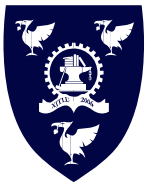

Xi'an Jiaotong-Liverpool University

西交利物浦大學

Image Processing EEE412

Lab 1

XUN XU

2019/10/8

Task1

(1)

Experimental result:

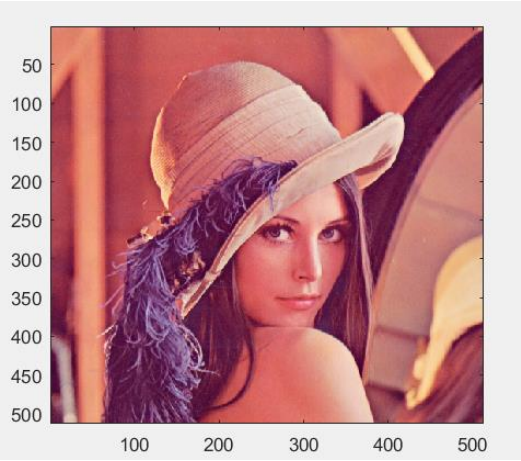

Figure 1: image displayed by image function

Figure 2: image of RGB component r

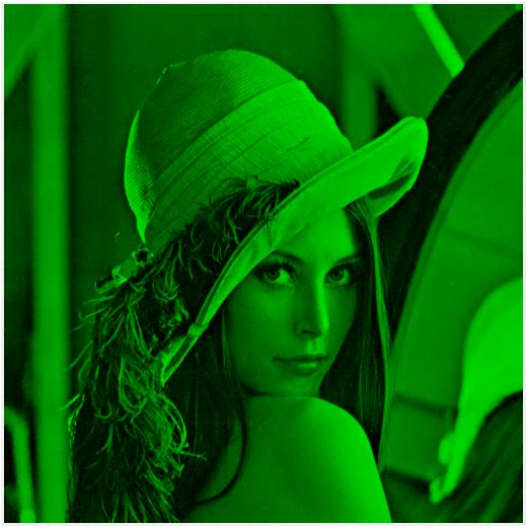

Figure 3: image of RGB component g

Figure 4: image of RGB component b

Code:

```
a=imread('C:\Users\xuxun\Desktop\image\Lab1-Materials\lenna512color.bmp');
figure;
image(a); %image function display image
rgb = im2double(a);
r = a(:, :, 1);                 % Extract r component
g = a(:, :, 2);                 % Extract g component
b = a(:, :, 3);                 % Extract b component
z=zeros(512,512);
m=cat(3,r,z,z);                 % Extract only the r component to the new blank image
n=cat(3,z,g,z);                 % Extract only the r component to the new blank image
p=cat(3,z,z,b);                 % Extract only the r component to the new blank image
figure;
imshow(m);
figure;
imshow(n);
figure;
imshow(p);
```

(2)

The formula for converting rgb to hsi is as follows：

$$\theta = \cos^{-1}\left\{\frac{[(R-G)+(R-B)]/2}{\sqrt{(R-G)^2+(R-B)(R+B)}}\right\}$$

$$H = \begin{cases} \theta & B \le G \\ 2\pi - \theta & B > G \end{cases}$$

$$S = 1 - \frac{3 * \min(R, G, B)}{R + G + B}$$

$$I = (R + G + B)/3$$

Experimental result:

Figure 5: image of rgb converted to hsi

Figure 6: image of h component of hsi

Figure 7: image s component of hsi image

Figure 8: image of i component of hsi image

Code:

```
  [hsi,H,S,I ] = rgb2hsi(a);          % extracts the his image and the three components of his
figure;
imshow(hsi);
figure;
imshow(H);
figure;
imshow(S);
figure;
imshow(I);
```

the function calculates rgb conversion to his:

```
function [hsi,H,S,I] = rgb2hsi( kkkk )
rgb = im2double(kkkk);
```

```
r = rgb(:, :, 1);                    % extract rgb image and three components
g = rgb(:, :, 2);
b = rgb(:, :, 3);
num = 0.5*((r - g) + (r - b));    % Perform conversion equation
den = sqrt((r - g).^2 + (r - b).*(g - b));
theta = acos(num./(den + eps));
H = theta; H(b > g) = 2*pi - H(b > g); H = H/(2*pi);
num = min(min(r, g), b);
den = r + g + b;
den(den == 0) = eps;
S = 1 - 3.* num./den;              %Calculate the three components of hsi
H(S == 0) = 0;
I = (r + g + b)/3;
hsi= cat(3, H, S, I);                % Combine 3 components into one HSI image
end
```

