Lab 2: JPEG Compression

Brian Hosler & Sarah Peachey

February 2, 2018

Abstract

Contents

1 JPEG Quality Factor

Peak Signal to Noise Ratio (PSNR) is a common method for measuring the quality of a image that has gone through compression. PSNR is a way to quanitize the quality of a decoded image as perceived by the human eye. (1) is how PSNR is calculated, where σ_e^2 is the variance of noise/error and A is the bit depth of the image. The value calculated can be interpretted as unnoticable noise if it is greater than 40dB, as low quality if it is less than 30dB, and as good quality otherwise.

$$PSNR = 10log_{10}(\frac{A^2}{\sigma_e^2}) \tag{1}$$

```
pep=imread('peppers.tif');                                          1
bab=imread('baboon.tif');                                           2
                                                                    3
imwrite(pep, 'pep90.jpg','Quality',90)                              4
imwrite(pep, 'pep70.jpg','Quality',70)                              5
imwrite(pep, 'pep50.jpg','Quality',50)                              6
imwrite(pep, 'pep30.jpg','Quality',30)                              7
imwrite(pep, 'pep10.jpg','Quality',10)                              8
                                                                    9
imwrite(bab, 'bab90.jpg','Quality',90)                              10
imwrite(bab, 'bab70.jpg','Quality',70)                              11
imwrite(bab, 'bab50.jpg','Quality',50)                              12
imwrite(bab, 'bab30.jpg','Quality',30)                              13
imwrite(bab, 'bab10.jpg','Quality',10)                              14
                                                                    15
pep_psnr=zeros(1,6);                                                16
pep_size=zeros(1,6);                                                17
                                                                    18
temp=imfinfo('peppers.tif');                                        19
pep_size(1)=temp.FileSize;                                          20
pep_psnr(1)= psnr(pep, pep);                                        21
                                                                    22
temp=imfinfo('pep90.jpg');                                          23
pep_size(2)=temp.FileSize;                                          24
pep_psnr(2)= psnr(imread('pep90.jpg'), pep);                        25
                                                                    26
temp=imfinfo('pep70.jpg');                                          27
