JPEG ALGORITHM

Hema Varshita

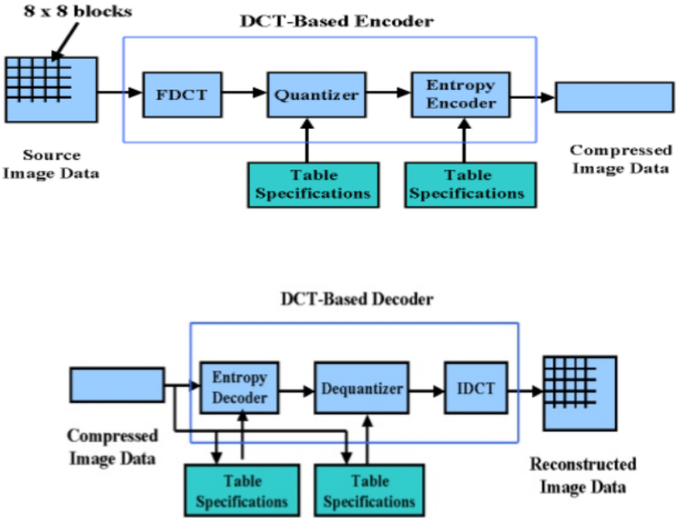
IIT Hyderabad

June 14, 2021

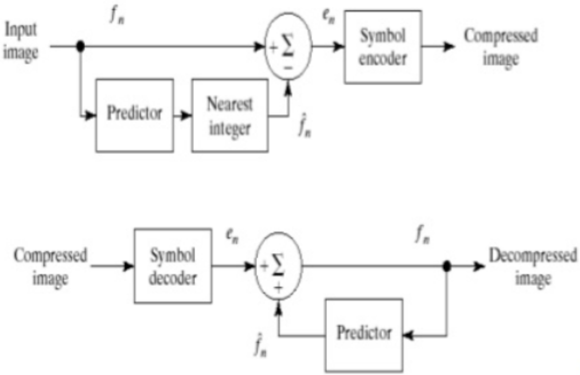
INTRODUCTION

- JPEG - Joint Photographic Experts Group
- The JPEG includes two basic compression methods for image compression, each with various modes of operation.
- DCT-based method - for "lossy" compression
- Predictive method - for "lossless" compression

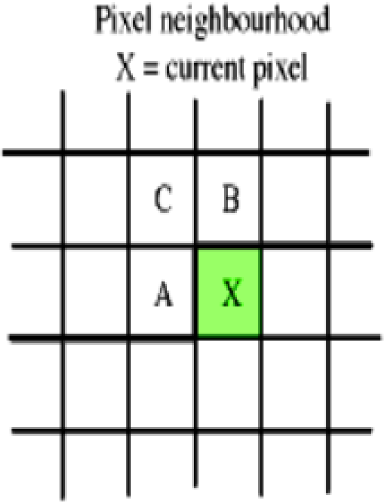
Block diagram Of Encoder and Decoder for DCT method



PREDICTIVE METHOD



PREDICTIVE METHOD



PREDICTIVE METHOD

Encoder type	Prediction method
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

MODES OF OPERATION

- For each mode, one or more distinct codecs(encoder/decoder) are specified depending on the precision or entropy coding technique used.
- Sequential encoding: each image component is encoded in a single left-to-right, top-to-bottom scan.
- Progressive encoding: the image is encoded in multiple scans where the image builds up in multiple coarse-to-clear passes.
- Lossless encoding: Encoding of the image is done for the exact recovery with low compression compared to lossy techniques.
- Hierarchical encoding: Image is encoded at multiple resolutions.

DCT based Baseline sequential codec - Steps Involved

Preprocessing

- At the input to encoder each image is grouped into samples of 8x8 blocks.
- The intensity values corresponding to each pixel of the sub block lie in the range $[0\ ,\ 2^P - 1]$. We shift these values to $[-2^P - 1\ ,\ 2^P - 1]$, since cosine values lie between 1 and -1.
- P can be 8-12 for DCT-based codecs, and 2-16 for Predictive codecs.
- P for Baseline Sequence codec is 8.

