数字图像处理　Homework1

1552746 崔鹤洁　　　**Date: 2017-10-10**

> ## Problem1

(1)PPT43

a. 求期望

当 $X \sim N(\mu, \sigma^2)$，换元, 令 $t = \frac{x-\mu}{\sigma}$

$$E(X) = \int_{-\infty}^{+\infty} x \frac{1}{\sqrt{2\pi}\sigma} exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}\mathrm{d}x$$

$$= \int_{-\infty}^{+\infty} (\sigma t + \mu) \frac{1}{\sqrt{2\pi}} exp\{-\frac{t^2}{2}\}\mathrm{d}t$$

$$= \sigma \int_{-\infty}^{+\infty} \frac{t}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\,\mathrm{d}t + \mu \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\,\mathrm{d}t$$

分析：

由于 $\frac{t}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}$ 是奇函数，在对称区间上积分值为0；

同时，$\frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}$ 为标准正态分布，即 $\mu = 0, \sigma = 1$ 时的正态分布概率函数的结果，所以其在 $-\infty$ 到 $+\infty$ 上积分值为1。

故上式最终结果为 μ，　即　　$E(X) = \mu$

b. 求方差

由数学期望的性质，可以得到

$$D(x) = E[x - E(x)]^2$$

$$= E\{X^2 - 2XE(X) + [E(x)]^2\}$$

$$= E(X^2) - 2[E(X)]^2 + [E(x)]^2$$

$$= E(X^2) - (E(X))^2$$

由（A）内的推导过程，可以知道，当$X \sim N(\mu, \sigma^2)$时，$E(X) = \mu$

又因为

$$E(X^2) = \int_{-\infty}^{+\infty} x^2 \frac{1}{\sqrt{2\pi}\sigma} exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}\mathrm{d}x$$

$$= \int_{-\infty}^{+\infty} (\sigma t + \mu)^2 \frac{1}{\sqrt{2\pi}} exp\{-\frac{t^2}{2}\}\mathrm{d}t$$

$$= \sigma^2 \int_{-\infty}^{+\infty} \frac{t^2}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\,\mathrm{d}t + \mu^2$$

$$= \sigma^2 + \mu^2$$

因此，正态分布的方差为

$$D(X) = (\sigma^2 + \mu^2) - \mu^2 = \sigma^2$$

(2) PPT120

❖result image:

Analysis:

The algorithm of function **adaptiveMedianFilter** is:

- Get the padding-width according to the cornel window size
- Get the size of the noise picture
- Fill pixels at the edge of the original image,which is assured by padding-width
- Get the temp window area Sxy, and calculate Zemd, Zmin, Zmax of this area
- For each pixel in the original figure, do the following operations:

  Stage A:

  A1 = Zmed – Zmin

  A2 = Zmed – Zmax

  If A1 >0 and A2 <0

    Go to stage B

  Else

    increase the window size

  If window size ≤ Smax

    repeat stage A

Else output Zmed

Stage B:

