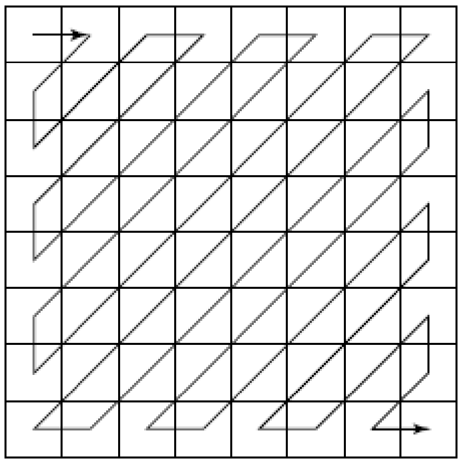
4. Zig-zag ordering and run-length encoding

The 64 quantized frequency coefficients (which are now integers) of each data unit are encoded using a combination of RLE and Huffman coding. To make it most likely to hit a long run of zeros: a *zig-zag scan* is used to turn the 8×8 matrix $\hat{F}(u, v)$ into a *64-vector*.



$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

$$\begin{array}{rrrrrrr} \mathbf{-26} \\ \mathbf{-3} & \mathbf{0} \\ \mathbf{-3} & \mathbf{-2} & \mathbf{-6} \\ \mathbf{2} & \mathbf{-4} & \mathbf{1} & \mathbf{-3} \\ \mathbf{1} & \mathbf{1} & \mathbf{5} & \mathbf{1} & \mathbf{2} \\ \mathbf{-1} & \mathbf{1} & \mathbf{-1} & \mathbf{2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{-1} & \mathbf{0} & \mathbf{0} \end{array}$$

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							

RLC aims to turn the $\hat{F}(u,v)$ values into sets *{#-zeros-to skip , next non-zero value}*.

From the above example

(32,6,-1,-1,0,-1,0,0,0,-1,0,0,1,0,0,……….,0)

First do not treat the DC component and the rest (AC component) will be

(0, 6)(0,-1)(0,-1)(1,-1)(3,-1)(2, 1)(0, 0)

A special pair (0,0) indicates the end of blocks after the last nonzero AC coefficient is reached.

1. **DPCM on DC coefficients**

DPCM *Differential Pulse Code Modulation* is a member differential encoding family. Differential Encoding methods calculates the difference between two consecutive data items, and encode the difference. It is depends on the concept that correlated values are generally similar, so their differences are small.

DC values reflects the average intensity of each block, but these coefficient in unlikely to change hardly within a short distance. This makes DPCM *Differential Pulse Code Modulation* an ideal scheme for coding the DC coefficients.
If the DC coefficients for the first 5 image blocks are

150, 155, 149, 152, 144

Then the DPCM would produce

150, 5, -6, 3, -8,

Assuming $d_i = DC_{i+1} - DC_i \ \ and \ \ d_0 = DC_0$

2. Huffman Coding of DC coefficients

Use DC as an example: each DPCM coded DC coefficient is represented by (SIZE, AMPLITUDE), where SIZE indicates how many bits are needed for representing the coefficient, and AMPLITUDE contains the actual bits. For the negative values, one's complement scheme is used. SIZE is Huffman coded since smaller SIZEs occur much more often. AMPLITUDE is not Huffman coded; its value can change widely so Huffman coding has no appreciable benefit.

In the example we're using, codes

150, 5, −6, 3, −8

Will be turned into

(8, 10010110), (3, 101), (3, 001), (2, 11), (4, 0111)

3. Huffman coding for AC coefficients

AC coefficients are run-length encoded (RLC). RLE pairs (Runlength, Value) are Huffman coded as with DC only on Value. So we get a triple: (Runlength, Size, Amplitude). However, Runlength, Size allocated 4-bits each (both will be one byte) and put into a single byte with is then Huffman coded. Again, Amplitude is not coded. So only Runlength an size are Huffman coded.

JPEG Image Decompression System

The JPEG decompression system is inverse of the JPEG compression system.

