```
AC Encoding Finshed
Encoding time:  0.1645 s
table size:  0 Bytes
code size:  2213 Bytes
 ----------------
AC Decoding Finshed
Decoding time:  0.1748 s
AC Lossy Compression Success!
Entropy:  1.77; Average code length:  1.82
```

3.3 图像无损压缩系统设计

在3.1和3.2的基础上，我们可以考虑针对图像数据，进行无损压缩系统的设计。`ImgLossyCodec_Encoder_Demo`为一个简单的图像压缩系统的函数实现示例，具体步骤包括：

- Step 1 预处理：读入原始图像数据，并将图像数据从RGB颜色空间转换到YCbCr颜色空间。完成参数设置，如码流文件的存放文件夹检查等。
- step 2 对图像数据的每个通道，利用matlab中的内置函数`hist`，完成图像信号的概率统计分析，调用自定义单通道编码函数`encode_channel`,完成二进制码流的生成和保存。并打印编码时间和压缩倍数。
- step 3 对图像数据的每个通道，调用自定义单通道解码函数`decode_channel`,完成图像数据的重建，再重新将图像数据转换到RGB颜色空间。并打印解码时间和判断图像数据是否无损。

以下分别为单通道编码函数`encode_channel`和单通道解码函数`decode_channel`的函数实现示例。其中，为了消除空间冗余，引入水平差分预测方法，并用参数`diffmode`来进行控制。当`diffmode`为0时关闭差分预测模式。

```
%%file encode_channel.m
function [size_dict, size_bitstream]=encode_channel(img, htreefile,
streamfile, diffmode)
%% 信号源的统计特性分析
if diffmode
    img = double(img);
    [Height,Width] = size(img);
    sig = img;
    sig(:,2:Width) = img(:,2:Width)-img(:,1:Width-1);
    sig = sig(:);
else
    sig = double(img(:));
end
[prob,symbols] = hist(sig,[min(sig):1:max(sig)]);% compute the histogram.
prob = prob/numel(sig);

%% 赫夫曼编码
dict = huffmandict(symbols,prob); %huffman tree
bitstream = huffmanenco(sig,dict);

%% 保存为码流文件（注意文件尺寸）
size_dict = savehtree(htreefile, dict);
size_bitstream = savestreamfile(streamfile, bitstream);
end
```

```
%%file decode_channel.m
function img = decode_channel(htreefile, streamfile, diffmode, imgsize)
%% 读取码流文件
```

```
dict_d = readhtree(htreefile);
bitstream_d = readstreamfile(streamfile);

dhsig = huffmandeco(bitstream_d,dict_d);
diffimg = reshape(dhsig, imgsize(1), imgsize(2));

%% 差分预测模式重建
if diffmode
    img = zeros(imgsize);
    img(:,1) = diffimg(:,1);
    for w = 2: imgsize(2)
        img(:,w) = diffimg(:,w)+img(:,w-1);
    end
else
    img = diffimg;
end
end
```

```
%%file ImgLossyCodec_Encoder_Demo.m
function ImgLossyCodec_Encoder_Demo(filename, ratio, diffmode)
%close all, clear all
%% 读入待压缩图像原始数据
%filename = 'data\\Lenna.png';
%filename = 'data\\Weeki_Wachee_spring.jpg';
img_rgb = imread(filename);
img_rgb = imresize(img_rgb, 1/ratio);
[h, w, d] = size(img_rgb);
if d > 1
    img_yuv = rgb2ycbcr(img_rgb);
end

% 码流存放路径
stream_path = 'stream';
if ~isdir(stream_path)
    mkdir(stream_path);
end

% 文件名解析
[pathstr, name, ext] = fileparts(filename);

% 预测模型参数设置：0/1：不开启/开启差分模式
%diffmode = 0; %
if diffmode
    disp(sprintf('Difference prediction mode is ON! \r\n-------------------
-----'));
else
    disp(sprintf('Difference prediction mode is OFF! \r\n-----------------
------'));
end
%% 对图像的各个通道进行单独编码，每个通道输出htree.bin和stream.bin两个文件。
tic
size_dict=0; % huffman tree的文件的大小(Byte)
size_bitstream=0; % 生成的码流文件的总大小(Byte)
if d == 1
    htreefile = sprintf('%s\\%s_m%d_htree.bin',stream_path,name,diffmode);
    streamfile =
sprintf('%s\\%s_m%d_stream.bin',stream_path,name,diffmode);
```

```
    [size_dict, size_bitstream]=encode_channel(img_rgb, htreefile,
streamfile, diffmode);
else
    for chl=1:d
        htreefile =
sprintf('%s\\%s_chl%d_m%d_htree.bin',stream_path,name,chl,diffmode);
        streamfile =
sprintf('%s\\%s_chl%d_m%d_stream.bin',stream_path,name,chl,diffmode);
        [size_dict_chl,
size_bitstream_chl]=encode_channel(img_yuv(:,:,chl), htreefile, streamfile,
diffmode);
        size_dict=size_dict+size_dict_chl;
        size_bitstream=size_bitstream+size_bitstream_chl;
    end
end
disp(sprintf('Image Encoding Finshed'));
disp(sprintf('Encoding time:  %6.4f s', toc));
disp(sprintf('Dict size:  %d Bytes', size_dict));
disp(sprintf('Code size:  %d Bytes', size_bitstream));

%%Calculation of compression ratio
B0 = numel(img_rgb);
B1 = size_dict+size_bitstream;
compressionratio=B0/B1;
disp(sprintf('Orginal:  %d Bytes; Compressed:  %d Bytes; Compression ratio:
%6.2f \r\n ----------------',B0, B1, compressionratio));

%% 对图像的各个通道进行单独解码，每个通道首先解析相应的htree.bin文件，生成Huffman
TreeB0,再对stream.bin文件进行解码和重建。
tic
% size_dict=0; % 读取的huffman tree的文件大小(Byte)
% size_bitstream=0; % 读取的码流文件的总大小(Byte)
img_yuv_rec = zeros([h,w,d]);
if d == 1
    htreefile = sprintf('%s\\%s_m%d_htree.bin',stream_path,name,diffmode);
    streamfile =
sprintf('%s\\%s_m%d_stream.bin',stream_path,name,diffmode);
    img_rec = decode_channel(htreefile, streamfile, diffmode, [h, w]);
else
    for chl=1:d
        htreefile =
sprintf('%s\\%s_chl%d_m%d_htree.bin',stream_path,name,chl,diffmode);
        streamfile =
sprintf('%s\\%s_chl%d_m%d_stream.bin',stream_path,name,chl,diffmode);
        img_yuv_rec_chl = decode_channel(htreefile, streamfile, diffmode,
[h, w]);
        img_yuv_rec(:,:,chl) = img_yuv_rec_chl;
    end
    img_yuv_rec = uint8(img_yuv_rec);
    img_rec = ycbcr2rgb(img_yuv_rec);
end
disp(sprintf('Image Decoding Finshed'));
disp(sprintf('Decoding time:  %6.4f s', toc));
mse_yuv= sum((double(img_yuv(:))-
double(img_yuv_rec(:))).^2)/numel(img_yuv);
mse_rgb= sum((double(img_rgb(:))-double(img_rec(:))).^2)/numel(img_rgb);

if mse_yuv
```

```
    disp('Image Lossy Compression Failed!')
else
    disp('Image Lossy Compression Success!')
end
subplot(121),imshow(img_rgb), title(sprintf('original: %s', name))
subplot(122),imshow(img_rec), title(sprintf('MSE YUV: %6.2f; RGB: %6.2f',
mse_yuv, mse_rgb))