pep_size(3)=temp.FileSize;                                          28
pep_psnr(3)= psnr(imread('pep70.jpg'), pep);                        29
                                                                    30
temp=imfinfo('pep50.jpg');                                          31
pep_size(4)=temp.FileSize;                                          32
pep_psnr(4)= psnr(imread('pep50.jpg'), pep);                        33
                                                                    34
temp=imfinfo('pep30.jpg');                                          35
pep_size(5)=temp.FileSize;                                          36
pep_psnr(5)= psnr(imread('pep30.jpg'), pep);                        37
                                                                    38
temp=imfinfo('pep10.jpg');                                          39
pep_size(6)=temp.FileSize;                                          40
pep_psnr(6)= psnr(imread('pep10.jpg'), pep);                        41
                                                                    42
figure                                                              43
subplot(2,1,1)                                                      44
plot(100*[1 .9 .7 .5 .3 .1],pep_size,'--o')                         45
hold on                                                             46
title('peppers -- file size vs quality')                           47
```

```matlab
subplot(2,1,2)                                                              48
plot(100*[1 .9 .7 .5 .3 .1],pep_psnr,'--o')                                49
hold on                                                                     50
title('peppers _-_ psnr _vs _quality')                                     51
                                                                            52
bab_psnr=zeros(1,6);                                                        53
bab_size=zeros(1,6);                                                        54
                                                                            55
temp=imfinfo('baboon.tif');                                                56
bab_size(1)=temp.FileSize;                                                  57
bab_psnr(1)= psnr(bab, bab);                                               58
                                                                            59
temp=imfinfo('bab90.jpg');                                                 60
bab_size(2)=temp.FileSize;                                                  61