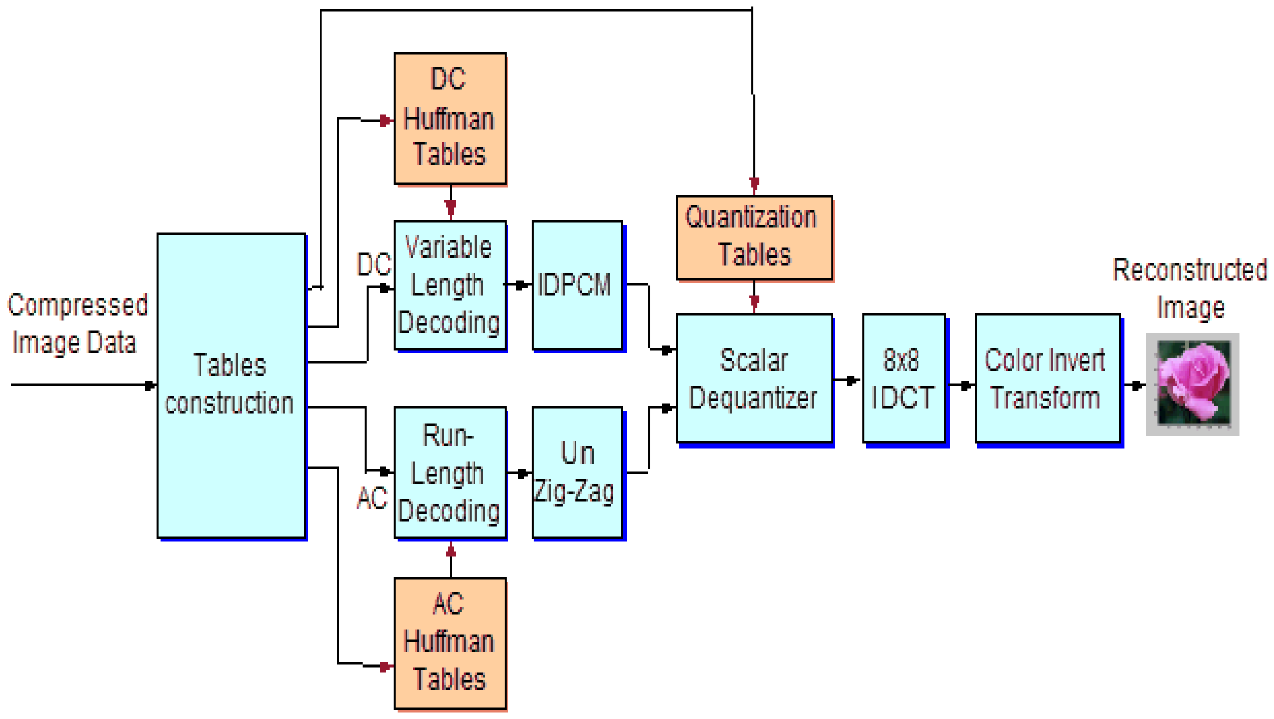


Figure 10. The JPEG decompression structure

As soon the code streams entered the decompression system, the all the received quantization and Huffman tables are reconstructed. The frame headers are also decoded to determine the size and precision of the image. The compressed stream for each 8×8 block is split into two parts. The DC code is decoded using the DC Huffman tables. The value output from DC decoder is, indeed, the difference between the DC value of the current and the previous 8×8 blocks. The IDPCM restores back the true DC value by adding the value obtained from the DC decoder with the DC value decoded from the previous block. The AC part is decoded using the AC Huffman tables to get the AC coefficients, which are organized in zig-zag order. Therefore, the unzigzag stage reorganizes the coefficients into 8×8 block. The dequantization stage performs the multiplications between the coefficients with the The IDCT performs the invert discrete cosine transform for each 8×8 block. Since the quantization generates quantization errors, the reconstructed block data is no longer identical to that of original image.

The data obtained at the IDCT output form the chrominance and luminance images, adding with the level offset and finally are converted into the RGB image before displaying on the screen.

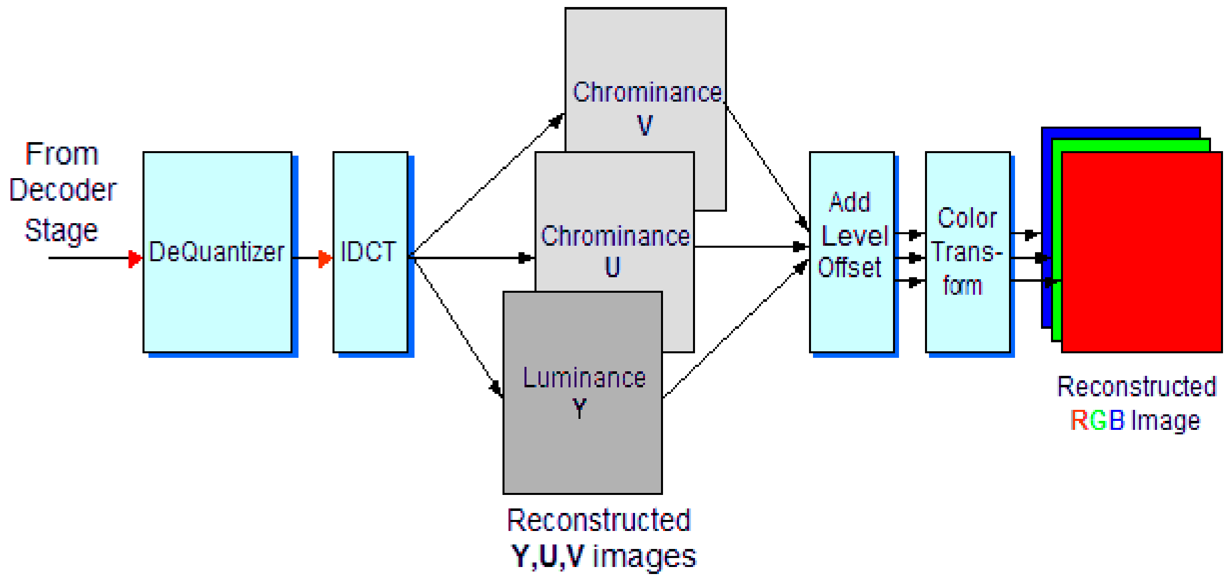


Figure 3. Color invert transformation at the decoder

The color invert transformation scheme is illustated in Figure 3.