数字图像处理第3次实验

数据科学与计算机学院　　17级计科7班（人工智能与大数据方向）

姚森舰　　17341189

实验要求

完成 Project 4-01~4-05，并以 figure4.4 为例检验谱平面随着图像旋转而旋转的性质。

实验过程及分析

PROJECT 04-01

(a) Multiply the input image by (-1)x+y to center the transform for filtering.

(b) Multiply the resulting (complex) array by a real function (in the sense that the the real coefficients multiply both the real and imaginary parts of the transforms). Recall that multiplication of two images is done on pairs of corresponding elements.

(c) Compute the inverse Fourier transform.

(d) Multiply the result by (-1)x+y and take the real part.

(e) Compute the spectrum.

由于MATLAB要求将函数放在代码尾部，所以此部分在代码文件的后面。

(a)

傅里叶变换的第一步就是用 $(-1)^{x+y}$ 乘以图像，做频谱中心化，这部分直接用循环即可，要注意的是必须先将图片转换为 `double` 类型，否则后面可能会出现奇怪的问题。此外，这部分其实可以用矩阵操作实现，但是感觉将矩阵放在指数上有点奇怪，所以还是仅使用了循环实现。

```
% 4.1.a
% 频谱中心化
% 输入为图像
% 输出为中心化后的图像
function  im = shift_to_center(img)
    [m, n] = size(img);
    % 不加double有问题，后面每个像素值会变成两倍？
    im = double(img);
    for i =  1:m
        for j = 1:n
            im(i,j) = im(i,j)*((-1)^(i+j));
        end
    end
end
```

(b)

这部分主要是题意理解的不是很清楚，我的理解是用滤波去乘以傅里叶频谱（复数）。

```
% 4.1.b
% 输入H 是频率滤波，F是傅里叶变换后的频谱
% 不是很理解题目的意思，这好像不用单独写一个函数...
function Y = mul(H,F)
    Y = H .* F;
end
```

(c)

直接调库，其余中心化，取实部的操作在函数外面完成。

```
% 4.1.c
% 计算傅里叶反变化
% 输入是图像矩阵，直接调库，返回傅里叶频谱，复数
function im2= my_ifft(im)
im2 = ifft2(im);
end
```

(d)

```
% 4.1.d
% 反变换的结果取实部后平移
function im = shift_ifft(res)
    im = shift_to_center(real(res));
end
```

(e)

计算傅里叶变换的全部操作， 返回的第一个参数是频谱，第二个是归一化频谱，为了使显示频谱时更加清晰而做。

```
% 4.1.e
% 正变换算频谱，整个流程
function [im, im_norm] = compute_spectrum(img)
% 中心化
im = shift_to_center(img);
% fft2 结果是复数，求模
im = abs(fft2(im));
% 这些值比较大，+1取log，或者做其他合适的操作
% im = double(log(im+1));
im = double( im .^ 0.2);
% 归一化
min_v = min(min(im));
max_v = max(max(im));
im_norm = double( (im-min_v) ./ (max_v-min_v) );
end
```

归一化使用以下方法：

$$N(i,j) = \frac{I(i,j) - min}{max - min} * 255 \tag{1}$$

PROJECT 04-02

Fourier Spectrum and Average Value Top

(a) Download Fig. 4.18(a) and compute its (centered) Fourier spectrum.

(b) Display the spectrum.

(c) Use your result in (a) to compute the average value of the image.