FDCT

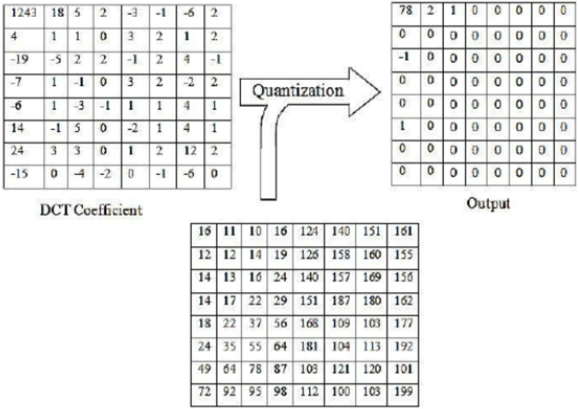
- Each 8x8 block of image is a discrete 64 point signal which is a function of both x and y axis
- The output of FDCT is a set of 64 orthogonal basis-signal amplitudes.
- The first element with zero frequency is called DC coefficient and the remaining 63 are called AC coefficients.
- The resultant matrix has higher energy values for lower frequencies and lower energy for higher frequencies since our source images are usually smooth, without any abrupt changes.

$$DCT(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{x=0}^{N-1} pixel(x,y) COS[\frac{(2x+1)i\pi}{2N}] COS[\frac{(2y+1)i\pi}{2N}] \quad (1)$$

$c(x) = \frac{1}{2}$ if x=0, else 1 if $x > 0$

Quantizer and Normalisation

- It's the principal source of lossines to the algorithm.
- Each DCT coefficient is divided by its corresponding quantisation value and rounded to its nearest integer given by the user.
- The aim is to compress the image as much as possible without seeing noticeable difference.



1243	18	5	2	-3	-1	-6	2
4	1	1	0	3	2	1	2
-19	-5	2	2	-1	2	4	-1
-7	1	-1	0	3	2	-2	2
-6	1	-3	-1	1	1	4	1
14	-1	5	0	-2	1	4	1
24	3	3	0	1	2	12	2
-15	0	-4	-2	0	-1	-6	0

DCT Coefficient

78	2	1	0	0	0	0	0
0	0	0	0	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
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

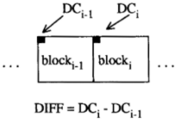
Output

16	11	10	16	124	140	151	161
12	12	14	19	126	158	160	155
14	13	16	24	140	157	169	156
14	17	22	29	151	187	180	162
18	22	37	56	168	109	103	177
24	35	55	64	181	104	113	192
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	199

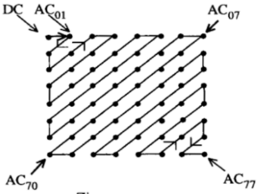
Quantization table

DC coding and Zig-Zag Sequence

- The first element of the quantized DCT matrix (DC coefficient) is a measure of average value of all the 64 elements and hence is encoded separately.
- The DC term is encoded as the difference from previous DC term.
- All the AC coefficients are stored in the zig-zag fashion to accommodate all the low frequency components first.



Differential DC encoding



Zig–zag sequence

Intermediate Entropy Coding Representation

The DC and AC coefficients are represented by two symbols (Symbol1,Symbol2)

- DC coefficient:
  - Symbol1: Represents size of bits used to represent the quantized differential DC amplitude. It can be at the max 11 bits.
  - Symbol2: Represents amplitude and can lie in the range $[\,-2^{11}, 2^{11}-1\,]$.
- AC coefficient:
  - Symbol1 - (RUN-LENGTH, SIZE):
    - Run-Length: Represents the length of zeros encountered lying between 0-15.
    - Size: Upto 10 bits are used to encode.
  - symbol2: Represents amplitude of non-zero quantised AC coefficients and can lie in the range $[\,-2^{10}, 2^{10}-1\,]$.

	SIZE					
	0	1	2	. . .	9	10
RUN LENGTH 0	EOB					
.	X		RUN-SIZE values			
.	X					
.	X					
15	ZRL					

Table 2. Baseline Huffman Coding Symbol-1 Structure

SIZE	AMPLITUDE
1	-1,1
2	-3,-2,2,3
3	-7..-4,4..7
4	-15..-8,8..15
5	-31..-16,16..31
6	-63..-32,32..63
7	-127..-64,64..127
8	-255..-128,128..255
9	-511..-256,256..511
10	-1023..-512,512..1023

Table 3. Baseline Entropy Coding Symbol-2 Structure

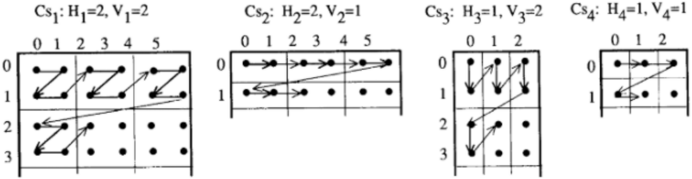
Entropy Encoder

- Entropy encoding in combination with the transformation and quantization results in significantly reduced data size
- It is a lossless process wherein data elements are replaced with coded representations
- Baseline codec uses a prefix free coding technique - “Huffman Encoding”
- It uses a model to accurately determine the probabilities for each quantized value and produces an appropriate code based on these probabilities so that the resultant output code stream will be smaller than the input stream.

