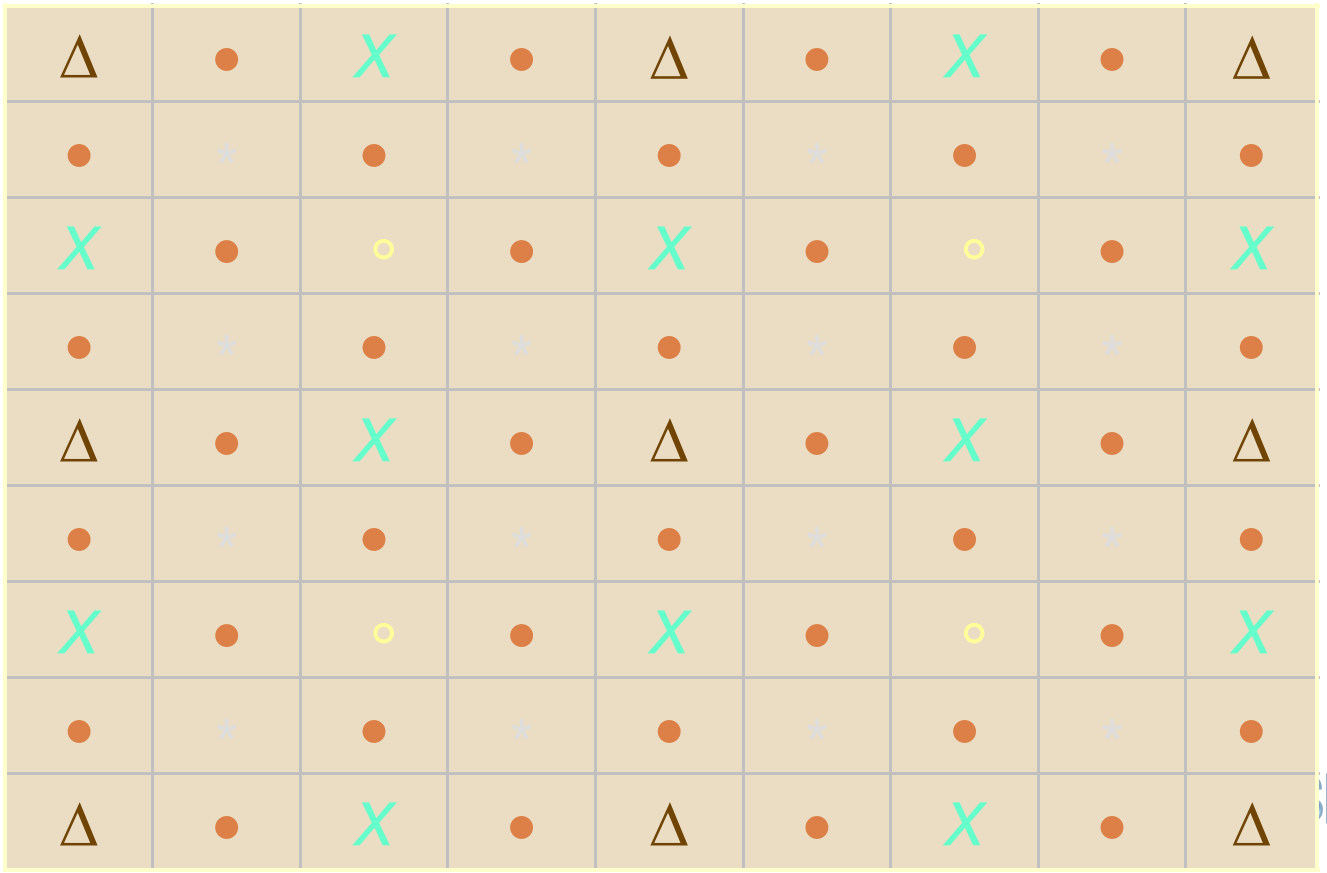
HINT (4)

- HINT (Hierarchical INTerpolation)
  - Δ → ◦ → X → * → ●



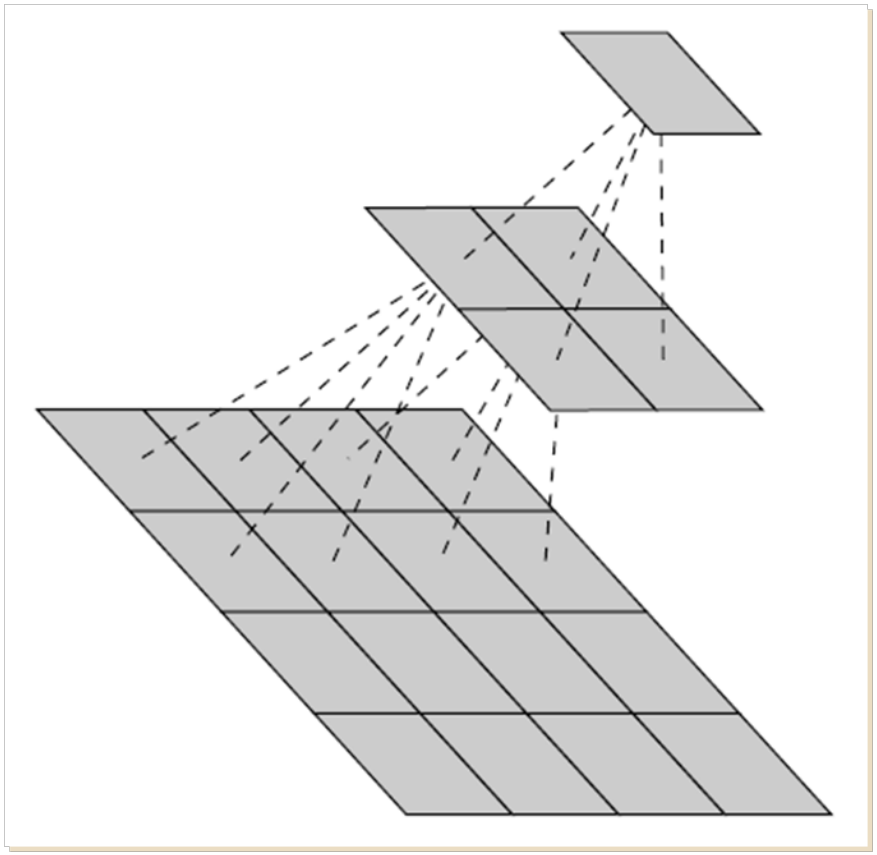
HINT (5)

