CS/EE 5590 / ENG 401 Special Topics

Multimedia Communication, Spring 2017

Lec 04
Variable Length Coding in JPEG

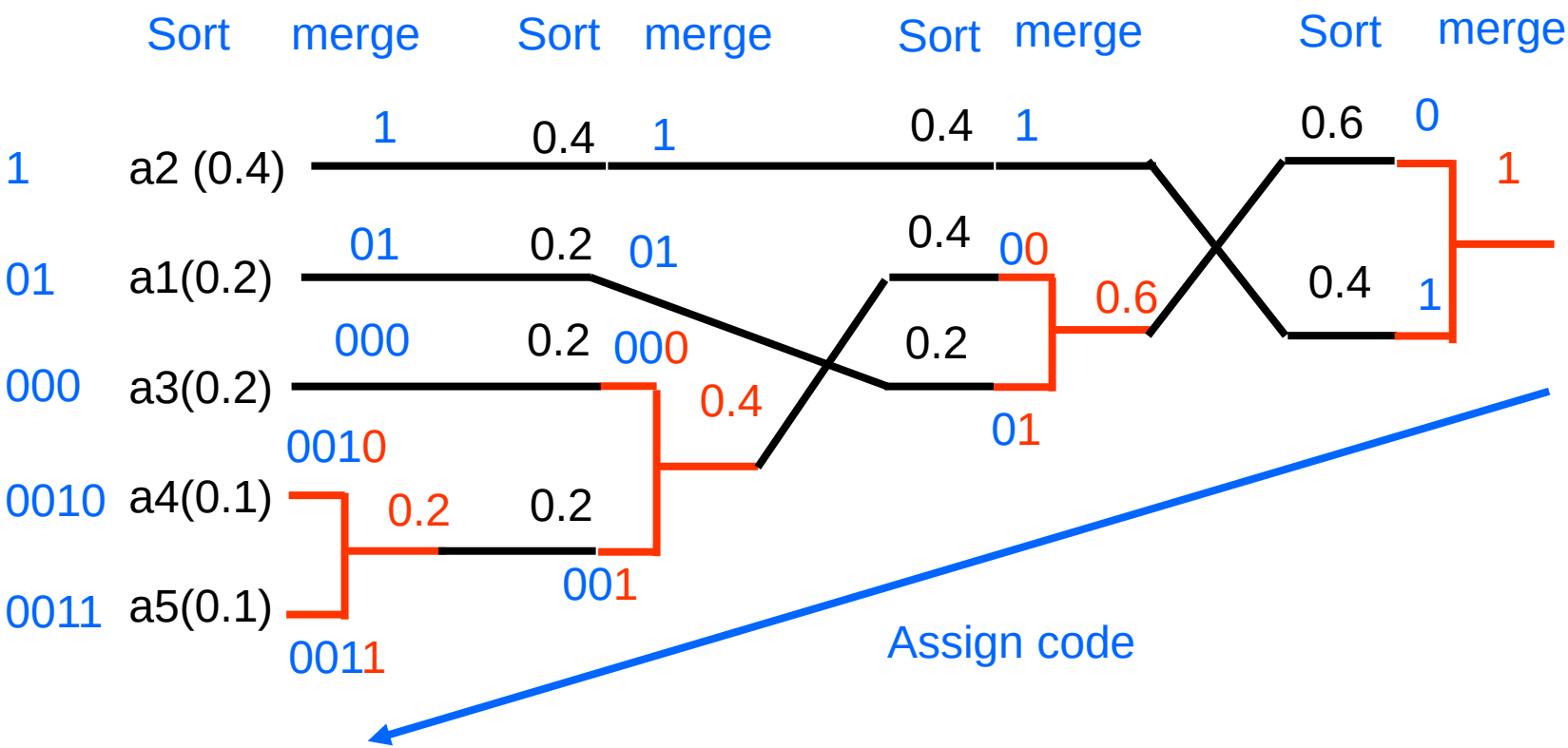
Zhu Li

Outline

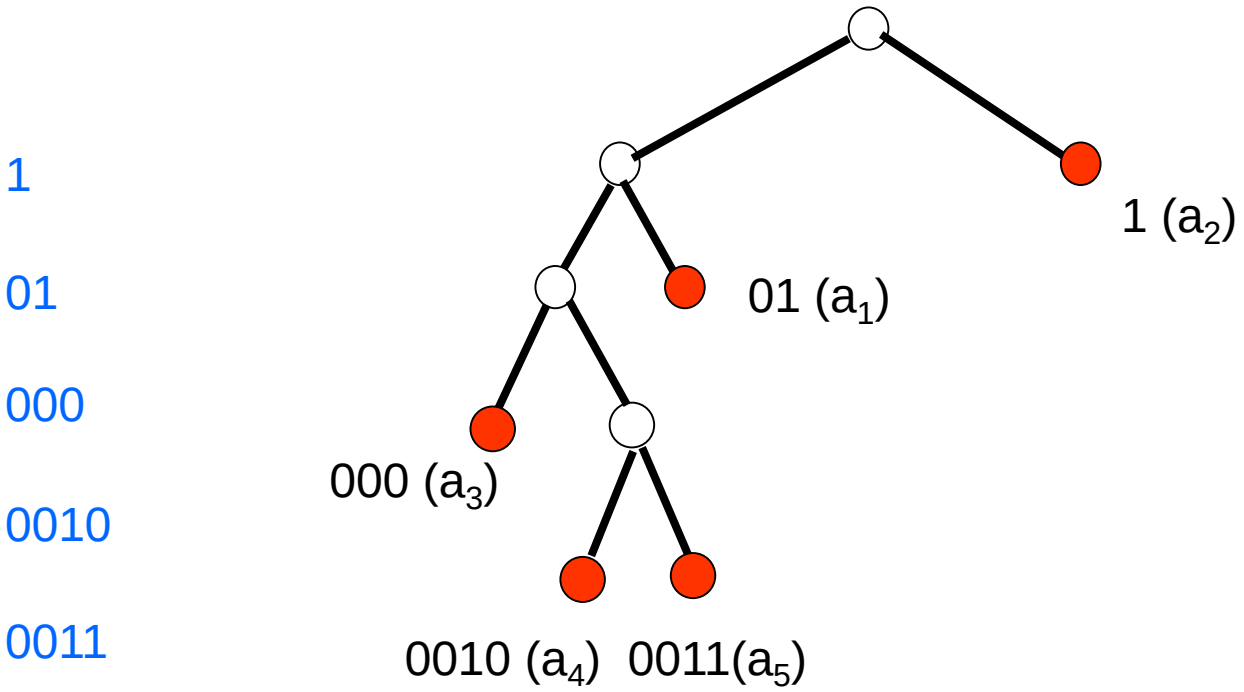
- Lecture 03 ReCap
- VLC
- JPEG Image Coding Framework

