AU 326 Digital Image Process

By: Shen Zhen (518021910149)

HW#: 1

October 10, 2020

I. PROBLEM 1

A. Introduction

High-definition television (HDTV) generates images with 1125 horizontal TV lines interlaced (where every other line is painted on the tube face in each of two fields, each field being 1¿60th of a second in duration). The width-to-height aspect ratio of the images is 16:9. The fact that the number of horizontal lines is fixed determines the vertical resolution of the images. A company has designed an image capture system that generates digital images from HDTV images. The resolution of each TV (horizontal) line in their system is in proportion to vertical resolution, with the proportion being the width-to-height ratio of the images. Each pixel in the color image has 24 bits of intensity resolution, 8 bits each for a red, a green, and a blue image. These three "primary" images form a color image. How many bits would it take to store a 2-hour HDTV movie?

B. Solution

The horizontal resolution is $1125 \times (16/9) = 2000$. It would take $1125 \times 2000 \times 24 \times 60 \times 7200 = 2.332 \times 10^{13} bits$.

II. PROBLEM 2

A. Introduction

Consider the two image subsets, S1 and S2, shown in the following figure. For V = 1, determine whether these two subsets are (a) 4-adjacent, (b) 8-adjacent, or (c) m-adjacent.

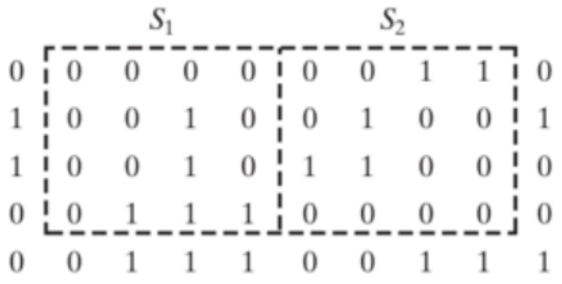

FIG. 1: Problem 2

B. Solution

As we can see in FIG.2:
(a)They are not 4-adjacent, because q is not in the set $N_4(p)$.
(b)They are 8-adjacent.
(c)They are m-adjacent, because q is in the set $N_D(p)$, and there is not a pixel from V in the set $N_4(p) \cap N_4(q)$.

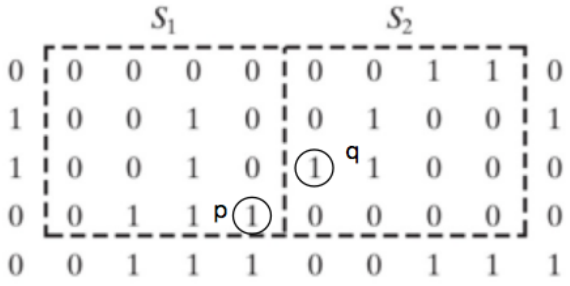

FIG. 2: Solution of Problem 2

III. PROBLEM 3

A. Introduction

Image subtraction is used often in industrial applications for detecting missing components in product assembly. The approach is to store a “golden” image that corresponds to a correct assembly; this image is then subtracted from incoming images of the same product. Ideally, the differences would be zero if the new products are assembled correctly. Difference images for products with missing components would be nonzero in the area where they differ from the golden image. What conditions do you think have to be met in practice for this method to work?

B. Solution

High accuracy of detection of the overall positions.Proper management of noise.Appropriate light sources.

IV. PROBLEM 4

A. Introduction

A CCD TV camera is used to perform a long-term study by observing the same area 24 hours a day, for 30 days. Digital images are captured and transmitted to a central location every 5 minutes. The illumination of the scene changes from natural daylight to artificial lighting. At no time is the scene without illumination, so it is always possible to obtain an image. Because the range of illumination is such that it is always in the linear operating range of the camera, it is decided not to employ any compensating mechanisms on the camera itself. Rather, it is decided to use image processing techniques to post process, and thus normalize, the images to the equivalent of constant illumination. Propose a method to do this. You are at liberty to use any method you wish, but state clearly all the assumptions you made in arriving at your design.