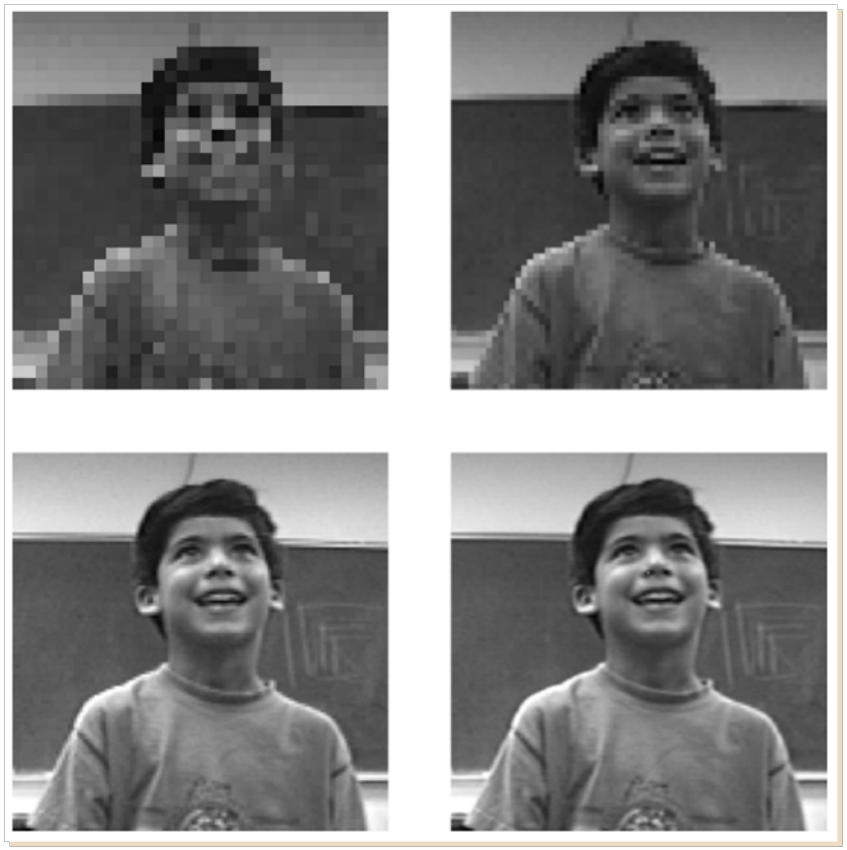
- HINT (Hierarchical INTerpolation)
  - Δ → ◦ → X → * → ●



Progressive Transmission

- Transmit a series of images at increasing resolution

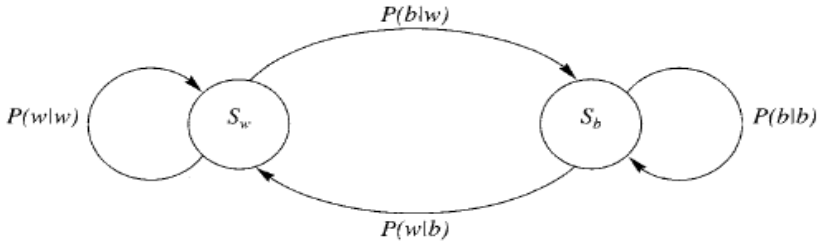




Facsimile Encoding

- Fax equipment
  - Group 1 (T.2): Analog, one A4 document (210x297mm) every 6 min over phone lines
  - Group 2 (T.3): Analog, one A4 page every 3 min.
  - Group 3 (T.4): Digital, compression, one A4/min.
  - Group 4 (T.6): Digital, compression, one A4/min.
  - Compression schemes
    - T.4/T.6 (MH/MR)
    - T.82 (JBIG)
    - T.88 (JBIG2)
    - T.44 (MRC)

Fax Coding



- Document scanned line-by-line
  - Sequence of black & white dots, called *pels* (Picture ELements)
  - Horizontal resolution:
    - 8.05 pels/mm (~205 pels/in)
    - 8.5in = 1728 pels, in reality 8.2in = 1664 pels
  - Vertical resolution:
    - 3.85/7.7/15.4 lines/mm == standard/fine/very fine modes
- One-dimensional coding (MH)
  - Scan lines coded independently as
  - Alternating runs of white/black/white/black/ ... pels
  - Modified Huffman (MH) = combination of RLE & Huffman
  - Code assignment based on statistics from 8(!) "typical" documents
    - CCITT copyrighted …

CCITT Training Documents

1. Typed business letter (English)
2. Circuit diagram (hand drawn)
3. Printed & typed invoice (French)
4. Densely typed report (French)
5. Printed technical article w/figures & equations (French)
6. Graph w/ printed captions (French)
7. Dense document (Kanji)
8. Handwritten memo w/ large white-on-black letters (English)

Modified Huffman (MH) Coding

- Most common pel sequences
  - 2, 3, 4 black pels 2/2/3-bits codes, followed by
  - 2-7 white pels: 4-bit codes
  - most of the rest coded with 12 bit codes
- Note
  - Runs can be up to 1728 pels long
- Solution
  - termination codes for sequences of 1-63 pels
  - make-up codes represent multiples of 64 pels

