CoE 3SK3

Project 2

Image Optimization in Linear Programming

Fengyi Song

1068106

Introduction

With the advanced technology developed today, people can take numerous methods to improve the quality of a picture. However, it is always human viewers that make ultimate judgments of visual quality. In this project, we want to enhance the contrast of pictures. Some pictures might be under expose and some might over expose. The purpose of this project is to optimize the pictures using linear programming.

Part (a) Formulate the problem of contrast enhancement.

Since we are only dealing with the grey scale picture, the range of pixels will only vary from 0 to 255.

$$T : \{0,1,\cdots,255\} \rightarrow \{0,1,\cdots,255\}$$

For an under expose picture, it means that there is a big probability around pixels with low intensities while it is another way around for over expose pictures. By increasing the larger intensity pixel while decreasing smaller intensity pixel, we can technically maximize the contrast of an image. In order to formulate this problem, fist thing we have to clarify is what kind of equation we are dealing with. Again, since this is a linear programming practice problem, we surely want to solve it by tools of linear programming. Between input image and output image, we need a transfer function which helps to enhance the contrast. Because the value of pixel is only integer, the transfer function is actually a function with many step-wise floors. By multiplying the probability density of images and transfer function, we can get a linear equation of C(s).

$$C(s) = \sum_{0 \le j < N} p_j s_j$$

This is the final linear equation that we want to maximize in order to enhance the contrast of images.

The shape of linear equation of n variables is a hyper plane. There are infinite solutions to maximization without constraints. Therefore, in order to achieve optimization, we have to set proper constrains on the linear equation. The first constraint, obviously, is

$$\sum_{0 \le j < N} s_j < 256$$

For the width of a single step, the constraint is $0 \le s_j < 256$.

Part (b) Initial Optimization

By applying the constraints above, we can see the visual effects of output image as following.

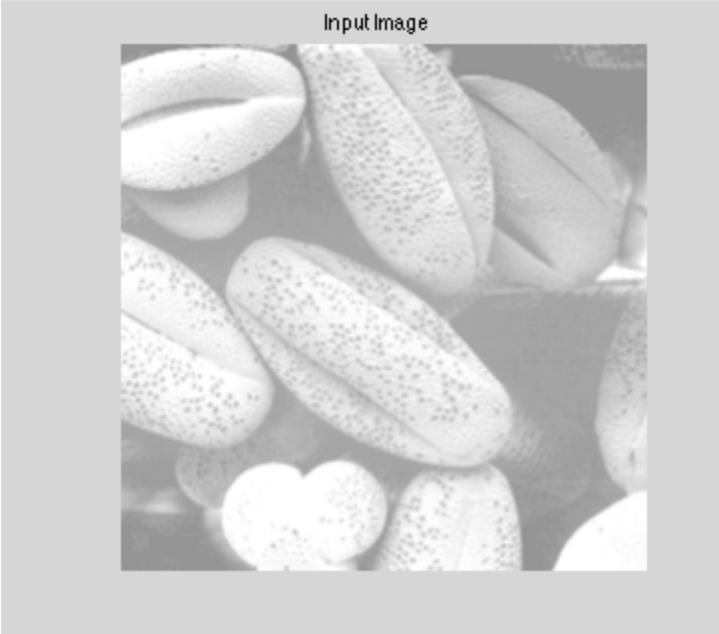

Figure1. overexpose InputImage(1)