(3)

Experimental result:

Figure 9: generated grey image using matlab's own functions

Code:

```
greyimg=rgb2gray(a);            % converts rgb images to grayscale images
figure;
imshow(greyimg);
imwrite(greyimg,'C:\Users\xuxun\Desktop\image\Lab1-Materials\mm.bmp','bmp');
```

(4)

Experimental result:

Figure 10: Binary image generated using matlab's own functions

Code:

```
binary_img = im2bw(a,0.5);    % Convert image a to binarized image, 0.5 represents
                               %threshold level for black and white recognition of pixel points
figure;
imshow(binary_img);
```

(5)

There are many standards for color display. RGB and HSI are our common color standards, and RGB contains almost all kinds of colors. It is more visually biased towards us human beings. We humans can only identify thousands of them. Colors of RGB standard , but its color features include other parameters, where color = brightness + chromaticity, and chromaticity = hue (color category) + saturation (color depth), which is not conducive to the processing of the computer. HSI extracts several parameters of the color separately, which is convenient for computer calculation. When manufacturing a product, we can convert it to RGB standard for our naked eye recognition.

Task2:

The calculation formula of Peak Signal to Noise Ratio (PSNR) is as follows:

$$PSNR(dB) = 10log_{10}(\frac{255\text{^}2}{mse})$$

$$mse = \frac{1}{N}\sum_{\forall ri}\sum_{\forall ci}\left(im(ri,ci) - im_2(ri,ci)\right)^2$$

Code:

```
function [psnr]=problem2m (im1,im2)
im1=im2double ( im1 ) ;
im2=im2double ( im2 ) ;
[ r1 , c1]= size ( im1 ) ;
[ r2 , c2]= size ( im2 ) ;
 p=0;
 if ( r1==r2&&c1==c2)
       for i =1:r1
             for j =1: c1
                   p=p+(im1(i ,j)-im2(i ,j))^2 ;
             end
       end
 else
       error('im1 and im2 must have same size');
 end
mse=p/(r1*c1) ;
psnr=10*log10( 255^2 /mse ) ;
end
```

Task3

A

Experimental result:

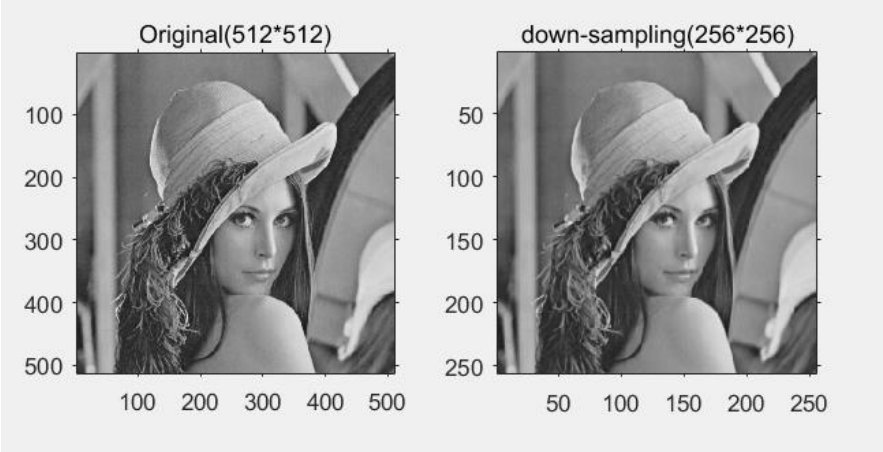

Figure 11: Image after down-sampling

Code idea:

Since the internal points are different from the surrounding points of the surrounding boundary points, I have adopted different processing methods. For the surrounding boundary points, because there are no other reference points around them to provide more accurate sampling information, I choose to keep their own grayscale value, while the inner point selects the nearest four neighbour points to take the mean value. Because the closer to the center point, their gray value has a greater impact on the gray value of the center point.