B. Solution

Assume that $f_o(x,y)$ means the image taken under artificial light of the environment.And the purpose is to normalize the images to it. To deal with possible objects that may move,we should take action to exclude these situations. We can choose some small areas of the whole image which are hardly affected by dynamic elements. Then we calculate their average maximum and minimum values, represented as $\bar{f}_{\max}$ and $\bar{f}_{\min}$. The next step is to find a function to transform any input $f(x,y)$ into an output confined to the boundaries of $\bar{f}_{\max}$ and $\bar{f}_{\min}$. Here is a feasible function:

$$f_{\text{out}}\ (x,y) = \text{af}(x,y) + b$$

Here

$$\text{a} = \frac{\bar{f}_{\max} - \bar{f}_{\min}}{f_{\max} - f_{\min}}$$

$$\text{b} = \frac{\bar{f}_{\min} f_{\max} - \bar{f}_{\max} f_{\min}}{f_{\max} - f_{\min}}$$

$f_{\text{out}}\ (x,y)$ is the output image,$f_{\max}$ and $f_{\min}$ mean the maximum and minimum value of the input.
Assumptions: (a) The range of the illumination is such that it is always in the linear operating range of the camera. (b) Few areas would be affected by the moving objects, otherwise $\bar{f}_{\max}$ and $\bar{f}_{\min}$ would lose their significance. And the input fetched by the camera which we assumed is the environment is not so accurate.

V. PROBLEM 5

A. Introduction

An image with intensities in the range [0, 1] has the PDF $p_r(r)$ shown in the following diagram. It is desired to transform the intensity levels of this image so that they will have the specific $p_z(z)$ shown. Assume continuous quantities and find the transformation (in terms of r and z) that will accomplish this.

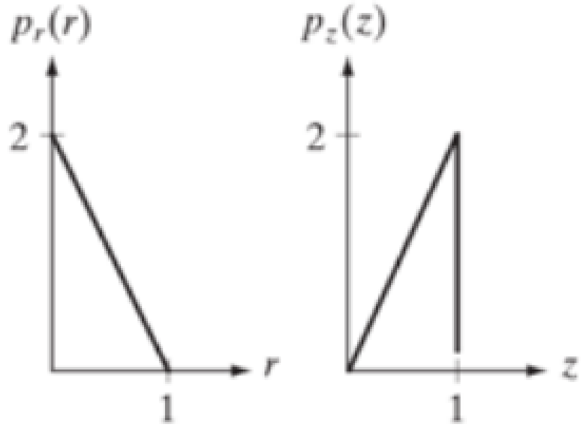

FIG. 3: Problem 5

B. Solution

Here is the solution:

$$s = T(r) = \int_0^t P_t(r)dr = \int_0^r (-2r+2)dr = -r^2 + 2r$$
$$v = G(z) = \int_0^z P_z(z)dz = \int_0^z 2zdz = z^2$$
$$z = G^{-1}(v) = \pm\sqrt{v}$$
$$v = s$$
$$z = \pm\sqrt{-r^2 + 2r}$$

VI. PROBLEM 6

A. Introduction

The images shown below are quite different, but their histograms are the same. Suppose that each image is blurred with a 3 * 3 averaging mask. (a) Would the histograms of the blurred images still be equal? Explain. (b) If your answer is no, sketch the two histograms.

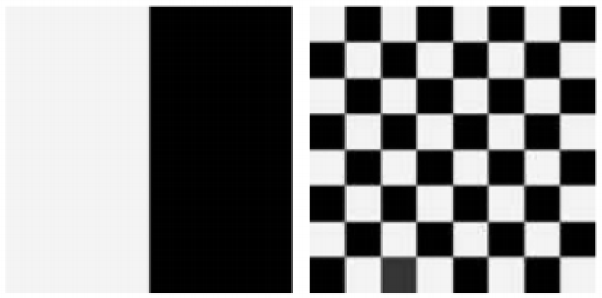

FIG. 4: Problem 6

B. Solution

(a)No,the histograms of the blurred images would be different. Because the total length of the boundaries of black and white on the right is larger than the left. When blurred, more dimensions of gray value are generated.

(b)Add a border of 0s to each image. Assume that each image is a N*N square. We use a 3*3 mask whose coefficients is $\frac{1}{9}$to implement the blurring. Table 1 and 2 show the number of points and the corresponding gray values of the blurred images of the original images, which is an illustration of the two histograms.