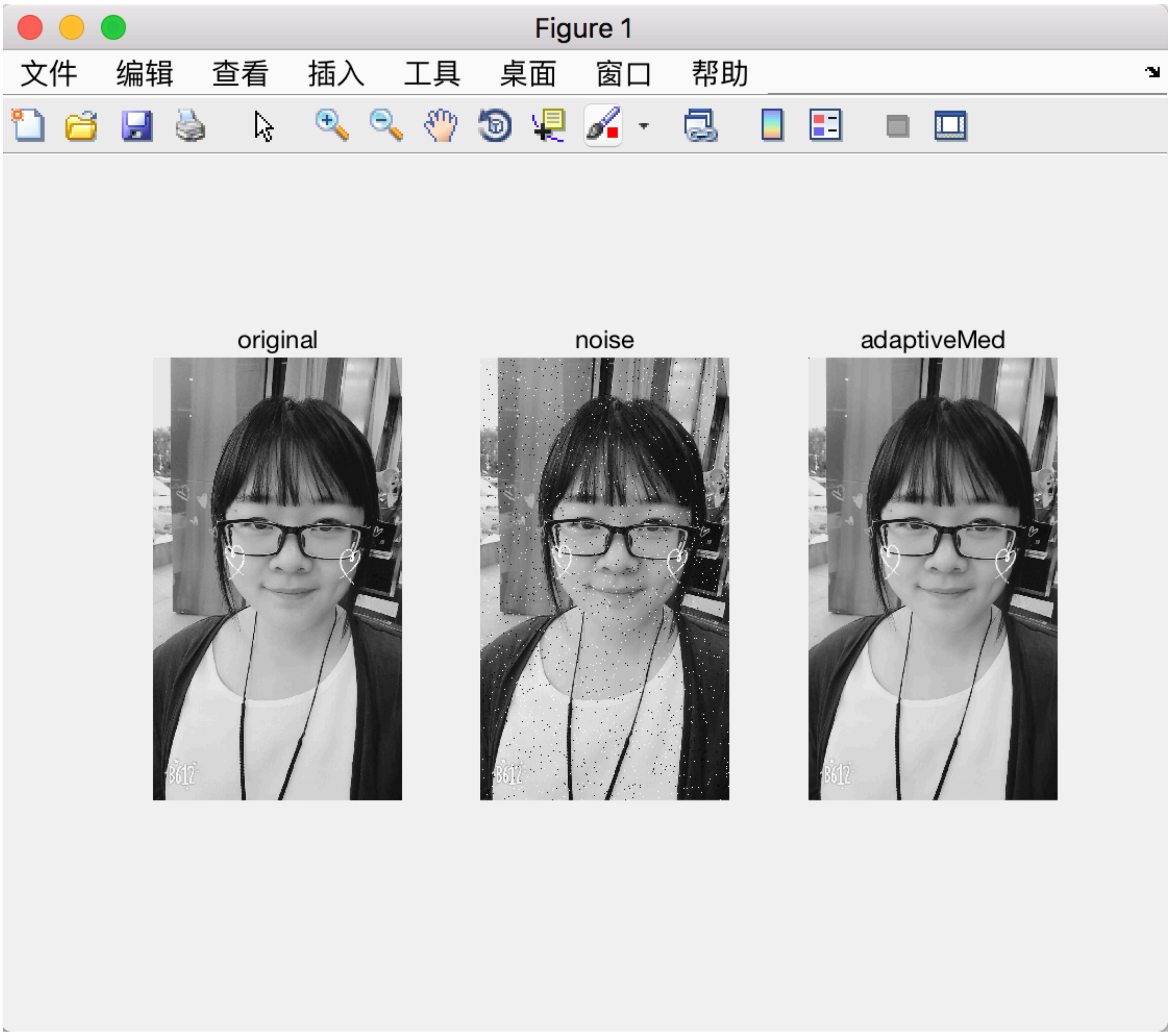
B1 = Zxy – Zmin

B2 = Zxy – Zmax
If B1 > 0 and B2 < 0

output Zxy

Else output Zmed

- If noise pixels in the window were more than half of the useful pixels, then it means you need to increase the size of the filter window Sxy, the increasing method is to make the window size + 2. For example, if the original filter window size is 3 * 3, so after increasing the value of the window, the filter window size changed into 5 x5, if Sxy < Smax, continue this operation, otherwise the gray levels of the pixel is Zmed. By analogy, ***the adaptive control of filter window size is formed***.
- ***The maximum size of filter window Smax*** should adjust with the spacial density of noise. Generally speaking, if the spacial density of the noise is big, we should choose a bigger Smax; if spacial density of the noise is small, Smax can be set a little smaller. Here we choose a fixed value, 11.



code:

```
clear
img = imread('cui.jpg');
img_gray = rgb2gray(img);
img_noi = imnoise(img_gray,'salt & pepper',0.02);
img_adaptive = adaptiveMedianFilter(img_noi,11);

figure;
% original image
subplot(1,3,1);
imshow(img_gray);
title('original')

% noised image
subplot(1,3,2);
imshow(img_noi);
title('noise')

% adaptive median filter image
subplot(1,3,3);
imshow(img_adaptive);
title('adaptiveMed')

function[img_result] = adaptiveMedianFilter(img_noi,max_window)

padding_width = (max_window-1)/2;
[m ,n] = size(img_noi);
% Initialize a image equal size with img_noi to receive pixels after
Adaptive Median Filter
img_result = img_noi;
% Fill pixels at the edge of the original image
img_noi = padarray(img_noi,[padding_width padding_width],0);

for i = padding_width+1:padding_width+m
    for j = padding_width+1:padding_width+n
        Zxy = img_noi(i,j);
        window = 3;
        while(window <= max_window)
            %Take the window of the filter plate
            tmp = img_noi(i-(window-1)/2:i+(window-1)/2,...
            j-(window-1)/2:j+(window-1)/2);
            Zmin = min(min(tmp));
            Zmax = max(max(tmp));
            Zmed = median(tmp(:));
            if((Zmed > Zmin) && (Zmed < Zmax))
                if((Zxy > Zmin) && (Zxy < Zmax))
                    img_result(i-padding_width,j-padding_width) = Zxy;
                else
                    img_result(i-padding_width,j-padding_width) = Zmed;
                end
```

```
                break;
            end
            window = window+2;
        end
        img_result(i-padding_width,j-padding_width) = Zmed;
    end
end
end
```

Problem2

❖result image from the output .gif:

Analysis:

The sigma is changed from 1 to 9; the center is (10,10);

The gaussian expression is

$$f(x,y) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{((x-x_0)^2+(y-y_0)^2)}{2\sigma^2}}$$

$(x_0, y_0)$is the center, σ is the sigma, which changed from 1 to 9 in this ***GaussianFunction.gif***

It can be seen form the .gif that as the sigma grows, the gaussian curve flattens out.