Outcome explanation：

It can be found that the downsampled image shows a certain degree of distortion.

Code:

```
xx=imread('C:\Users\xuxun\Desktop\image\Lab1-Materials\lenna512.bmp');
[r1,c1]=size(xx); % take image row and column values
img=im2double(xx);
%(a)Down_Sampling mean value
for i=1:2:r1;
    for j=1:2:c1;
        if(i>1&&j>1)
            img2(i,j)=(img(i-1,j)+img(i,j+1)+img(i+1,j)+img(i,j-1))/4;
        else
            img2(i,j)=img(i,j);
        end

    end
end
img2=img2(1:2:end,1:2:end);
```

```
[r2,c2]=size(img2);
figure;
subplot(121);
imshow(img);
axis on;
title(['Original','(',num2str(r),'*',num2str(c) ,')']);
subplot(122);
imshow (img2);
axis on;
title(['down-sampling' ,'(',num2str(r2),'*' ,num2str( c2),')']);
```

B

Experimental result:

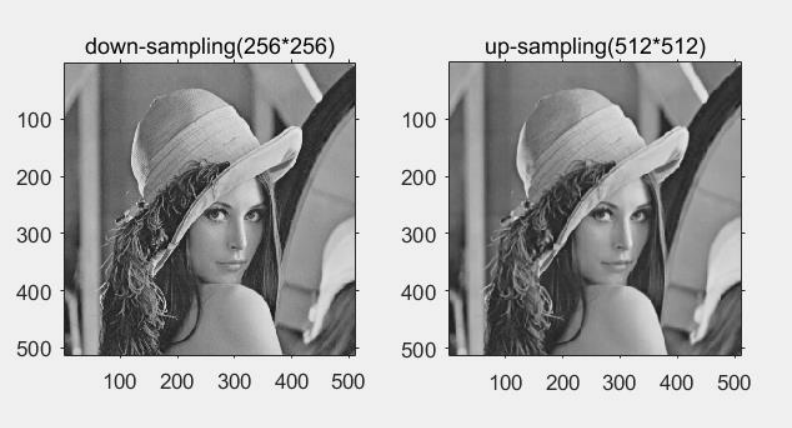

Figure 12: Image after Up-sampling:nearest neighbor interpolation

Code idea:

Here the up-round function is used to upsample the added points.

Outcome explanation：

It can be seen that the upsampled image shows a certain degree of enhancement.

Code:

```
T=2; % Define image sampling multiples
img3=zeros(T*r2,T*c2);

for i=1:r2*T;
      for j=1:c2*T;
            m=ceil(i/T);
            n=ceil(j/T);
            img3(i,j)=img2(m,n);
      end
end
[r3,c3]=size(img3);
figure;
subplot(1,2,1);
```

```
imshow (img);
axis on;
title(['down-sampling' ,'(',num2str(r2),'*' ,num2str( c2),')']);
subplot(1,2,2);
imshow(img3);
axis on;
title(['up-sampling','(',num2str(r3),'*',num2str(c3),')']);
```