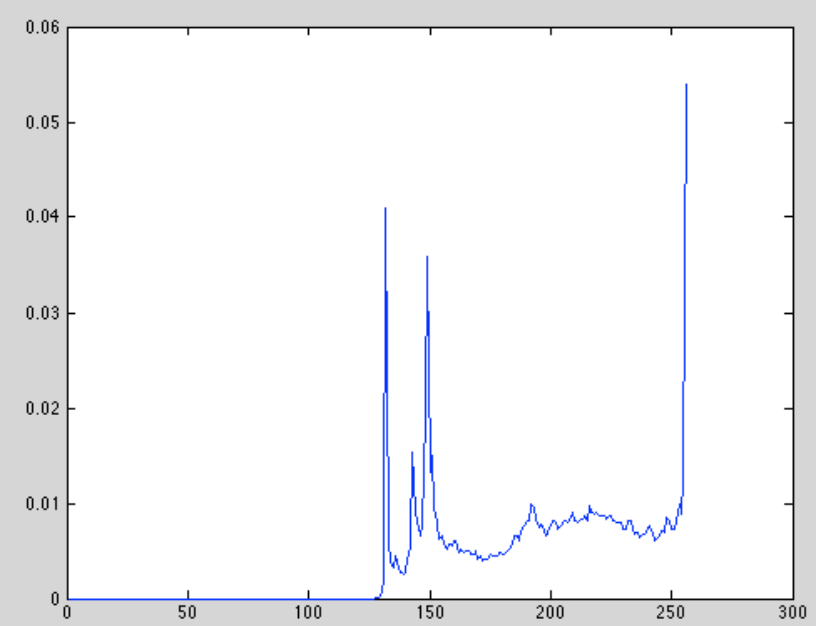

Figure2. Probability Density Function of InputImage(1)

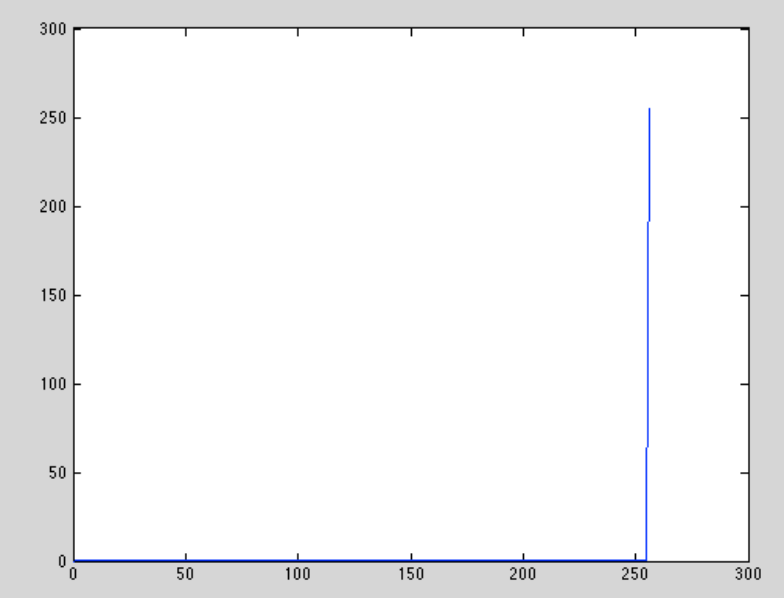

Figure3. Transfer function for InputImage(1)

Figure4. output of InputImage(1)

At first glance, the visual effects seem to be awful, but on the second thoughts, it is quite reasonable. This is because we want to maximize the difference between lower intensity pixel and larger intensity pixel. After applying those constraints, the computer would set a big jump at 255 while push all the other pixels back to origin. Therefore, a black and white image is shown. However, this is undesirable because it

does not comply with human's standards. Another example is listed below to further discuss this problem.

Figure5. InputImage(2)