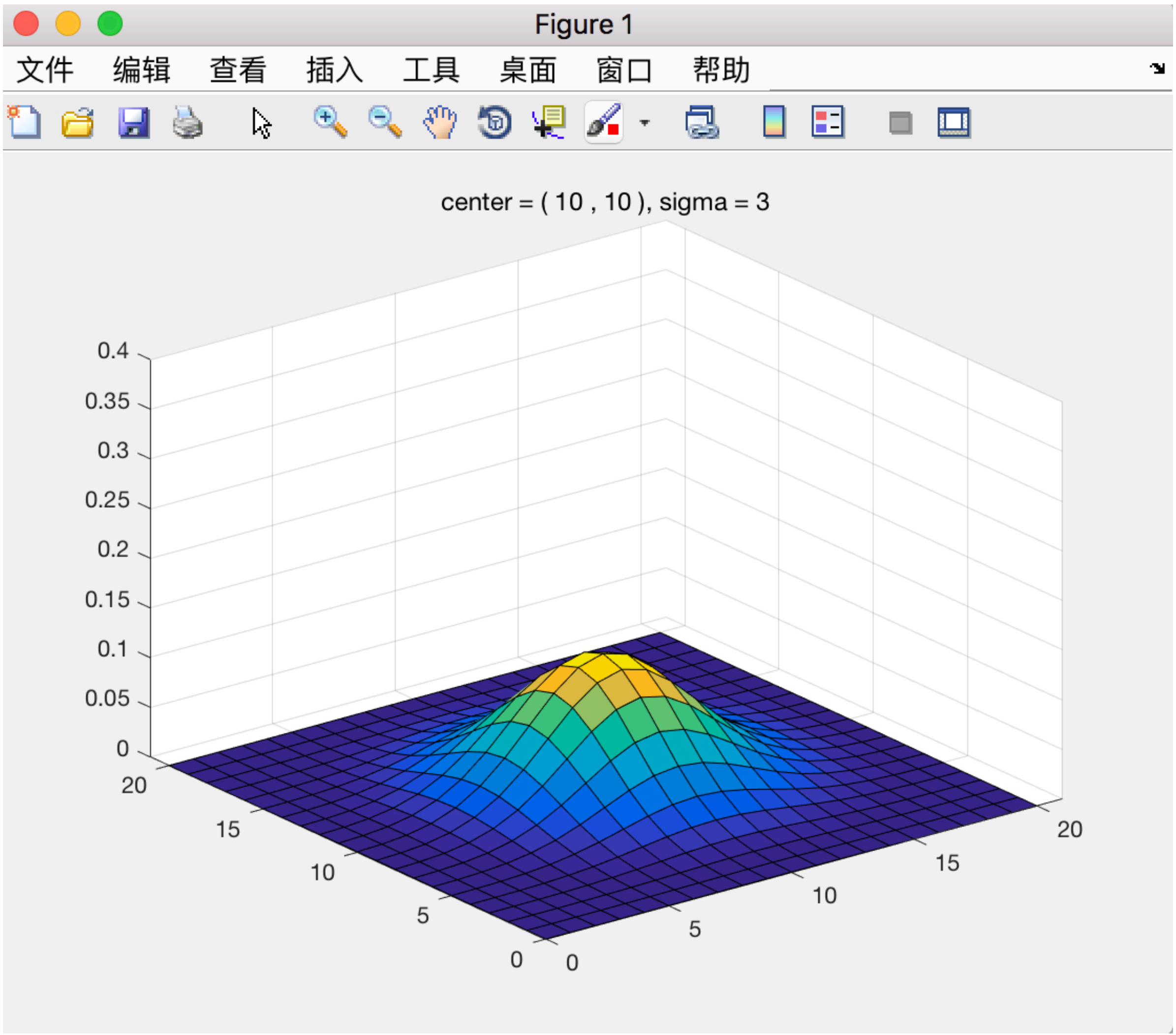


Figure 1
文件
编辑
查看
插入
工具
桌面
窗口
帮助
center = (10 , 10), sigma = 3
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
20
15
10
5
0
0
5
10
15
20

❖code:

```
clear all
clc
X = 0 : 1 : 20;
Y = 0 : 1 : 20;
figure
filename = 'GaussianFunction.gif';

for sigma = 1:9
    % calculate the value of Z
    Z = zeros(21, 21);
    for row = 1 : 1 : 21
        for col = 1 : 1 : 21
            Z(row, col) = (X(row) - 10) .* (X(row)-10) + (Y(col) - 10) .*
(Y(col) - 10);
        end
    end
    Z = -Z/(2*sigma*sigma);
    Z = exp(Z) / (sqrt(2*pi) * sqrt(sigma*sigma));

    % show the gaussian surface
    surf(X, Y, Z);
    title(sprintf(' center = ( 10 , 10 ), sigma = %d ',sigma));
    axis([0 21 0 21 0 0.1]);

    drawnow
    % get the frame
    frame = getframe(gcf);

    %%To make GIF files, images must be index images
    im = frame2im(frame);
    [A,map] = rgb2ind(im,256);

    %create it at the first time
    if sigma == 1
        imwrite(A,map,filename,'gif','LoopCount',Inf,'DelayTime',0.2);
    else
        %DelayTime is used to set up GIF files to play fast or slow
        imwrite(A,map,filename,'gif','WriteMode','append','DelayTime',0.2);
    end
end
```

