Xi'an Jiaotong-Liverpool University

西交利物浦大學

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE412 Image and Video Processing

Lab1 - Image Processing

Student Name Zhiyu.Xu

Student ID 1926905

September 2019

Contents

Task 1

1.1 show RGB

1.1.1 Image result and code

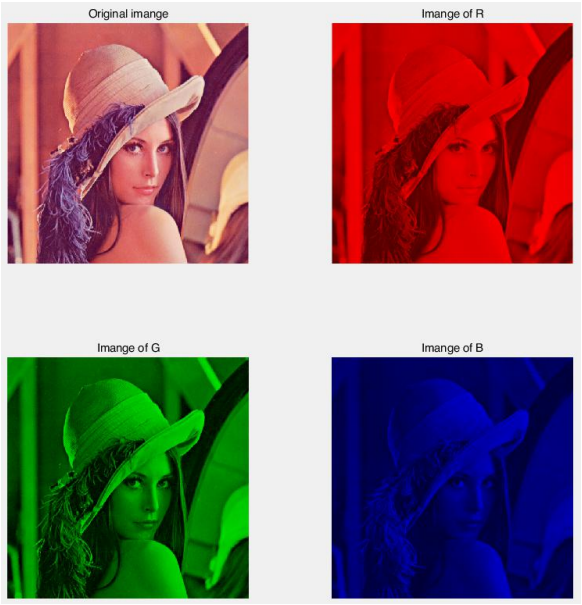

Figure 1 RGB Image

```
% Task(1) show RGB
Im = imread('lenna512color.bmp');
null = zeros(512,512);
subplot(2,2,1),imshow(Im),title('Original imange');
subplot(2,2,2),imshow(cat(3,Im(:,:,1),null,null)),title('Imange of R');
subplot(2,2,3),imshow(cat(3,null,Im(:,:,2),null)),title('Imange of G');
subplot(2,2,4),imshow(cat(3,null,null,Im(:,:,3))),title('Imange of B');
truesize;
```

Code 1

1.1.2 Discussion

In figure1, we split the original image into three parts(red part,green part and blue part). i.e. it can prove tat any colours can be produced by adding three primary colours together.

1.2 change RGB to HSI

1.2.1 Image result and code

Figure 2 HSI Image

```
% Task(2) change RGB to HSI
Im = imread('lenna512color.bmp');
rgb=im2double(Im);%change uint8 to double
%Extract imange components;
r = rgb(:, :, 1);
g = rgb(:, :, 2);
b = rgb(:, :, 3);
%RGB to HSI function
num = 0.5*((r - g) + (r - b));
den = sqrt((r - g).^2 + (r - b).*(g - b));
theta = acos(num./(den + eps)); %The "eps":Prevent divisor equal to zero;
%H
H = theta;
H(b > g) = 2*pi - H(b > g);
H = H/(2*pi);
%S
num = min(min(r, g), b);
den = r + g + b;
den(den == 0) = eps; %Prevent divisor equal to zero;
S = 1 - 3.* num./den;
H(S==0)=0;
%I
I = (r+g+b)/3;
subplot(2,2,1);imshow(Im);title('Original imange');
subplot(2,2,2);imshow(H);title('Imange of H');
subplot(2,2,3);imshow(S);title('Imange of S');
subplot(2,2,4);imshow(I);title('Imange of I');
truesize;
```

Code 2

1.2.2 Discussion

HSI model is a digital image model，which divides the color into three features: the first feature is the H (Hue), which can represent the kinds of colors such as RGB; the second feature is used to indicate the light and dark, called I (Intensity); the third feature is used to indicate the vividness of the color, that is S (Saturation). Color images can be described by color models such as RGB or HSI, which have a strict mathematical relationship and can be converted to each other. So this code is to convert RGB images into HIS images using known formulas.