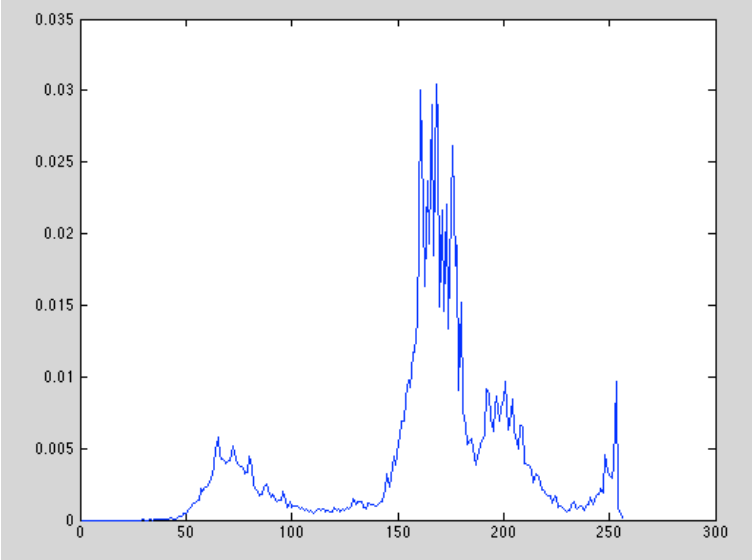

Figure6. Probability Density Function of InputImage(20

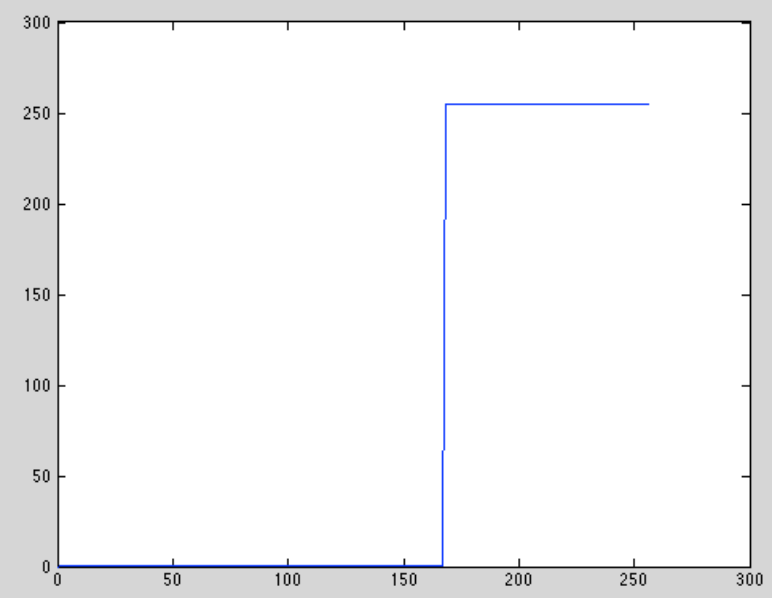

Figure7. Transfer Function of InputImage(2)

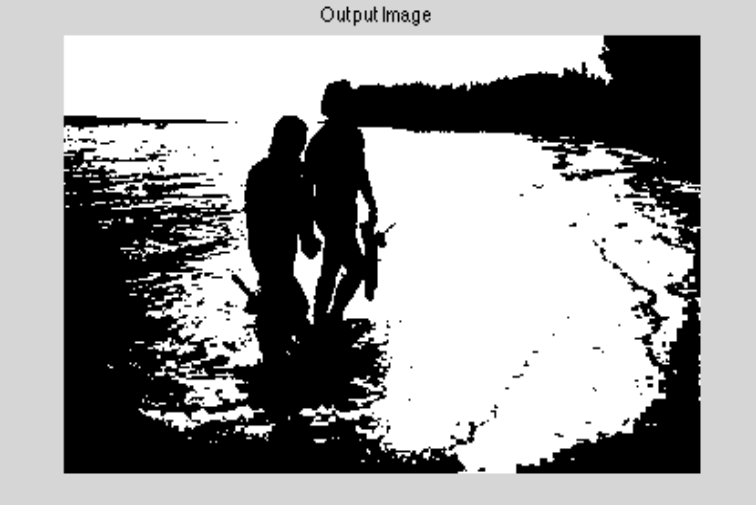

Figure8. OutputImage(2)

Part(c) Final Optimization

The reason why we have black and white picture is stated before. In order to avoid this problem, we must maintain the original shape of probability density function of Input image. It is known that the output of y=x transfer function would be the image itself. Based on this principle, a minimum slope and maximum slope should be bounded. In other words, we have to obey the general trend of y=x.
Therefore, we need extra constraints. In this step, I force each step size should not be smaller than 1/d, which is 0.5 and not bigger than 3 as illustrated in Dr. Wu's paper. By doing this, we could guarantee that there will not be any unreasonable jump and therefore, it would largely comply with a uniform transfer function while maximizing the linear equation.