MH Termination Codes

Run length	White code-word	Black code-word	Run length	White code-word	Black code-word
0	00110101	0000110111	32	00011011	000001101010
1	000111	010	33	00010010	000001101011
2	0111	11	34	00010011	000011010010
3	1000	10	35	00010100	000011010011
4	1011	011	36	00010101	000011010100
5	1100	0011	37	00010110	000011010101
6	1110	0010	38	00010111	000011010110
7	1111	00011	39	00101000	000011010111
8	10011	000101	40	00101001	000001101100
9	10100	000100	41	00101010	000001101101
10	00111	0000100	42	00101011	000011011010
11	01000	0000101	43	00101100	000011011011
12	001000	0000111	44	00101101	000001010100
13	000011	00000100	45	00000100	000001010101
14	110100	00000111	46	00000101	000001010110
15	110101	000011000	47	00001010	000001010111
16	101010	0000010111	48	00001011	000001100100
17	101011	0000011000	49	01010010	000001100101
18	0100111	0000001000	50	01010011	000001010010
19	0001100	00001100111	51	01010100	000001010011
20	0001000	00001101000	52	01010101	000000100100
21	0010111	00001101100	53	00100100	000000110111
22	0000011	00000110111	54	00100101	000000111000
23	0000100	00000101000	55	01011000	000000100111
24	0101000	00000010111	56	01011001	000000101000
25	0101011	00000011000	57	01011010	000001011000
26	0010011	000011001010	58	01011011	000001011001
27	0100100	000011001011	59	01001010	000000101011
28	0011000	000011001100	60	01001011	000000101100
29	00000010	000011001101	61	00110010	000001011010
30	00000011	000001101000	62	00110011	000001100110
31	00011010	000001101001	63	00110100	000001100111

MH Make-up Codes

Run length	White code-word	Black code-word	Run length	White code-word	Black code-word
64	11011	0000001111	1344	011011010	0000001010011
128	10010	000011001000	1408	011011011	0000001010100
192	010111	000011001001	1472	010011000	0000001010101
256	0110111	000001011011	1536	010011001	0000001011010
320	00110110	000000110011	1600	010011010	0000001011011
384	00110111	000000110100	1664	011000	0000001100100
448	01100100	000000110101	1728	010011011	0000001100101
512	01100101	0000001101100	1792	00000001000	same as
576	01101000	0000001101101	1856	00000001100	white
640	01100111	0000001001010	1920	00000001101	from this
704	011001100	0000001001011	1984	000000010010	point
768	011001101	0000001001100	2048	000000010011	
832	011010010	0000001001101	2112	000000010100	
896	011010011	0000001110010	2176	000000010101	
960	011010100	0000001110011	2240	000000010110	
1024	011010101	0000001110100	2304	000000010111	
1088	011010110	0000001110101	2368	000000011100	
1152	011010111	0000001110110	2432	000000011101	
1216	011011000	0000001110111	2496	000000011110	
1280	011011001	0000001010010	2560	000000011111	

MH Coding Examples & Rules

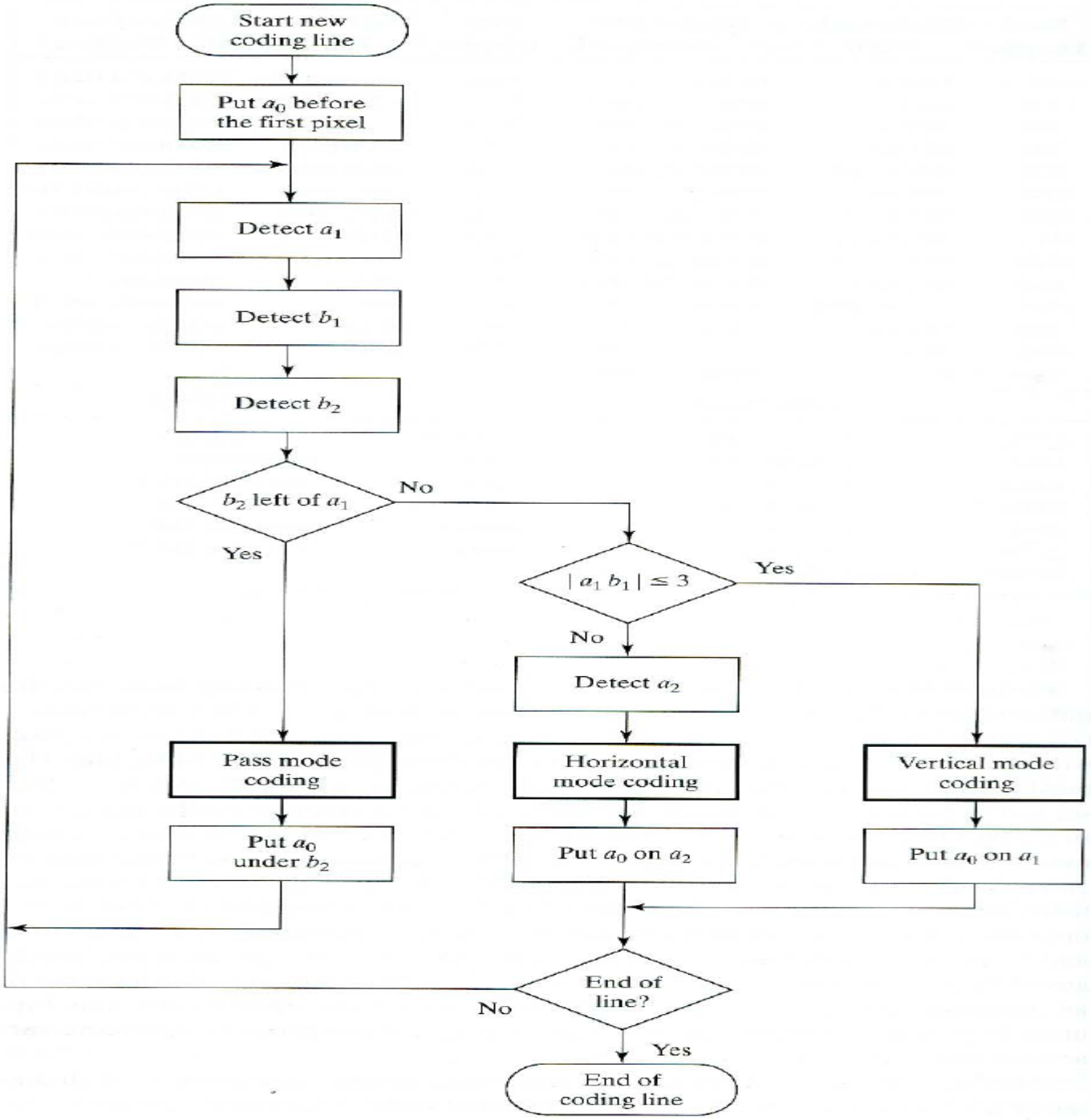
- 12 white pels
  - 001000
- 76 white pels (64 + 12)
  - 11011 001000
- 140 white pels (128 + 12)
  - 10010 001000
- 64 black pels (64 + 0)
  - 0000001111 0000110111
- Each line
  - is coded separately;
  - gets an extra white pel on the left;
  - ends with a 12-bit EOL: 000000000001