Quantized DCT Matrix $\rightarrow$ Entropy Encoder $\rightarrow$ Compressed “data”

VLC and VLI Entropy Coding

- For both DC and AC coefficients:
  - Symbol1 is encoded with a variable-length code (VLC) from Huffman table.
  - Symbol2 is encoded with variable-length integer (VLI) code determined by SIZE and not Huffman. Positive values are represented as unsigned magnitude binary numbers.Negative values represented as the 1's complement of the corresponding positive values.
- For encoding of symbol 1 Huffman tables are provided to encoders externally.
- For encoding symbol 2 VLI codes are hardwired into standard.



$$
\begin{aligned}
MCU_1 &= d^1_{00}\, d^1_{01}\, d^1_{10}\, d^1_{11} \quad d^2_{00}\, d^2_{01} \quad d^3_{00}\, d^3_{10} \quad d^4_{00}, \\
MCU_2 &= d^1_{02}\, d^1_{03}\, d^1_{12}\, d^1_{13} \quad d^2_{02}\, d^2_{03} \quad d^3_{01}\, d^3_{11} \quad d^4_{01}, \\
MCU_3 &= d^1_{04}\, d^1_{05}\, d^1_{14}\, d^1_{15} \quad d^2_{04}\, d^2_{05} \quad d^3_{02}\, d^3_{12} \quad d^4_{02}, \\
MCU_4 &= \underbrace{d^1_{20}\, d^1_{21}\, d^1_{30}\, d^1_{31}}_{Cs_1 \text{ data units}} \quad \underbrace{d^2_{10}\, d^2_{11}}_{Cs_2} \quad \underbrace{d^3_{20}\, d^3_{30}}_{Cs_3} \quad \underbrace{d^4_{10}}_{Cs_4},
\end{aligned}
$$

Figure 8. Generalized Interleaved Data Ordering Example

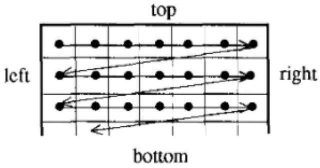
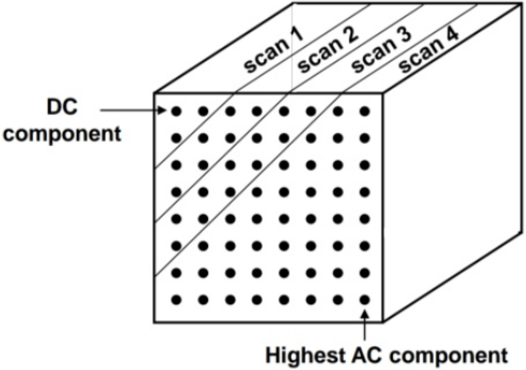


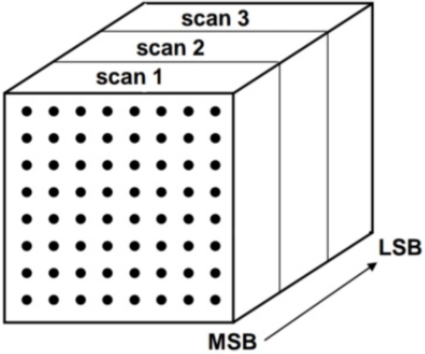
Figure 7. Noninterleaved Data Ordering

Progressive mode of operation

- Multiple scans are used to compress the image having long transmission time.
- In the initial scan the image is roughly encoded but still being recognisable.
- Further scans are performed until desired picture is obtained as specified by the quantisation table.
- The following are two progressive methods:
    - Spectral selection
    - Successive approximation