Figure9 optimization of OutputImage(2)

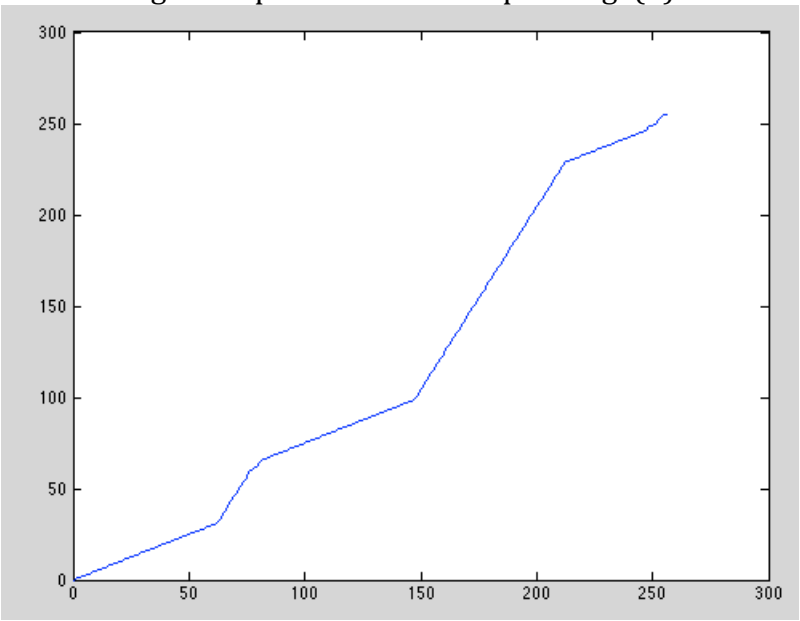

Figure10 Transfer Function after Optimization

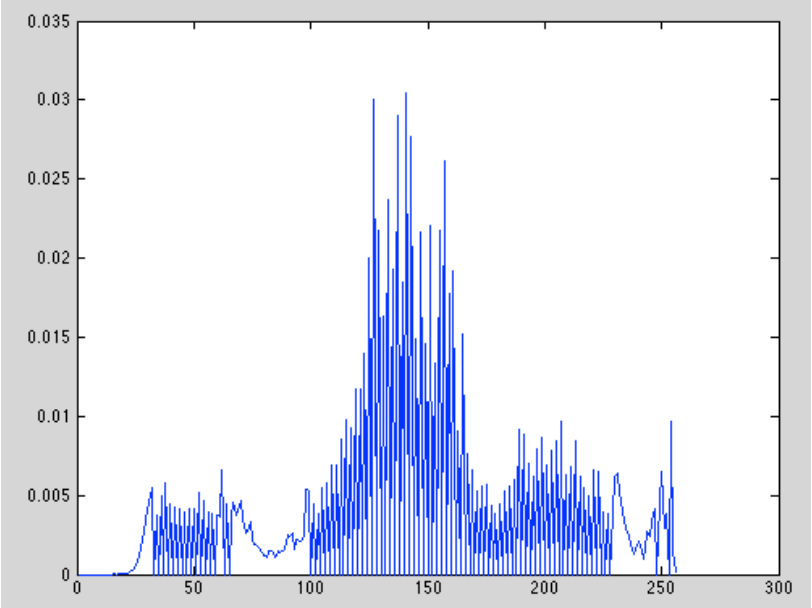

Figure11 Probability Density Function after Optimization

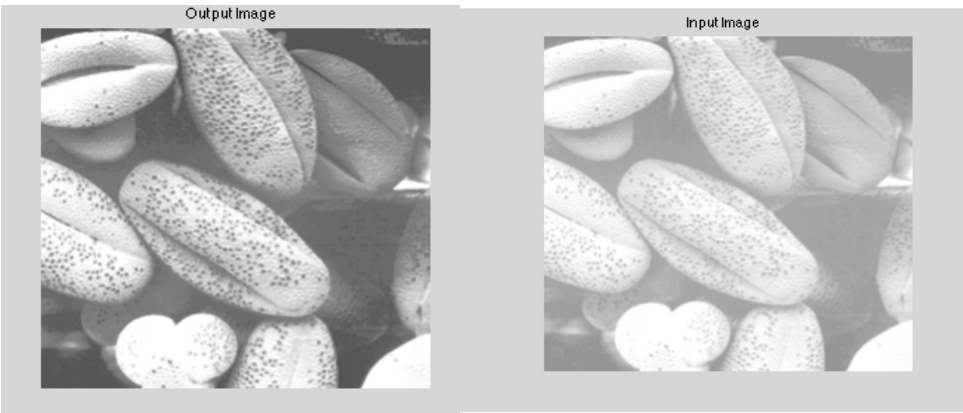

Figure12. Optimization of OutputImage(1)