Hoffmann Coding

- Hoffman Coding: iterative sort and merge to create a binary tree, assigning the bit value along the way, reverse for code
- Example
  - Source alphabet A = {a_1, a_2, a_3, a_4, a_5}, Probability distribution: {0.2, 0.4, 0.2, 0.1, 0.1}



Huffman code is prefix-free



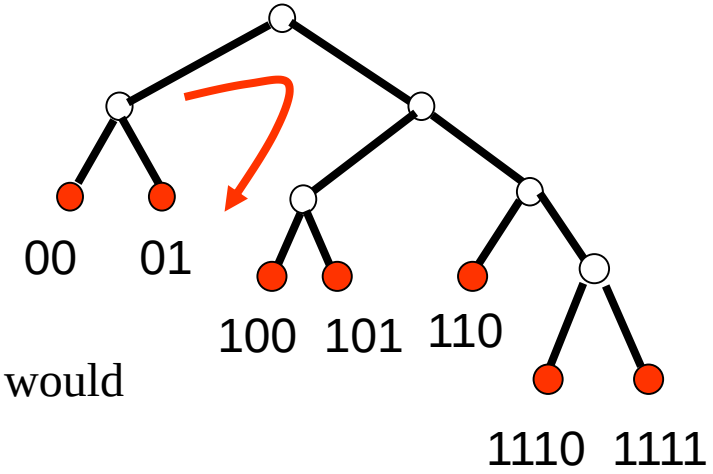
- All codewords are *leaf nodes*
  - ➔ No code is a prefix of any other code. (Prefix free)

Canonical Huffman

- Properties:
  - The first code is a series of 0
  - Codes of same length

    are consecutive: 100, 101, 110
  - If we pad zeros to the right side such that all codewords have the same length, shorter codes would have lower value than longer codes:

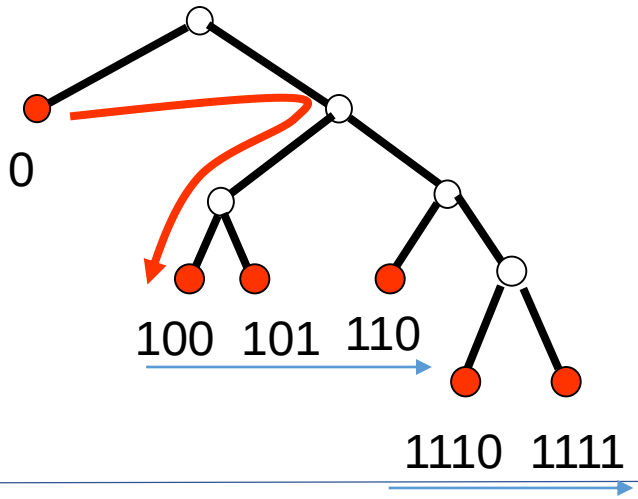
    0000 < 0100 < 1000 < 1010 < 1100 < 1110 < 1111



- Coding from length level *n* to level *n*+1:
  - C(*n*+1, 1) = 2 (C(*n*, last) + 1): append a 0 to the next available level-n code

    First code of length n+1

    Last code of length n



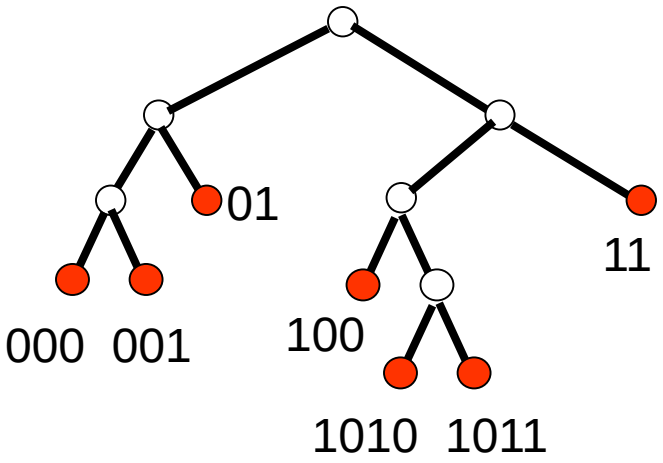
- If from length n to n + 2 directly:

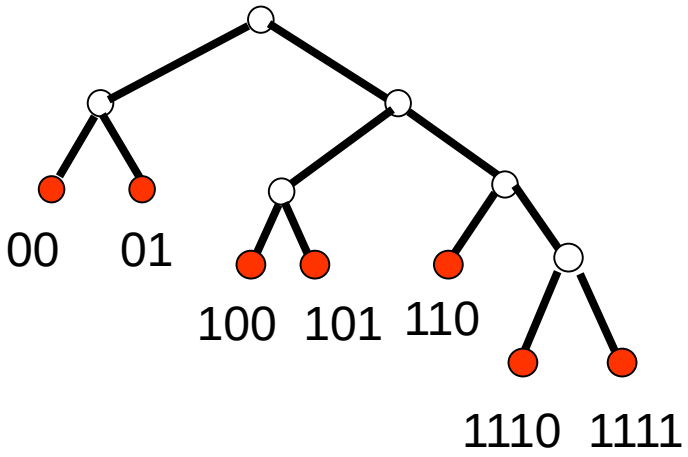
  e.g., 1, 3, 3, 3, 4, 4

  C(n+2, 1) = 4(C(n, last) + 1)

Advantages of Canonical Huffman

1. Reducing memory requirement





- Non-canonical tree needs:
  - All codewords
  - Lengths of all codewords
- Need a lot of space for large table

- Canonical tree only needs:
  - Min: shortest codeword length
  - Max: longest codeword length
  - Distribution:
    - Number of codewords in each level (consecutive)
    - Min=2, Max=4,# in each level: 2, 3, 2

Golomb Code

$$n = qm + r = \left\lfloor \frac{n}{m} \right\rfloor m + r$$

- q: Quotient, used unary code

q	Codeword
0	0
1	10
2	110
3	1110
4	11110
5	111110
6	1111110
…	…

- r: remainder, "fixed-length" code
- K bits if m = 2^k
  - m=8: 000, 001, ……, 111
- If m ≠ 2^k: (not desired)

  $\lfloor \log_2 m \rfloor$ bits for smaller r

  $\lceil \log_2 m \rceil$ bits for larger r

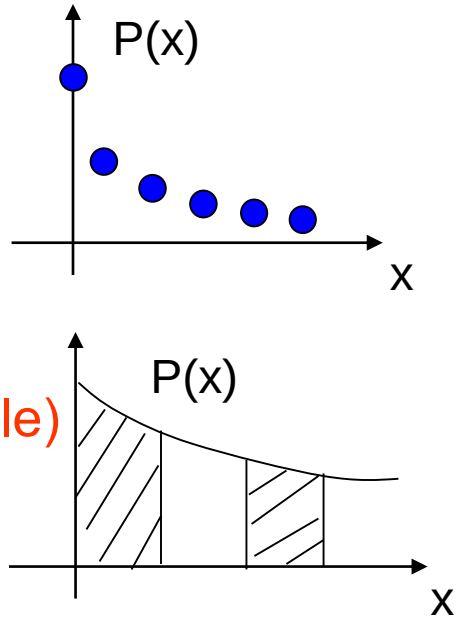
  m = 5: 00, 01, 10, 110, 111

Golomb is Optimal for Geometric Distribution

- Geometric distribution with parameter ρ:
  - P(X=n) = $\rho^n (1-\rho)$
- Unary code is optimal prefix code when ρ ≤ 1/2.
  - Also optimal among all entropy coding for ρ = 1/2.
- How to design the optimal code when ρ > 1/2 ?
  - Transform into GD with ρ ≤ 1/2 (as close as possible)

    How? By grouping m events together!

Each x can be written as $x = x_q m + x_r$



$$P_{X_q}(q) = \sum_{r=0}^{m-1} P_X(qm+r) = \sum_{r=0}^{m-1} (1-\rho)\rho^{qm+r} = (1-\rho)\rho^{qm}\frac{1-\rho^m}{1-\rho} = \rho^{mq}(1-\rho^m)$$

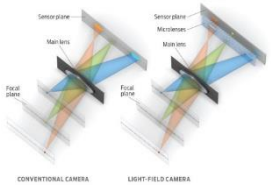
➔ x_q has geometric dist with parameter ρ^m.

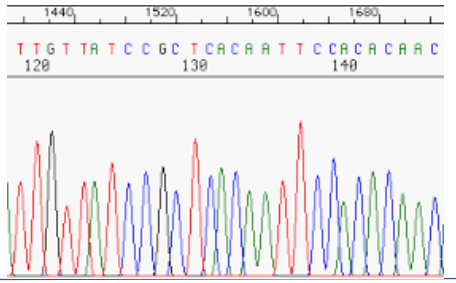
Unary code is optimal for x_q if $\rho^m \le 1/2$ ➔

$$m \ge -\frac{1}{\log_2 \rho} \qquad m = \left\lceil -\frac{1}{\log_2 \rho} \right\rceil \text{ is the minimal possible integer.}$$

Tentative Topics for Course Projects

- Compression Part:
  - Key Points Compression – SIFT compression (my seminar this Friday)
  - Point Cloud Compresison
  - Light Field Compression
  - Immersive Video Compression
  - DNA Sequence Compression
  - Super Resolution
- Objective
  - Deep dive to understand the technical details
  - Replicate data set, test model results
  - Innovation (25% extra credits)