Number of points	values
$\mathrm{N}\left(\frac{\mathrm{N}}{2}-1\right)$	0
2	$\frac{2}{9}$
N-2	$\frac{3}{9}$
4	$\frac{4}{9}$
3N-8	$\frac{6}{9}$
$(N-2)\left(\frac{N}{2}-1\right)$	1

TABLE I: The histogram of the left image

Number of points	values
$\frac{N^2}{2} - 14N + 98$	0
28	$\frac{2}{9}$
14N-224	$\frac{3}{9}$
128	$\frac{4}{9}$
98	$\frac{5}{9}$
16N-256	$\frac{6}{9}$
$\frac{N^2}{2} \cdot 16N + 128$	1

TABLE II: The histogram of the right image

VII. PROBLEM 7

A. Introduction

Develop a program to calculate histogram and make histogram equalization for given images. Test the program on suitable images. Display histogram plots of the original and equalized images.

B. Code

This section will consist of the important code blocks which were implemented in order to meet the requirements of the lab.

1. Using inserted functions

The code block below shows the inserted functions of calculating histogram (*imhist()*) and making histogram equalization (*histeq()*) in Matlab.

```matlab
clear;close all;clc;
H=rgb2gray(imread('image7.jpg'));

subplot(2,2,1);
imshow(H);
title('The original image');

subplot(2,2,2);
imhist(H);
title('The histogram of the original image');

subplot(2,2,3);
H2=histeq(H);
imshow(H2);
title('The equalized image');

subplot(2,2,4);
imhist(H2);
title('The histogram of the equalized image');
```

2. Using functions of my own

The code block below shows the functions of my own to calculate histogram and making histogram equalization in Matlab.

```matlab
clear;close all;clc;
H=rgb2gray(imread('image7.jpg'));
[row, col] = size(H);

subplot(2,2,1), imshow(H), title('The original image');
subplot(2,2,2), imhist(H), title('The histogram of the original image');

PMF = zeros(1, 256);
for i = 1:row
    for j = 1:col
        PMF(H(i,j) + 1) = PMF(H(i,j) + 1) + 1;
    end
end
PMF = PMF / (row * col);

CDF = zeros(1,256);
CDF(1) = PMF(1);
for i = 2:256
    CDF(i) = CDF(i - 1) + PMF(i);
end

Sk = zeros(1,256);
for i = 1:256
    Sk(i) = CDF(i) * 255;
end

Sk = round(Sk);
```

```
for i = 1:row
    for j = 1:col
        H(i,j) = Sk(H(i,j) + 1);
    end
end

subplot(2,2,3), imshow(H), title('The equalized image');
subplot(2,2,4), imhist(H), title('The histogram of the equalized image');
```

C. Result

I implemented histogram equalization in Matlab, and I called the existing functions to meet my needs in FIG.5. Also, in FIG.6, I implemented my own functions and get the same result.We can see the original image and its histogram along with the equalized image and the corresponding histogram.Through histogram equalization, the image has an interface with more contrast and decisive lines compared with the original one.Also, from the two histograms we can conclude that gray values have a meaner distribution on the entire axis after histogram equalization.

