中国地质大学（武汉）

数字图像处理实验报告

学　　院：自动化学院

课　　程：数字图像处理

指导老师：上官星辰

学　　号：20201000128

班　　级：231202

姓　　名：刘瑾瑾

2023 年 11 月 20 日

目录

一、任务一

1.1 实验要求

读入一幅灰色图像，利用 MATLAB 编程实现采用前值预测（$x_N = a_1 x_{N-1}$）进一阶无损预测编码，设置 a_1=0.8，并显示出解码图像。

1.2 实验原理

预测编码是根据离散信号之间存在着一定关联性的特点，利用前面一个或多个信号预测下一个信号进行，然后对实际值和预测值的差（预测误差）进行编码。如果预测比较准确，误差就会很小。在同等精度要求的条件下，就可以用比较少的比特进行编码，达到压缩数据的目的。

一阶无损预测编码通常应用于灰度图像，依赖于对当前像素值和其邻近像素值之间关系的建模,并基于对图像中像素值的预测。

1.3 实验结果

原始灰度图像与解码灰度图如图 1 所示，通过原图像和解码图的直方图看以看出图像数据得到了压缩，预测误差趋近于零，压缩效果良好。

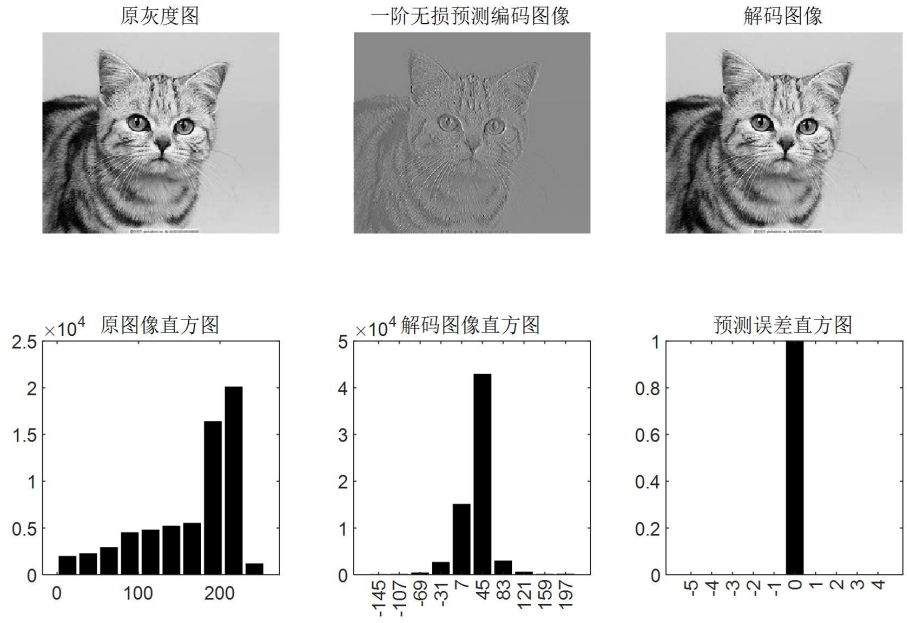

图 1 任务一实验结果

二、任务二

2.1 实验要求

设有一图像（256 灰度级）分成了很多 8*8 的不重叠的像素块，其中一个亮度数据块如下，请将其进行 JPEG 编码。

62	51	61	64	77	61	64	73
63	55	66	93	107	85	68	75
62	58	68	119	149	100	69	74
73	68	61	126	150	116	72	71
47	71	60	114	128	90	70	73
65	75	64	70	75	62	51	74
85	70	60	52	55	68	69	86
78	71	65	68	61	76	70	99

2.2 实验原理

JPEG 编码是一种广泛用于图像压缩的标准，包括图像编码和解码过程以及压缩图像数据的编码表示。JPEG 图像压缩主要包括离散余弦变换（DCT）、量化、熵编码等步骤，通过去除图像的冗余信息来减小图像文件的大小。

JPEG 压缩算法分为两大类:无失真压缩和有失真压缩。使用无失真压缩算法将源图像数据转变为压缩数据，该压缩数据经对应的解压缩算法处理后可获得与源图像完全一致的重建图像。有失真压缩算法基于离散余弦变换，所生成的压缩图像数据经解压缩生成的重建图像与源图像在视觉上保持一致。