Problem3

❖result image:

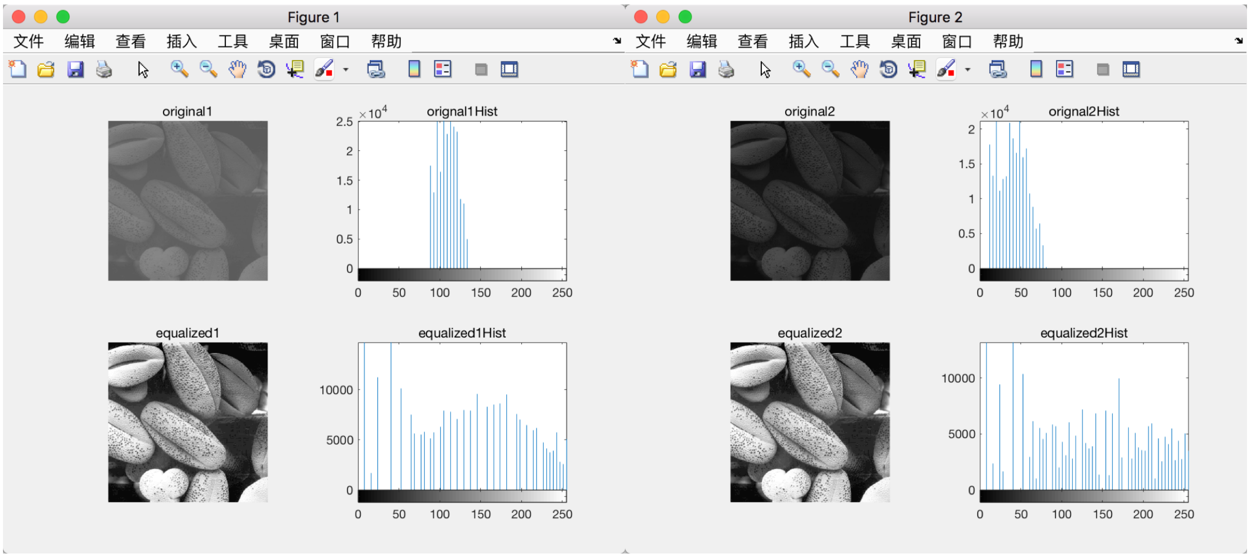
On the left is figure and Hist distribution for imageQ_1_1 before and after histogram equalization;

On the right is figure and Hist distribution for imageQ_1_2 before and after histogram equalization.

Analysis:

- Since it is discrete, equalized figures are not strictly uniform.

- Statistics from local image histograms can be used for local image enhancement; it can enhance dark areas while leaving the light area as unchanged as possible.



✥code:

```
img1=imread('Q_1_1.tif');
img2=imread('Q_1_2.tif');

%Implement the histogram equalization
im_histeq1=histeq(img1);
im_histeq2=histeq(img2);

%show figure1 before equalized
figure(1);
subplot(2,2,1);
imshow(img1);
title('original1')

%show figure1's Hist before equalized
subplot(2,2,2);
imhist(img1,64);
title('orignal1Hist')

%show figure1 after equalized
subplot(2,2,3);
imshow(im_histeq1);
title('equalized1')

%show figure1's Hist after equalized
subplot(2,2,4);
imhist(im_histeq1,64);
title('equalized1Hist')

%show figure2 before equalized
figure(2);
subplot(2,2,1);
imshow(img2);
title('original2')

%show figure2's Hist before equalized
subplot(2,2,2);
imhist(img2,64);
title('orignal2Hist')

%show figure2 after equalized
subplot(2,2,3);
imshow(im_histeq2);
title('equalized2')

%show figure2's Hist after equalized
subplot(2,2,4);
imhist(im_histeq2,64);
title('equalized2Hist')
```

Problem4

✥result image:

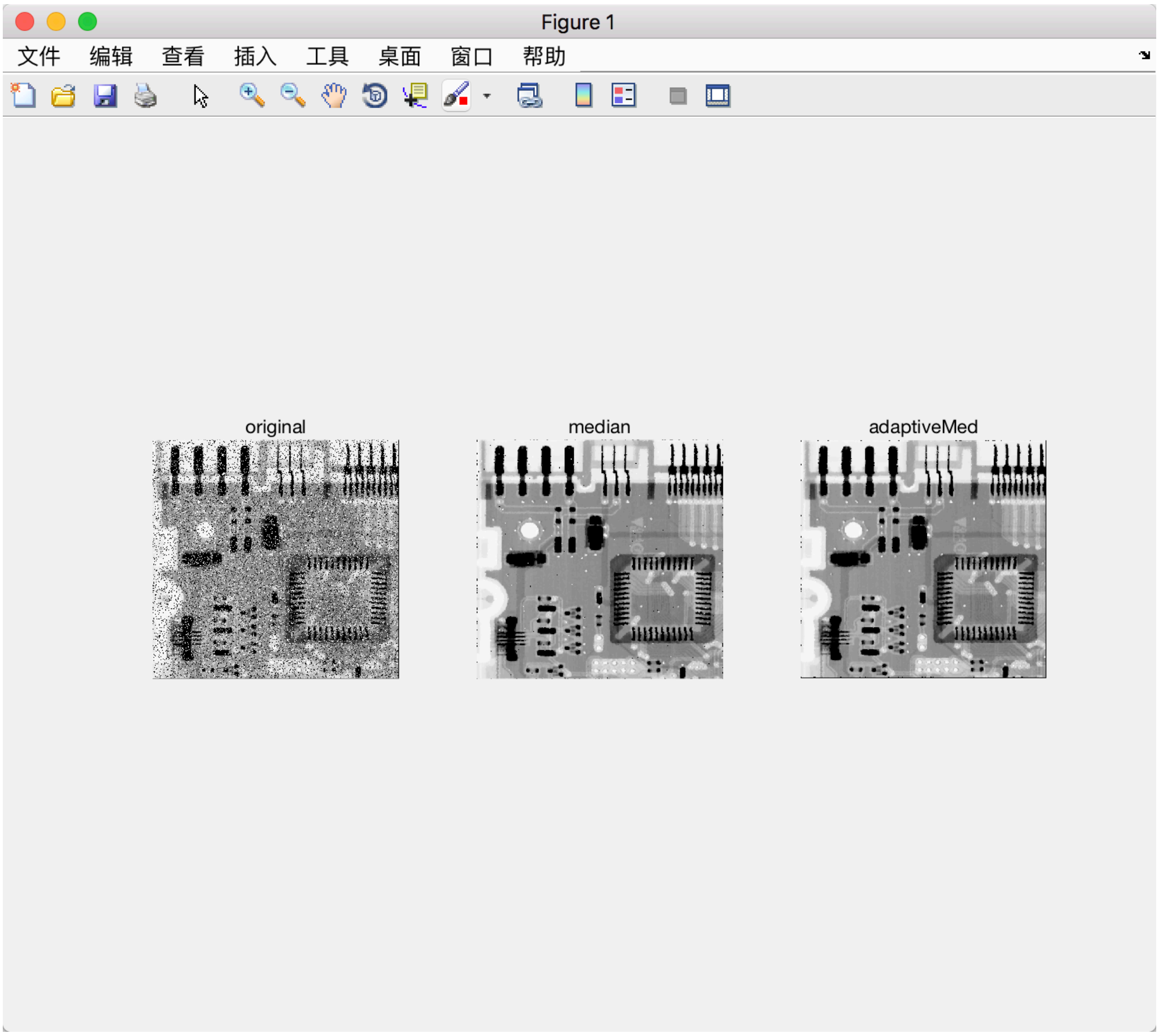
The picture in the middle uses median filter in Matlab;

the picture on the right uses adaptive median filter implemented by myself in ***homework1_(2)***

Analysis:

we can see the result of adaptive median filter have a better result than median filter provided by Matlab.

- The adaptive median filter can handle much more spatially dense impulse noise, and also performs some smoothing for non-impulse noise
- The key insight in the adaptive median filter is that the filtersize changes depending on the characteristics of the image



✥code:

```
img=imread('Q_2.tif');
figure;
% Using median filtering
img_med = medfilt2(img);
```

```matlab
% Using adaptiveMedian filtering
img_adaptive = adaptiveMedianFilter(img,11);

% original image
subplot(1,3,1);
imshow(img);
title('original')

% noised image
subplot(1,3,2);
imshow(img_med);
title('median')

% adaptive median filter image
subplot(1,3,3);
imshow(img_adaptive);
title('adaptiveMed')

function[img_result] = adaptiveMedianFilter(img_noi,max_window)

padding_width = (max_window-1)/2;
[m ,n] = size(img_noi);
% Initialize a image equal size with img_noi to receive pixels after
Adaptive Median Filter
img_result = img_noi;
% Fill pixels at the edge of the original image
img_noi = padarray(img_noi,[padding_width padding_width],0);

for i = padding_width+1:padding_width+m
    for j = padding_width+1:padding_width+n
        Zxy = img_noi(i,j);
        window = 3;
        while(window <= max_window)
            %Take the window of the filter plate
            tmp = img_noi(i-(window-1)/2:i+(window-1)/2,...
            j-(window-1)/2:j+(window-1)/2);
            Zmin = min(min(tmp));
            Zmax = max(max(tmp));
            Zmed = median(tmp(:));
            if((Zmed > Zmin) && (Zmed < Zmax))
                if((Zxy > Zmin) && (Zxy < Zmax))
                    img_result(i-padding_width,j-padding_width) = Zxy;
                else
                    img_result(i-padding_width,j-padding_width) = Zmed;
                end
                break;
            end
            window = window+2;
        end
```

```
        img_result(i-padding_width,j-padding_width) = Zmed;
    end
end
end
```

