Mcmaster Unviersity
COMP ENG 3SK3
IMAGE PROCESSING
Project 1

Name:Fengyi Song
Student ID:1068106

Part A. Introduction of Bicubic-Interpolation Algorithm

Model:

$$f(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}a_{ij}x^{i}y^{j}$$

$$f(x,y)=\begin{bmatrix}x^3 & x^2 & x & 1\end{bmatrix}\begin{bmatrix}a_{3,3} & a_{3,2} & a_{3,1} & a_{3,0}\\ a_{2,3} & a_{2,2} & a_{2,1} & a_{2,0}\\ a_{1,3} & a_{1,2} & a_{1,1} & a_{1,0}\\ a_{0,3} & a_{0,2} & a_{0,1} & a_{0,0}\end{bmatrix}\begin{bmatrix}y^3\\ y^2\\ y\\ 1\end{bmatrix}$$

Given 4*4 known data points, we have F= $B*A*B^T$. That is, within any given 4 by 4 matrix, we are able to derive a euqation described by the formula above and draw estimated points according to it.

And the steps to calculate coefficient is as following.

$$F=BAB^{\mathrm{T}}\Rightarrow A=B^{-1}F(B^{\mathrm{T}})^{-1}=B^{-1}F(B^{-1})^{\mathrm{T}}$$

Therefore, for a coordinate with x starting from -1,0,1 to 2 and y from -1,0,1,2, we always have

$$B=\begin{bmatrix}-1 & 1 & -1 & 1\\ 0 & 0 & 0 & 1\\ 1 & 1 & 1 & 1\\ 8 & 4 & 2 & 1\end{bmatrix};\quad B^{-1}=\begin{bmatrix}-1/6 & 1/2 & -1/2 & 1/6\\ 1/2 & -1 & 1/2 & 0\\ -1/3 & -1/2 & 1 & -1/6\\ 0 & 1 & 0 & 0\end{bmatrix}$$

In order to interpolate the center missing point, we could plug (0.5,0.5) into the equation

$$f(\frac{1}{2},\frac{1}{2})=\begin{bmatrix}\frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 1\end{bmatrix}_x B^{-1}F(B^{-1})^{\mathrm{T}}\begin{bmatrix}\frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 1\end{bmatrix}_y^{\mathrm{T}}$$

$$=\begin{bmatrix}\frac{-1}{16} & \frac{9}{16} & \frac{9}{16} & \frac{-1}{16}\end{bmatrix}\begin{bmatrix}f(-1,-1) & f(-1,0) & f(-1,1) & f(-1,2)\\ f(0,-1) & f(0,0) & f(0,1) & f(0,2)\\ f(1,-1) & f(1,0) & f(1,1) & f(1,2)\\ f(2,-1) & f(2,0) & f(2,1) & f(2,2)\end{bmatrix}\begin{bmatrix}\frac{-1}{16}\\ \frac{9}{16}\\ \frac{9}{16}\\ \frac{-1}{16}\end{bmatrix}$$

For each 4 by 4 frame we take from the original matrix, we do not change the coordinate of matrix such that we do not have to change the coefficients each time we move it. For the efficiency of the algorithm and the correlation of the neighbor points, I interpolate three points in a single frame, (0.5,0.5),(0.5,1) and (0,0.5) respectively. However, the coefficient are not same, we have to do the same steps for the other two points as center point.

Part B. Complexity

Since I used a nested loop each with iteration of n, the complexity is n^2.
For each interpolated point, the number of operations are as following.
Multiplication: 20
Addition: 20

The number of operations required to interpolate the center missing point is same as to interpolate other two points.

Part C. Demonstration of Interpolation

Figure(a) shows the original gray scale image. In this project, we are only dealing with one channel image.

Figure(a) Input Image

Figure(b) describes the image just before interpolation. I insert zero row/column vector every other row/colums in order to hold interpolation point.

Figure(b) Enlarged Picture before Interpolation

Figure(c) is the interpolated image from which we can see that the missing pixels are interpolated based upon the surrounding pixels.

Figure(c) Output Image after Interpolation

Part D. Down Sampling and Interpolation

In this part, we are down sampling a sample image and interpolate that image by bi-cubic algorithm. First of all, let's simply use the same image as Part c for comparison.

Figure(d) Image before Down Sampling

Figure(e) Image after Down Sampling

We can clearly see that the new image is decreased due to the effect of down sampling.

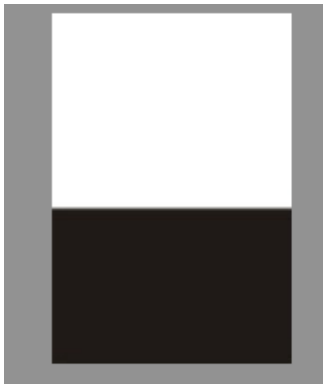
Figure(f) Interpolated Image

Figure(f) is the output interpolated image which has the same size as original image. The standard square error compared with original one for this image is

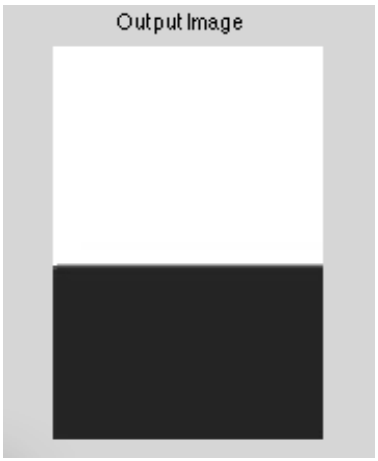
```
mse =

    35.1556

>>
```

Part E. Discussion and Observation

Firstly, for the purpose of limiting variables, let us just interpolate images with black and white.



