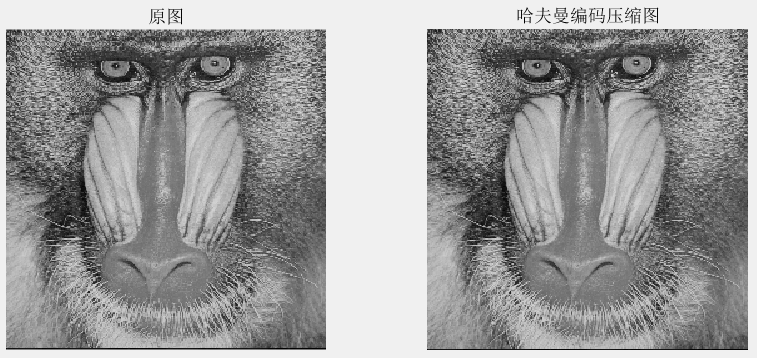
**作业4 数字图像编码实验**

高鹏昺 计科3班15051317

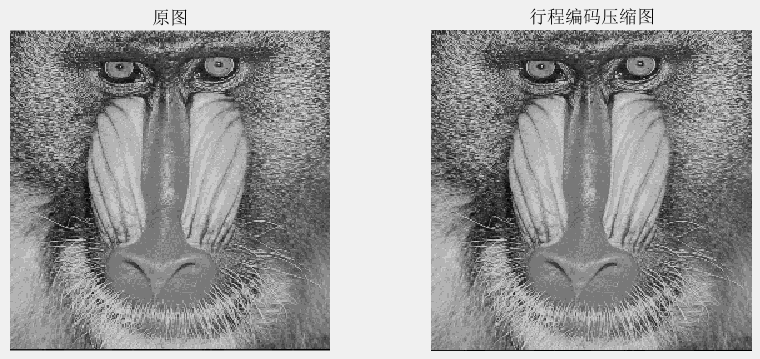
**题目1 哈夫曼编码和行程编码压缩处理**

**测试图1结果：**

Huffman哈夫曼编码压缩：



RLE行程编码压缩：



压缩信息比较分析：

|  |  |  |  |
| --- | --- | --- | --- |
| **编码压缩方式** | **原图尺寸** | **压缩后尺寸** | **压缩率** |
| **哈夫曼编码** | 256\*256 | 1\*60522 | 0.9235 |
| **行程编码** | 256\*256 | 41931\*2 | 1.2796 |

**测试图2结果：**

Huffman哈夫曼编码压缩：



RLE行程编码压缩：



压缩信息比较分析：

|  |  |  |  |
| --- | --- | --- | --- |
| **编码压缩方式** | **原图尺寸** | **压缩后尺寸** | **压缩率** |
| **哈夫曼编码** | 256\*256 | 1\*63495 | 0.9689 |
| **行程编码** | 256\*256 | 23872\*2 | 0.7285 |

**测试图3结果：**

Huffman哈夫曼编码压缩：



RLE行程编码压缩：



压缩信息比较分析：

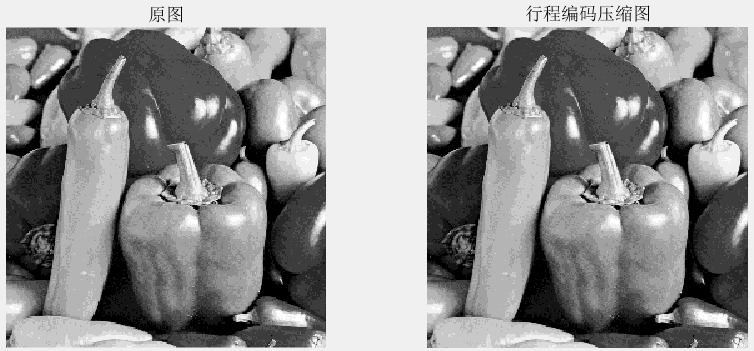
|  |  |  |  |
| --- | --- | --- | --- |
| **编码压缩方式** | **原图尺寸** | **压缩后尺寸** | **压缩率** |
| **哈夫曼编码** | 256\*256 | 1\*58623 | 0.8945 |
| **行程编码** | 256\*256 | 16339\*2 | 0.4986 |

**测试图4结果：**

Huffman哈夫曼编码压缩：



RLE行程编码压缩：



压缩信息比较分析：

|  |  |  |  |
| --- | --- | --- | --- |
| **编码压缩方式** | **原图尺寸** | **压缩后尺寸** | **压缩率** |
| **哈夫曼编码** | 256\*256 | 1\*63352 | 0.9667 |
| **行程编码** | 256\*256 | 23231\*2 | 0.7090 |