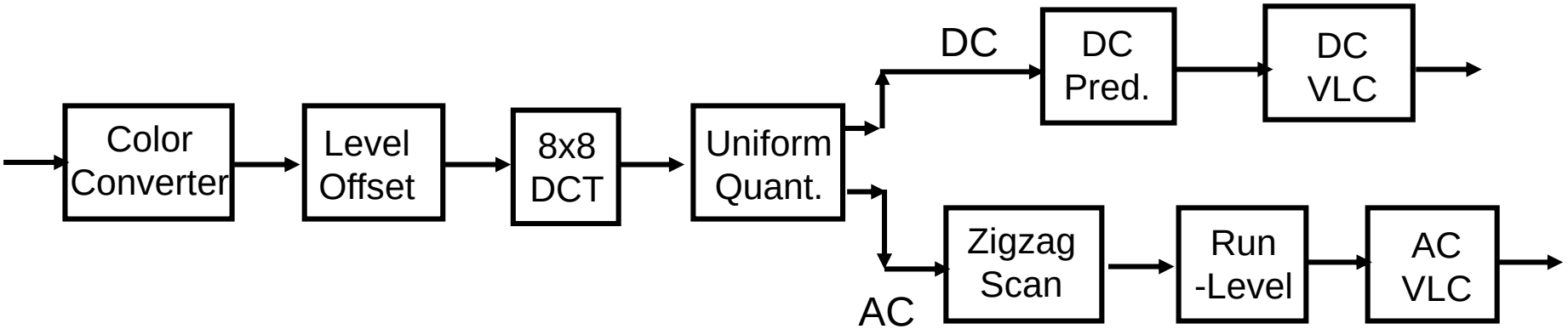


Outline

- VLC: variable length coding
  - Examples: Huffman coding, Golumb coding
- VLC Coding in
  - JPEG
  - H.264

Overall Structure of JPEG

- Color converter: RGB to YUV
- Level offset: subtract 2^(N-1). N: bits / pixel.
- Quantization: Different step size for different coeffs
- DC: Predict from DC of previous block
- AC:
  - Zigzag scan to get 1-D data
  - Run-level: joint coding of non-zero coefficients and number of zeros before it.



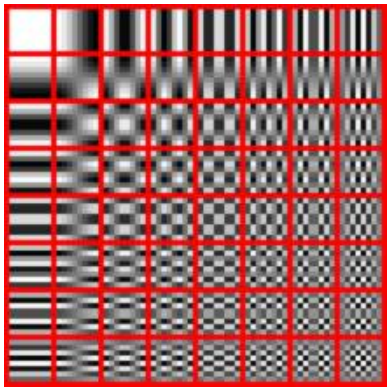
JPEG DCT and Quantization Examples

❑ DCT – Discrete Cosine Transform

$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

- $\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } u = 0 \\ 1, & \text{otherwise} \end{cases}$

  - Matlab: coef = dct2(im);

