Xi'an Jiaotong-Liverpool University
西交利物浦大学

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE330

Image compression

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Engineering

Student Name : Xiangxing. Li
Student ID : 1510108

Contents

Chapter 1

Task 1

By using function imshow(I,[]), we can see the profile of the image more clearly. imshow(I,[]) displays the grayscale image I, scaling the display based on the range of pixel values in I. The output of function e(r,c) is shown below.

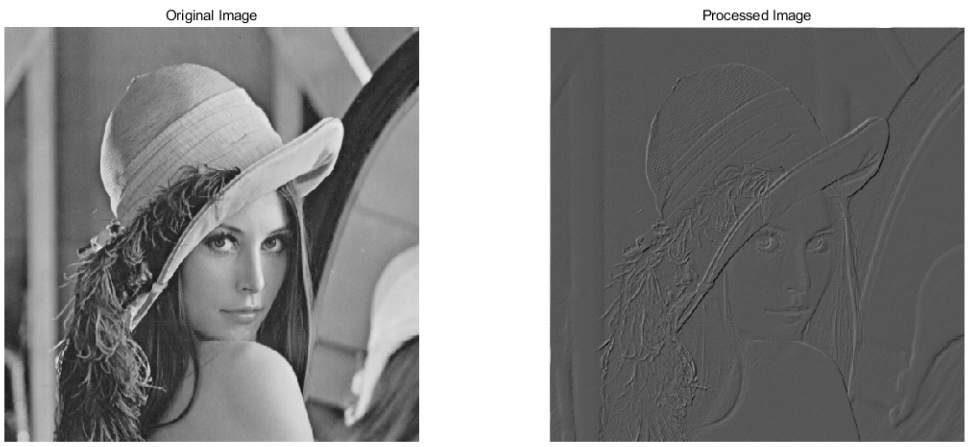

Figure 1.1: Original and Processed image

1.0.1 Raster-Scan DPCM Function

The running code is shown below.

```
%Task1
im = imread('lenna512.bmp','bmp');
%(1)error function
[im_e_DPCM] = Raster_Scan_DPCM(im);
subplot(1,2,1);
imshow(im);title('Original Image');
```

```
subplot(1,2,2);
imshow(im_e_DPCM,[]); title('Processed Image');

%(2)Entropy
E_in  = entropy(im)
E_out = entropy(uint8(im_e_DPCM))
```

The Raster-scan DPCM function code is shown below.

```
function [e,p] = Raster_Scan_DPCM(im)
[r,c] = size(im);
im_d = double(im);
p = zeros(r, c);
e = zeros(r, c);
for i = 2:r
for j = 2:c
p(i,j) = (im_d(i,j-1)+im_d(i-1,j-1)+im_d(i-1,j))./3;
end
end
%Calculate difference
for i = 1:r
for j = 1:c
e(i,j) = im_d(i,j) - p(i,j);
end
end
end
```

1.0.2 Entropy Function

The output image is easier to be compressed since it has the smaller entropy. Since the larger entropy, the more information the image contains, which will make it more difficult to compress

$$EntropyOfInputImage"im" = 7.3775 \tag{1.1}$$

$$EntropyOfOutputImage"im" = 2.7860 \tag{1.2}$$

The function code which calculates entropy is shown below.

```
function [en] = entropy(im)
count=imhist(im);
[r,c]=size(im)
en=0;%en is entropy
for i=1:256
p(i)=count(i)/(r*c);
if p(i)~=0
en=en-p(i)*log2(p(i));
end
```

```
end
end
```

Chapter 2

Task 2

2.0.1 2DCT Function

The original image has 512*512 pixels and the processed image has 64*64 pixels.

Figure 2.1: Compressed image

The running code is shown below.

```
im = imread('lenna512.bmp','bmp');
ims = DCT2_8x8(im);
imshow(ims,[]);
title('Lenna8*8')
```

The function code which calculates realize 2DCT is shown below.

```
%Task2_1
function [im_64]=DCT2(im)
[r, c] = size(im);
im_64 = zeros(64,64);
```

```
im_dct2 = zeros(r,c);
for i=1:8:r
for j=1:8:c
im_64 =dct2(im(i:i+7,j:j+7));
im_dct2(i,j)=im_64(1,1);
end
end
%Recommbination
for i=1:64
for j=1:64
im_64(i,j)=im_dct2(8*i-7,8*j-7);
end
end

end
```

2.0.2 2DCT with Quantization

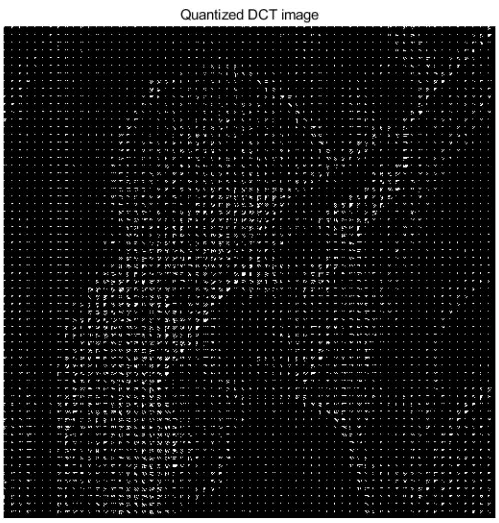

Figure 2.2: Quantized image with 2DCT

The running code is shown below.

```
im =imread('lenna512.bmp');
quantized_image =quan(im,30);
figure(1)
imshow(quantized_image);
title('Quantized DCT image');
```

The function code for quantization is shown below.

```
function [quan]= quantize(im,QP)
[r, c] = size(im);
%Write Q matrix
Q=[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;];
%Decide S value according to QP
if (QP>50)
S= (100-QP)/50;
elseif (QP<=50)
S= 50/QP;
end
%Get quantized result
for i=1:8:r
for j=1:8:c
im64 = dct2(im(i:i+7,j:j+7));
ims = round(im64./(S*Q));
quan(i:i+7,j:j+7) = ims;
end
end
quan;
end
```

2.0.3 Image Depression & PSNR

The main figure results and code is shown below. The PSNR are

```
SNR_ouput =

17.2018    31.9226    34.1539    35.3327    36.2212    37.3853    39.4034
55.2879
```

which corresponds to QP = 1:14:99.

Figure 2.3: Original image and Depressed image

```
%Task2_3_4
im = imread('lenna512.bmp');
QP = 30;
quan_result = quan(im,QP);
imo = decompress(quan_result,QP);
PSNR = psnr(im,uint8(imo));
PSNR_i=zeros(1,8);
j=1;

for QP_i = 1:14:99
quan_result = quan(im,QP_i);
imo = decompress(quan_result,QP_i);
PSNR_i(j) = psnr(im,uint8(imo));
j=j+1;
end
SNR_ouput=PSNR_i

figure(1)
subplot(1,2,1);imshow(im);title('Original Image');
subplot(1,2,2);imshow(imo,[]);title('Processed Image');
```

Where the function code for image depression is shown below.

```
%Task2_3_4
function [decom]= decompress(quantized,QP)
Q=[
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;
];
if (QP>50)
S= (100-QP)/50;
elseif (QP<=50)
S= 50/QP;
end
[r,c]=size(quantized);
for i=1:8:r
for j=1:8:c
decomp = quantized(i:i+7,j:j+7).*(S*Q);
decom(i:i+7,j:j+7) = idct2(decomp);
end
end
end
```