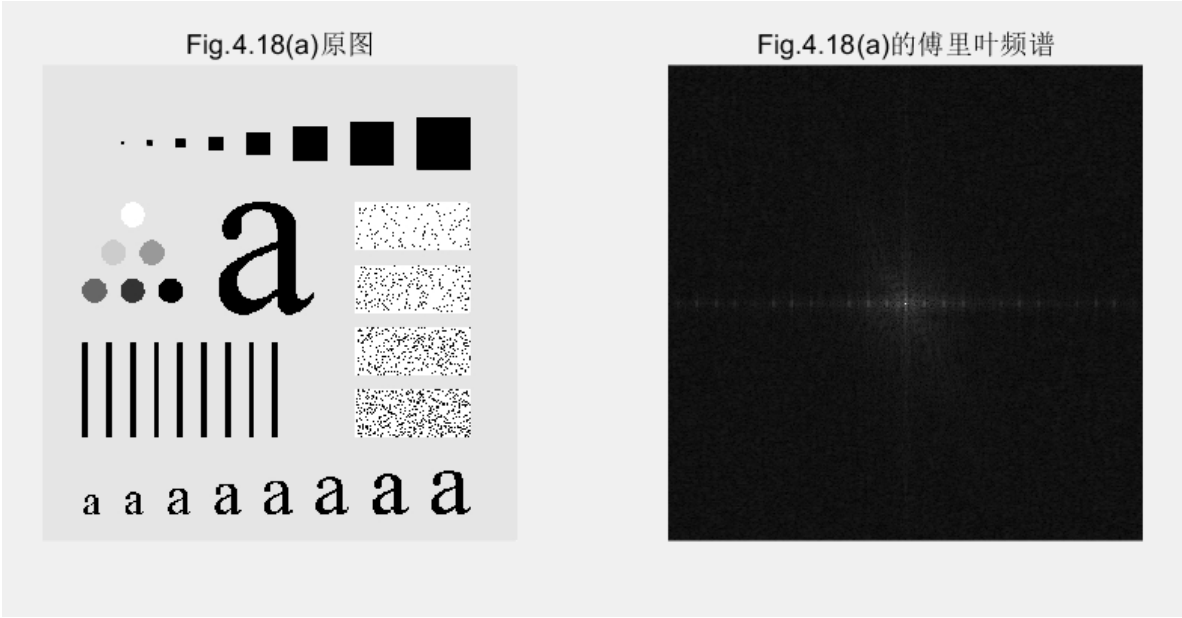
(a)

直接调用写好的函数：

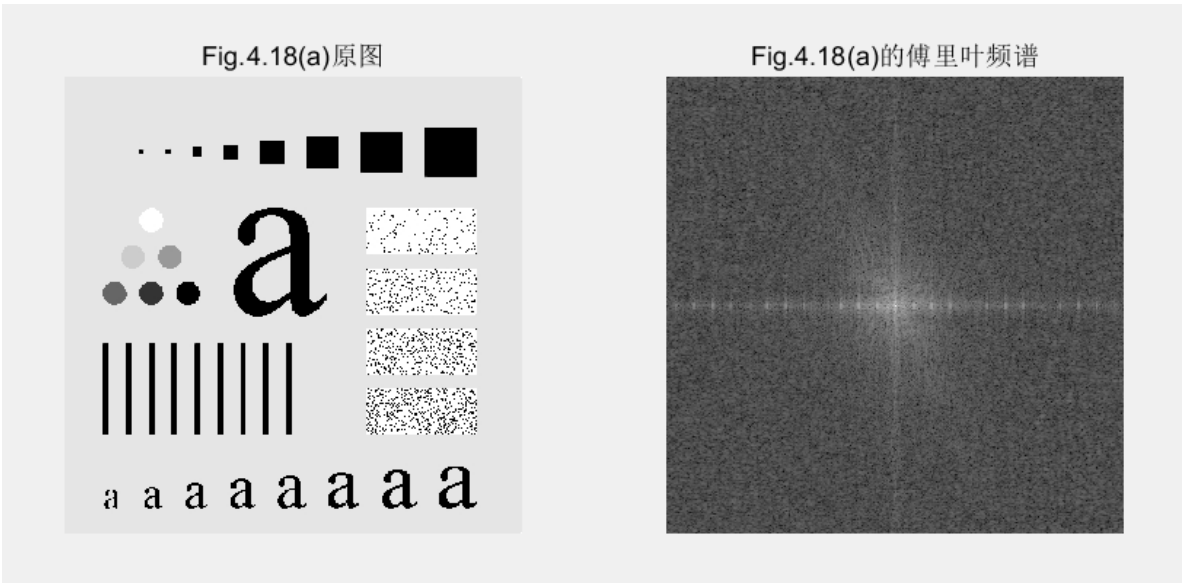
```
% 4.2.a
im = imread('Fig4.18(a).jpg');
F = im;
F_norm = im;
% 计算频谱，第二个是归一化的频谱
[F, F_norm] = compute_spectrum(im);
```

(b)

显示频谱没什么特别要注意的地方，运行结果如下：



为了显示更清晰，显示 `F_norm`，归一化后的频谱：



(c)