1.3 gray image

1.3.1 Image result and code

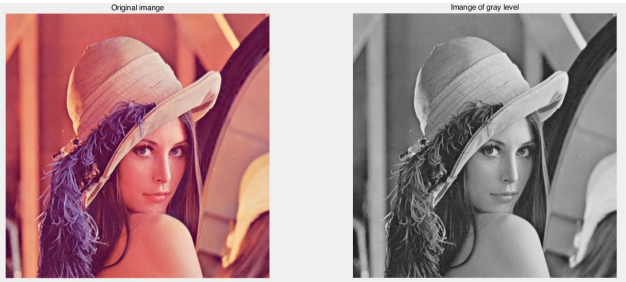

Figure 3 Gray Image

```
% Task1_3 gray image
Im = imread('lenna512color.bmp');
Im_gray = rgb2gray(Im);
subplot(1,2,1);imshow(Im);title('Original imange');
subplot(1,2,2);imshow(Im_gray);title('Imange of gray level');
truesize;
```

Code 3

1.3.2 Discussion

As for gray level, there is an image with only one sample color per pixel. Such images are usually displayed as grayscale from the darkest black to the brightest white. A complete image consists of three channels: red, green and blue. Thumbnails for the three channels of red, green, and blue are displayed in grayscale. Different gray scales are used to represent the proportion of "red, green, blue" in the image. The pure white in the channel represents the highest brightness here, and the brightness level is 255.

1.4 binary image

1.4.1 Image result and code

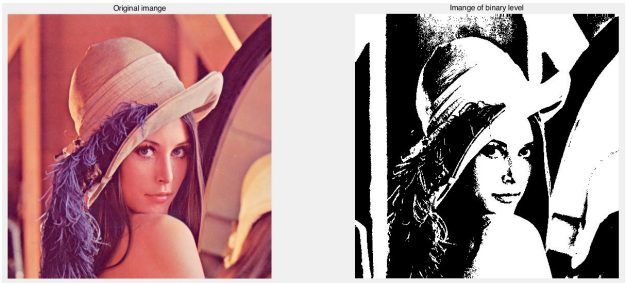

Figure 4 Binary Image

```
% Task1_4 binary image
Im = imread('lenna512color.bmp');
Im_binary = im2bw(Im);
subplot(1,2,1);imshow(Im);title('Original imange');
subplot(1,2,2);imshow(Im_binary);title('Imange of binary level');
truesize;
```

Code 4

1.4.2 Discussion

Compared with the grayscale image, the binary image means that there are only two gray levels in the image, that is, the gray value of any pixel in the image is 0 or 255, which represents black and white, respectively.

Task 2

2.1 PSNR

PSNR is the abbreviation of “Peak Signal to Noise Ratio”, which is an objective standard for evaluating images. Usually after image compression, the output image will be different to some extent from the original image. To measure the quality of the processed image, we usually refer to the PSNR value to measure the satisfaction of a certain process. The function code is shown as below.

```
% Task2 PSNR function
function [PSNR] = PSNR1(im_1,im_2)
%im_1 = imread('iamgename1.***');
%im_2 = imread('iamgename2.***');
im1 = double(im_1);
im2 = double(im_2);
[r1,c1] = size(im1);
[r2,c2] = size(im2);
t = 0;
if (r1 == r2 && c1 == c2)
    for i=1:r1
        for j=1:c2
            t=t+(im1(i,j)-im2(i,j))^2;
        end
    end
end
mse = t/(r1 * c1);
PSNR = 10*log10(255^2/mse);
end
```

Code 5

Task 3

3.1 Down-sampling by using mean value

3.1.1 Image result and code

Figure 5 Down-sampling by using mean value

```
%% Task3_1 Down-sampling by using mean value;
I0 = imread('lenna512.bmp');
foo = @(mat) mean(mat.data,'all');
I1 = blockproc(I0,[2 2],foo);
[r0,c0] = size(I0);
[r1,c1] = size(I1);
figure(1)
subplot(1,2,1);imshow(I0),title(['Original','(',num2str(r0),'*',num2str(c0),')']);
subplot(1,2,2);imshow(uint8(I1)),title(['Down-sampling','(',num2str(r1),'*',num2str(c1),')']);
truesize;
```

Code 6

3.1.2 Discussion

The principle in the code is to average each adjacent four pixel values in the original image and save it to a new image. So shrink a 512*512 image to a 256*256 image.

