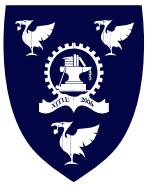

Xi'an Jiaotong-Liverpool University

西交利物浦大學

Image Processing EEE412

Lab 4：Image compression

XUN XU 1926930

2019/11/23

Task1

1.1

Code:

```
function E = entropy(I)


p = imhist(I(:));% calculate histogram counts


p(p==0) = [];% remove zero entries in p


p = p ./ numel(I);% normalize p so that sum(p) is one.


E = -sum(p.*log2(p));


end
```

The formula used to calculate the entropy of a picture is:

$$H(X) = \sum_{i=1}^{M} p_i * log_2 \frac{1}{p_i}$$

In order to avoid too many for loops, the image is first for histogram, the probability of these points appearing in the picture is calculated, and then the entropy sum of all points is calculated by the formula.

1.2

To avoid excessive for loops, block handle functions are used here.

Code:

```
%calculate the entropy of the original image
im=imread('lenna512.bmp');
outcome1=entropy(im);
disp('original image');
disp(outcome1);

%calculate the entropy of the image reduced the to the half size
fun = @(block_struct)mean(block_struct.data,'all');
img2= blockproc(im,[2 2],fun);
outcome2=entropy(uint8(img2));
disp('image reduced the to the half size');
disp(outcome2);

%calculate the entropy of the image reduced the gray level of “lenna512.bmp” to 16 values
fun = @(block_struct)block_struct.data/16;
img3= blockproc(im,[1 1],fun);
outcome3=entropy(img3);
disp('image reduced the gray level of “lenna512.bmp” to 16 values ');
disp(outcome3);
```

Result:

```
original image

    7.3775


image reduced the to the half size

    7.4231


image reduced the gray level of “lenna512.bmp” to 16 values

    3.4846
```

Figure1: Entropy of different pictures

Results analysis:

From the outcome, entropy of the image reduced the to the half size is almost the same with the outcome of the original image.The entropy of the image reduced the gray level to 16 values is much smaller than the original image.

This is because reducing the dimensions (size) of the image does not reduce the amount of information contained in the image, but reducing the pixel level to 16 greatly reduces the amount of information contained in the image.

1.3

Code:

```
%Task1_3
```

```
im = imread('lenna512.bmp','bmp');
%error function
[im_DPCM,k] = Raster_Scan_DPCM(im);
subplot(1,2,1);
imshow(im);title('Original Image');
subplot(1,2,2);
imshow(im_DPCM,[]);title('DPCM Image');
%Entropy
outcome1=entropy(im);
outcome2=entropy(uint8(im_DPCM));
disp(outcome1);
disp(outcome2);
```

```
function [e,p] = Raster_Scan_DPCM(image)
%use the same value as its neighbor to pad the left and up side of the image
im= padarray(image,[1 1],'replicate','pre');
[r,c] = size(im);
im_d = double(im);
p=zeros(r-1, c-1);
e=zeros(r-1, c-1);
%Calculate the predicted value
p=(2*im_d(2:r,1:(c-1))+im_d(1:(r-1),1:(c-1))+2*im_d(1:(r-1),2:c))/5;
%Calculate difference
e = double(image)- p;
end
```

7.3775

2.6890

Figure2:entropy of original image and the DPCM image

Results analysis:

Obviously, the original image is easier to compress. The image entropy value processed by DPCM is much smaller than the original image. The smaller the entropy value is, the less information the image contains. Compression means to reduce the information, the original image contains more information, can be compressed by the original space is larger.

Task2:

2.a

Code:

```
%Task2_a
```

```
im = imread('lenna512.bmp','bmp');

ims = DCT2(im);

subplot(121);

imshow(im);

title('Original Image 512*512');

subplot(122);

% imshow(ims,[]);

imshow(ims,[]);

title('Compressed image 64*64');
```

```
function[img_DCT]=DCT2(im)

fun = @(block_struct)dct2(block_struct.data);

imgDCT2= blockproc(im,[8 8],fun);

fun = @(block_struct)block_struct.data(1,1);

img_DCT= blockproc(imgDCT2,[8 8],fun);

end
```

Result:

Figure3: Original image and DCT transformed image

Results analysis:

Each 8*8 block in the original image is processed with DCT, and then the upper left corner elements of each block are taken out respectively to form an image. As can be seen from figure3, the rearranged image is consistent with the original image as a whole. Due to the reduction of pixels, the clarity of the image decreases a lot. In general, it is a compression of the original image. The DCT processing process roughly retains the dc information in the image, while the ac component is basically removed to retain the main information and compress the image. The rearrangement is the recombination of these dc components, so as to obtain the results we have seen.