C

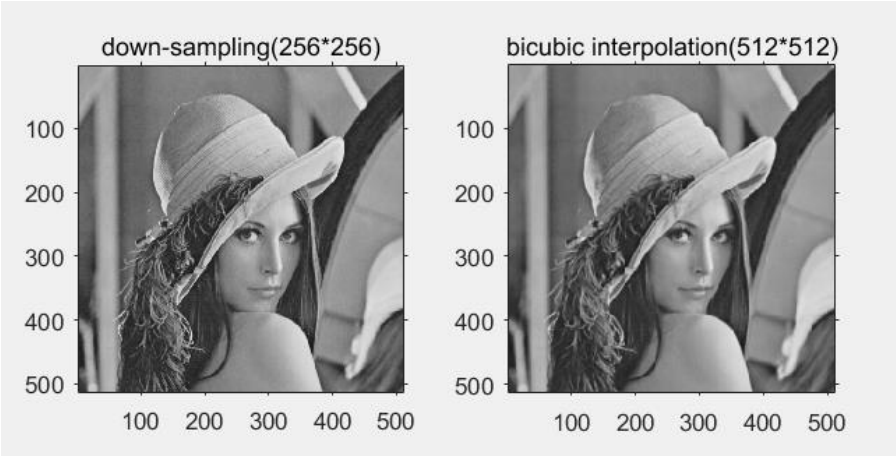

Figure 13:Image after bicubic interpolation

We can see that the image after bicubic interpolation has hardly changed

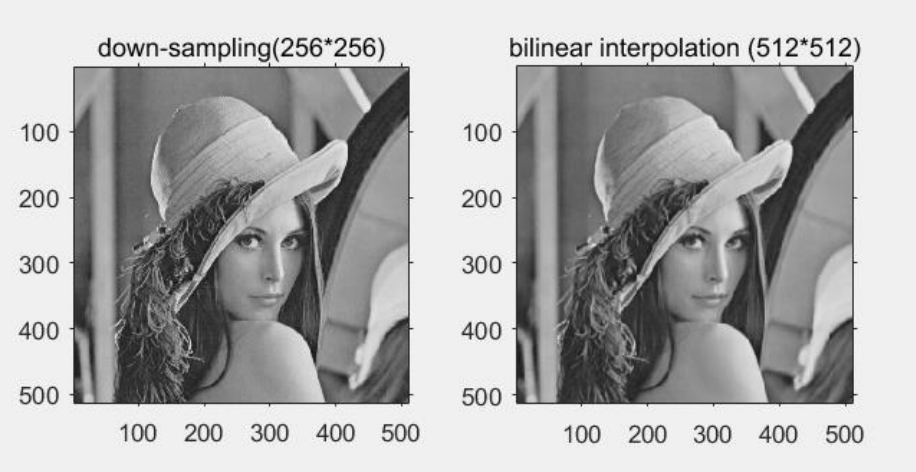

Figure 14:Image after bilinear interpolation

We can see that the image after bilinear interpolation has hardly changed

Code:

```
img4=imresize(img2,2,'nearest');% bilinear interpolation
img5=imresize(img2,2,'bicubic' );% bicubic interpolation
[r4,c4]= size(img4);
[r5,c5]= size(img5);

figure;
subplot(1,2,1);
imshow (img);
axis on;
title(['down-sampling' ,'(',num2str(r2),'*' ,num2str( c2),')']);
subplot(1,2,2);
imshow(img3);
axis on;
title(['bicubic interpolation','(',num2str(r5),'*',num2str(c5),')']);

figure;
subplot(1,2,1);
imshow (img);
axis on;
title(['down-sampling' ,'(',num2str(r2),'*' ,num2str( c2),')']);
subplot(1,2,2);
imshow(img3);
axis on;
title(['bilinear interpolation ','(',num2str(r4),'*',num2str(c4),')']);
```

D

the psnrs between the original image and the up-sampled images respectively are as follows：

```
>> problem21
   77.2996

   78.5458

   77.2996
```

It can be found that the images processed by several methods have a certain degree of distortion, but the results of the three methods are similar, nearest and bilinear interpolation are the same, and the bicubic interpolation method is slightly better.

Code:

```
psnr1=problem2m( img , img3 );%
psnr2=problem2m( img , img4 );%
psnr3=problem2m( img , img5 );%

  disp(psnr1);
  disp(psnr2);
  disp(psnr3);
```

Task4

Experimental result:

Figure 15:Image after quantization

As a result, it can be found that severe quantization on images can cause severe distortion of the image. As a result, it can be found that severe quantization on images can cause severe distortion of the image. So reducing gray levels is not a good strategy, although higher gray levels require more storage, but it also means higher image clarity.

Code:

```
xx=imread('C:\Users\xuxun\Desktop\image\Lab1-Materials\lenna512.bmp');
x1=im2double(xx);
figure(1);
subplot(2,2,1);
imshow(x1)
title('512*512')
x2=x1(1:4:end,1:4:end);% 1 bit for every 4 digits
subplot(2,2,2)
imshow(x3)
title('128*128
x3=histeq(x3,16);% Change the gray level of the image to 16
subplot(2,2,3);
imshow(x16)
title('16*16')
```