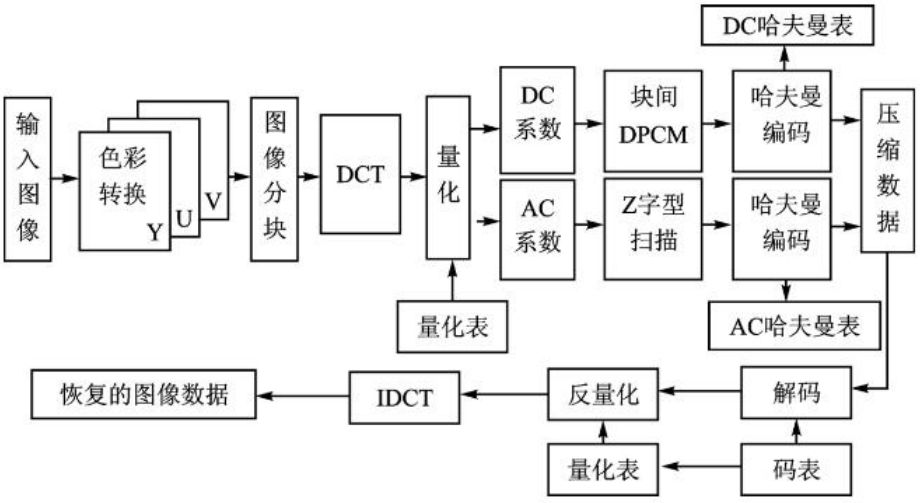

图 2 JPEG 基本系统的编解码方框图

2.3 实验结果

对原始数据进行 DCT 编码后量化，量化结果和 Z 字形扫描结果如下：

Quantized Data:

-26	-3	-6	1	2	-1	0	0
1	-1	-4	1	1	0	0	0
-4	1	5	-1	-1	0	0	0
-4	1	1	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

zigzag_A =

-26	1	-3	-6	-1	-4	-4	1
-4	1	2	1	5	1	1	0
0	1	-1	1	-1	0	0	-1
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

对量化后结果进行哈夫曼编码，编码结果，即 JPEG 编码结果如下：

Ecoded Data:

行 1至15														
1	0	0	0	1	1	0	1	0	0	1	0	1	0	1
行 16 至 30														
0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
行 31 至 45														
1	0	0	0	0	0	1	1	0	0	0	0	0	0	1
行 46 至 60														
0	1	0	0	0	0	0	1	0	0	0	1	0	1	1
行 61 至 75														
1	1	0	0	0	0	0	1	0	1	1	0	0	0	0
行 76 至 90														
1	1	0	1	0	0	0	0	0	1	0	0	1	1	1
行 91 至 105														
0	0	0	0	0	0	1	0	0	1	1	1	0	0	0
行 106 至 119														
0	0	0	1	1	0	1	0	1	0	0	0	0	0	

进行解码、反量化、反 DCT 变换后，数据如下所示，与原始数据相比有所变换，原因在于 DCT 变换属于有失真压缩算法，损失部分信息。

Reconstructed Data:

56.2186	64.3558	60.8529	64.0703	72.1647	58.0889	52.6161	75.2606
59.5110	59.7509	60.9775	84.5473	105.9453	84.0953	61.3407	73.0857
66.4204	58.4351	66.0538	112.0635	145.3877	112.3370	71.8409	74.8711
70.1181	61.0740	72.2289	123.3741	154.5376	114.2206	71.9971	78.9268
64.7306	62.3020	70.7584	105.7154	123.5691	86.7647	59.5592	78.4583
59.1098	65.3556	64.9117	73.9515	81.0559	59.9786	51.7464	77.0949
65.5688	77.3542	65.7145	53.2791	60.1635	60.4032	63.7045	83.9444
77.5428	91.1640	71.8173	48.6817	60.2809	75.7400	82.4600	94.3504