bab_psnr(2)= psnr(imread('bab90.jpg'), bab);                               62
                                                                            63
temp=imfinfo('bab70.jpg');                                                 64
bab_size(3)=temp.FileSize;                                                  65
bab_psnr(3)= psnr(imread('bab70.jpg'), bab);                               66
                                                                            67
temp=imfinfo('bab50.jpg');                                                 68
bab_size(4)=temp.FileSize;                                                  69
bab_psnr(4)= psnr(imread('bab50.jpg'), bab);                               70
                                                                            71
temp=imfinfo('bab30.jpg');                                                 72
bab_size(5)=temp.FileSize;                                                  73
bab_psnr(5)= psnr(imread('bab30.jpg'), bab);                               74
                                                                            75
temp=imfinfo('bab10.jpg');                                                 76
bab_size(6)=temp.FileSize;                                                  77
bab_psnr(6)= psnr(imread('bab10.jpg'), bab);                               78
                                                                            79
figure                                                                      80
subplot(2,1,1)                                                              81
plot(100*[1 .9 .7 .5 .3 .1],bab_size,'--o')                                82
hold on                                                                     83
title('baboon _-_ file _size _vs _quality')                                84
subplot(2,1,2)                                                              85
plot(100*[1 .9 .7 .5 .3 .1],bab_psnr,'--o')                                86
hold on                                                                     87
title('baboon _-_ psnr _vs _quality')                                      88
```
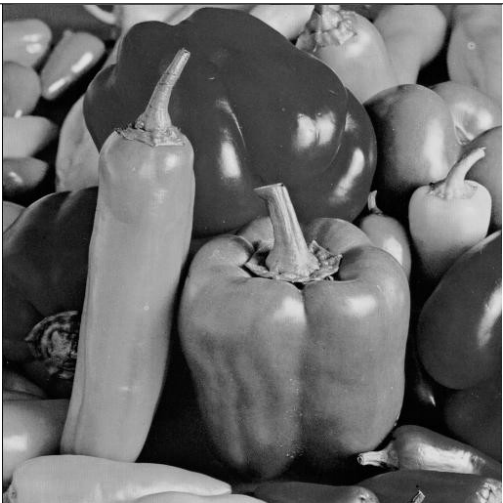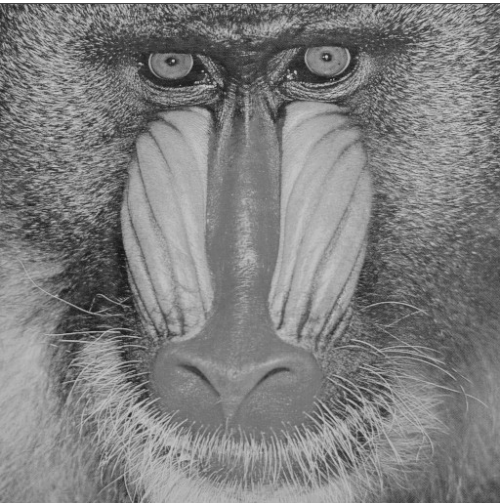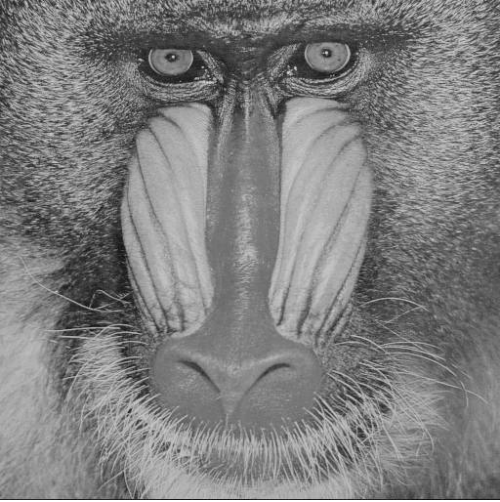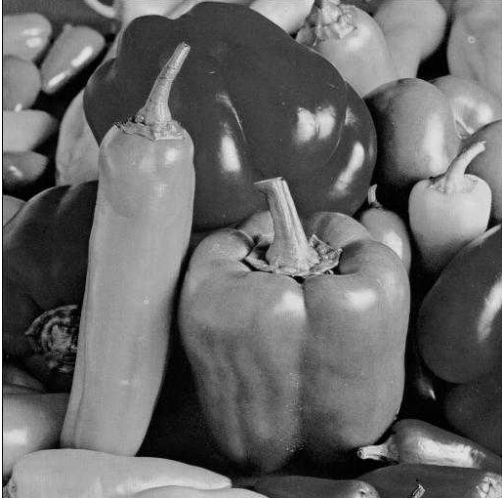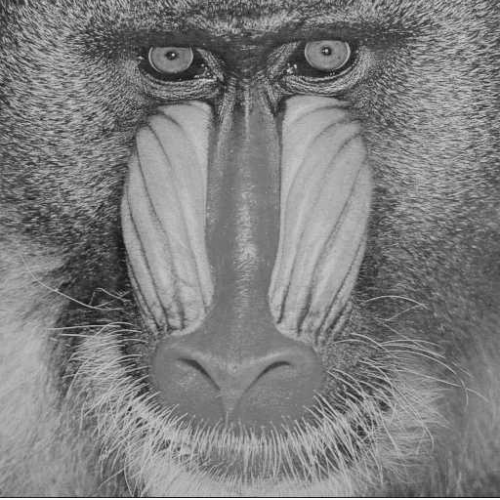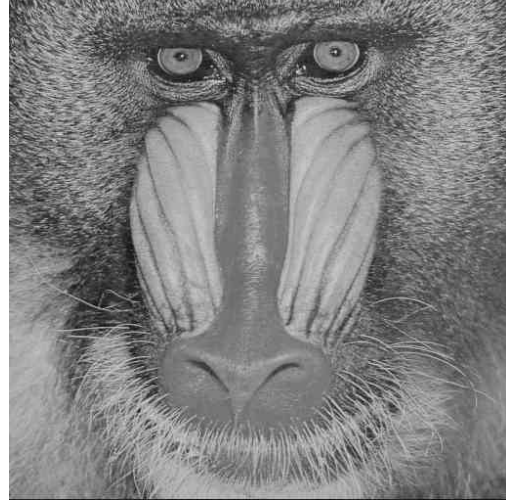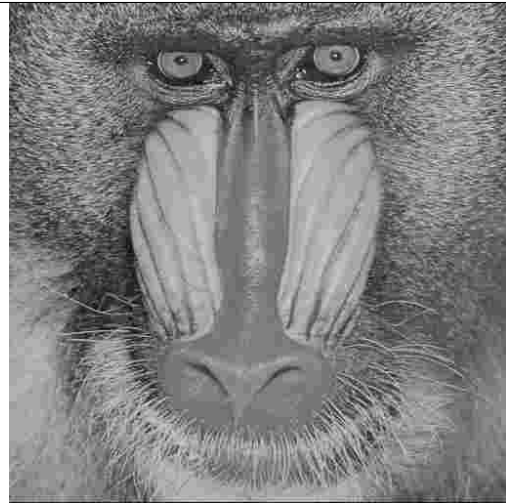
In the following table the peppers.tif and baboon.tif were compressed with image quality factors of 90, 70, 50, 30, & 10.

