IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 3, October 2015)

"JPEG-A Compression Tool" Implemented In MATLAB

Julie Antony K
MTech Student
College Of Engineering, Kidangoor
Kidangoor south PO
Kerala, India

Sanoj R
Assistance Professor, Dept of IT
College Of Engineering, Kidangoor
Kidangoor South PO
Kerala, India

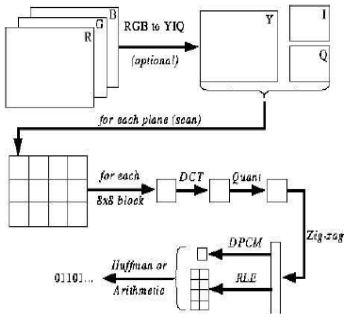
***Abstract*— The goal of image compression is to reduce the amount of data required to represent a image and doing this it reduce time for sending an image. *The* best tool for the image compression is"JPEG". It yields an excellent quality image with high compression rates. Its Stands for Joint Photographic Experts Group. It's used with .jpg,. .tiff and .eps file format. It can be used in 24-bit color images. It can be used in satellite, photography, medical forensics and many more applications.**

***Keywords*— JPEG, DCT, QUANTIZATION.**

I. INTRODUCTION

Steps In Jpeg Compression. It consists of mainly 6 steps. After this process the image will be compresses considerably. By doing this the time for sending such file get reduced**.**

- Encoding



1. If the color is represented in RGB mode, translate it to YUV.
2. Divide the file into 8 X 8 blocks.
3. To obtain frequency domain values apply Discrete Cosine Transform.
4. Quantize the resulting values.
5. Arrange the resulting coefficients in a zigzag order.
6. Do a run-length encoding of the coefficients follow by Huffman coding

Step1. If The Color Is Represented In RGB Mode, Translate It To Yuv Mode**.**

This is an optional step but by doing this it gives a better compression rate. Color image is formed by stacking the three planes ie red, green and blue plane like a sandwich model. (FIG1).YUV color mode stores color in terms of its luminance and chrominance .Where Y stands for brightness and U,V stand for color intensity .YUV is not an actual way to represent a color intensity it is just a way for incorporating the RGB information

FIG1: A)ORIGINAL IMAGE B)RED PLANE C)GREEN PLANE D)BLUE PLANE

STEP2: Divide The File Into 8 X 8 Blocks.
By diving the file into 8*8 we may reach every part of the image. and for the further process we may take each of this block one by one.

```
. for i=0:15;
```

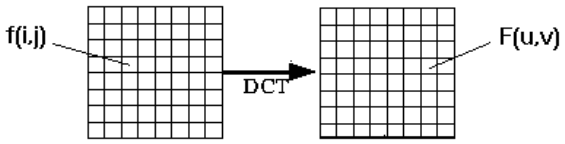
IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 3, October 2015)

```
    for j=0:15;
       qq=m( i*8+[1:8],j*8+[1:8]);
    end
end
```

Step3: To Obtain Frequency Domain Values Apply Discrete Cosine Transform



Discrete Cosine Transform (DCT):

$$F(u,v) = \frac{\Lambda(u)\Lambda(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{(2i+1)\cdot u\pi}{16}\cdot\cos\frac{(2j+1)\cdot v\pi}{16}\cdot f(i,j)$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$

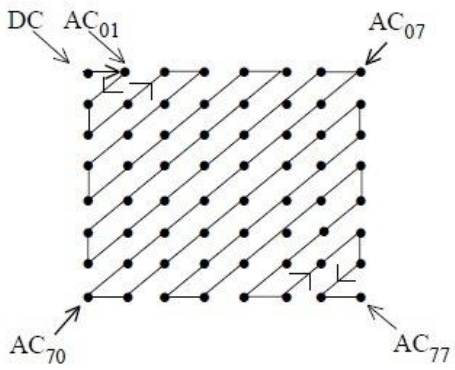
The DCT transforms the data from the spatial domain to the frequency domain that is color intensity or its amplitude value can be seen as a wave form its is now transforming into frequencies components. By representing in wave form or showing the value in xy plane we may not be able to accurately specify its value. So that by representing in the frequency domain we can identity variation of the color from one pixel to other pixel in a given image. The frequency domain is a better representation for the data.