Spectral selection



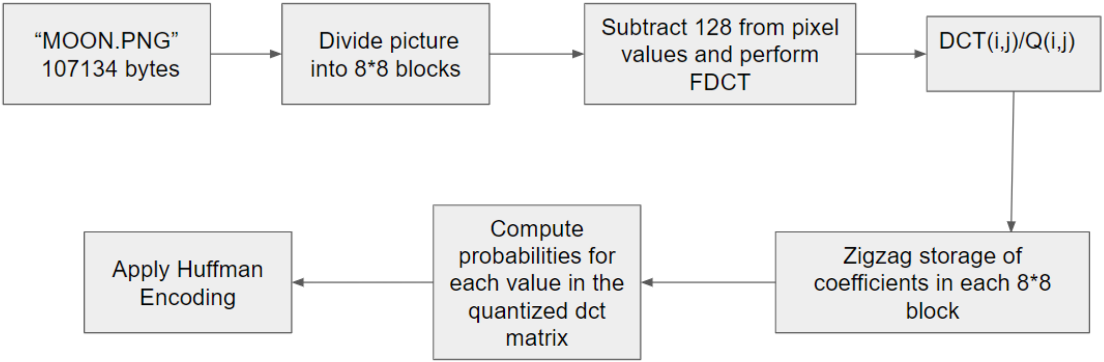
Successive approximation

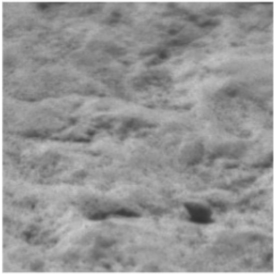
Hierarchical mode of operation

The image is pyramidal encoded such that the difference in resolution of adjacent encoding is two in one or both the dimensions.

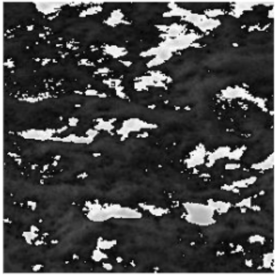
- Image is filtered and downsampled by a multiple of 2.
- Encode this image
- Now decode this image,pass it through same filter used at receiver and upsample by 2.
- Use the up-sampled image as a prediction of the original, and encode the difference image.
- This is repeated until full image resolution is encoded.

IMPLEMENTATION

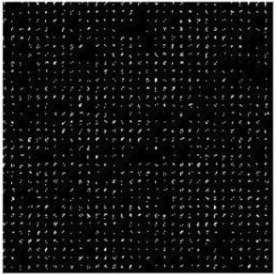




Original Image



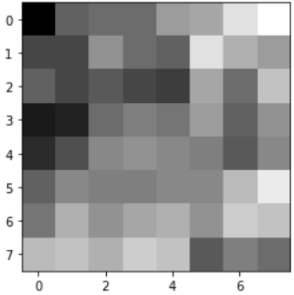
After subtracting
128 from each pixel.



Quantized DCT matrix

```
[[140. 150. 151. 151. 156. 157. 163. 166.]
 [147. 147. 155. 151. 150. 163. 158. 156.]
 [150. 147. 149. 147. 146. 157. 151. 160.]
 [142. 143. 151. 153. 152. 156. 150. 155.]
 [144. 148. 154. 155. 154. 153. 149. 154.]
 [150. 154. 153. 153. 154. 154. 159. 164.]
 [152. 158. 155. 157. 158. 155. 161. 160.]
 [159. 160. 158. 161. 160. 149. 153. 151.]]
```

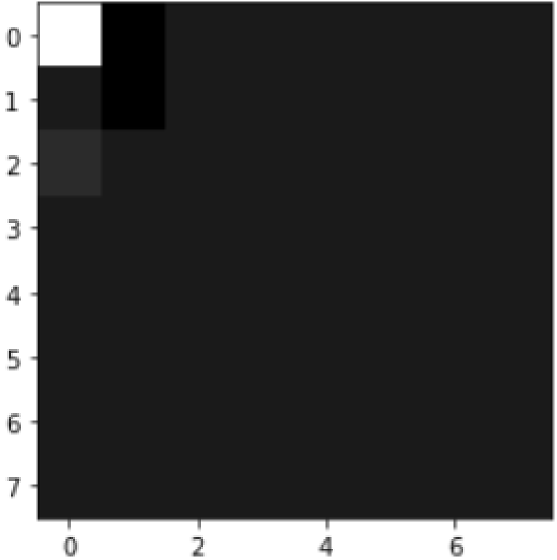
8x8 block of the original image



Quantized DCT matrix of 8X8 BLOCK

```
[[12. -1. -0. -0. -0. -0.  0. -0.]
 [-0. -1.  0.  0. -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.]
 [-0.  0.  0. -0.  0. -0. -0. -0.]
 [-0.  0.  0. -0.  0. -0. -0. -0.]]
```

Most of the frequency is captured in the DC coefficient of the matrix.