Quality Factor	Peppers	Baboon
90		
70		
50		

30

10

As seen in figures 1, and 2 as the image file size gets smaller the PSNR of the image drops lower. Also, as seen in the images as the quality gets lower the image becomes more blocky and noise in added in the high freqency sections. Which is normally fine because the human eye is less perceptive to changes in high frequency regions, but edge also count as high frequency and noise can be perceived on edged of smooth regions. The blockiness occurs because jpeg divides the image into 8 x 8 block and then runs the algorithm on each block, therefore, some block may throw away or keep more information than the one next to it. Also the jpeg algorithm takes into account the humans do not perceive change in high frequency regions as well, so it will discrad this information.

There is a clear drop in image quality around a quality factor of 30 or 10. The image becomes far more blocky because of the loss in information. The PSNR graphs also reflect this, because the peppers at quality 10 have a PSNR of about 30, and the baboon at quality of 30 drops below 30dB. Since less than 30dB is the qualification for low quality it make sense that these images have visually perceived error.
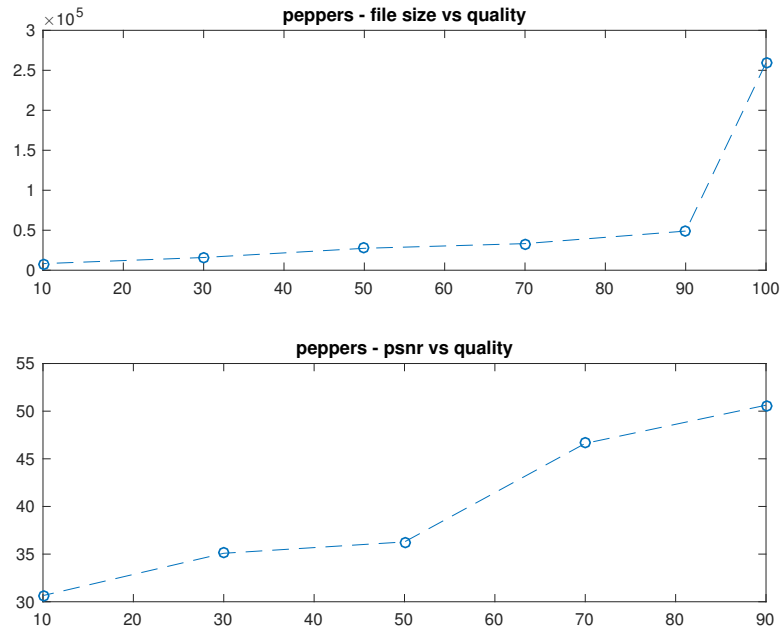
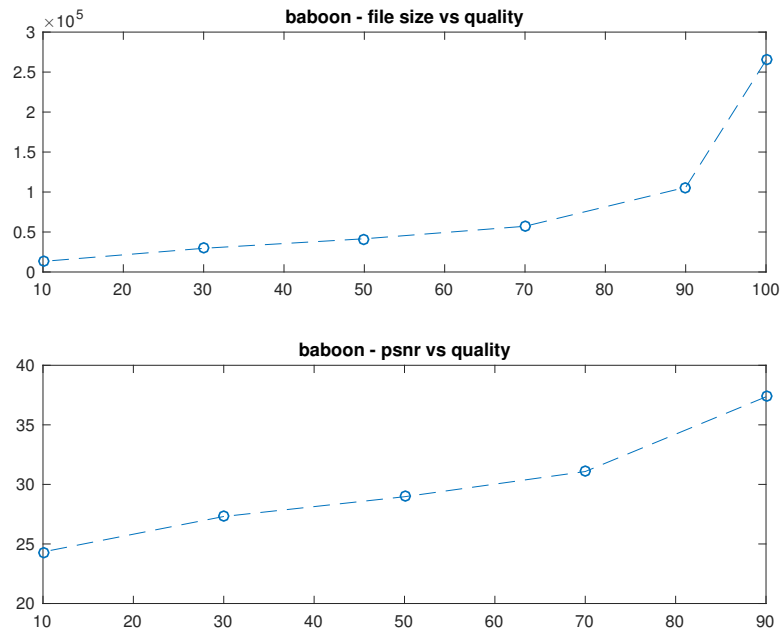Figure 1: Peppers file size and PSNR as the quality changes

Figure 2: Baboon file size and PSNR as the quality changes

2 Writing JPEG compression in matlab

The following sections are the code the implements the jpeg encoding algorithm by segmenting the image into 8 x 8 pixel blocks, computing the discrete cosine tranform, and quanitizing each block. Then using the provided functions to further format and entropy encode the jpeg. That process is then reversed for decoding the image.

myJpgEncode.m

```matlab
function [result] = myJpgEncode( pep,Q )                      1
%myJpgEncode implement my own jpeg algorithm                   2
%    using the notes                                           3
A=zeros(size(pep));                                            4
stor=[];                                                       5
for i=1:512/8                                                  6
    for j=1:512/8                                              7
        tempA=dct2(pep(8*(i-1)+1:i*8,8*(j-1)+1:j*8));          8
        pep_quan=round(tempA./Q).*Q;                           9
        stor=[stor;ZigzagMtx2Vector(pep_quan)];              10
    end                                                       11
end                                                           12
result=JPEG_entropy_encode(512,512,8,Q,stor,'./',1);         13
end                                                          14
```

myJpgDecode.m

```matlab
function [pep] = myJpgDecode()                                1
%myJpgEncode implement my own jpeg algorithm                  2
%    using the notes                                          3
[rowN,colN,dct_block_size,iQ,iZZDCTQIm]=JPEG_entropy_decode('  4
    ./');
pep=zeros(512);                                               5
ndx=1;                                                        6
for i=1:512/8                                                 7
    for j=1:512/8                                             8
        pep(8*(i-1)+1:i*8,8*(j-1)+1:j*8)=idct2(               9
            Vector2ZigzagMtx(iZZDCTQIm(ndx,:)));
        ndx=ndx+1;                                           10
    end                                                      11
end                                                         12
end                                                         13
```