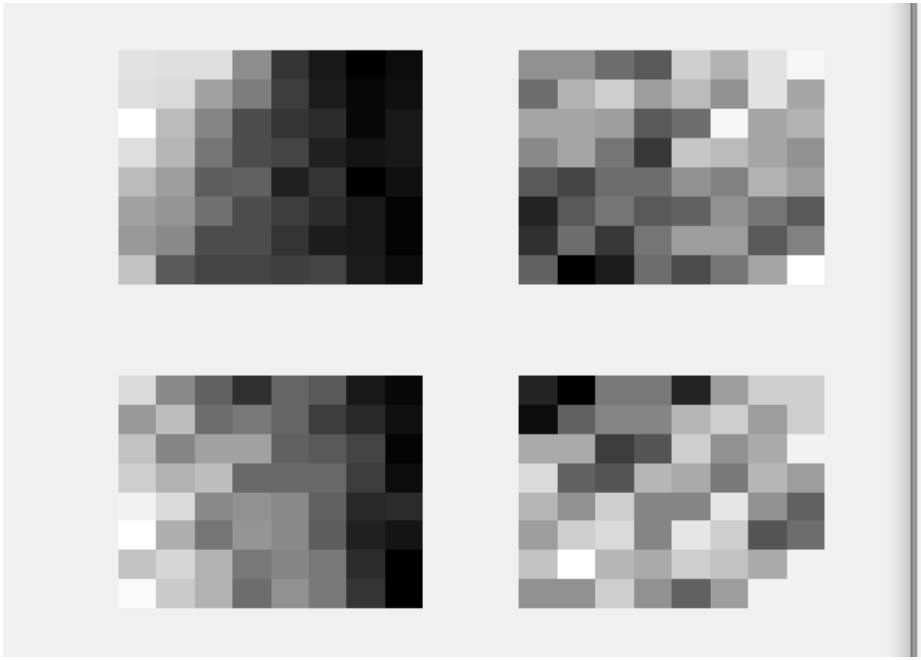


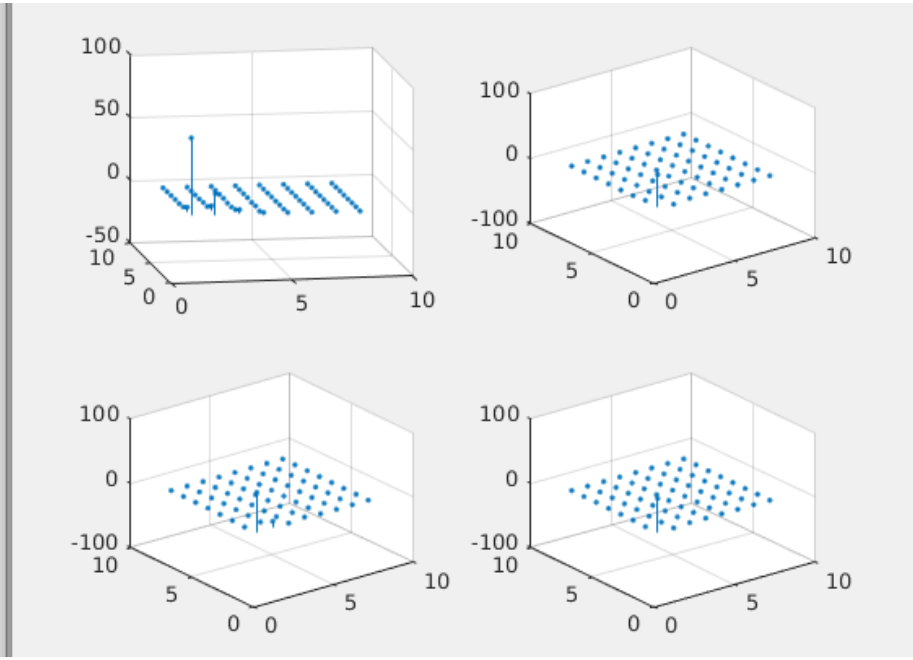
DCT Basis

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

DCT Quant

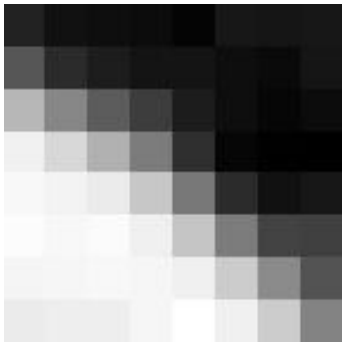
❑ Quantization in JPEG





2-D 8-point DCT Example

- Original Data:



89	78	76	75	70	82	81	82
122	95	86	80	80	76	74	81
184	153	126	106	85	76	71	75
221	205	180	146	97	71	68	67
225	222	217	194	144	95	78	82
228	225	227	220	193	146	110	108
223	224	225	224	220	197	156	120
217	219	219	224	230	220	197	151

- 2-D DCT Coefficients (after rounding to integers):

Most energy is in the upper-left corner

1155	259	-23	6	11	7	3	0
-377	-50	85	-10	10	4	7	-3
-4	-158	-24	42	-15	1	0	1
-2	3	-34	-19	9	-5	4	-1
1	9	6	-15	-10	6	-5	-1
3	13	3	6	-9	2	0	-3
8	-2	4	-1	3	-1	0	-2
2	0	-3	2	-2	0	0	-1

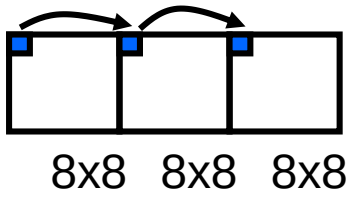
Coefficient Category – ExpGolomb Like

- Divide coefficients into categories of exponentially increased sizes
- Use Hoffman code to encode category ID (unary in Golomb)
- Use fixed length code within each category
- ➔ Similar to Exponential Golomb code

Ranges	Range Size	DC Cat. ID	AC Cat. ID
0	1	0	N/A
-1, 1	2	1	1
-3, -2, 2, 3	4	2	2
-7, -6, -5, -4, 4, 5, 6, 7	8	3	3
-15, …, -8, 8, …, 15	16	4	4
-31, …, -16, 16, …, 31	32	5	5
-63, …, -32, 32, …, 63	64	6	6
…	…	…	…
[-32767, -16384], [16384, 32767]	32768	15	15

Coding of DC Coefficients

❑Encode e(n) = DC(n) – DC(n-1)



DC Cat.	Prediction Errors	Base Codeword	Codeword
0	0	010	010
1	-1, 1	011	011 x
2	-3, -2, 2, 3	100	100 x x
3	-7, -6, -5, -4, 4, 5, 6, 7	00	00 x x x
4	-15, …, -8, 8, …, 15	101	101 x x x x
5	-31, …, -16, 16, …, 31	110	110 x x x x x
6	-63, …, -32, 32, …, 63	1110	1110 x x x x x x
…	…	…	…

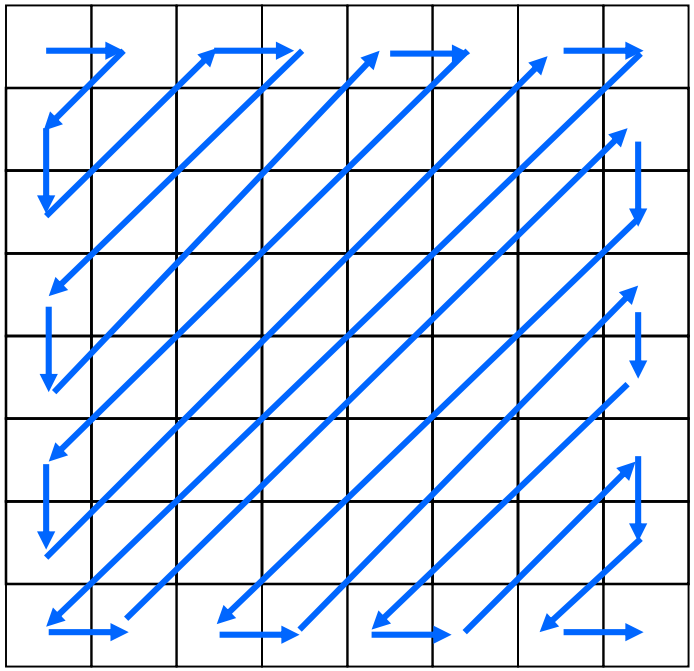
Our example:

DC: 8. Assume last DC: 5 ➔ e = 8 – 5 = 3.