**题目2 JPEG编码压缩处理**

**结果图：**



压缩信息比较分析：

|  |  |  |  |
| --- | --- | --- | --- |
| **质量因子** | **均方根误差** | **压缩比** | **图像大小** |
| **20** | 1042.2516 | 22.04 | 11894 |
| **60** | 709.3750 | 10.89 | 24076 |
| **80** | 571.5449 | 6.91 | 37937 |

**实验心得总结**

通过本次实验编程和图像对比分析我对图像压缩中的哈夫曼编码、行程编码和JPEG编码这3种图像压缩编码方式都有了更加深入的理解和掌握。

哈夫曼编码和行程编码属于无损编码，压缩率存在一个极限。行程编码对于有大面积色块的图像也缩效果很好，对于纷杂的图像压缩效果不好。当图像灰度分布不均匀时，哈夫曼编码的编码效率较高。JPEG静止图像编码为有损压缩，压缩比较高，失真程度较小。

**程序代码**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **程序编号** | **P1** | **文件名称** | **Huffman.m** | **说明** | **题目1哈夫曼编码代码** |
| function Huffman()  clear  close all;  X=imread('../IMAGE/Image4.jpg');  X = rgb2gray(X);  data=uint8(X);  [zipped,info]=huffencode(data);  unzipped=huffdecode(zipped,info);  figure;  subplot(121);imshow(data); title('原图');  subplot(122);imshow(unzipped); title('哈夫曼编码压缩图');  erms=0  info  cr=info.ratio  whos data unzipped zipped  function [zipped,info]=huffencode(vector)  if ~isa(vector,'uint8')  error('input argument must be a uint8 vector');  end  [m,n]=size(vector);  vector=vector(:)';  f=frequency(vector);  simbols=find(f~=0);  f=f(simbols);  [f,sortindex]=sort(f);  simbols=simbols(sortindex);  len=length(simbols);  simbols\_index=num2cell(1:len);  codeword\_tmp=cell(len,1);  while length(f)>1  index1=simbols\_index{1};  index2=simbols\_index{2};  codeword\_tmp(index1)=addnode(codeword\_tmp(index1),uint8(0));  codeword\_tmp(index2)=addnode(codeword\_tmp(index2),uint8(1));  f=[sum(f(1:2)) f(3:end)];  simbols\_index=[{[index1,index2]} simbols\_index(3:end)];  [f,sortindex]=sort(f);  simbols\_index=simbols\_index(sortindex);  end  codeword=cell(256,1);  codeword(simbols)=codeword\_tmp;  len=0;  for index=1:length(vector)  len=len+length(codeword{double(vector(index))+1});  end  string=repmat(uint8(0),1,len);  pointer=1;  for index=1:length(vector)  code=codeword{double(vector(index))+1};  len=length(code);  string(pointer+(0:len-1))=code;  pointer=pointer+len;  end  len=length(string);  pad=8-mod(len,8);  if pad>0  string=[string uint8(zeros(1,pad))];  end  codeword=codeword(simbols);  codelen=zeros(size(codeword));  weights=2.^(0:23);  maxcodelen=0;  for index=1:length(codeword)  len=length(codeword{index});  if len>maxcodelen  maxcodelen=len;  end  if len>0  code=sum(weights(codeword{index}==1));  code=bitset(code,len+1);  codeword{index}=code;  codelen(index)=len;  end  end  codeword=[codeword{:}];  cols=length(string)/8;  string=reshape(string,8,cols);  weights=2.^(0:7);  zipped=uint8(weights\*double(string));  huffcodes=sparse(1,1);  for index=1:nnz(codeword)  huffcodes(codeword(index),1)=simbols(index);  end  info.pad=pad;  info.huffcodes=huffcodes;  info.ratio=cols./length(vector);  info.length=length(vector);  info.maxcodelen=maxcodelen;  info.rows=m;  info.cols=n;  function vector=huffdecode(zipped,info,image)  if ~isa(zipped,'uint8')  error('input argument must be a uint8 vector');  end  len=length(zipped);  string=repmat(uint8(0),1,len.\*8);  bitindex=1:8;  for index=1:len  string(bitindex+8.\*(index-1))=uint8(bitget(zipped(index),bitindex));  end  string=logical(string(:)');  len=length(string);  string((len-info.pad+1):end)=[];  len=length(string);  weights=2.