3 Evaluating Quantization Tables

Our jpeg encoding and decoding algorithms are then tested by using the standard jpeg luminance quantization table. Then a custom quantization table is created based off of the values from the DCT of the peppers.tif image. This custom table should better compress the image, without losing too much information as to distort the image.

```matlab
Q=[16  11  10  16  24  40  51   61;...          1
    12  12  14  19  26  58  60   55;...          2
    14  13  16  24  40  57  69   56;...          3
    14  17  22  29  51  87  80   62;...          4
    18  22  37  56  68  109  103  77;...         5
    24  35  55  64  81  104  113  92;...         6
    49  64  78  87  103  121  120  101;...       7
    72  92  95  98  112  100  103  99];          8
                                                 9
tempQ=zeros(8);                                  10
for  i=1:512/8                                   11
    for  j=1:512/8                               12
        tempQ=tempQ+abs(dct2(pep(8*(i-1)+1:i*8,8*(j-1)+1:j*8))  13
            );
    end                                          14
end                                              15
DCTs=tempQ/4096;                                 16
nrm1=max(max(DCTs))./DCTs;                       17
Q2=double(uint8(nrm1.^.65));                     18
                                                 19
stor=[];                                         20
for  i=1:512/8                                   21
    for  j=1:512/8                               22
        tempA=dct2(pep(8*(i-1)+1:i*8,8*(j-1)+1:j*8));  23
        stor=[stor;ZigzagMtx2Vector(tempA)];     24
    end                                          25
end                                              26
vrnce=Vector2ZigzagMtx(var(stor));               27
nrm2=max(max(vrnce))./vrnce;                     28
Q3=double(uint8(log(nrm2).^2))+1;                29
                                                 30
A=myJpgEncode( pep,Q );                          31
jpg1=uint8(myJpgDecode());                       32
psnr(pep,jpg1)                                   33
                                                 34
B=myJpgEncode( pep,Q2 );                         35
jpg2=uint8(myJpgDecode());                       36
psnr(pep,jpg2)                                   37
                                                 38
C=myJpgEncode( pep,Q3 );                         39
jpg3=uint8(myJpgDecode());                       40
psnr(pep,jpg3)                                   41
                                                 42
figure                                           43
imshow([pep,jpg1;jpg2,jpg3])                     44
```

The first quantization table used was the standard JPEG table, which resulted in a file size of 44kB, and a signal to noise ratio around 36. In order to improve upon this, a table should be constructed and tailored to the image it's representing. With this in mind, a quantization table was derived by taking the average magnitude of each DCT coefficient over the whole image. These average magnitudes were inverted, so that a high mean would become a small quantization value, and a small magnitude would result in a large quantization step size. This table was scaled to between 1 and 300, so that step sizes were more realistic. Finally, a third table was constructed using the varience of the DCT coefficients.

Similar normalization was applied. The compressed file sizes and psnr's are reported in Table 1.

The varience based table compressed the image the most, will little loss in the psnr. Comparing the compressed image with original side-by-side, makes the loss in quality apparent. The Mean-based table did not compress the image as well as the variance-based table, but resulted in a higher psnr than the standard JPEG table.

Q-Table	Compressed Size(kB)	PSNR(dB)
Standard	44.7	36.27
Mean	33.1	36.33
Variance	23.7	34.40

Table 1: Comparison of Different JPEG Quantization Tables Used

Figure 3: Resultant image after no JPEG compression(top-left), compression using the JPEG standard quantization table(top-right), compression using a quantization table based on the mean magnitude of the DCT coefficients(bottom-left), and compression using a quantization table derived from the DCT coefficients' variences(bottom-right).