2.b

Code:

```
function [quan]= quantize(im,QP)
%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
%block processing used to quantize
fun = @(block_struct)dct2(block_struct.data);
imgDCT2= blockproc(im,[8 8],fun);
fun = @(block_struct)round(block_struct.data./(S*Q));
quan= blockproc(imgDCT2,[8 8],fun);
end
```

Result:

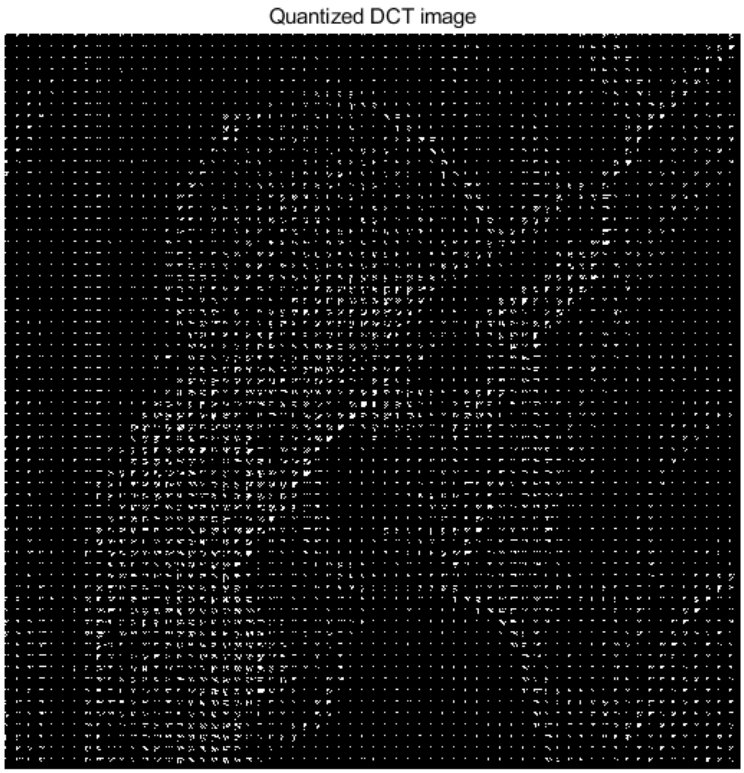

figure4: Quantized image

We can see that the quantified image turns into a bunch of white highlights

2.c&2.d

Code:

```
%Task2_d

im = imread('lenna512.bmp');

QP = 20;

quan_result = quantize(im,QP);

imdecompress = decompress(quan_result,QP);

PSNR = psnr(im,uint8(imdecompress));

PSNR_i=zeros(1,8);

j=1;
```

```
for QP = 1:14:99
    quan_result = quantize(im,QP);
    imdecompress = decompress(quan_result,QP);
    PSNR_i(j) = psnr(im,uint8(imdecompress));
    j=j+1;
end
PSNR_out=PSNR_i;
disp(PSNR_out);
```

```
%Task2_c
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
%decompress process
fun = @(block_struct)block_struct.data.*(S*Q);
imgDCT2= blockproc(quantized,[8 8],fun);
```

```
fun = @(block_struct)idct2(block_struct.data);

decom= blockproc(imgDCT2,[8 8],fun);

end
```

QP	1	15	29	43	57	74	85	99
PSNR (dB)	17.2018	31.9226	34.1539	35.3327	36.2212	37.3853	39.4043	55.2879

Table: PSNR results

Results analysis:

As can be seen from the results, the larger the QP value is, the larger the PSNR value is. As the value of QP increases, the value of S keeps decreasing. In the program, the smaller S is, the larger the result is, and the more weight obtained after round is retained. If S is large, the result after round is close to 0. So the result is easy to be ignored. The more weight is retained, the closer the image is to the original