^(0:51);  vector=repmat(uint8(0),1,info.length);  vectorindex=1;  codeindex=1;  code=0;  for index=1:len  code=bitset(code,codeindex,string(index));  codeindex=codeindex+1;  byte=decode(bitset(code,codeindex),info);  if byte>0  vector(vectorindex)=byte-1;  codeindex=1;  code=0;  vectorindex=vectorindex+1;  end  end  vector=reshape(vector,info.rows,info.cols);  function codeword\_new=addnode(codeword\_old,item)  codeword\_new=cell(size(codeword\_old));  for index=1:length(codeword\_old)  codeword\_new{index}=[item codeword\_old{index}];  end  function f=frequency(vector)  if ~isa(vector,'uint8')  error('input argument must be a uint8 vector');  end  f=repmat(0,1,256);  len=length(vector);  for index=0:255  f(index+1)=sum(vector==uint8(index));  end  f=f./len;  function byte=decode(code,info)  byte=info.huffcodes(code); %%程序结束 | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **程序编号** | **P2** | **文件名称** | **RLE.m** | **说明** | **题目1行程编码代码** |
| function RLE()  clear;  close all;  I = imread('../IMAGE/Image4.jpg');  [r\_, c\_, d\_] = size(I);  if d\_ > 1  I0=rgb2gray(I);  else  I0=I;  end  I0=round(I0/20)\*20;  [zipped,info]=RLEncode(I0);%%调用RLE进行编码  unzipped=RLEdecode(zipped,info); %%调用解码程序进行编码  figure;  subplot(121);imshow(I0); title('原图');  subplot(122);imshow(unzipped); title('行程编码压缩图');  %erms=compare(I(:),unzipped(:))  info  cr=info.ratio  whos I unzipped zipped    function[zipped,info]=RLEncode(vector)  [m,n]=size(vector);  vector=vector(:)';  L=length(vector);  c=vector(1);e(1,1)=c;e(1,2)=0;%e(:,1)存放灰度，e(:,2)存放行程  t1=1  for j=1:L  if((vector(j)==c))  e(t1,2)=double(e(t1,2))+1;  else  c=vector(j);  t1=t1+1;  e(t1,1)=c;  e(t1,2)=1;  end  end  zipped=e;  info.rows=m;  info.cols=n;  [m,n]=size(e);  info.ratio=m\*n/(info.rows\*info.cols);    function unzipped=RLEdecode(zip,info)  zip=uint8(zip);  [m,n]=size(zip);  unzipped=[];  for i=1:m  section=repmat(zip(i,1),1,double(zip(i,2)));  unzipped=[unzipped section];  end  unzipped=reshape(unzipped,info.rows,info.cols);%程序结束 | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **程序编号** | **P3** | **文件名称** | **JPEG.m** | **说明** | **题目2主程序代码** |
| clear;  close all;  path1 = '../IMAGE/lena.png';  I0 = imread(path1);  factor = [20, 60, 80];  figure;  subplot(2,2,1); imshow(I0); title('原始图像');  [I1, K1] = jpeg(20);  subplot(2,2,2); imshow(I1); title(['质量因子=', num2str(20)]);  [I2, K2] = jpeg(60);  subplot(2,2,3); imshow(I2); title(['质量因子=', num2str(60)]);  [I3, K3] = jpeg(80);  subplot(2,2,4); imshow(I3); title(['质量因子=', num2str(80)]); | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **程序编号** | **P4** | **文件名称** | **jpeg.m** | **说明** | **题目2 JPEG编码代码** |
| function [Image, k] = jpeg(factor)  path1 = '../IMAGE/lena.png';  path2 = ['../IMAGE/lena\_', num2str(factor), '.jpg'];  I0=imread(path1);  imwrite(I0,path2,'quality', factor); %根据题目要求修改质量因子    I1=imread(path2);  k = imfinfo(path2)  s = k.FileSize/1024;  e = I0(:) - I1(:); %减去原图得到误差  [m, n] = size(e);  rmse = sqrt(sum(e(:)).^2/(m\*n));%求均方根误差  i\_size = k.Width\*k.Height\*k.BitDepth / 8;  i\_compress = k.FileSize; %压缩后图像大小  ratio = i\_size/i\_compress; %压缩比  %imshow(I1);  fprintf('质量因子:%d\n',factor);  fprintf('均方根误差:%f\n',rmse);  fprintf('压缩比：%f\n',ratio);  fprintf('压缩后图像大小：%f\n',i\_compress);  Image = I1; | | | | | |