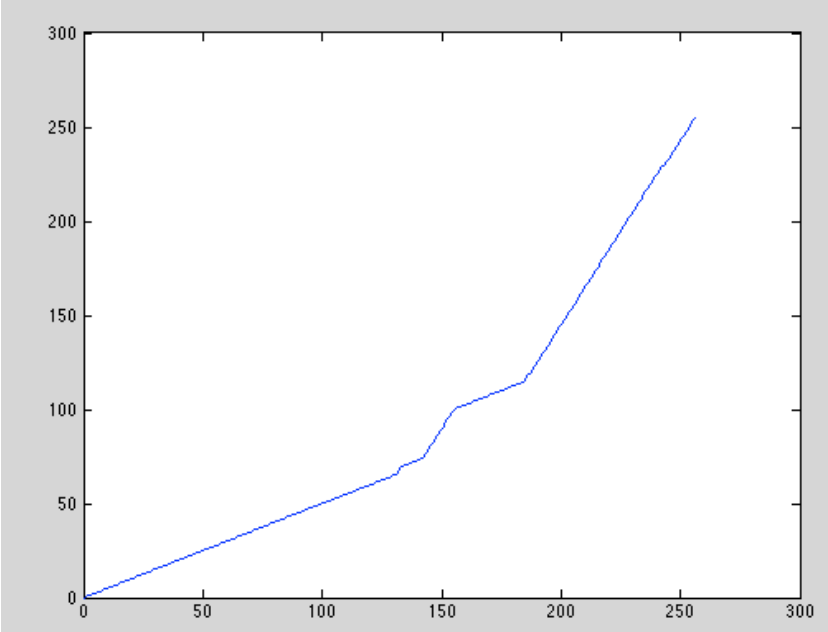

Figure13 Transfer Function for InputImage(2) after Optimization

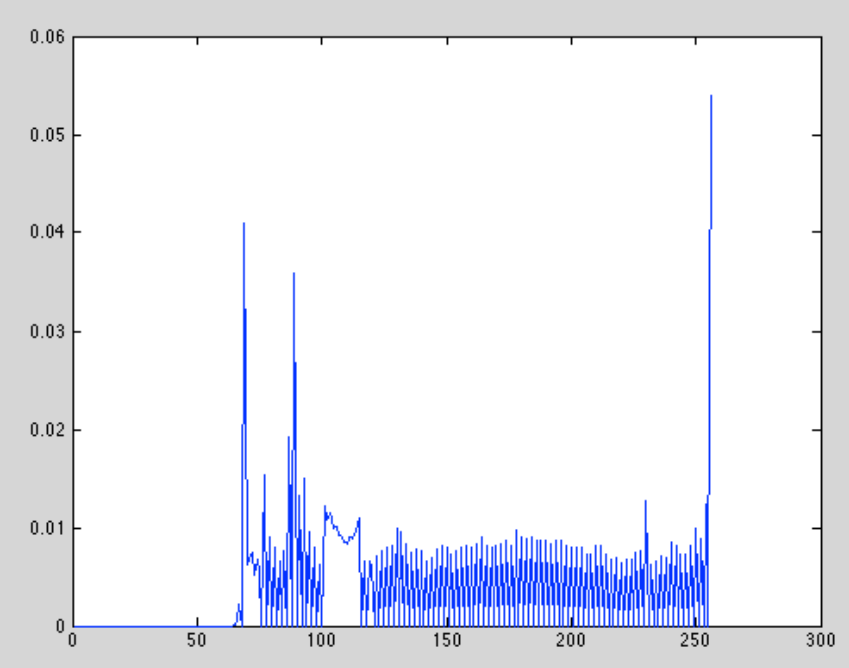

Figure14. Probability Density Function of OutputImage(2) after Optimization

Part(D) Minimization in Linear Programming

To observe the effects of minimization of contrast, let's use the same picture. Intuitively, since we want to minimize C(s) defined before, and the pixel intensity still lie between (0,255), the transfer function should be 0. In other words, all the pixel intensity will be set as 0 and the image would become black. Detailed image visualization is shown below.

