JPEG格式分析

姓名：邹宇航 学号：1801210778

JPEG 是Joint Photographic Experts Group（联合图像专家小组）的缩写，是第一个国际图像压缩标准。JPEG图像压缩算法能够在提供良好的压缩性能的同时，具有比较好的重建质量，被广泛应用于图像、视频处理领域。

JPEG的整体流程遵循几个大步骤：1.把数据分为“重要”和“不重要”部分。2.过滤掉不重要部分。3.保存。

详细的压缩流程如下：

（1）图像分割：JPEG算法的第一步，图像被分割成大小为NN的小块，这些小块在整个压缩过程中都是单独被处理的。

（2）颜色空间转换RGB->YCbCr：所谓“颜色空间”，是指表达颜色的数学模型，比如我们常见的“RGB”模型，就是把颜色分解成红绿蓝三种分量，这样一张图片就可以分解成三张灰度图，数学表达上，每一个NN的图案，可以表达成三个NN的矩阵，其中的数值的范围一般在[0,255]之间。不同的颜色模型各有不同的应用场景，例如RGB模型适合于像显示器这样的自发光图案，而在印刷行业，使用油墨打印，图案的颜色是通过在反射光线时产生的，通常使用CMYK模型，而在JPEG压缩算法中，需要把图案转换成为YCbCr模型，这里的Y表示亮度(Luminance)，Cb和Cr分别表示绿色和红色的“色差值”。 JPEG把图像转换为YCbCr之后，就可以针对数据得重要程度的不同做不同的处理。这就是为什么JPEG使用这种颜色空间的原因。

（3）离散余弦变换：JPEG算法中的核心内容，离散余弦变换（Discrete cosine transform），简称DCT。

（4）数据量化：在可以损失一部分精度的情况下，如何用更少的空间存储这些浮点数？结果是使用量子化（Quantization），简称量化。是把方向分成16个区间，用0到16这样的整数来表示，这样只用4个bit就足够了。JPEG提供的量子化算法如下：

https://thecodeway.com/blog/wp-content/uploads/2014/09/jpeg_031.gif

（5）哈夫曼编码：JPEG压缩的最后一步是对数据进行哈弗曼编码(Huffman coding)，它的基本原理是根据数据中元素的使用频率，调整元素的编码长度，以得到更高的压缩比。

网上下载Matlab代码并且调通（实现JPEG压缩）：

1.compress.m

function JPEGEncodeDecode

%UNTITLED7 Summary of this function goes here

%   Detailed explanation goes here

img=imread('bridge.bmp');

subplot(121);imshow(img);title('原图');          %显示原图

img\_ycbcr = rgb2ycbcr(img);             % rgb->yuv

[row,col,~]=size(img\_ycbcr);       % 取出行列数，~表示3个通道算1列

%对图像进行扩展

row\_expand=ceil(row/16)\*16;        %行数上取整再乘16，及扩展成16的倍数

if mod(row,16)~=0            %行数不是16的倍数，用最后一行进行扩展

    for i=row:row\_expand

        img\_ycbcr(i,:,:)=img\_ycbcr(row,:,:);

    end

end

col\_expand=ceil(col/16)\*16;  %列数上取整

if mod(col,16)~=0         %列数不是16的倍数，用最后一列进行扩展

    for j=col:col\_expand

        img\_ycbcr(:,j,:)=img\_ycbcr(:,col,:);

    end

end

%对Y,Cb,Cr分量进行4:2:0采样

Y=img\_ycbcr(:,:,1);                    %Y分量

Cb=zeros(row\_expand/2,col\_expand/2);        �分量

Cr=zeros(row\_expand/2,col\_expand/2);        %Cr分量

for i=1:row\_expand/2

    for j=1:2:col\_expand/2-1          %奇数列

        Cb(i,j)=double(img\_ycbcr(i\*2-1,j\*2-1,2));

        Cr(i,j)=double(img\_ycbcr(i\*2-1,j\*2+1,3));

    end

end

for i=1:row\_expand/2

    for j=2:2:col\_expand/2            %偶数列

        Cb(i,j)=double(img\_ycbcr(i\*2-1,j\*2-2,2));

        Cr(i,j)=double(img\_ycbcr(i\*2-1,j\*2,3));

    end

end

%分别对三种颜色分量进行编码

Y\_Table=[16  11  10  16  24  40  51  61

    12  12  14  19  26  58  60  55

    14  13  16  24  40  57  69  56

    14  17  22  29  51  87  80  62

    18  22  37  56  68 109 103  77

    24  35  55  64  81 104 113  92

    49  64  78  87 103 121 120 101

    72  92  95  98 112 100 103  99];%亮度量化表

CbCr\_Table=[17, 18, 24, 47, 99, 99, 99, 99;

    18, 21, 26, 66, 99, 99, 99, 99;

    24, 26, 56, 99, 99, 99, 99, 99;