三、任务三

3.1 实验要求

读入一幅带有椒盐噪声的图像，使用 MATLAB 编程进行开和闭运算，要求：

（1）用二阶单位矩阵的结构元素进行开、闭运算；

（2）用半径为 1 的平坦圆盘形结构元素进行开、闭运算；

（3）使用开和闭运算实现形态学滤波；

（4）显示所有开、闭运算和形态学滤波的结果。

3.2 实验原理

开运算和闭运算是形态学图像处理中的两种基本操作，它们通常用于改善或修改图像中的形状。

开运算为先腐蚀后膨胀，对于图像 X 及结构元素 S，用符号 X○S 表示 S 对图像 X 作开运算。

$$XOS = (X \quad \Theta S) \oplus S$$

闭运算为先膨胀后腐蚀，对于图像 X 及结构元素 S，用符号 X·S 表示 S 对图像 X 作开运算。

$$X \cdot S = (X \oplus S)\Theta S$$

由于开、闭运算所处理的信息分别与图像的凸、凹处相关，因此，它们本身都是单边算子，可以利用开、闭运算去除图像的噪声、恢复图像，也可交替使用开、闭运算以达到双边滤波目的。一般，可以将开、闭运算结合起来构成形态学噪声滤波器，例如(XOS)·S 或(X·S)OS 等。

整个过程是先做开运算再做闭运算，可以写为：

$$\{[(X\Theta S) \oplus S] \oplus S\}\Theta S = (X \ \mathrm{O} \ S) \cdot S$$

在一般情况下，噪声往往由信号上下凸起的尖峰组成。只要这些噪声是很好分离的,则可以利用开运算和闭运算的迭代运算或闭运算和开运算的迭代运算将其消除。从统计学角度看,以开运算作为滤波器,存在这样的问题:除非噪声图像位于非噪声图像的上方，例如存在极大噪声的情况;否则，滤波器的输出将会产生偏移现象。这是因为做过开运算的图像总是位于噪声图像下方的缘故，闭运算也存在同样的问题。使用迭代运算的目的之一就是要减弱这些偏移现象。

3.3 实验结果

如图 3 所示，添加椒盐噪声后，图像出现许多黑色和白色的点，这些点分别代表极端暗度值（黑色点）和极端亮度值（白色点）。

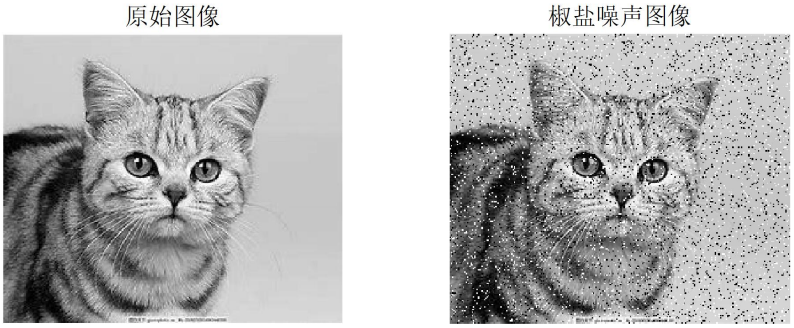

图 3 增加椒盐噪声的图像

对椒盐噪声图像分别使用二阶单位矩阵的结构元素和半径为 1 的平坦圆盘形结构元素分别进行开运算、闭运算和形态学滤波，结果如图 4 所示。进行开运算后，图像中黑色噪声点增加；进行闭运算后，图像中白色噪声点增加。进行形态学滤波后的图像与原始灰度图基本相似，清晰度稍有降低。

图 4 任务三实验结果

四、任务四

4.1 实验要求

读入一幅 RGB 图像，使用 MATLAB 编程实现最大信息熵阈值分割，并显示图像分割处理后的结果。

4.2 实验原理

一维最大熵阈值分割。目标区域 O 的概率分布和背景区域 B 的概率分布分别是：

$$P_O = p_i/p_t \qquad i = 0,1, \ldots, t$$

背景区域 B 的概率灰度分布为：

$$P_B = p_i/(1 - p_t) \qquad i = t+1, t+2, \ldots, L-1$$

式中：$p_t = \sum_{i=0}^{t} p_i$，目标区域熵的定义为$H_O(t) = -\sum_{i=0}^{t} P_O \log_2 P_O$；背景区域熵的定义$H_B(t) = -\sum_{i=t+1}^{L-1} P_B \log_2 P_B$

