JPEG Baseline: DPCM Coding

DPCM of DC Coefficient= difference of two DC coefficients of two consecutive blocks.

Example: If DC coefficients of consecutive blocks are give by: 13, 13, 10, 11, 11, 10

DPCM values become **13, 0, -3, +1, 0, -1, ...**

DPCM Code is expressed by (**SSSS**, **value**):

SSSS denotes the number of bits needed to encode the value

Value is the actual bits that represent the value

DPCM Codes in JPEG Baseline

If the **DPCM Values** of consecutive DCT blocks are given by

13, 0, -3, +1, 0, -1, ...

the corresponding DPCM codes are encoded as

	Value	***SSS***	***Value***	
	13	4	1100	
	0	0		
negative	-3	2	00	Complement of value
	1	1	1	
	0	0		
negative	-1	1	0	Complement of value

VLC for Luminance DC Differences

Category	Code length	Code word
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

VLC Coded of DPCM Codes

Value	*SSS*	*Huffman Code*	*Value*	*Encoded Bits*
13	4	101	1100	1011100
0	0	00		00
-3	2	011	00	01100
1	1	010	1	0101
0	0	00		00
-1	1	010	0	0100

VLC for Chrominance DC Differences

Category	Code length	Code word
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

RLC Values of AC coefficients

Since the AC coefficients contain long strings of zeros,thus AC is encoded in the form **(skip, value):**

- **Skip**: the number of zeros proceeding the **Value**
- **Value**: the next non-zero coefficient, the value field is encoded as: **SSS/value**.

Example: 63 AC coefficients are given by

6, 7, 0, 0, 0, 3, -1, 0, 0, 0,…,0.

AC Coefficients in RLC Values

- (0, 6), (0, 7), (3, 3), (0, -1), (0, 0)
- The last (0,0) indicates the end of the string for this block. (The rest AC coefficients are all zeros)

RLC Codes for AC coefficients

AC Coefficients in RLC Values are given by:

(0, 6), (0, 7), (3, 3), (0, -1), (0, 0)

RLC Codes (in Binary format): (Skip, SSS, Value)

AC Coefficients	Skip	SSS	Value
0, 6	0	3	110
0, 7	0	3	111
3, 3	3	2	11
0, -1	0	1	0
0, 0	0	0	

VLC for AC Differences

Run/Size	Code length	Code word
0/0 (EOB)	14	1010
0/1	12	00
0/2	12	01
0/3	13	100
0/4	14	1011
0/5	15	11010
0/6	17	1111000
0/7	18	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011
1/1	14	1100
1/2	15	11011
1/3	17	1111001
1/4	19	111110110
1/5	11	11111110110
1/6	16	1111111110000100
1/7	16	1111111110000101
1/8	16	1111111110000110
1/9	16	1111111110000111
1/A	16	1111111110001000
2/1	15	11100
2/2	18	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	1111111110001001
2/6	16	1111111110001010
2/7	16	1111111110001011
2/8	16	1111111110001100
2/9	16	1111111110001101
2/A	16	1111111110001110
3/1	16	111010
3/2	19	111110111
3/3	12	111111110101
3/4	16	1111111110001111
3/5	16	1111111110010000
3/6	16	1111111110010001
3/7	16	1111111110010010
3/8	16	1111111110010011
3/9	16	1111111110010100
3/A	16	1111111110010101

VLC Coded of RLC Codes

Skip and **SSS** are considered as one symbol and encoded by a **Huffman codeword:**

RLC Value	Composite Symbol (Skip ,	SSS)	Huffman Codeword	Non-zero Value
0, 6	0	3	100	110
0, 7	0	3	100	111
3, 3	3	2	111110111	11
0,-1	0	1	00	-0
0, 0	0	0	1010	

The Huffman-encoded bitstream is then derived by adding the run-length encoded value to each of Huffman codewords:

100110 100111 11111011111 000 1010

JPEG BitStream Format

Level 1: frame header and frame contents (level 2)

Frame Header
- The overall width and height of the image in pixel
- The number and type of components to represent the image: (R/G/B, Y/Cr/Cb, ...)
- The digitizing format used (4:2:2; 4:2:0 etc.)

Level 2: scan header, and scan components (level 3)

Scan header
- Identity of the components(RGB etc.)
- # of bit used in each digitize component
- Quantization table and values have been used

A **scan/component** comprises one or more segments

Level 3: Segment header, and segments

Segment header contains
- Huffman table and values used in encoding

Segments contains a groups of 8x8 blocks of images