    47, 66, 99 ,99, 99, 99, 99, 99;

    99, 99, 99, 99, 99, 99, 99, 99;

    99, 99, 99, 99, 99, 99, 99, 99;

    99, 99, 99, 99, 99, 99, 99, 99;

    99, 99, 99, 99, 99, 99, 99, 99];%色差量化表

Qua\_Factor=0.5;%量化因子,最小为0.01,最大为255,建议在0.5和3之间,越小质量越好文件越大

%对三个通道分别DCT和量化

Y\_dct\_q=Dct\_Quantize(Y,Qua\_Factor,Y\_Table);

Cb\_dct\_q=Dct\_Quantize(Cb,Qua\_Factor,CbCr\_Table);

Cr\_dct\_q=Dct\_Quantize(Cr,Qua\_Factor,CbCr\_Table);

%对三个通道分别反量化和反DCT

Y\_in\_q\_dct=Inverse\_Quantize\_Dct(Y\_dct\_q,Qua\_Factor,Y\_Table);

Cb\_in\_q\_dct=Inverse\_Quantize\_Dct(Cb\_dct\_q,Qua\_Factor,CbCr\_Table);

Cr\_in\_q\_dct=Inverse\_Quantize\_Dct(Cr\_dct\_q,Qua\_Factor,CbCr\_Table);

%恢复出YCBCR图像

YCbCr\_in(:,:,1)=Y\_in\_q\_dct;

for i=1:row\_expand/2

    for j=1:col\_expand/2

        YCbCr\_in(2\*i-1,2\*j-1,2)=Cb\_in\_q\_dct(i,j);

        YCbCr\_in(2\*i-1,2\*j,2)=Cb\_in\_q\_dct(i,j);

        YCbCr\_in(2\*i,2\*j-1,2)=Cb\_in\_q\_dct(i,j);

        YCbCr\_in(2\*i,2\*j,2)=Cb\_in\_q\_dct(i,j);

        YCbCr\_in(2\*i-1,2\*j-1,3)=Cr\_in\_q\_dct(i,j);

        YCbCr\_in(2\*i-1,2\*j,3)=Cr\_in\_q\_dct(i,j);

        YCbCr\_in(2\*i,2\*j-1,3)=Cr\_in\_q\_dct(i,j);

        YCbCr\_in(2\*i,2\*j,3)=Cr\_in\_q\_dct(i,j);

    end

end

I\_in=ycbcr2rgb(YCbCr\_in);

I\_in(row+1:row\_expand,:,:)=[];%去掉扩展的行

I\_in(:,col+1:col\_expand,:)=[];%去掉扩展的列

subplot(122);imshow(I\_in);title('重构后的图片');

end

 function [Matrix]=Dct\_Quantize(I,Qua\_Factor,Qua\_Table)

%UNTITLED Summary of this function goes here

%   Detailed explanation goes here

I=double(I)-128;   %层次移动128个灰度级，详见书本P401

�t2变换：把ImageSub分成8\*8像素块，分别进行dct2变换，得变换系数矩阵Coef

I=blkproc(I,[8 8],'dct2(x)');

Qua\_Matrix=Qua\_Factor.\*Qua\_Table;              %量化矩阵

I=blkproc(I,[8 8],'round(x./P1)',Qua\_Matrix);  %量化，四舍五入

Matrix=I;          %得到量化后的矩阵

end

 function [ Matrix ] = Inverse\_Quantize\_Dct( I,Qua\_Factor,Qua\_Table )

%UNTITLED3 Summary of this function goes here

%   Detailed explanation goes here

Qua\_Matrix=Qua\_Factor.\*Qua\_Table;     %反量化矩阵

I=blkproc(I,[8 8],'x.\*P1',Qua\_Matrix);%反量化，四舍五入

[row,column]=size(I);

I=blkproc(I,[8 8],'idct2(x)');   �T反变换

I=uint8(I+128);

for i=1:row

    for j=1:column

        if I(i,j)>255

            I(i,j)=255;

        elseif I(i,j)<0

            I(i,j)=0;

        end

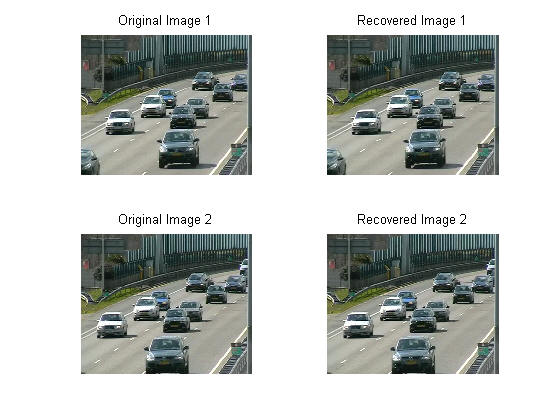
    end

end

Matrix=I;       %反量化和反Dct后的矩阵

end

实验结果：



压缩率===9.17%