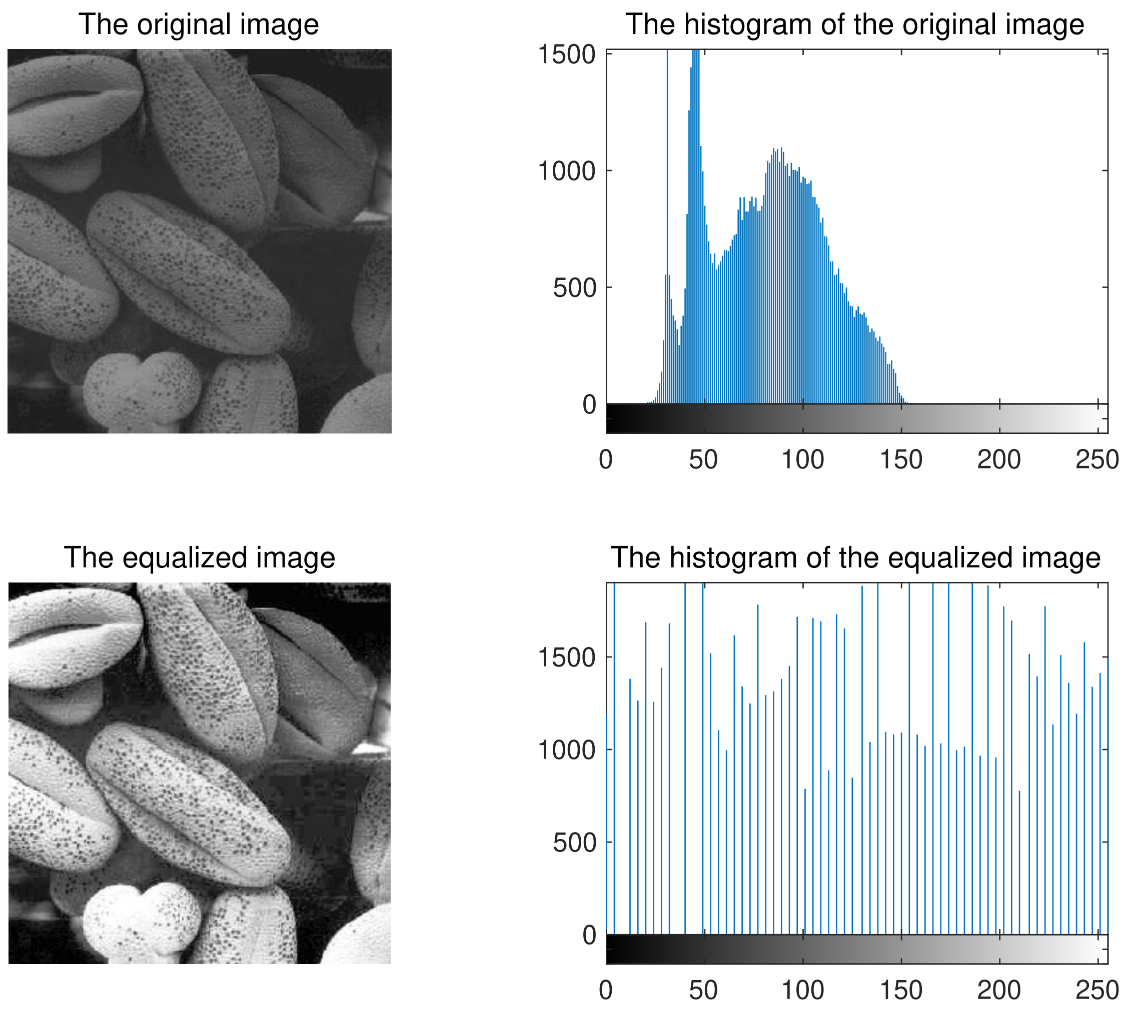

FIG. 5: The result of Problem 7 in Matlab using existing functions

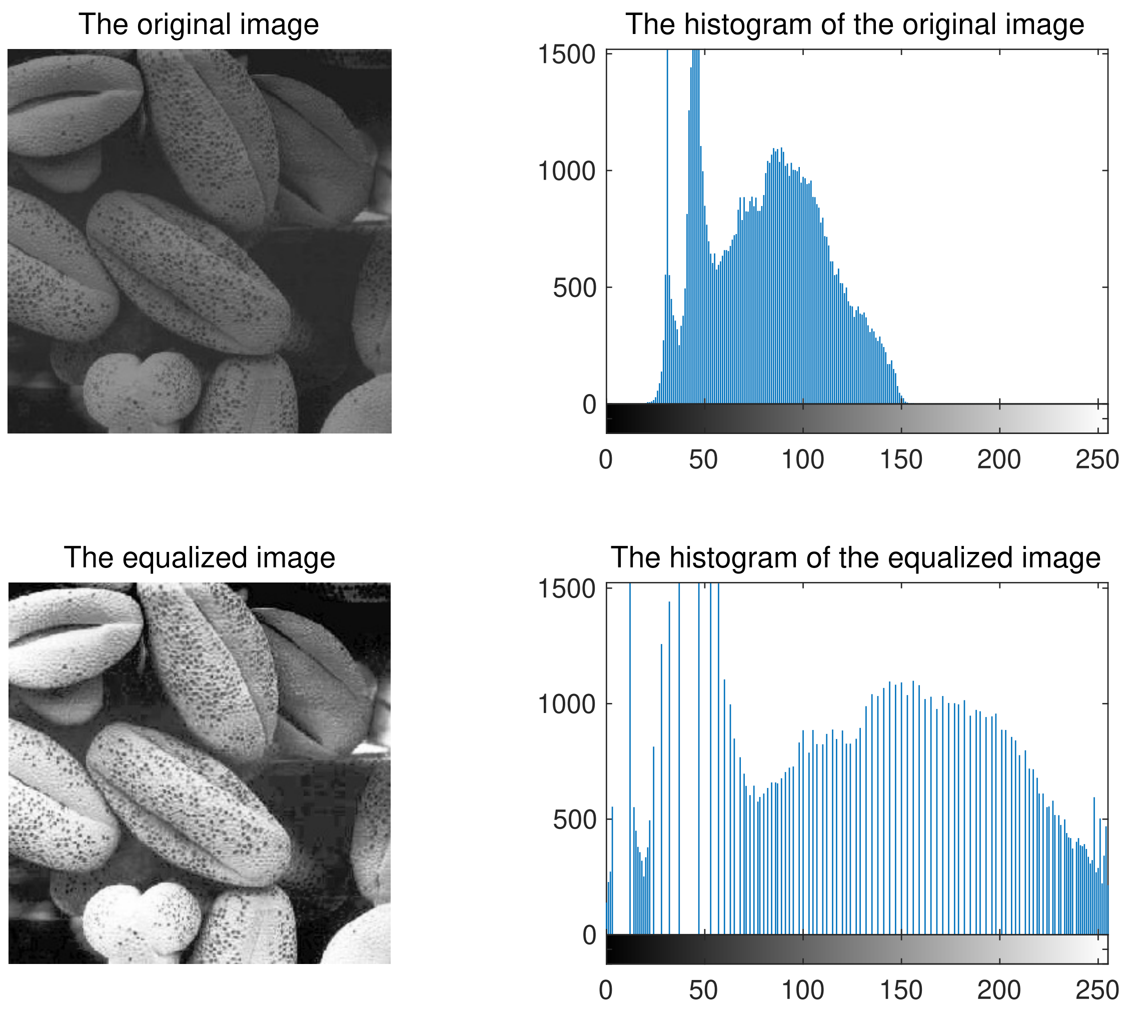

FIG. 6: The result of Problem 7 in Matlab using functions of my own

VIII. PROBLEM 8

A. Introduction

Add different levels of Gaussian noises to a given image, observe the change of corresponding histograms (based on the program you developed above). And then process the Gaussian noise polluted images using your designed kernels to achieve desired effect.

B. Code

This section will consist of the important code blocks which were implemented in order to meet the requirements of the lab.

1. *Using inserted functions*

The code block below uses inserted function in Matlab to add Gaussian noise (*imnoise()*) and process it (*fspecial()* and *imfilter()*).

```
clear;close all;clc;

A=rgb2gray(imread('image8.jpg'));
B=imnoise(A,'gaussian',0,0.01);
C=imnoise(A,'gaussian',0,0.03);
D=imnoise(A,'gaussian',0.2,0.01);
E=imnoise(A,'gaussian',0.2,0.03);