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

INPUT DCT MATRIX

186	-18	15	-9	23	-9	-14	19
21	-34	26	-9	-11	11	14	7
-10	-24	-2	6	-18	3	-20	-1
-8	-5	14	-15	-8	-3	-3	8
-3	10	8	1	-11	18	18	15
4	-2	-18	8	8	-4	1	-7
9	1	-3	4	-1	-7	-1	-2
0	-8	-2	2	1	4	-6	0

OUTPUT DCT MATRIX



ZIG ZAG ORDER

By comparing the both figure ie output matrix and zig -zag order figure we can conclude that leftmost top corner element show the DC component and rest represent AC component. DC have the average value of all AC components .AC represent the intensity of the each pixel value. Comparing the value as we move away from the DC element the magnitude of pixel value is going down. So it shows that representation of the image is concentrated on the upper left coefficients of the output matrix, with the lower right coefficients of the DCT matrix containing less useful information.

```
for i=0:15;
   for j=0:15;
      qq=m( i*8+[1:8],j*8+[1:8]);
      y = DCT_8X8(qq);
     transformed_image ( i*8+[1:8],j*8+[1:8])=y;
   end
end
N = size(d,1);
```

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 3, October 2015)

DCT_8*8

```
n = 0:N-1;
for k=0:N-1
 if (k>0)
  C(k+1,n+1)=cos(pi*(2*n+1)*k/2/N)/sqrt(N)*sqrt(2);
 Else
  C(k+1,n+1)=cos(pi*(2*n+1)*k/2/N)/sqrt(N);
end
end
out = C*d*(C');
end
```

4. Quantize The Resulting Values

In this it mainly discards the data that is less useful in the image .By doing this the image get compressed. In quantization process we divide each element of DCT value output to fixed number. Since the DCT value is in matrix format we can take a constant matrix to divide that value. In JPEG an inbuilt matrix are given for this purpose .There is choice of an inbuilt matrix. One matrix is based on luminance matrix and other is based on chrominance

Luminance

$$\begin{pmatrix} 8 & 6 & 6 & 7 & 6 & 5 & 8 & 7 \\ 7 & 7 & 9 & 9 & 8 & 10 & 12 & 20 \\ 13 & 12 & 11 & 11 & 12 & 25 & 18 & 19 \\ 15 & 20 & 29 & 26 & 31 & 30 & 29 & 26 \\ 28 & 28 & 32 & 36 & 46 & 39 & 32 & 34 \\ 44 & 35 & 28 & 28 & 40 & 55 & 41 & 44 \\ 48 & 49 & 52 & 52 & 52 & 31 & 39 & 57 \\ 61 & 56 & 50 & 60 & 46 & 51 & 52 & 50 \end{pmatrix}$$

Chrominance

$$\begin{pmatrix} 9 & 9 & 9 & 12 & 11 & 12 & 24 & 13 \\ 13 & 24 & 50 & 33 & 28 & 33 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \end{pmatrix}$$

F'(u,v)=round((F(u,v)\Q(u,v))
F(u,v) –DCT coffient,
Q(u,v)-Quantiztion matrix
F'(u,v)-Quantized DCT coeffient

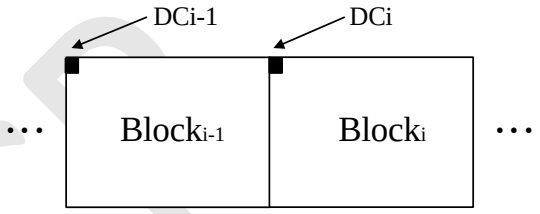
The main loss of JPEG compression will take place here, by looking the Q(u,v) we can see that larger value are at the lower right part. By doing this more loss of data or image will occur at the higher frequencies area.
Q_8x8 is the luminance matrix.

```
quantization_matrix_128x128 = repmat(Q_8x8,16,16);
quantized_image_128x128 =
round(transformed_image./quantization_matrix_1288)
Single_column_quantized_image=im2col(quantized_image
_128x128, [8 8],'distinct');
DCcomponent=ZigZaged_Single_Column_Image(1,:)
ACcomponent=ZigZaged_Single_Column_Image(2:64,:)
```

5.Encoding The Pixel

In this case we first code DC components using the differential coding. and later we encode the AC components using the run length coding and Huffman coding.