Cat.: 2, index 3 ➔ Bitstream: 10011

Coding of AC Coefficients

- Most non-zero coefficients are in the upper-left corner
- Zigzag scanning:



- Example

8	24	-2	0	0	0	0	0
-31	-4	6	-1	0	0	0	0
0	-12	-1	2	0	0	0	0
0	0	-2	-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	0	0	0	0	0

- Zigzag scanning result (DC is coded separately):

  24 -31 0 -4 -2 0 6 -12 0 0 0 -1 -1 0 0 0 2 -2 0 0 0 0 0 -1 EOB

EOB: End of block symbol. The remaining coeffs are all 0.

Coding of AC Coefficients

- Many AC coefficients are zeros:
  - Huffman coding is not efficient for symbol with prob. > 1/2
- Run-level coding: Jointly encode a non-zero coefficient and the number of zeros before it (run of zeros): (run, level) event
- Disadvantage: Symbol set is enlarged: #Run x #Level
- Tradeoff:
  - Run: encode up to 15 zeros. Apply escape coding for greater values.
  - Level: Divide level into 16 categories, as in DC.
  - Apply Huffman coding to the joint Run / Category event:
    - Max symbol set size: 16 x 16 = 256.
  - Followed by fixed length code to signal the level index within each category
- Example: zigzag scanning result

  24 -31 0 -4 -2 0 6 -12 0 0 0 -1 -1 0 0 0 2 -2 0 0 0 0 0 -1 EOB
- (Run, level) representation:
- (0, 24), (0, -31), (1, -4), (0, -2), (1, 6), (0, -12), (3, -1), (0, -1), (3, 2), (0, -2), (5, -1), EOB

Coding of AC Coefficients

Run / Catg.	Base codeword	Run / Catg.	Base Codeword	…	Run / Cat.	Base codeword
EOB	1010	-	-	…	ZRL	1111 1111 001
0/1	00	1/1	1100	…	15/1	1111 1111 1111 0101
0/2	01	1/2	11011	…	15/2	1111 1111 1111 0110
0/3	100	1/3	1111001	…	15/3	1111 1111 1111 0111
0/4	1011	1/4	111110110	…	15/4	1111 1111 1111 1000
0/5	11010	1/5	11111110110	…	15/5	1111 1111 1111 1001
…	…	…	…	…	…	…

- ZRL: represent 16 zeros when number of zeros exceeds 15.
  - Example: 20 zeros followed by -1: (ZRL), (4, -1).
- (Run, Level) sequence: (0, 24), (0, -31), (1, -4), ……
- Run/Cat. Sequence: 0/5, 0/5, 1/3, …

24 is the 24-th entry in Category 5 ➔ (0, 24): 11010 11000

-4 is the 3-th entry in Category 3 ➔ (1, -4): 1111001 011

A complete Example (Sayood pp. 392)

- Original data:

124	125	122	120	122	119	117	118
121	121	120	119	119	120	120	118
126	124	123	122	121	121	120	120
124	124	125	125	126	125	124	124
127	127	128	129	130	128	127	125
143	142	143	142	140	139	139	139
150	148	152	152	152	152	150	151
156	159	158	155	158	158	157	156

2-D DCT

39.8	6.5	-2.2	1.2	-0.3	-1.0	0.7	1.1
-102.4	4.5	2.2	1.1	0.3	-0.6	-1.0	-0.4
37.7	1.3	1.7	0.2	-1.5	-2.2	-0.1	0.2
-5.6	2.2	-1.3	-0.8	1.4	0.2	-0.1	0.1
-3.3	-0.7	-1.7	0.7	-0.6	-2.6	-1.3	0.7
5.9	-0.1	-0.4	-0.7	1.9	-0.2	1.4	0.0
3.9	5.5	2.3	-0.5	-0.1	-0.8	-0.5	-0.1
-3.4	0.5	-1.0	0.8	0.9	0.0	0.3	0.0

- Quantized by basic table:

2	1	0	0	0	0	0	0
-9	0	0	0	0	0	0	0
3	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
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

- Zigzag scanning:

  2 1 -9 3 EOB

A complete Example (Sayood pp. 392)

- Zigzag scanning:
  2 1 -9 3 EOB
  - DC: 2 (assume last DC = -1)
    e = 3: Category 2, base code 100,
    index 11 ➔ 10011
  - (Run, Level) sequence of AC:
    (0, 1), (0, -9), (0, 3), EOB
  - Run/Cat. sequence:
    0/1, 0/4, 0/2, EOB
  - All bit stream (24 bits):

10011 001 10110110 0111 1010

124 125 122 120 122 119 117 118
121 121 120 119 119 120 120 118
126 124 123 122 121 121 120 120
124 124 125 125 126 125 124 124
127 127 128 129 130 128 127 125
143 142 143 142 140 139 139 139
150 148 152 152 152 152 150 151
156 159 158 155 158 158 157 156

Raw: 8x8x8 = 512 bits !

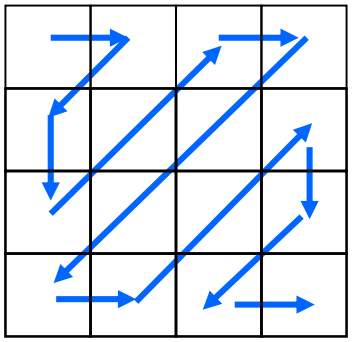
Outline

- VLC Coding in
  - JPEG
  - H.264

Context-Adaptive VLC (CAVLC)

- Entropy coding methods in H.264: CAVLC, CABAC(arithmetic)
- Differences of H.264 from JPEG:
  - Use 4-point DCT instead of 8-point DCT
  - CAVLC codes levels and runs separately:
    - VLC table size reduced.
    - Important contribution: also used in CABAC and HEVC
  - Switch VLC tables according to context information
  - DC: Several cases. Code together with AC in this lecture.
- HEVC: CAVLC not supported, Arithmetic coding instead.