MH Coding Examples & Rules (2)

- Examples
  - ■■■□□■■□□□□□□□
    1w 3b 2w 2b 7w
  - □□■■■■■□□□□□■■
    3w 5b 5w 2b
- Error correction
  - Not explicit: based on EOL & knowledge of valid codes
- Page separators
  - Beginning: 1x EOL
  - End: 6x EOL
- T4 allows for padding b/w data & EOL

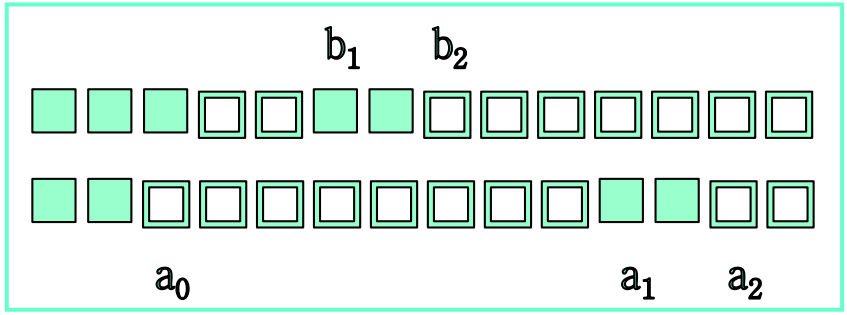
Two-dimensional Coding (MR)

- **M**odified **R**EAD
  - READ = **R**elative **E**lement **A**ddress **D**esignate
- Reference line
  - Last decoded line
- Code line
  - Current line being encoded
- Notation
  - a_0: first pel of a new codeword (either black or white)
  - a_1: first pel to the right a_0 of opposite color
  - a_2: first pel to the right a_1 of opposite color
  - b_1: first pel to the right a_0 on the ***reference line*** of opposite color
  - b_2: first pel to the right b_1 on the ***reference line*** of opposite color


Start new coding line
Put a_0 before the first pixel
Detect a_1
Detect b_1
Detect b_2
b_2 left of a_1
No
Yes
$|a_1 b_1| \leq 3$
Yes
No
Detect a_2
Pass mode coding
Horizontal mode coding
Vertical mode coding
Put a_0 under b_2
Put a_0 on a_2
Put a_0 on a_1
End of line?
No
Yes
End of coding line

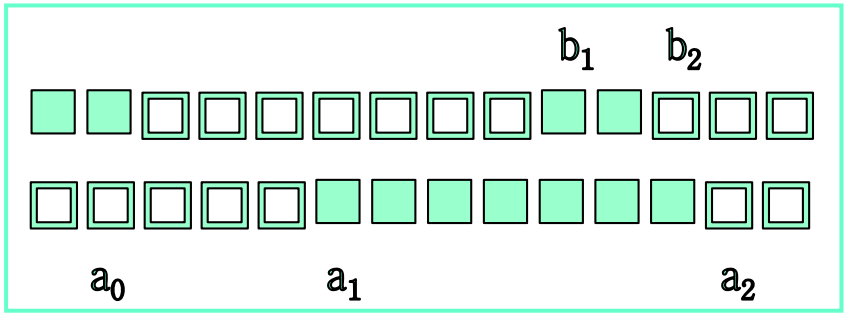
MR Coding

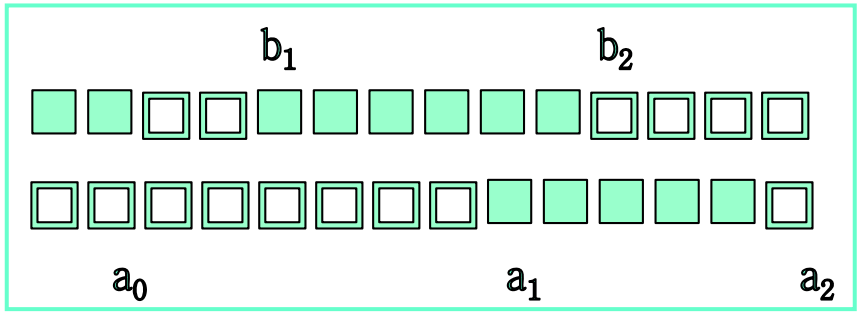
- Three modes
  - Pass, Vertical, Horizontal
  - Based on comparing (a_0a_1), (a_1a_2), and (b_1b_2)
- Pass mode
  - Condition: (b_1b_2) is to the left of (a_1a_2) and b_2 is strictly to the left of a_1
  - Run encoded: (b_1b_2)
  - Codeword (P): 0001 + coded length of (b_1b_2)
  - Update: $a_0 = b_2$ & the rest according to definitions
  - Example:



MR Coding (2)

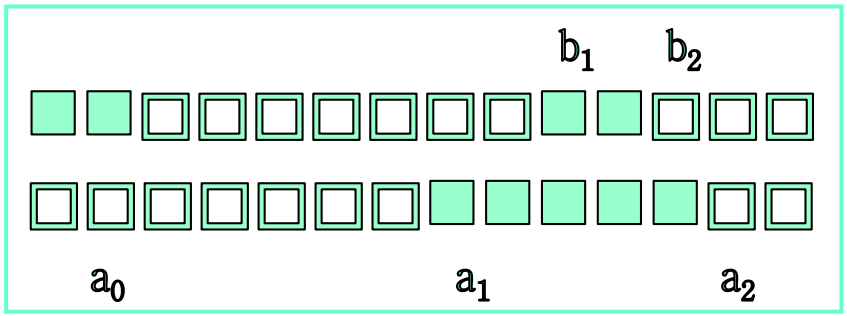
- Horizontal mode
  - Condition: (b_1b_2) overlaps (a_1a_2) by ***more than 3*** pels
    - as determined by $|a_1\text{-}b_1|$
  - Runs encoded: (a_0a_1), (a_1a_2)
  - Codeword (H): 001 + coded lengths of (a_0a_1), (a_1a_2)
  - Update: $a_0 = a_2$ & the rest according to definitions
  - Examples:

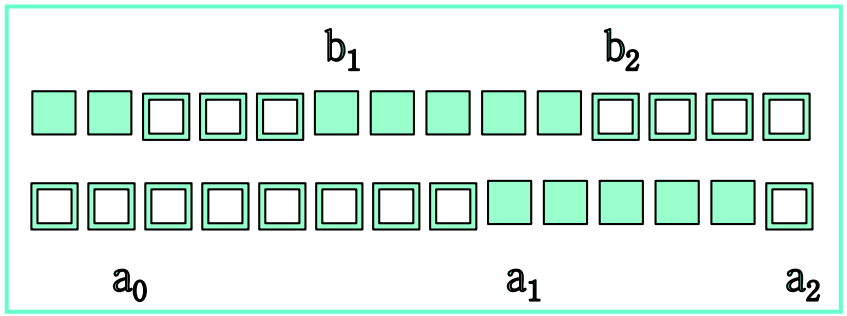




MR Coding (3)

- Vertical mode
  - Condition: (b_1b_2) overlaps (a_1a_2) by ***no more than 3*** pels
  - Expected as most common (assuming small differences)
  - Run encoded: (a_1b_1)
  - Codeword:
    - -3/-2/-1/0/1/2/3 → VR(3)/VR(2)/VR(1)/V(0)/VL(1)/VL(2)/VL(3)
    - 1/000001/00001/011/1/010/000010/0000010
  - Update: $a_0 = a_1$ & the rest according to definitions
  - Examples:





Two-dimensional Coding (Modified Read: MR)

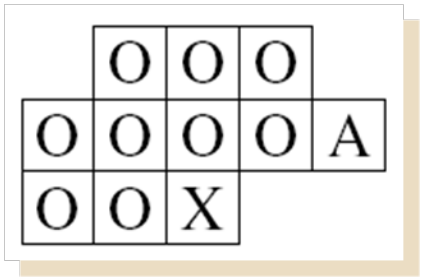
Mode	Code Word
Pass	0001
Horizontal	$001 + M(a_0a_1) + M(a_1a_2)$
Vertical	
a_1 below b_1	1
a_1 one to the right of b_1	011
a_1 two to the right of b_1	000011
a_1 three to the right of b_1	0000011
a_1 one to the left of b_1	010
a_1 two to the left of b_1	000010
a_1 three to the left of b_1	0000010
Extension	0000001××××

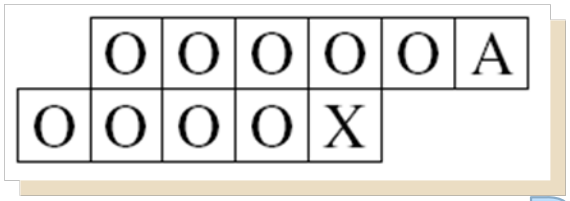
MR Coding (4)

- Extension
  - Code: 0000001000
  - Used to switch compression method in the middle of the page
- Error correction
  - Errors can propagate in a cascading fashion
  - T.4: After coding a line with a 1D code, at most K - 1 should 2D coded
    - $K = 2$, $K = 4$
- T.6:
  - Same as T.4 but no provision for 1D coding
    - Add a 'white' line at the top
  - A.k.a., **M**odified **M**odified **R**EAD (MMR)

JBIG /T.82/

- Joint Bi-Level Image Processing Group
- Basic ideas
  - Utilize context-based prediction (10/12-bit Markov models)
  - Allow context to be anchored anywhere in decoded area
  - Use arithmetic coding with 1024/4096 coders
  - Define the format & decoder, leave encoder open
- Pixel neighborhoods
  - Serve as context
    - 10 pixels = 2^{10} contexts





QM Coder

- AC variant
  - Instead of interval boundaries ($I^{(n)}$ & $u^{(n)}$), keep track of lower bound and interval length ($I^{(n)}$ & $A^{(n)}$)
    - $A^{(n)} = u^{(n)} - I^{(n)}$

$$A^{(n)} = A^{(n-1)}(F_X(x_n) - F_X(x_n - 1)) = A^{(n-1)}P(x_n).$$

$$l^{(n)} = l^{(n-1)} + A^{(n-1)}F_X(x_n - 1).$$

Coding

- Given context C, conditional probability is denoted as q_c
- More probable symbol (MPS) vs. Less probable symbol (LPS)

$$l^{(n)} = l^{(n-1)}$$
$$A^{(n)} = A^{(n-1)}(1 - q_c)$$

$$l^{(n)} = l^{(n-1)} + A^{(n-1)}(1 - q_c)$$
$$A^{(n)} = A^{(n-1)}q_c.$$

QM Coder (2)

- Simplifications
  - x * a ≈ a, when a ≈ 1
  - MPS
    - $I^{(n)} = I^{(n-1)}$
    - $A^{(n)} = 1 - q_C$
  - LPS:
    - $I^{(n)} = I^{(n-1)} (1 - q_C)$
    - $A^{(n)} = q_C$
- Rescaling
  - If $A^{(n)}$ drop below 0.75, coder rescales (double) until $A^{(n)} > 0.75$
  - Observation:
    - LPS always triggers rescaling, MPS: not necessarily
- Adaptation
  - q_C is updated in every rescaling
- Reversal
  - If $q_C > (A^{(n)} - q_C)$ then LSP & MPS are swapped

MH vs. MR vs. MMR vs. JBIG

Source Description	Original Size (pixels)	MH (bytes)	MR (bytes)	MMR (bytes)	JBIG (bytes)
Letter	4352 × 3072	20,605	14,290	8,531	6,682
Sparse text	4352 × 3072	26,155	16,676	9,956	7,696
Dense text	4352 × 3072	135,705	105,684	92,100	70,703

JBIG2 /T.88/ [2000]

- Features
  - Better performance
    - 3-5x MMR, 2-4x JBIG
  - Specialized methods for text, half-tones, and bi-level
  - Lossy & lossless
  - Two modes of progressive compression based on
    - Quality
    - Content
  - Multipage document compression
  - Flexible format designed for embedding
  - Fast decompression
  - Decoder specified (like JBIG)