After the quantization process we may obtain one DC components and 63 AC components in each block .we may considered only DC and the rest of 63 AC coefficient are not considered in this part. In differential coding the final output will be difference between current DC block with that of the previous block. So in final output of JPEG first value of each block is the difference of this DC value.
Differential Coding :DC




```
b=size(DCcomponent);
 z=zeros(b);
 z(1)=DCcomponent(1);
 for i=2:255
    z(i)= DCcomponent(i+1)- DCcomponent(i);
   end
 z(256)=DCcomponent(256)-DCcomponent(255);
 for i=1:256
    if(z(i)<0)
       flag(i)=1;
    else
       flag(i)=0;
    end
end
 p=abs(z);
```

In run length coding grouping of the elements are taking place here. To make RLC more useful the pixel value should contain the same repeated values then only we obtain maximum compression. If the same value are repeated it is grouped as a single value. It is lossless compression technique. It is better suited in the situation that have the repeating values in the in the input given.

Encoder – Algorithm
1. Scan the on the first value of input
2. Check the next value

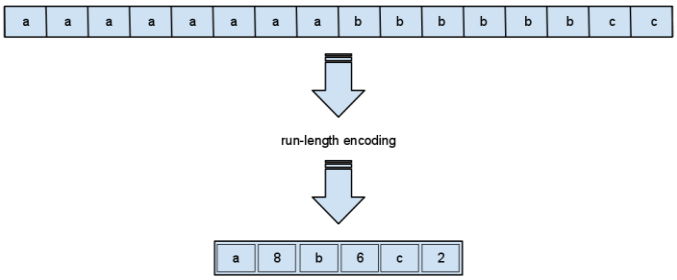
If same as preceding value. set a counter. Then increment the counter value until a different value or end of the input reach. Output the value followed by the counter. Repeat this process.

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 3, October 2015)

If not same as preceding value. Output the value followed by '1'.



Example: EOB (End of Block) signifies the rest of the elements are zeros.
57, 45, 0, 0, 0, 0, 23, 0, -30, -16, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ..., 0
0,57) ; (0,45) ; (4,23) ; (1,-30) ; (0,-16) ; (2,1) ; EOB
(0,57) ; (0,45) ; (4,23) ; (1,-30) ; (0,-16) ; (2,1) ; (0,0)
Code

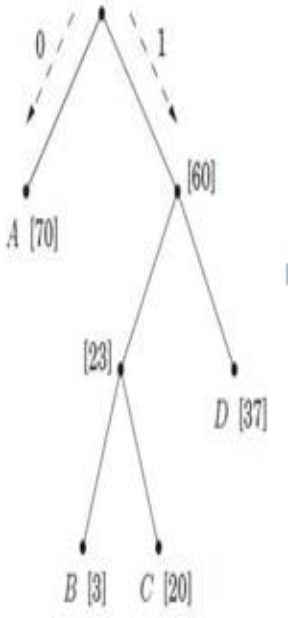
```
run_level=double([]);
 for coloum_index=1:256
 run_level_pairs=double([]);
 single_block_image_vector_64(1:63)=0;
 for row_Vector_Index=1:63
single_block_image_vector_64(row_Vector_Index)=
 ACcomponent(row_Vector_Index, coloum_index);

 end
 non_zero_value_index_array = find(single_block_image_vector_64~=0);
 number_of_non_zero_entries = length(non_zero_value_index_array);
  if number_of_non_zero_entries~=0
 if non_zero_value_index_array(1)==1
 run=0;
run_level_pairs=cat(1,run_level_pair,run,
single_block_image_vector_64(non_zero_value_index_array(1)));
    else
   run=non_zero_value_index_array(1)-1;
run_level_pairs=cat(1,run_level_pairs,,run,
single_block_image_vector_64(non_zero_value_index_array(1));
          end
   if number_of_non_zero_entries>1
              for n=2:number_of_non_zero_entries
run=non_zero_value_index_array(n)-
non_zero_value_index_array(n-1)-1;
  run_level_pairs=cat(1,run_level_pairs,run,
single_block_image_vector_64(non_zero_value_index_array(n)))
;
      end
          end
       end
       % Case 3: "End of Block" mark insertion
       run_level_pairs=cat(1, run_level_pairs, 255, 255);

run_level(1:size(run_level_pairs),coloum_index)=run_level_pair;
end
```

Huffman encoding is last stage of the JPEG compression. It is mainly focused on representing the data in smaller number of bits than the original form.