0	3	-1	0
0	-1	1	0
1	0	0	0
0	0	0	0



- Example: After zigzag scan: 0 3 0 1 -1 -1 0 1 0 … 0
- Observations:
  - Trailing ones (T1): The last few non-zero coefficients are usually 1 or -1
  - Many blocks only have 1 or -1 after quantization.
  - Larger absolute values near DC and smaller values towards high freq.

    ➔Encode non-zero coefficients in reverse order facilitates adaptivity.

    Another important contribution, also adopted by CABAC and HEVC.

Main Steps

0	3	-1	0
0	-1	1	0
1	0	0	0
0	0	0	0

- After zigzag scan: 0 3 0 1 -1 -1 0 1 0 … 0
- Five Steps: CoeffToken ➔ SignTrail ➔ Levels ➔ TotalZeros ➔ Runs
  1. CoeffToken: Jointly encode the total number of non-zero coeffs (TC) and the number of trailing ones (T1s) (up to three):
     - (TC, T1s) = (5, 3)
  2. Encode the signs of trailing ones in reverse order, if any (0 for positive, 1 for negative)
     - 0, 1, 1
  3. Encode the remaining non-zero coeffs in reverse order:
     - 1, 3
  4. Encode total zeros from the beginning to the last non-zero coeff:
     - 3 : 0 3 0 1 -1 -1 0 1 0 … 0
  5. Encode the number of zeros before each non-zero coeff in reverse order, except before the first non-zero coeff:
     - 1, 0, 0, 1: 0 3 0 1 . -1 . -1 0 1 0 … 0
- Adaptive VLC table switching is used in most steps.

Step 1: CoeffToken: Number of non-zero coeffs and trailing ones

- Huffman coding for the joint event (TC, T1s)
  - TC: 0 to 16.
  - T1s: 0 to 3.
  - ➔ Table size: 17 x 4.

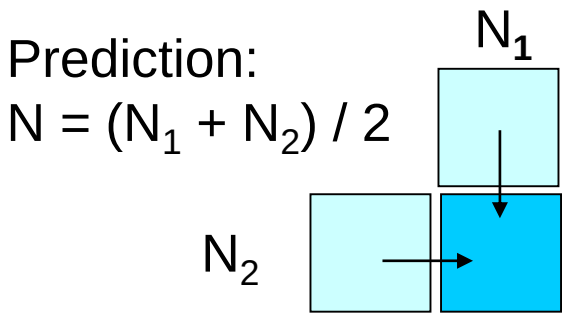
CoeffToken VLC 0

TC\T1s	0	1	2	3
0	1	-	-	-
1	000011	01	-	-
2	00000111	0001001	001	-
3	000001001	00000110	0001000	00011
4	000001000	000001011	000000101	000010
5	0000000111	000001010	000000100	0001011
...	...	...	...	...
16	0000000000000000	0000000000000001	0000000000000010	0000000000000100

- If TC = 0, no further info is coded for this block.

Adaptive coding of NumTrail

- Select from three Huffman tables and a VLC for CoeffToken:
  - Depending on the average TC of the top and left blocks: $N = (N_1 + N_2) / 2$.
  - Reason: TC of neighboring blocks have strong correlation.
  - $0 \leq N < 2$: Use Table 0 (favors small TC)
  - $2 \leq N < 4$: Use Table 1 (favors middle TC)
  - $5 \leq N < 8$: Use Table 2 (favors large TC)
  - $N \geq 8$: Use fixed-length code xxxxyy
    - xxxx: TC
    - yy: T1s.



More NumTrail Huffman Tables

CoeffToken VLC 2: for 5 ≤ N < 8

T1s TC	0	1	2	3
0	0011	-	-	-
1	0000011	0010	-	-
2	0000010	101110	1101	-
3	000011	101001	010110	1100
4	000010	101000	010001	1111
5	101101	101011	010000	1110
...	...	...	...	...
15	0000000010	00000000011	00000000010	00000000001
16	0000000000001	000000000001	0000000000001	0000000000000

Step 2, 3: Coding of Level Information

- After zigzag scan: 0 3 0 1 -1 -1 0 1 0 … 0

❑Send signs of trailing ones first (up to three T1s), 0 for positive, 1 for negative:

- 0, 1, 1

❑Encode the remaining levels in reverse order:

- Observations: Absolute values of non-zero coeffs are generally lower at high frequencies, and become higher near DC.
- If coded forwardly, it's difficult to predict the first few big coeffs, and to decide the best VLC table.
- Reverse coding:
  - Recently-coded level is used to decide which Huffman table to use for the next level:
    - Start from VLC 0.
    - Switch to the next VLC if previous level is greater than a threshold.
    - Use Golomb-Rice(N) code with larger N after each switching:
      - Larger levels are coded more efficiently.

Coding of Level Information

❑Level VLC 0

(Unary code)

Level	Code
1	1
-1	01
2	001
-2	0001
3	00001
-3	000001
..	..
-7	00000000000001
±8 to ±15	00000000000001xxxx
±16 ->	00000000000001xxxxxx xxxxxx

Level VLC 1

(Golomb-Rice(2) Code)

Level	Code
1	10
-1	11
2	010
-2	011
3	0010
-3	0011
..	..
14	00000000000010
-14	00000000000011
±15 to ±22	00000000000001x xxx
±23 ->	00000000000001x xxxxxxxxxx

Coding of Level Information

- Level VLC 2 (Golomb-Rice(4) Code)

Level	Code
1	100
-1	101
2	110
-2	111
3	0100
-3	0101
4	0110
-4	0111
5	00100
..	..
±37 ->	000000000000001xxxxxxxxxxxx