Problem5

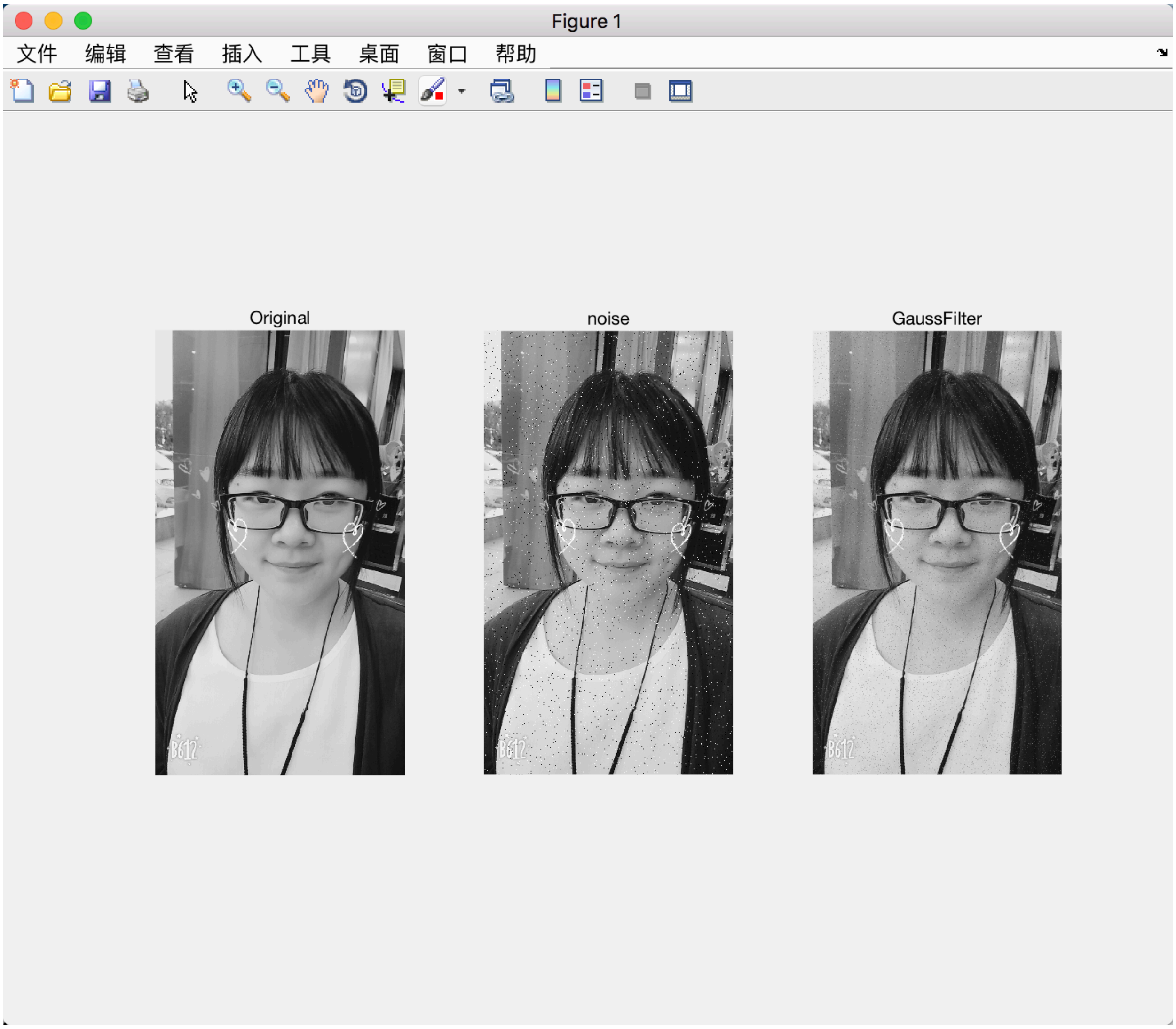
(1) Create a gauss filter of size 4*4, and apply the filter to the image with convolution, padding

✥result image:

Analysis:

According to ***the rule of thumb***: set filter half-width to about 3σ, and the required size is 4, so I set the sigma to 2/3;

The picture on the right is the result image after GaussFilter.