Algorithm.

1 .For each pixel unit associate a frequency that show how much times it is repeated in that section. We can represent the frequencies as in terms of probability or in percentage form.
2. Next is creating a binary tree in that leaf node represents the smaller frequencies and root node will be the sum of the frequencies of child nodes.
3. Repeat this procedure until all the units are represented in the binary tree format.

Symbol	Codeword
A	0
B	100
C	101
D	11



IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 3, October 2015)

```
f=histc(ACPreparehuffuman(:), 0:255);
fprobability=f(:)'/sum(f);
simbols=find(fprobability~=0);
fprobabilityb=fprobability(simbols);
[fprobabilitybsort,sortindex]=sort(fprobabilityb);
simbolsort=simbols(sortindex);
len=length(simbolsort);
simbols_index=num2cell(1:len);
codeword_tmp=cell(len,1);
while length(fprobabilitybsort)>1,
index1=simbols_index{1};
index2=simbols_index{2};
codeword_tmp(index1)=addnode(codeword_tmp(index1),double(0));
codeword_tmp(index2)=addnode(codeword_tmp(index2),double(1));
fprobabilitybsort=[sum(fprobabilitybsort(1:2)) fprobabilitybsort(3:end)];
simbols_index=[{[index1 index2]} simbols_index(3:end)];
[f,sortindex]=sort(fprobabilitybsort);
simbols_index=simbols_index(sortindex);
end
codeword=cell(256,1);
codeword(simbolsort)=codeword_tmp;
for index=1:length(ACPreparehuffuman),
lendata(1,index)=length(codeword{double(ACPreparehuffuman(index))+1});
end
string=repmat(double(0),1,sum(lendata));
pointer=1;
for index=1:length(ACPreparehuffuman),
 code=codeword{double(ACPreparehuffuman(index))+1};
   if(numel(code)<=8)
      leng=8;
   else
        leng=16;
   end
   len=length(code);
   re=leng-len;
   code=[code round(rand(1,re))];
   string(pointer+(0:leng-1))=code;
pointer=pointer+leng;
end
```

```
codeword=cell(256,1);
codeword(simbolsort)=codeword_tmp;
for index=1:length(ACPreparehuffuman),
lendata(1,index)=length(codeword{double(ACPreparehuffuman(index))+1});
end
string=repmat(double(0),1,sum(lendata));
pointer=1;
for index=1:length(ACPreparehuffuman),
 code=codeword{double(ACPreparehuffuman(index))+1};
   if(numel(code)<=8)
      leng=8;
   else
        leng=16;
   end
   len=length(code);
   re=leng-len;
   code=[code round(rand(1,re))];
   string(pointer+(0:leng-1))=code;
pointer=pointer+leng;
end
codelen=zeros(size(codeword));
weights=2.^(0:51);
for index=1:length(codeword),
len=length(codeword{index});
code=sum(weights(codeword{index}==1));
codeword{index}=code;
codelen256(index)=len;
end
end
Table1(1:size(codeword))=codeword;
Table2=codelen256;
```

CONCLUSION

I have demonstrated that conversion from a color image to the JPEG encoded binary bit stream .It is a fairly simple and straightforward process. I have mentioned main part of the program that is implemented in MATLAB.I hope this paper will make you fell more user-friendly with JPEG compression tool.

REFERENCES

1"LOSSLESS HUFFMAN CODING TECHNIQUE FOR IMAGE COMPRESSION AND RECONSTRUCTION USING BINARY TREES": ISSN: 2229-6093MridulKumaMathur et al,Int.J.Comp.Tech.Appl,Vol 3 (1), 76-79

2." A NEW LOSSLESS METHOD OF IMAGE COMPRESSION AND DECOMPRESSION USING HUFFMAN CODING TECHNIQUES": Journal of Theoretical and Applied Information Technology

3." IMAGE COMPRESSION USING THE DISCRETE COSINE TRANSFORM" Mathematica Journal, 4(1), 1994, p. 81-88