Coding of Level Information

- A special case:
  - If number of trailing ones is less than 3, the next non-zero coeff must be > 1 or < -1.
  - Shift the next level to reduce the codeword length:
    - If the next level > 0: it is coded as Level – 1.
    - If the next level < 0: it is coded as Level + 1.
- Example: -2 4 3 -3 0 0 -1 0 … 0
  - T1 = 1.
  - -3 will be coded as -2
    - -3 in VLC_0: 000001
    - -2 in VLC_0: 0001
    - 2 bits are saved

Step 4: Coding of Total Zeros

- After zigzag scan: 0 3 0 1 -1 -1 0 1 0 … 0

❑Maximal total zeros is 16 – TC:

0 0 … 0 X X X

Zeros non zeros

❑Two trivial cases:

- If TC = 16, TotalZeros must be 0.
- If TC = 0, no further info is needed.

❑15 non-trivial cases: for TC = 1 to 15.

❑Each has a Huffman table for TotalZeros:

- Huffman table size: (16 – TC) + 1.

Coding of Total Zeros

Different Huffman tables

TC *TotalZeros*	1	2	3	4	5	6	7
0	1	111	0010	111101	01000	101100	111000
1	011	101	1101	1110	01010	101101	111001
2	010	011	000	0110	01011	1010	11101
3	0011	001	010	1010	1110	001	1001
4	0010	000	1011	000	011	010	1111
5	00011	1000	1111	100	100	000	00
6	00010	0101	011	110	1111	110	01
7	000011	1001	100	1011	110	111	101
8	000010	1100	0011	010	101	100	110
9	0000011	01000	1110	001	001	011	100
10	0000010	11011	1010	0111	000	10111	–
11	00000001	11010	11000	1111	01001	–	–
12	00000000	010010	110011	111100	–	–	–
13	00000011	0100111	110010	–	–	–	–
14	000000101	0100110	–	–	–	–	–
15	000000100	–	–	–	–	–	–

Coding of Total Zeros

TC *TotalZeros*	8	9	10	11	12	13	14	15
0	101000	111000	10000	11000	1000	100	00	0
1	101001	111001	10001	11001	1001	101	01	1
2	10101	11101	1001	1101	101	11	1	-
3	1011	1111	101	111	0	0	-	-
4	110	00	01	0	11	-	-	-
5	00	01	11	10		-	-	-
6	111	10	00	-	-	-	-	-
7	01	110	-	-	-	-	-	-
8	100	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-

Step 5: Coding of Runs

- What we have known so far:
  - TC – total non zero coefficients
  - All levels: e.g, {3 -2 }
  - Total Zeros
- To reconstruct the sequence, only need to know the positions of all zeros before the last non-zero coefficient.
- A typical example: 3 -2 1 0 1 0 0 0 -1 0 … 0
- JPEG (run, level) approach (excluding DC):

  (0, -2), (0, 1), (1, 1), (3, -1)

  Many runs are 0 at the beginning of the sequence.
- Observation:
  - Most of the run of zeros are near the end of the sequence.
- CAVLC: Send runs before each non-zero coeff in reverse order.

Coding of Runs

- Sequence: 3 -2 1 0 1 0 0 0 -1 0 … 0
  - TotalZeros: 4
  - ZerosLeft (ZL): Number of zeros that have not been coded.
  - Step 1: ZerosLeft = TotalZeros = 4, RunsBefore = 3.

    Possible zeros before the non-zero coefficient: 0, 1, 2, 3, 4

    ➔ Need a Huffman table with 5 symbols, for example:

    ➔ 01, 00, 11, 101, 100.

    ➔ Code 101 (3 | 4).

    ➔ Update ZerosLeft: ZerosLeft = 4 – 3 = 1.
  - Step 2: ZerosLeft = 1, RunsBefore = 1.

    Possible zeros before the non-zero coefficient: 0, 1.

    ➔ Need a Huffman table with 2 symbols: 1, 0

    ➔ Code 0 (1 | 1).

    ➔ Update ZerosLeft: ZerosLeft = 1 – 1 = 0.

    ➔ Stop if ZerosLeft = 0 (The positions of all zeros have been coded).

Coding of Runs

- Special case:
  - Runs before the first non-zero coeff does not need to be coded:
- Example: 0 0 2 0 0 0 -1 0 … 0
  - TotalZeros = 5, TC = 2
  - Step 1: ZerosLeft = TotalZeros = 5, RunsBefore = 3
    - Coding (3 | 5)
    - ZerosLeft = 5 – 3 = 2.
    - But there is only one non-zero coeff left:
      - ➔ There must be 2 zeros before the next non-zero coeff.

VLC Table for Coding of Runs

Zeros Left *Run Before*	1	2	3	4	5	6	>6
0	1	1	01	01	01	01	000
1	0	01	00	00	00	00	010
2	–	00	11	11	11	101	101
3	–	–	10	101	101	100	100
4	–	–	–	100	1001	111	111
5	–	–	–	–	1000	1101	110
6	–	–	–	–	–	1100	0011
7	–	–	–	–	–	–	0010
8		–	–	–	–	–	00011
9	–	–	–	–	–	–	00010
10	–	–	–	–	–	–	00001
11	–	–	–	–	–	–	0000011
12	–	–	–	–	–	–	0000010
13	–	–	–	–	–	–	0000001
14	–	–	–	–	–	–	00000001

Summary

- VLC is the real world image coding solution
- Elements of Hoffman and Golomb coding schemes are incorporated
- JPEG: introduced DC prediction , AC zigzag scan, run-level VLC
- H264: introduced reverse order coding.