JBIG2 Regions

- Text
  - Any letters, symbols, hieroglyphs, musical notes, etc.
    - Usually organized in rows or columns
  - A symbol is a rectangular bitmap
  - Symbols are placed in a dictionary
  - Dictionary is either AC/MQ- or MMR-compressed and written to the file
  - Coding
    - Relative coordinated w.r.t. previous symbol
    - Pointer to dictionary entry
  - Advanced coding
    - Multiple bitmaps and binary operations on them: AND, OR, XOR
  - Lossy compression
    - (Carefully!) mapping of multiple symbols to the same bitmap

JBIG2 Regions (2)

- Halftone
  - Consists of halftone ***cells*** that are 3x3 or 4x4
  - Dictionary:
    - Treat cells as integers
    - Replace cells with pointers in the dictionary
  - Same dictionary can be used for many pages
- Generic (everything else)
  - Either AC- or MMR-coded
- Page
  - Contains any number of (potentially overlapping) regions
    - Determined by coder
- Lossy compression
  - Decoder is generally unaware of loss
- Refinements
  - Regions can be decoded into buffers and reused for subsequent pages

JBIG2 Regions Illustration

1. Text regions. These contain text, normally arranged in rows. The text does not have to be in any specific language, and may consist of unknown symbols, dingbats, music notes, or hieroglyphs. The encoder treats each symbol as a rectangular bitmap which it places in a dictionary. The dictionary is compressed by either arithmetic coding or MMR and written, as a segment, on the compressed file. Similarly, the text region itself is encoded and written on the compressed file as another segment. To encode a text region the encoder prepares the relative coordinates of each symbol (the coordinates relative to the preceding symbol), and a pointer to the symbol's bitmap in the dictionary. (It is also possible to have pointers to several bitmaps and specify the symbol as an aggregation of those bitmaps, suchas logical AND, OR, or XOR.) This data also is encoded before being written on the compressed file. Compression is achieved if the same symbol occurs several times. Notice that thesymbol may occur in different text regions

Example of shearing

2. Halftone regions. A bi-level image may contain a grayscale image done in halftone. The encoder scans such a region cell by call (halftone cells, also called patterns, are typically. A halftone dictionary is created by collecting all the different cells, and converting each to an integer (typically 9 bits or 16 bits).

Lena in halftone (2 by 2 inches)

JBIG2 Procedures

- Start
  - Initialize page to all zeroes or ones
- Decoding procedures for
  - Segment headers (segments encode regions)
  - Generic regions
  - Generic refinement regions
  - Symbol dictionary
    - Refinements, aggregates
  - Symbol region
    - <dx, dy, pointer<
  - Halftone dictionary
  - Halftone region

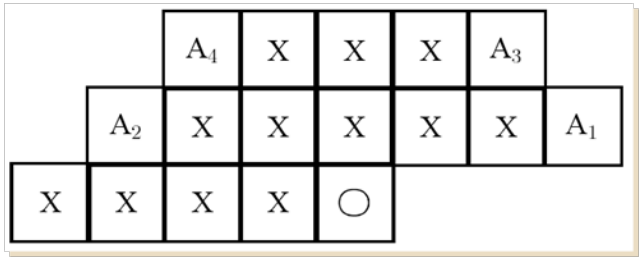
JBIG2 Generic Region Decoding

Name	Type	Description
MMR	UINT-1	MMR or AC?
GBW	UINT-32	Region width
GBH	UINT-32	Region height
GBTEMPLATE	UINT-2	Template number
TPGDON	UINT-1	Typical prediction?
USESKIP	UINT-1	Skip pixels?
SKIP	BITMAP	Bitmap for skipping
$GBATX_1$	INT-8	Relative ***X*** coordinate for A_1
$GBATY_1$	INT-8	Relative ***Y*** coordinate for A_1
$GBATX_2$	INT-8	Relative ***X*** coordinate for A_2
$GBATY_2$	INT-8	Relative ***Y*** coordinate for A_2
$GBATX_3$	INT-8	Relative ***X*** coordinate for A_3
$GBATY_3$	INT-8	Relative ***Y*** coordinate for A_3
$GBATX_4$	INT-8	Relative ***X*** coordinate for A_4
$GBATY_4$	INT-8	Relative ***Y*** coordinate for A_4

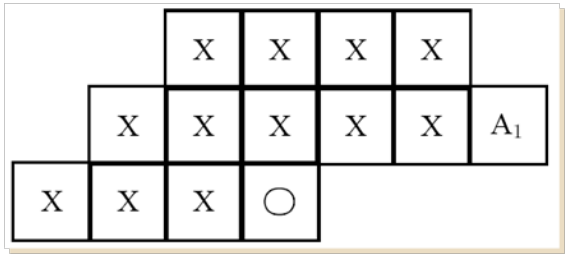
JBIG2 Generic Region Decoding (2)