由于上面的函数内部做了中心化，取模等等操作，数值可能有缩放，所以重新拿原始图片做傅里叶变换的结果来计算均值，同时与常规方法的结果做了一下对比。

```
[m,n] = size(im);
sum0 = sum(sum(im));
avg = sum0 / m / n
im = abs(fft2(im));
avg2 = im(1,1) / m / n
```

运行结果：

```
avg =

 207.3635


avg2 =

 207.3635
```

PROJECT 04-03

Lowpass Filtering Top

(a) Implement the Gaussian lowpass filter in Eq. (4.3-7). You must be able to specify the size, *M* x *N*, of the resulting 2D function. In addition, you must be able to specify where the 2D location of the center of the Gaussian function.

(b) Download Fig. 4.11(a) [this image is the same as Fig. 4.18(a)] and lowpass filter it to obtain Fig. 4.18(c).

上面提到的高斯低通滤波的式(4.3.7)为：

$$H(u,v) = e^{\frac{D^2(u,v)}{-2\sigma^2}} \tag{2}$$

其中：

$$D(u,v) = \sqrt{(u-\frac{M}{2})^2 + (v-\frac{N}{2})^2} \tag{3}$$

M,N为图像尺寸，σ 作为参数输入，高斯分布参数σ决定了高斯函数的宽度，也就决定了能通过的频率范围。在频率域上使用滤波对图像处理的步骤如下：

(1) 用$(-1)^{x+y}$乘以输入图像, 做频谱中心化处理;
(2) 计算(1)结果的DFT, 即F(u, v);
(3) 用滤波器函数H(u, v)乘以F(u, v)(在频谱域处理图像);
(5) 得到(4)结果中的实部;
(6) 用$(-1)^{x+y}$乘以(5)中的结果。

(a)

按上面的步骤实现：

```
% 4.3.a
% 高斯低通滤波
```

```
% 输入的参数依次为图像矩阵，平移坐标，高斯低通滤波的标准差
function im_after = gaussian_lowpass(img,x0,y0,sig)
    [m,n] = size(img);
    % 频谱中心化
    img = shift_to_center(img);
    % 正变换
    F= fft2(img);
    % 必须转成double，不然后 H .* F 说不支持复整数
    H = double(img);
    im_after = img;
    % 生成滤波
    for i = 1:m
        for j = 1:n
            D2 = (i-x0)^2 + (j-y0)^2;
            H(i,j) = exp(-0.5* ( D2 / sig^2 ) );
        end
    end
    % 滤波
    im_after = H .* F;
    % 反变换
    im_after = ifft2(im_after);
    % 平移
    im_after = shift_ifft(im_after);
end
```

(b)