Figure(g) Image just in black and white.

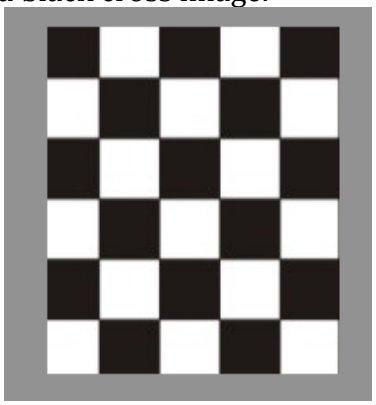


Figure(h) Output Image in Black and White

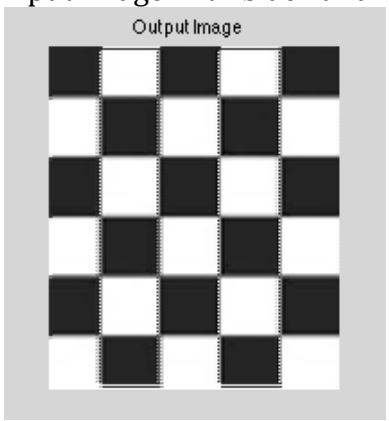
The MSE is

```
mse =

    0.2145
```

Next, let us check a white and black cross image.



Figure(i) Input Image with black and white grid



Figure(j) Interpolated image of black and white grid

The MSE is

```
mse =

   13.0655
```

In conclusion, since we are actually making estimations, the more changing elements, larger MSE we will get. Regardless of color density, even though we only have two color, the combination makes big difference. If the image has more details, our interpolation would have more error and therefore, the output image would become more distorted.

Appendix. Matlab Code

```
clc;clear;close all;
InputImage=imread('cross.png');
imshow(InputImage);
title('Input Image');
%[row,col]=size(InputImage);
t = cputime;
%m=[1 2 3 4;5 6 7 8;9 10 11 12;]
InputImage=rgb2gray(InputImage);
row=size(InputImage,1);
col=size(InputImage,2);
channel=size(InputImage,3);
OutputImage_2=uint8(zeros(row/2,col/2,channel));

for i=1:row/2
    for j=1:col/2
        OutputImage_2(i,j)=InputImage(2*i-1,2*j-1);
    end
end
[row_2, col_2, channel_2]=size(OutputImage_2);

row=row_2;
col=col_2;
channel=channel_2;
Original=InputImage;
InputImage=OutputImage_2;


imshow(OutputImage_2);
whos InputImage
OutputImage=uint8(zeros(2*row,2*col,channel));% Expand the output image
by a factor of 2

for k=1:row
    for j=1:col
        OutputImage(2*k-1,2*j-1)=InputImage(k,j);% copy the element
from original image
        %OutputImage(2*k-1,2*j-1)=OutputImage_2(k,j);
    end
end
figure(2);
imshow(OutputImage);
title('Before Interpolation');
%return;

%OutputImage=double(OutputImage);

B=[-1.0 1.0 -1.0 1.0;0 0 0 1.0;1.0 1.0 1.0 1.0;8.0 4.0 2.0 1.0;];
a=[1.0/8.0 1.0/4.0 1.0/2.0 1.0]*inv(B);% the coefficient of f(0.5,0.5)
b=inv(B)'*[1.0;1.0;1.0;1.0];% the coefficient of f(0.5,1)
c=[0 0 0 1.0]*inv(B);% the coefficient of f(0,0.5)

for i=1:(row-3)
    for j=1:(col-3)
        F=double(InputImage(i:i+3,j:j+3));
        f_mid=a*F*a';       %Interpolate f(0.5,0.5)
```

```
        f_midu=a*F*b;%Interpolate f(0.5 1)
        f_midl=c*F*a'; %Interpolate f(0,0,5)
        OutputImage(2*i+2,2*j+2)=f_mid;
        OutputImage(2*i+2,2*j+1)=f_midu;
        OutputImage(2*i+1,2*j+2)=f_midl;
    end
end
[newrow, newcol]=size(OutputImage);

for k=1:newrow % This loop is dealing with the fram of the image
    for j=1:newcol
        if(OutputImage(k,j)==0&&j~=1)
            OutputImage(k,j)=OutputImage(k,j-1);
        end
        if(OutputImage(k,j)==0&&k~=1)
            OutputImage(k,j)=OutputImage(k-1,j);
        end

    end
end
[r,cl]=size(OutputImage);
Diff=OutputImage-Original(1:r,1:cl);
s=sum(sum(Diff.^2));
mse=s/(r*cl);



figure(3);
imshow(OutputImage);title('Output Image');
TimeElapsed = cputime-t;
```