3.2 Up-sampling by using nearest neighbor interpolation

3.2.1 Image result and code

Figure 6 Up-sampling by using nearest neighbor interpolation(enlarged)

```
%% Task3_2  Up-sampling by using nearest neighbor interpolation;
N = 2; %the 'N' is magnification;
I2 = zeros(N*r1,N*c1);
for i=1:r1*N
    for j=1:c1*N
        r2=round(i/N);
        c2=round(j/N);
        I2(i,j)=I1(r2,c2);
    end
end
[r3,c3]=size(I2);
figure(2);
subplot(1,2,1);imshow(uint8(I1)),title(['Down-sampling','(',num2str(r1),'*',num2str(c1),')']);
subplot(1,2,2);imshow(uint8(I2)),title(['Up-sampling','(',num2str(r3),'*',num2str(c3),')']);
truesize;
```

Code 7

3.2.2 Discussion

Up-sampling the down-sampled image by nearest neighbor interpolation method, The enlarged result image is shown as figure 6. Based on the results, we can see that although the processed image pixels are already consistent with the original image, the image quality is between down-sampling image and original image.

3.3 Up-sampling by using bilinear and bicubic interpolation

3.3.1 Image result and code

Figure 7 Up-sampling by using bilinear and bicubic interpolation(enlarged)

```
%% Task3_3 Up-sampling by using bilinear interpolation and bicubic interpolation;
I3 = imresize(I1,2,'bilinear');
I4 = imresize(I1,2,'bicubic');
[r4,c4] = size(I3);
[r5,c5] = size(I4);
figure(3);
subplot(1,3,1);imshow(uint8(I1)),title(['Down-sampling','(',num2str(r1),'*',num2str(c1),')']);
subplot(1,3,2);imshow(uint8(I3)),title(['Up-sampling by bilinear interpolation','(',num2str(r4),'*',num2str(c4),')']);
subplot(1,3,3);imshow(uint8(I4)),title(['Up-sampling by bicubic interpolation','(',num2str(r5),'*',num2str(c5),')']);
truesize;
```

Code 8

3.3.2 Discussion

The code uses the matlab function to get “bilinear” and “bicubic” images. According to Figure 7, it seems that “bicubic” images is clearer and have best quality.

3.4 Calculate the PSNR

3.4.1 Result and code

```
%% Task3_4 Calculate the psnr;
% Useing PSNR1 function in task2;
PSNR_N0   = PSNR1(I0,I2);%Nearest;
PSNR_Bil0 = PSNR1(I0,I3);%Bilinear;
PSNR_Bic0 = PSNR1(I0,I4);%Bicubic;
% Useing matlab function;
PSNR_N   = psnr(I0,uint8(I2));%Nearest;
PSNR_Bil = psnr(I0,uint8(I3));%Bilinear;
PSNR_Bic = psnr(I0,uint8(I4));%Bicubic;
```

Code 9

PSNR \ Calculation method	PSNR1 function in task2	matlab function
nearest neighbor interpolation	31.5546	31.5546
bilinear interpolation	32.7468	32.7368
bicubic interpolation	34.3052	34.2903

3.4.2 Discussion

According to the above result table, Although two different calculation methods are used, the result error is small and can be ignored. Also, they have a same result, is that PSNR(bicubic) > PSNR(bilinear) > PSNR(nearest). As we know, the PSNR(Peak Signal to Noise Ratio) reflects the quality of processed image. the larger PSNR is, the higher image quality is. So the quality of bicubic interpolation is best.

Task 4

4.1 Quantization

4.1.1 Image result and code

Figure 8 256 gray level to 16 gray level by quantization

```
% Task4 256 gray level to 16 gray level by quantization
im_256 = imread('lenna512.bmp');
[r,c] = size(im_256);
im_16 = zeros(r,c);
for i = 1:r
    for j = 1:c
        im_16(i,j) = floor(im_256(i,j)/16);%rounding
    end
end
subplot(1,2,1);imshow(im_256),title('Original Imange');colorbar;
subplot(1,2,2);imshow(uint8(im_16),[0,15]),title('Quantized Imange');colorbar;
```

Code 10

4.1.2 Discussion

The quantization process is to convert the 256 gray value to 16 gray value by dividing the value of each pixel of the original image by 16. According to figure 8, the quantized image color hierarchy is single and the grayscale is clearly defined.