调用上面的函数，取 $\sigma = 15, x_0 = M/2, y_0 = N/2$，并显示图像：

Fig4.11(a)原图

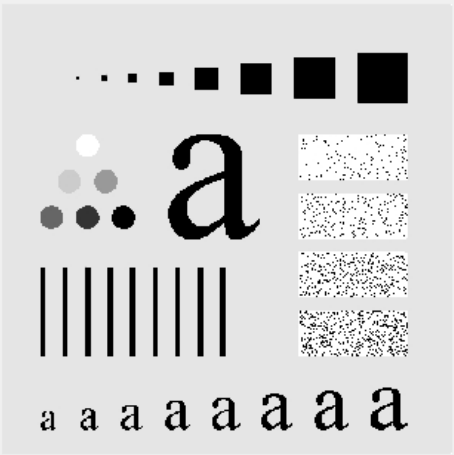


Fig4.11(a)经过高斯低通滤波后

可以看到高斯低通滤波处理后的图片变模糊了，图片中低频部分，也就是变化缓慢的部分留了下来。

PROJECT 04-04

Highpass Filtering Using a Lowpass Image *Top*

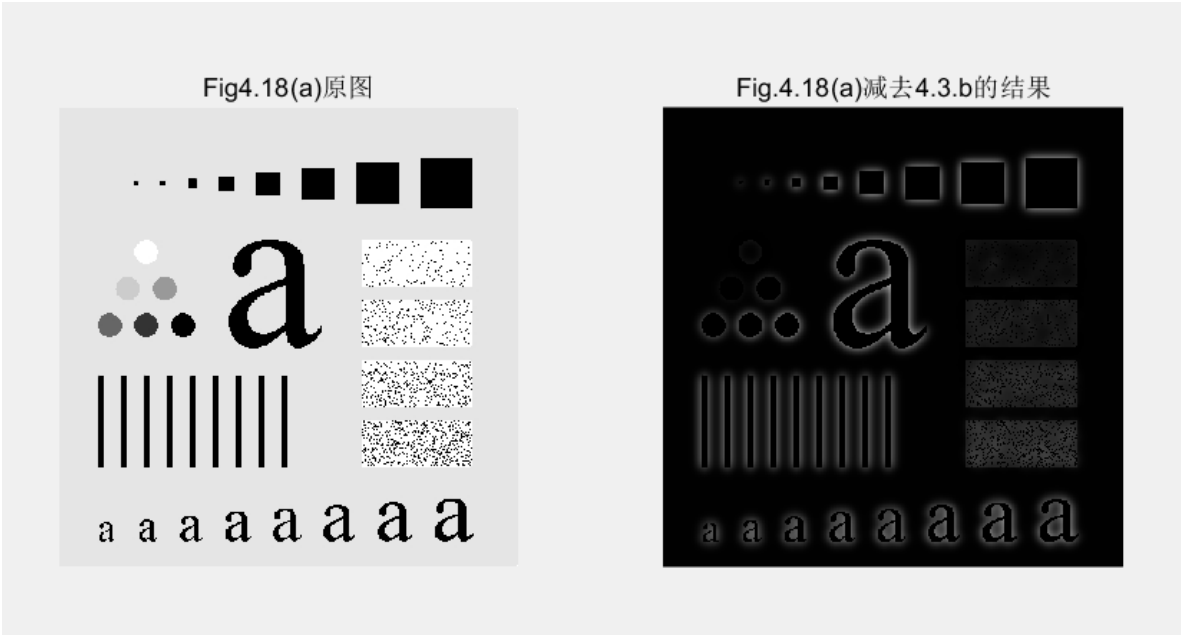
(a) Subtract your image in Project 04-03(b) from the original to obtain a sharpened image, as in Eq. (4.4-14). You will note that the resulting image does not resemble the Gaussian highpass results in Fig. 4.26. Explain why this is so.

(b) Adjust the variance of your Gaussian lowpass filter until the result obtained by image subtraction looks similar to Fig. 4.26(c). Explain your result.

(a)

```
% 4.4.a
% 减去上面的结果
img44a = uint8(im) - uint8(im_after);
figure;
subplot(121);
imshow(im);
title('Fig4.18(a)原图');
subplot(122);
imshow(img44a);
title('Fig.4.18(a)减去4.3.b的结果');
```

结果如下：



与题目中的图对比看好像是差不多的，差异可能来源于计算中的舍入误差。

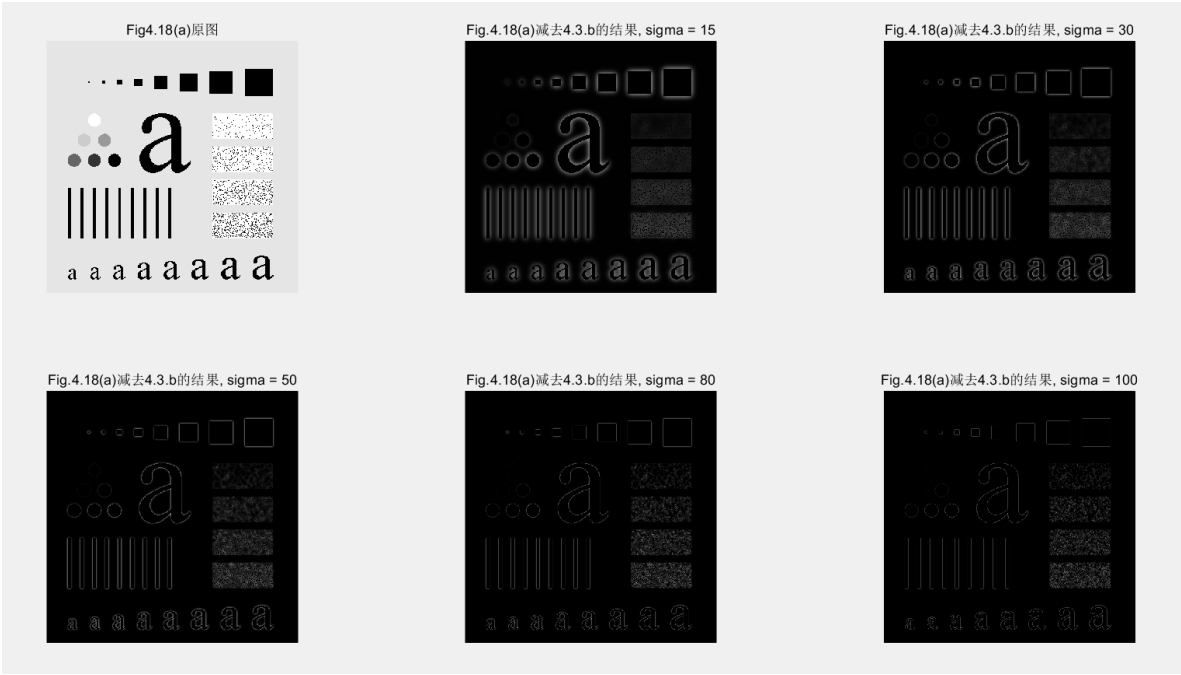
(b)