B1= fspecial('gaussian',[9,9],1);
B2 = imfilter(A,B1,'replicate');
C1= fspecial('gaussian',[9,9],1);
C2 = imfilter(A,C1,'replicate');
D1= fspecial('gaussian',[9,9],1);
D2 = imfilter(A,D1,'replicate');
E1= fspecial('gaussian',[9,9],1);
E2 = imfilter(A,E1,'replicate');

subplot(5,3,1),imshow(A),title('The original image');
subplot(5,3,2),imhist(A);
subplot(5,3,4),imshow(B),title('mean value=0,variance=0.01');
subplot(5,3,5),imhist(B);
subplot(5,3,6),imshow(B2),title('Replica of the left polluted image');
subplot(5,3,7),imshow(C),title('mean value=0,variance=0.03');
subplot(5,3,8),imhist(C);
subplot(5,3,9),imshow(C2),title('Replica of the left polluted image');
subplot(5,3,10),imshow(D),title('mean value=0.2,variance=0.01');
subplot(5,3,11),imhist(D);
subplot(5,3,12),imshow(D2),title('Replica of the left polluted image');
subplot(5,3,13),imshow(E),title('mean value=0.2,variance=0.03');
subplot(5,3,14),imhist(E);
subplot(5,3,15),imshow(E2),title('Replica of the left polluted image');
```