- GBTEMPLATE
  - Which of four templates are used

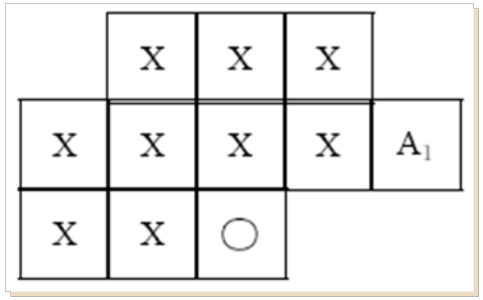
GBTEMPLATE==0



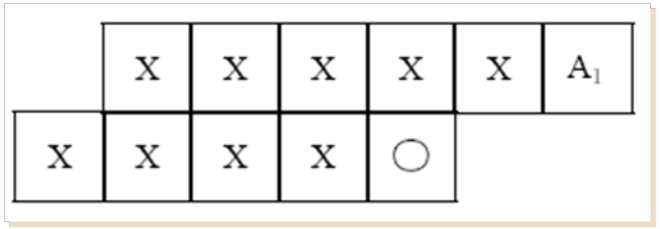
GBTEMPLATE==1



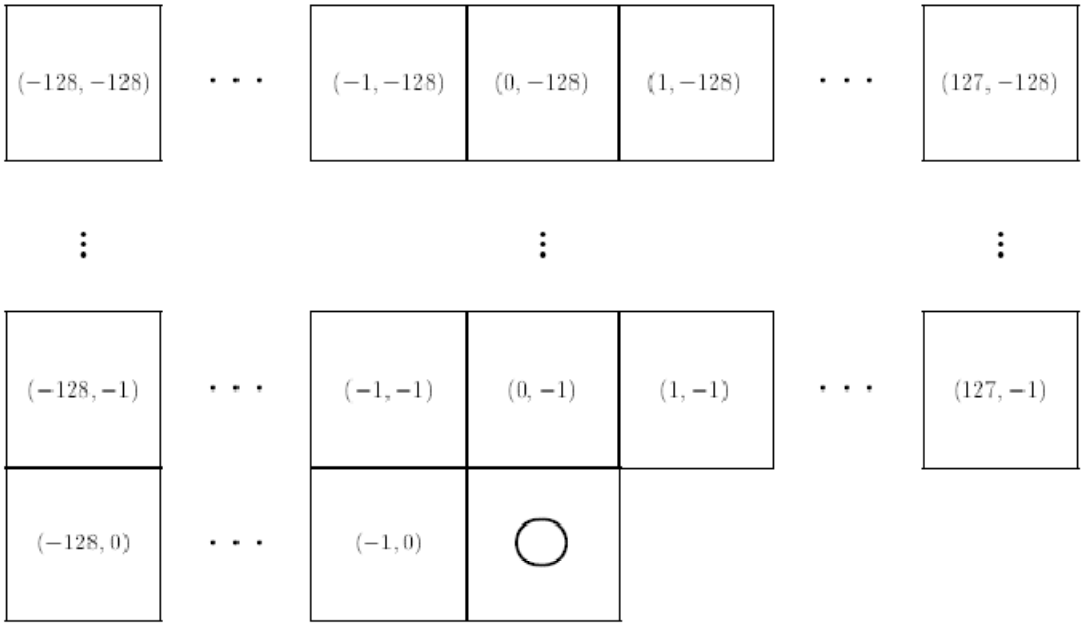
GBTEMPLATE==2



GBTEMPLATE==3



JBIG2 Generic Region Decoding (3)



Allowed AT pixel locations

GBTEMPLATE	$GBATX_1$ $GBATY_1$	$GBATX_2$ $GBATY_2$	$GBATX_3$ $GBATY_3$	$GBATX_4$ $GBATY_4$
0	3 -1	-3 -1	2 -2	-2 -2
1	3 -1	NA NA	NA NA	NA NA
2	2 -1	NA NA	NA NA	NA NA
3	2 -1	NA NA	NA NA	NA NA

Nominal AT pixel locations

PSP LAB

JBIG2 Generic Region Decoding (4)

- Typical prediction for generic direct coding (TPGD)
  - Typical prediction--row is identical to last
- Skipped pixels
  - if USESKIP then SKIP is a GTWxGTH bitmap
  - AC-coded using templates

Mixed Raster Content MRC /T.44/

- General approach
  - Define a horizontal partitioning scheme
  - Use appropriate existing methods for compression
- Three layers
  - Background, foreground, and mask/selector
- Three targeted data types
  - color images (continuous tone or color-mapped)
  - bi-level data
  - multilevel (multicolor) data.
- Wherever the mask layer pixel is black (I) we pick the corresponding pixel from the foreground layer. Wherever the mask pixel is white (0) we use the pixel from the background layer.

MRC Example

It Will soon be June 4

That's Ruby's Birthday!

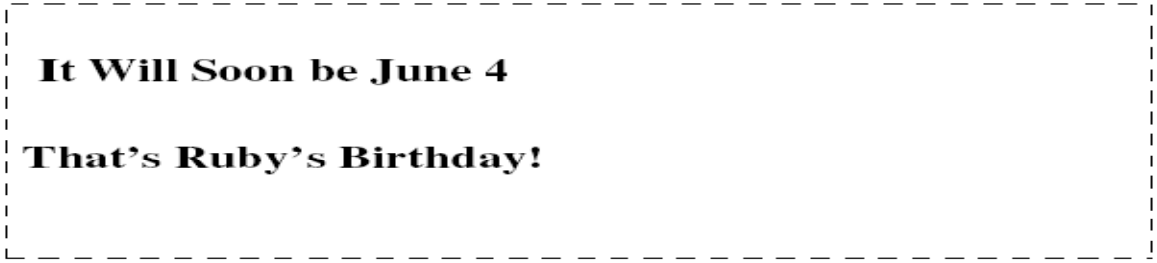
You are invited to a PARTY

to CELEBRATE

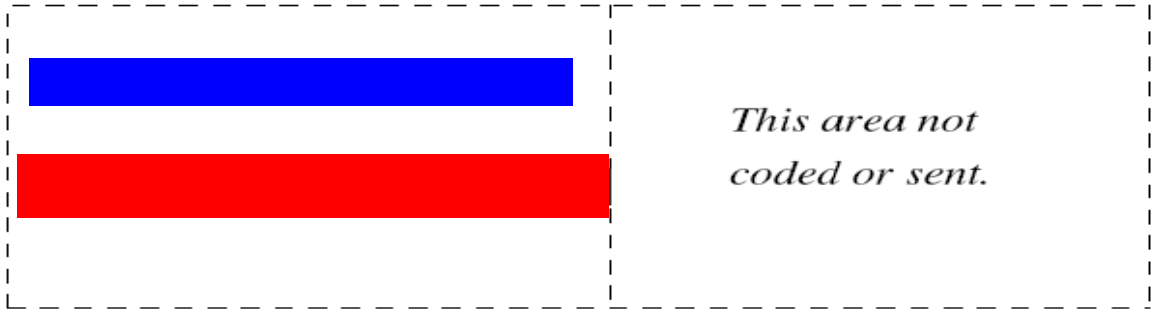
with Ruby and Hanna

MRC Example: Layers

Background



Mask/selector (b/w)