由目标区域和背景区域熵、得到熵函数定义为：

$$\phi(t) = H_0 + H_B$$

当熵函数$\phi(t)$取得最大值时，对应的灰度值就是所求的最佳阈值：

$$t^* = \max_{0<t<L-1}[\phi(t)]$$

当求取的熵最大时，从图像中获得最大信息量。

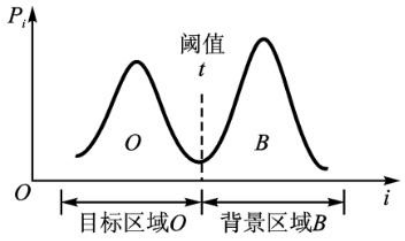

图 5 一维最大熵阈值分割

4.3 实验结果

原始图像与最大信息熵分割结果如图 6 所示，最大信息熵阈值分割将图像边沿很好的显示了出来。

原始图像

最大信息熵图像分割结果

图 6 任务四实验结果

五、实验代码

```
%% 任务一
%读入一幅灰色图像，利用 MATLAB 编程实现采用前值预测进一阶无损预测编码，设置
a1=0.8，并显示出解码图像。
I=imread('gray.jpg');
x=double(I);
%编码
y=LPCencode(x,0.8);%设置 a=0.8
%解码
xx=LPCdecode(y,0.8);
%显示结果
figure(1);
subplot(2,3,1);
imshow(I);
title('原灰度图');
subplot(2,3,2);
imshow(mat2gray(y));
title('一阶无损预测编码图像');
subplot(2,3,3);
imshow(mat2gray(xx));
title('解码图像');
%计算误差
e=double(x)-double(xx);
[m, n]=size(e);
erms=sqrt(sum(e(:).^2)/(m*n))
 %显示直方图
subplot(2,3,4);
[h, f]=hist(x(:));
bar(f, h, 'k');
title('原图像直方图');
subplot(2,3,5);
[h, f]=hist(y(:));
bar(f, h,'k');
title('解码图像直方图');
subplot(2,3,6);
[h, f]=hist(erms(:));
bar(f, h,'k');
title('预测误差直方图');
%% 任务二
%设有一图像（256 灰度级）分成了很多 8*8 的不重叠的像素块，其中一个亮度数据块如
下，请将其进行 JPEG 编码。
```

```
clc;clear;
% 给定的亮度数据块
data = [
    62, 51, 61, 64, 77, 61, 64, 73;
    63, 55, 66, 93, 107, 85, 68, 75;
    62, 58, 68, 119, 149, 100, 69, 74;
    73, 68, 61, 126, 150, 116, 72, 71;
    47, 71, 60, 114, 128, 90, 70, 73;
    65, 75, 64, 70, 75, 62, 51, 74;
    85, 70, 60, 52, 55, 68, 69, 86;
    78, 71, 65, 68, 61, 76, 70, 99
];
% 归一化至 [0, 255]
normalized_data = data - 128;
% DCT 变换
dct_data = dct2(normalized_data);
% 量化
quantization_matrix = [
    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
];
quantized_data = round(dct_data ./ quantization_matrix);
% 输出量化后的系数
disp('Quantized Data:');
disp(quantized_data);
% 哈夫曼编码
[M, N] = size(quantized_data);
I1 = quantized_data(:);
P=zeros(1,31);
%获取各符号的概率
for i = 0:31
    P(i+1) = length(find(I1 == (i-26)))/(M*N);
end
k=-26:5;
% Huffman 编码
dict = huffmandict(k,P); %根据灰度级 k 和概率数组 P 生成 Huffman 字典
enco = huffmanenco(I1,dict); %哈夫曼编码
% 检查解码后的数据
```

```
disp('Ecoded Data:');
disp(enco');
deco = huffmandeco(enco,dict); %哈夫曼解码
Ide = col2im(deco,[M,N],[M,N],'distinct'); %把向量重新转换成图像块
% 逆量化
dequantized_data = Ide .* quantization_matrix;
% 逆 DCT 变换
inverse_dct_data = idct2(dequantized_data);
% 反归一化
reconstructed_data = inverse_dct_data + 128;
% 输出重建的数据
disp('Reconstructed Data:');
disp(reconstructed_data);
%% 任务三
%读入一幅带有椒盐噪声的图像，使用 MATLAB 编程进行开和闭运算
clc;clear;
grayImage = imread('gray.jpg');
% 加入噪声密度为 0.1 的椒盐噪声
saltNoise = imnoise(grayImage, 'salt & pepper', 0.1);
%1)用二阶单位矩阵的结构元素进行开、闭运算