2. *Using functions of my own*

This section consists of the important code blocks of my own which were implemented in order to meet the requirements of the lab.

The code block below illustrates how to add Gaussian noise.By using Gassian distribution matrix, noise is created and added to the original image.

```
function [B] = myGaussianNoise (mean,var)
    A=rgb2gray(imread('image8.jpg'));
    [m,n,~]=size(A);
    y=mean+sqrt(var)*randn(m,n); %Gaussian distribution matrix
    B=double(A)/255;%for easier calculation
    B=B+y;%add noise
    B=B*255;%transform to 0~255
    B=uint8(B);%transform to uint8
end
```

The code block below shows how to filter Gaussian noise.A Gaussian template(kernel) is arranged to implement convolution over the image.By replace every pixel with the convolution result, we can get the filtered image.

```matlab
function[B] = myGaussianFilter(A)
    [row,col]=size(A);
    sigma = 1.6;
    N = 9;
    N_row = 2*N+1;

    %Gaussian template
    H = [];
    for i=1:N_row
        for j=1:N_row
            numerator=double((i-N-1)^2+(j-N-1)^2);
            H(i,j)=exp(-numerator/(2*sigma*sigma))/(2*pi*sigma);
        end
    end
    H=H/sum(H(:));  %normalization

    %processed image
    B=zeros(row,col);
    M=zeros(row+2*N,col+2*N);
    for i=1:row
        for j=1:col
            M(i+N,j+N)=A(i,j);
        end
    end

    temp=[];
    for ai=N+1:row+N
        for aj=N+1:col+N
            temp_row=ai-N;
            temp_col=aj-N;
            temp=0;
            for bi=1:N_row
                for bj=1:N_row
                    temp= temp+(M(temp_row+bi-1,temp_col+bj-1)*H(bi,bj));
                end
            end
            B(temp_row,temp_col)=temp;
        end
    end
    B=uint8(B);

end
```

The code block below displays how to get the result.

```matlab
A=rgb2gray(imread('image8.jpg'));
B=myGaussianNoise (0,0.01);
C=myGaussianNoise (0,0.03);
D=myGaussianNoise (0.2,0.01);
E=myGaussianNoise (0.2,0.03);
B1= myGaussianFilter(B);
C1= myGaussianFilter(C);
D1= myGaussianFilter(D);
E1= myGaussianFilter(E);

subplot(5,3,1),imshow(A),title('The␣original␣image');
subplot(5,3,2),imhist(A);
subplot(5,3,4),imshow(B),title('mean␣value=0,variance=0.01');
subplot(5,3,5),imhist(B);
subplot(5,3,6),imshow(B1),title('Replica␣of␣the␣left␣polluted␣image');
subplot(5,3,7),imshow(C),title('mean␣value=0,variance=0.03');
subplot(5,3,8),imhist(C);
subplot(5,3,9),imshow(C1),title('Replica␣of␣the␣left␣polluted␣image');
subplot(5,3,10),imshow(D),title('mean␣value=0.2,variance=0.01');
subplot(5,3,11),imhist(D);
subplot(5,3,12),imshow(D1),title('Replica␣of␣the␣left␣polluted␣image');
subplot(5,3,13),imshow(E),title('mean␣value=0.2,variance=0.03');
subplot(5,3,14),imhist(E);
subplot(5,3,15),imshow(E1),title('Replica␣of␣the␣left␣polluted␣image');
```

C. Result

I implemented Gaussian noises addition and Gaussian filtering in Matlab, and I called the existing functions to meet my needs in FIG.7.Also, in FIG.8,I wrote my own functions and implemented it.From both figures, we can see the original image and its histogram along with the images that have been added into different Gaussian noises.With the mean value increasing, the whole image becomes brighter, and the histogram moves right along the axis entirely in the original shape.With the variance increasing, the granular noises get more obvious, and the histogram possesses a wider distribution along the axis. I used the inserted Gaussian filtering function together with my own filter function to replicate the images, and it works well.