Figure 15 InputImage for Minimization

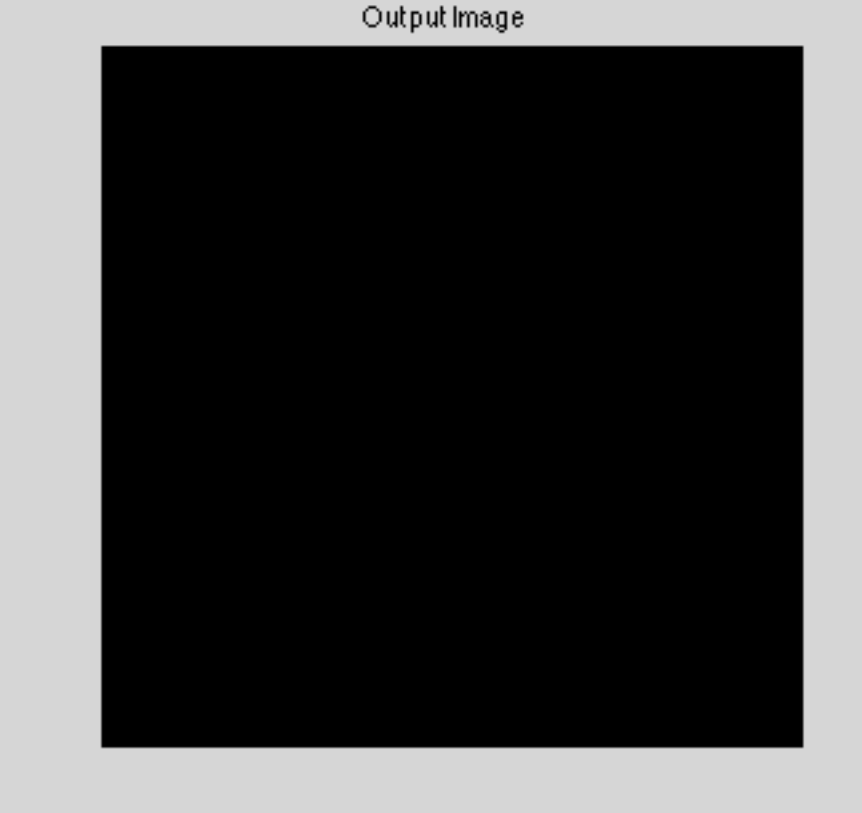

Figure16 OutputImage for Minimization

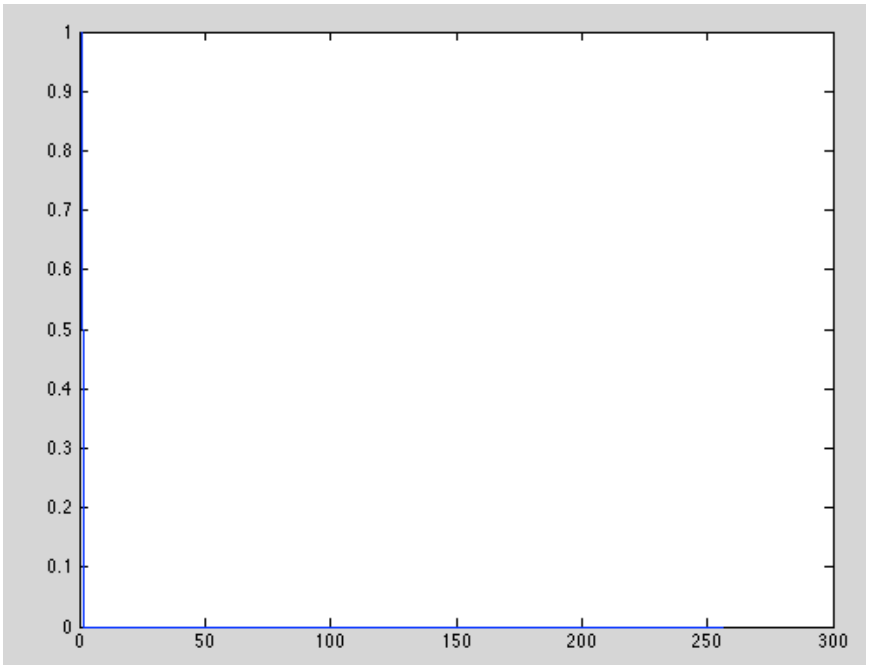

Figure17 Probability Density Function of OutputImage

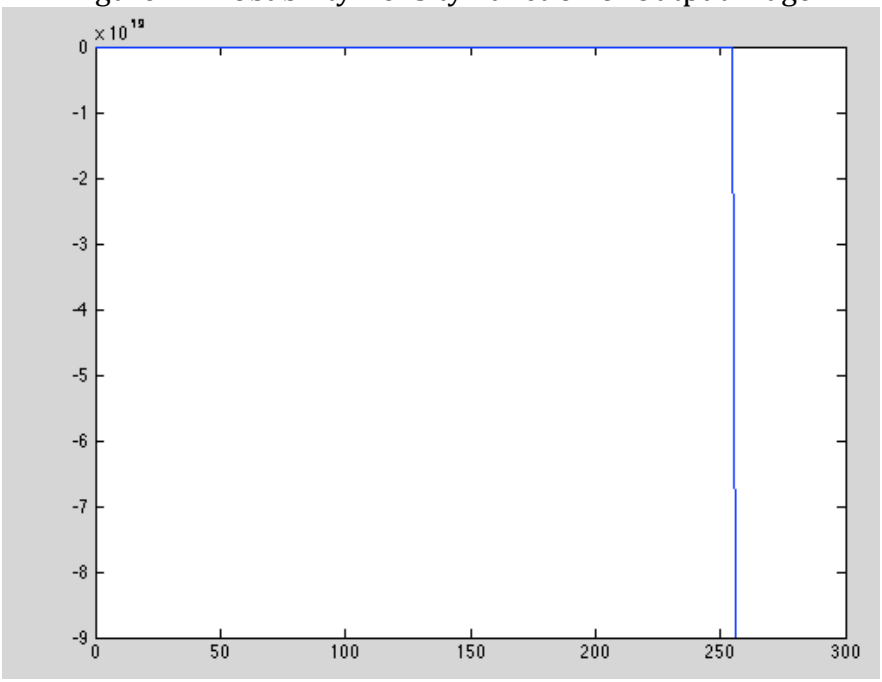

Figure18 Transfer Function of Minimization

Part(E) Matlab Code

```
clc;clear;close all;
InputImage=imread('test_4.png');


t = cputime;

InputImage=rgb2gray(InputImage); %not necessary for tif
figure(1)
imshow(InputImage);
title('Input Image');
InputImage=double(InputImage);
row=size(InputImage,1);
col=size(InputImage,2);
channel=size(InputImage,3);
X=linspace(0,255,256);
figure(2)
u=reshape(InputImage,row*col,1);
N=HIST(u,X);
p=N/(row*col);
plot(p);

%hist(u,X);
f=-p';
A=zeros(254,256);


for m=2:254

    A(m,m-1)=-1;
    A(m,m)=-1;
    A(m,m+1)=-1;
%     A(m,254-m+4)=1;
%     A(m,254-m+3)=1;
%     A(m,254-m+2)=1;

end
%A((255:507),(1:256))=A((2:254),(1:256));
A(1,:)=ones(1,256);
B=-1*ones(254,1);
B(1,:)=255;
%B=[B;ones(253,1)];
lb=0.5*ones(256,1);
ub=3*ones(256,1);

A1=ones(1,256);


s=linprog(f,A1,255,[],[],lb,ub);
OutputImage=zeros(row,col);

[r,c]=size(s);
gradient=0;
figure(3)
for i=1:r
```

```
    plot([i-1,i],[gradient,gradient+s(i)])
    hold on;
    gradient=gradient+s(i);

    for j=1:row
        for k=1:col
            if(InputImage(j,k)==(i-1))

                OutputImage(j,k)=gradient;
                %fprintf('Input is %g, output is
%g\n',InputImage(j,k),OutputImage(j,k));
            end

        end
    end

end


X=linspace(0,255,256);
figure(4)
u_out=reshape(OutputImage,row*col,1);
N_out=HIST(u_out,X);
p_out=N_out/(row*col);
plot(p_out)



whos OutputImage
figure(5)
imshow(uint8(OutputImage));title('Output Image');

TimeElapsed = cputime-t;
```