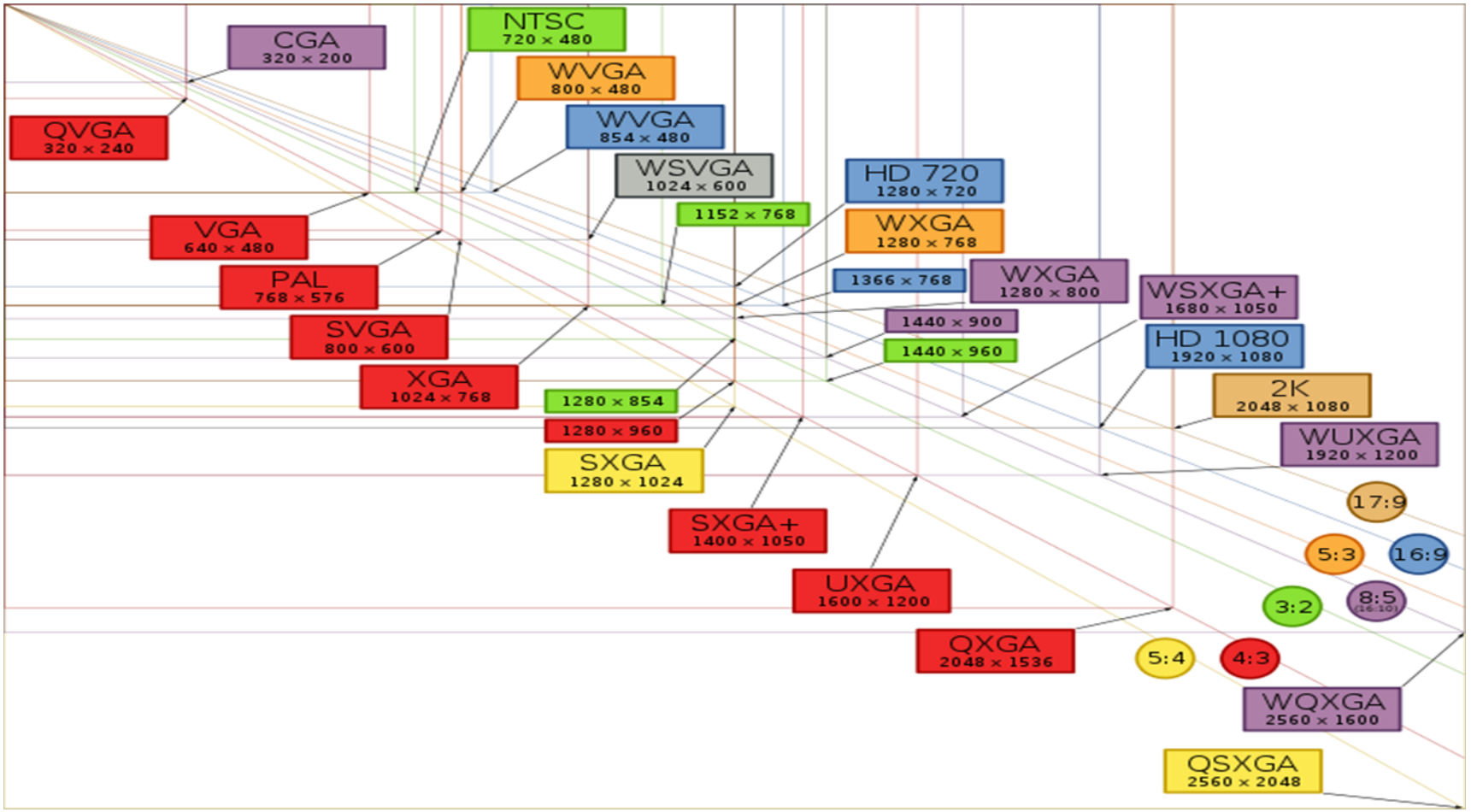


Foreground

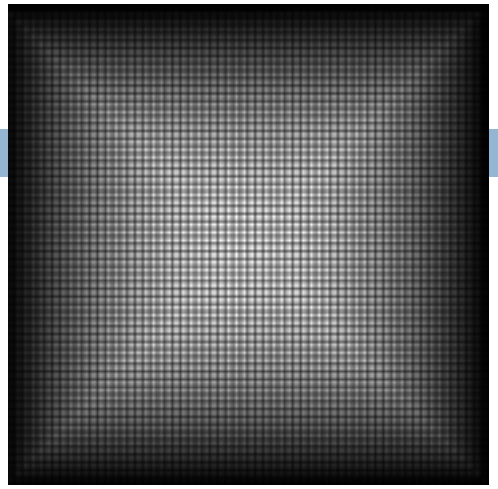
Homeworks & References

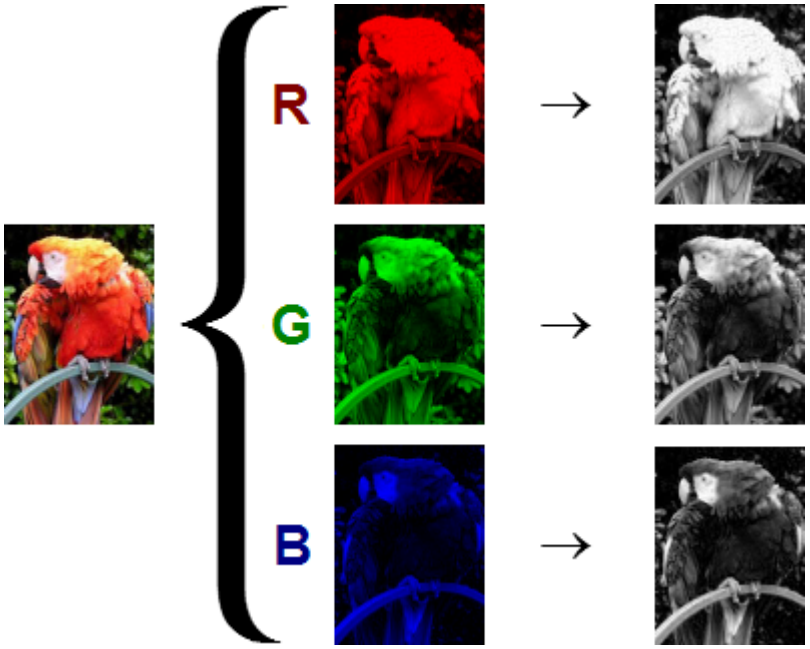
- Preparation problems (Sayood 3rd, pp. 193-194)
  - 1, 2, 3, 4
- References
  - http://hfrp.umm.edu/People/Hao/ENEE728I/ENEE728I_CALIC.pdf.

Resolution v.s. Aspect Ratio



PSP LAB




R
G
B