DICT OF ELEMENTS OF QUANTIZED DCT MATRIX AND IT'S PROBABILITIES

['12.0': 0.0008544921875, '0.0': 0.9849700927734375, '14.0': 0.000762939453125, '15.0': 0.000762939453125, '6.0': 0.0007171630859375, '-16.0': 3.0517578125e-05, '-14.0': 4.57763671875e-05, '-7.0': 0.0001068115234375, '1.0': 0.000396728515625, '-1.0': 0.00018310546875, '11.0': 0.0006866455078125, '7.0': 0.000762939453125, '8.0': 0.0008544921875, '10.0': 0.0006561279296875, '20.0': 0.0005035400390625, '3.0': 0.000518798828125, '-2.0': 0.0002288818359375, '-8.0': 0.0001220703125, '-4.0': 0.00018310546875, '5.0': 0.0006103515625, '-9.0': 4.57763671875e-05, '4.0': 0.00054931640625, '13.0': 0.00079345703125, '-6.0': 0.0001220703125, '9.0': 0.0006561279296875, '-3.0': 0.0001068115234375, '-11.0': 6.103515625e-05, '-13.0': 6.103515625e-05, '17.0': 0.00048828125, '2.0': 0.0003662109375, '16.0': 0.000518798828125, '18.0': 0.0004119873046875, '-10.0': 3.0517578125e-05, '-5.0': 0.0001220703125, '24.0': 0.0001220703125, '26.0': 9.1552734375e-05, '27.0': 7.62939453125e-05, '23.0': 0.0001983642578125, '22.0': 0.0002288818359375, '21.0': 0.0003814697265625, '25.0': 0.0001373291015625, '28.0': 3.0517578125e-05, '19.0': 0.0002899169921875, '-12.0': 1.52587890625e-05, '-29.0': 1.52587890625e-05, '-27.0': 1.52587890625e-05, '30.0': 1.52587890625e-05, '-30.0': 1.52587890625e-05, '-38.0': 1.52587890625e-05, '-19.0': 1.52587890625e-05, '29.0': 3.0517578125e-05, '31.0': 1.52587890625e-05]

CORRESPONDING HUFFMAN CODES

0.0 - 0	8.0 - 11000	12.0 - 10111
13.0 - 10101	15.0 - 10010	7.0 - 10011
14.0 - 10001	11.0 - 111110	6.0 - 111111
10.0 - 111100	9.0 - 111101	5.0 - 111011
16.0 - 111000	4.0 - 111001	20.0 - 110110
3.0 - 110111	0.0 - 0	8.0 - 11000
12.0 - 10111	13.0- 10101	15.0- 10010
7.0- 10011	14.0- 10001	11.0- 111110
6.0 -111111	10.0- 111100	9.0- 111101
5.0 -111011	16.0- 111000	4.0- 111001
20.0- 110110	3.0 -110111	17.0- 110100
18.0- 101101	21.0- 101000	1.0 -101001
2.0 -100000	19.0- 1110100	-2.0- 1100100
22.0- 1100101	23.0- 1011000	-1.0- 1000010
-4.0- 1000011	-12.0- 11010111000	25.0- 11101010
24.0- 11010100	-6.0- 11001110	-5.0- 11001111
-3.0- 11001100	-8.0- 11001101	-7.0- 10110011

CORRESPONDING HUFFMAN CODES

27.0- 111010110	26.0- 111010111	-11.0- 110101010
-13.0- 110101011	-14.0- 101100100	-9.0- 101100101
28.0- 1101011010	29.0- 1101011011	-16.0- 1101011000
-10.0- 1101011001	-19.0- 11010111110	31.0- 11010111111
-30.0- 11010111100	-38.0- 11010111101	-27.0- 11010111010
30.0- 11010111011	-29.0- 11010111001	

Results

In [32]:
```
s1 = os.path.getsize("dct.txt")

s2 = os.path.getsize("compress.dat")
s3 = os.path.getsize("moon.png")
ratio = s1/s2
```

In [33]:
```
print("Original Image:",s3)
print("DCT Matrix :",s1)
print("Compressed form:",s2)
print("Compression Ratio achieved :",ratio)
```

```
Original Image: 107134
DCT Matrix : 265393
Compressed form: 11919
Compression Ratio achieved : 22.266381407836228
```