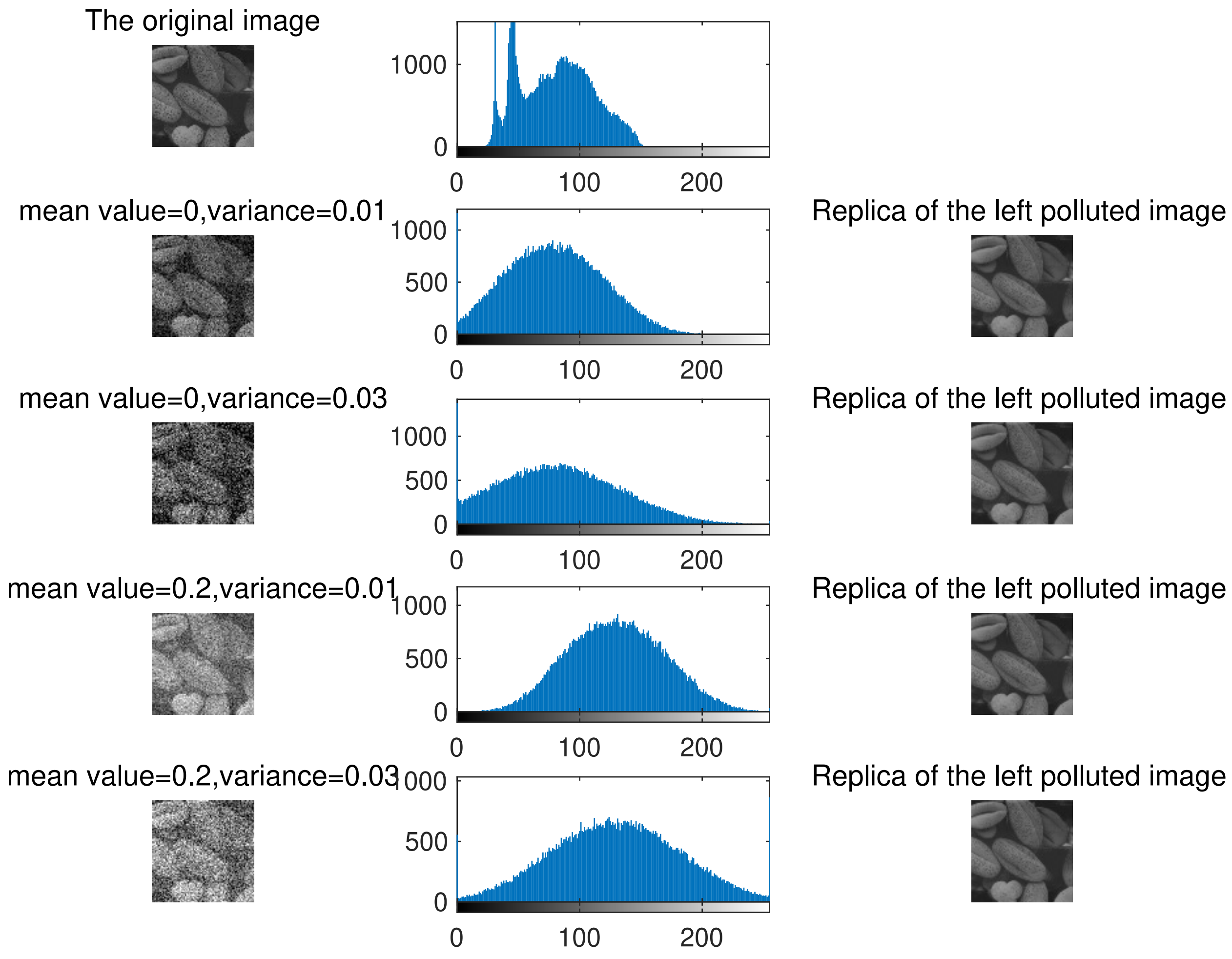

FIG. 7: The result of Problem 8 in Matlab using existing functions