✥code:

Here I provide 3 implements code using different methods:

- Impletment using ***imfilter***:

```
img=imread('cui.jpg');
img_gray = rgb2gray(img);
figure;

% add noise
img_noi = imnoise(img_gray,'salt & pepper',0.02);

% Rule of thumb: set filter half-width to about 3σ
sigma = 0.5;
% create a Gauss filter of size 4x4
gausFilter = fspecial('gaussian', [3,3], sigma);

% Gauss filtering
gaus= imfilter(img_noi, gausFilter, 'replicate');

subplot(1,3,1);
imshow(img_gray);
title('Original')

subplot(1,3,2);
imshow(img_noi);
title('noise')

subplot(1,3,3);
imshow(gaus);
title('GaussFilter')
```

- Impletment using ***conv2***:

```
img=imread('cui.jpg');
img_gray = rgb2gray(img);
figure;

% add noise
img_noi = imnoise(img_gray,'salt & pepper',0.02);

% Rule of thumb: set filter half-width to about 3σ
sigma = 0.5;
% create a Gauss filter of size 4x4
gausFilter = fspecial('gaussian', [3,3], sigma);
% convolution and padding
img_convo = conv2(double(img_noi),gausFilter,'same');
img_convo = uint8(img_convo);

subplot(1,3,1);
imshow(img_gray);
title('Original')

subplot(1,3,2);
imshow(img_noi);
title('noise')

subplot(1,3,3);
imshow(img_convo);
title('GaussFilter')
```

- Impletment convolution and padding by myself:

```
img=imread('cui.jpg');
img_gray = rgb2gray(img);
figure;

% add noise
img_noi = imnoise(img_gray,'salt & pepper',0.02);
img_gaus = img_noi;

% Rule of thumb: set filter half-width to about 3σ
sigma = 0.5;
% create a Gauss filter of size 4x4
gausFilter = fspecial('gaussian', [3,3], sigma);

[m ,n] = size(img_noi);
% padding
padding_width = 2;
img_noi = padarray(img_noi,[padding_width padding_width],0);
% rotate the gausFilter
gausFilter = rot90(gausFilter,2);
% convolution
for i = padding_width+1:padding_width+m
    for j = padding_width+1:padding_width+n
        temp = img_noi(i-1:i+1,j-1:j+1);
        img_gaus(i-padding_width,j-padding_width) =
sum(sum(double(temp).*gausFilter));
    end
end

subplot(1,3,1);
imshow(img_gray);
title('Original')

subplot(1,3,2);
imshow(img_noi);
title('noise')

subplot(1,3,3);
imshow(img_gaus);
title('GaussFilter')
```

(2) Give one intensity transformation function forspreading the intensities of an image such that the lowestis Imin and the highest is Imax , (0Imin Imax 255). Denote by Fmax and Fmin the maximum and minimum intensities values of the input image.

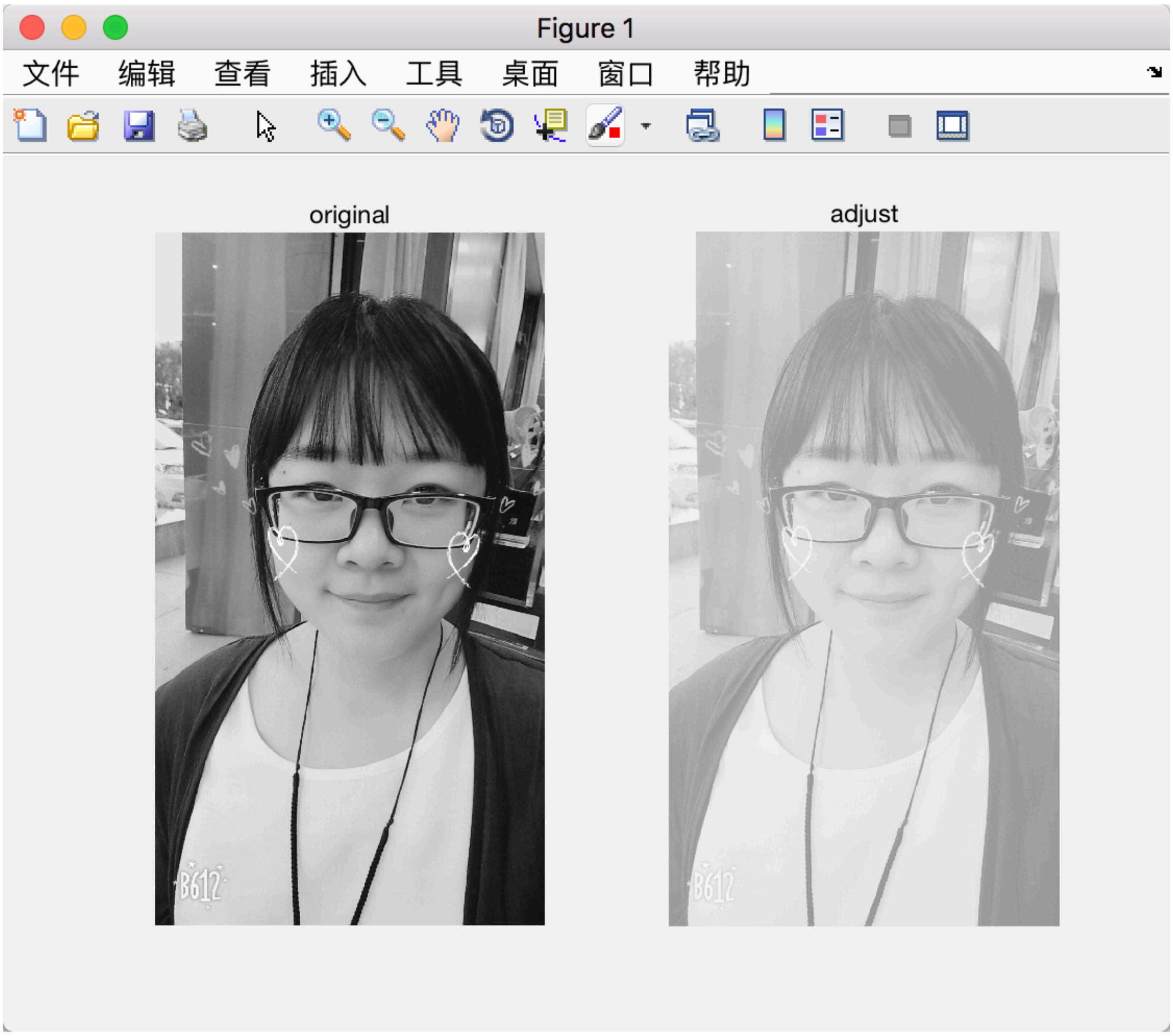
❖result image:

Analysis:

Running the program, it will first let the user input the target range of intensities values(0~255);

Then, it will cast the maximum and minimum intensities values of the input image to the target.

In the code part, I provide 2 implements, one is using Matlab function ***imadjust***, the other is a linear mapping (sigma=1) without using Matlab Function.



✤code:

Here I provide two implement code using different methods:

- Impletment Matlab function ***imadjust:***

```
img=imread('cui.jpg');
img_gray = rgb2gray(img);

% let the user input the target range of intensities values
targetMax = input('Please input target I_max(0~255): ');
targetMin = input('Please input target I_min(0~255): ');
% Change the target range from 0 to 1 to use the adjust function
targetMax = targetMax/256;
targetMin = targetMin/256;

% find the maximum and minimum intensities values of the input image
counts = imhist(img_gray);
minBinValue = find(counts>0, 1, 'first');
minBinValue = minBinValue/256;
maxBinValue = find(counts>0, 1, 'last');
maxBinValue = maxBinValue/256;

% spreading the intensities of the input image
img_adj = imadjust(img_gray,[minBinValue,maxBinValue],
[targetMin,targetMax]);

figure;

subplot(1,2,1);
imshow(img_gray);
title('original')

subplot(1,2,2);
imshow(img_adj);
title('adjust')
```

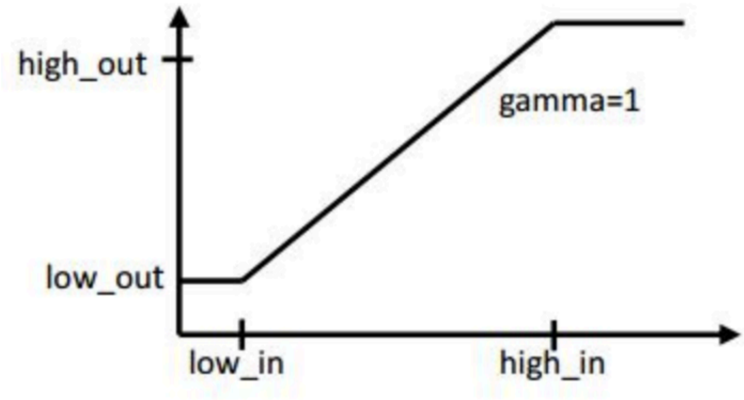
- Impletment without Matlab function ***imadjust***:

Analysis:

Suppose *** J = imadjust(I,[low_in; high_in],[low_out; high_out],gamma)***

Map the brightness value in image I to the new value in J, and the value between low_in and hige_in is mapped to the value between low_out and high_out. The values below low_in and above are cut off, which means that the values below low_in are mapped to low_out, and the values above high_in are mapped to high_out.

The gray mapping process is shown as the followed picture:




```
img=imread('cui.jpg');
img_gray = rgb2gray(img);
img_adj = img_gray;

% let the user input the target range of intensities values
targetMax = input('Please input target I_max(0~255): ');
targetMin = input('Please input target I_min(0~255): ');

% Linear projection imadjust
[m,n] = size(img_gray);
for i = 1:m
    for j = 1:n
        if img_gray(i,j) < targetMin
            img_adj(i,j) = targetMin;
        elseif img_gray(i,j) > targetMax
            img_adj(i,j) = targetMax;
        end
    end
end

figure;
subplot(1,2,1);
imshow(img_gray);
title('original')

subplot(1,2,2);
imshow(img_adj);
title('adjust')
```