```

读入图像数据，开启差分预测模式，执行以下代码：

```
%lec4exp7
close all, clear all
%% 读入待压缩图像原始数据
filename = 'data\\Lenna.png';
%filename = 'data\\Weeki_Wachee_spring.jpg';
ratio = 1;
ImgLossyCodec_Encoder_Demo(filename, ratio, 1)
```

以下为差分预测模式开启后的输出结果：

```
Difference prediction mode is ON!
------------------------
Image Encoding Finshed
Encoding time:  10.4095 s
Dict size:  22275 Bytes
Code size:  412967 Bytes
Orginal:  786432 Bytes; Compressed:  435242 Bytes; Compression ratio:
1.81
 ----------------
Image Decoding Finshed
Decoding time:  360.4501 s
Image Lossy Compression Success!
```

关闭差分预测模式，执行以下代码：

```
ImgLossyCodec_Encoder_Demo(filename, ratio, 0)
```

以下为差分预测模式关闭启后的输出结果：

```
Difference prediction mode is OFF!
------------------------
Image Encoding Finshed
Encoding time:  10.4310 s
Dict size:  5990 Bytes
Code size:  596815 Bytes
Orginal:  786432 Bytes; Compressed:  602805 Bytes; Compression ratio:
1.30
 ----------------
Image Decoding Finshed
Decoding time:  326.5554 s
Image Lossy Compression Success!
```

从上面的实验结果可以看出，差分预测模式的开启，可以大大提高压缩性能。但由于字典变大，字典文件所需要的存储空间显著增大。同时，解码复杂度也会提升。

-
-