se1=strel([1,0;0,1]);
open1=imopen(saltNoise,se1);
close1=imclose(saltNoise,se1);
%2）用半径为 1 的平坦圆盘形结构元素进行开、闭运算
se2=strel('disk',1);
open2=imopen(saltNoise,se2);
close2=imclose(saltNoise,se2);
%3）使用开和闭运算实现形态学滤波
morphFiltered1=imclose(imopen(grayImage, se1), se1);
morphFiltered2=imclose(imopen(grayImage, se2), se2);
% 显示图像
figure(2);
subplot(1, 2, 1);
imshow(grayImage);
title('原始图像');
subplot(1, 2, 2);
imshow(saltNoise);
title('椒盐噪声图像');
figure(3);
subplot(2, 3, 1);
imshow(open1);
title('二阶单位矩阵开运算');
subplot(2, 3, 2);
imshow(close1);
```

```
title('二阶单位矩阵闭运算');
subplot(2, 3, 3);
imshow(morphFiltered1);
title('二阶单位矩阵形态学滤波');
subplot(2, 3, 4);
imshow(open2);
title('平坦圆盘形结构开运算');
subplot(2, 3, 5);
imshow(close2);
title('平坦圆盘形结构闭运算');
subplot(2, 3, 6);
imshow(morphFiltered2);
title('平坦圆盘形结构形态学滤波');
%% 任务四
%读入一幅 RGB 图像，使用 MATLAB 编程实现最大信息熵阈值分割，并显示图像分割处理后的结果
clc;clear;
a=imread('cat.jpg');
figure(4);
subplot(1,2,1);
imshow(a);
title('原始图像');
count=imhist(a);
[m,n]=size(a);
N=m*n;
L=256;
count=count/N;%%每一个像素的分布概率
for i=1:L
    if count(i)~=0
        st=i-1;
        break;
    end
end
for i=L:-1:1
    if count(i)~=0
        nd=i-1;
        break;
    end
end
f=count(st+1:nd+1);  %f 是每个灰度出现的概率
size(f)
E=[];
for Th=st:nd-1   %%%设定初始分割阈值为 Th
av1=0;
```

```
av2=0;
Pth=sum(count(1:Th+1));
%%%第一类的平均相对熵为
for i=0:Th
    av1=av1-count(i+1)/Pth*log(count(i+1)/Pth+0.00001);
end
%%%第二类的平均相对熵为
for i=Th+1:L-1
    av2=av2-count(i+1)/(1-Pth)*log(count(i+1)/(1-Pth)+0.00001);
end
E(Th-st+1)=av1+av2;
end
position=find(E==(max(E)));
th=st+position-1;
 for i=1:m
    for j=1:n
        if a(i,j)>th
            a(i,j)=255;
        else
            a(i,j)=0;
        end
    end
end
subplot(1,2,2);
imshow(a);
title('最大信息熵图像分割结果');
function y = LPCencode(x, f)
%LPCencode 函数用一维无损预测编码压缩图像 x，f 为预测系数，如果 f 默认，则 f=1，
%就是前值预测。

error(nargchk(1, 2, nargin))
if nargin < 2
    f = 1;
end
x = double(x);
[m, n] = size(x);
p = zeros(m, n);    %存放预测值
xs = x;
zc = zeros(m, 1);
for j = 1: length(f)
    xs = [zc xs(:, 1: end-1)];
    p = p + f(j) * xs;
end
y = x - round(p);
```

```
end
function x = LPCdecode(y, f)
%LPCdecode 函数是解码程序，与编码程序用的是一个预测器。
error(nargchk(1, 2, nargin));
if nargin < 2
    f = 1;
end
f = f(end: -1: 1);
[m, n] = size(y);
order = length(f);
f = repmat(f, m, 1);
x = zeros(m, n + order);
for j = 1: n
    jj = j + order;
    x(:, jj) = y(:, j) + round(sum(f(:, order: -1: 1).* x(:,
(jj-1):-1:(jj-order)), 2));
end
x = x(:, order + 1: end);
end
```