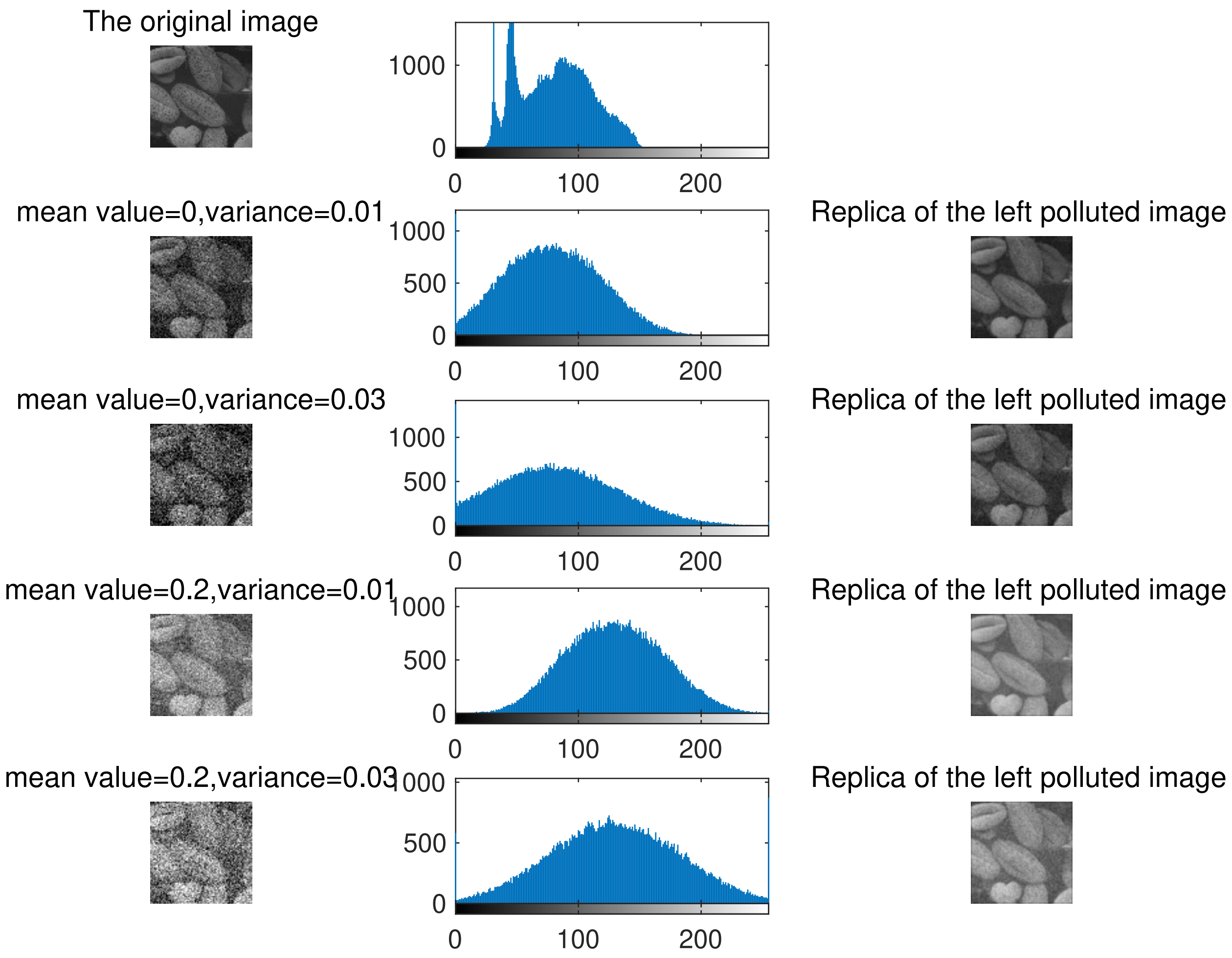

FIG. 8: The result of Problem 8 in Matlab using functions of my own