因为PDF不清晰， Fig. 4.26(c)看得不是很清楚，多试了几个σ， 相对来说σ 越大似乎越接近。从原理上讲，σ 越大，高斯分布越“矮胖”，作用范围越广，留下的成分也就越多，相减得到的细节就相对来说变少。

```
% 4.4.b
% 更改不同的 sigma
figure;
```

```
subplot(231);
imshow(im);
title('Fig4.18(a)原图');
sigmas = [15, 30, 50, 80, 100];
for i = 1:5
    im_after_i = gaussian_lowpass(im, m/2, n/2, sigmas(i));
    img44b = uint8(im) - uint8(im_after_i);
    subplot(2,3,i+1);
    imshow(img44b);
    title(['Fig.4.18(a)减去4.3.b的结果, sigma = ', num2str(sigmas(i))]);
end
```

结果如下：



PROJECT 04-05

Correlation in the Frequency Domain Top

Download Figs. 4.41(a) and (b) and duplicate Example 4.11 to obtain Fig. 4.41(e). Give the (*x*,*y*) coordinates of the location of the maximum value in the 2D correlation function. There is no need to plot the profile in Fig. 4.41(f).

大致的步骤还是差不多的，就是最开始要先算出P，Q，做延拓，然后对两张图片分别做傅里叶变换，把其中一个的结果取共轭后相乘，再做傅里叶反变换得到结果，图中最亮的地方就是相关性最大的地方。用到的原理主要是：

$$f(x,y) \circ h(x,y) = F^*(u,v)H(u,v) \tag{4}$$

实现如下：

```
% 4.5
```

```
im_a = imread('Fig4.41(a).jpg');
im_b = imread('Fig4.41(b).jpg');
[m1,n1] = size(im_a)
[m2,n2] = size(im_b)
P = m1 + m2;
Q = n1+ n2;
im_expand1 = zeros(P,Q);
im_expand2 = zeros(P,Q);
% 扩展图片
im_expand1(1:m1,1:n1) = im_a;
im_expand2(1:m2,1:n2) = im_b;
% 中心化
im_expand1 = shift_to_center(im_expand1);
im_expand2 = shift_to_center(im_expand2);
% 傅里叶正变化
F1 = fft2(im_expand1);
F2 = fft2(im_expand2);
% 其中一个取共轭
% 不过交换后结果不同，F1取共轭更接近真实情况
correlation = F2 .* conj(F1);
% 反变化后移动到中心
result = shift_ifft(ifft2(correlation));

figure;
subplot(121);
imshow(uint8(im_a));
title('Fig4.41(a)原图');
subplot(122);
imshow(uint8(im_b));
title('Fig4.41(b)原图');
figure;
imshow(result,[]);
title('Fig4.41(a)(b)相关性')
% 相关性最大的地方
maximum  = max(max(result));
[row,col] = find(result == maximum);
% 取出相关性最高的一块
% pic = result(160:220,[160:220]);
% imshow(pic,[]);
disp([row,col]);
```

结果:

Fig4.41(a)原图

Fig4.41(b)原图

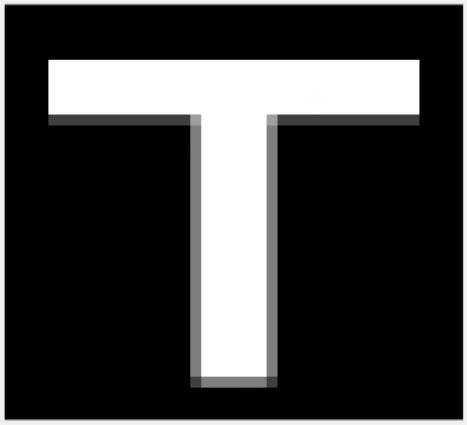


Fig4.41(a)(b)相关性

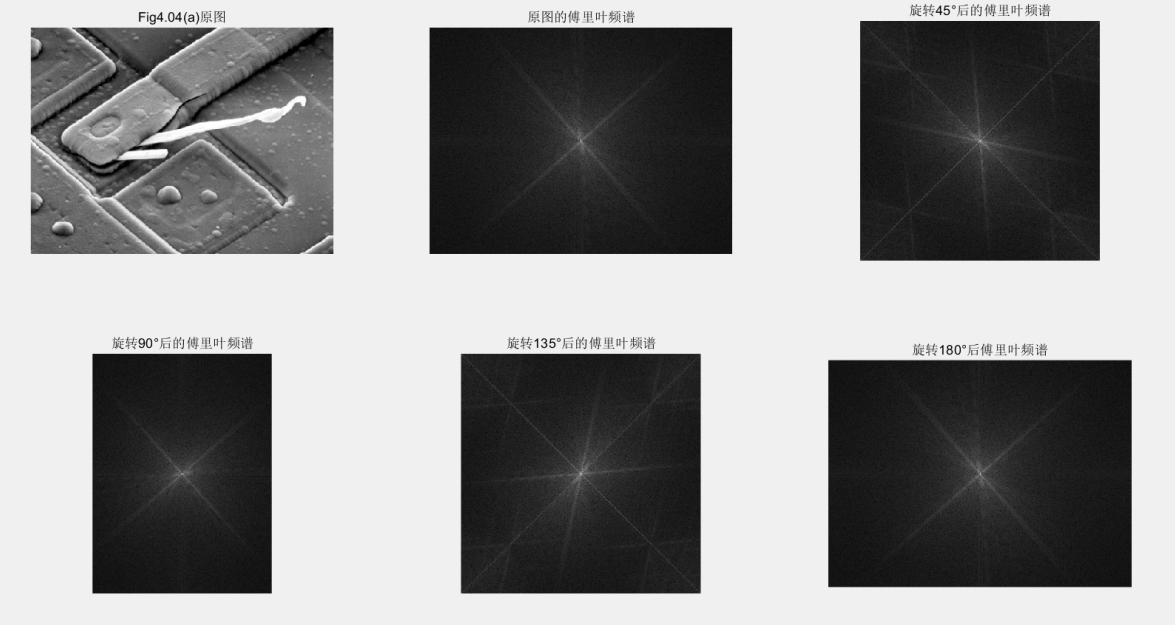
最后得到的相关性最大的点坐标为`[187,193]`.

检验谱平面随着图像旋转而旋转的性质

先将图片旋转不同角度，然后做傅里叶变换，计算频谱。

```
% 检测旋转后傅里叶频谱的变化
im404 = imread('Fig4.04(a).jpg');
figure,
subplot(231),imshow(im404);
title('Fig4.04(a)原图')
[F0,F00] = compute_spectrum(im404);
subplot(232);
imshow(F00);
title('原图的傅里叶频谱')
for i = 1:4
    im404_45 = imrotate(im404,45*i);
    [F1,F11] = compute_spectrum(im404_45);
    subplot(2,3,i+2);
    imshow(F11);
    title(['旋转',num2str(i*45), '°后的傅里叶频谱'])
end
```

结果如下：



这里在显示频谱时，为了看得更清晰，对灰度值做了一些缩放得处理。可以看到，频谱的图片也会随着原图的旋转而旋转。

总结

通过实验对傅里叶变换的整个流程熟悉了很多，遇到得主要问题是，做运算前要将图片转换为`double`类型，一开始有的地方用`double()`，有的地方用`im2double()`,但是后者会对灰度做缩放，所以有时结果有问题，后来统一改成了`double()`，需要时自己做归一化，然后在显示图片的时候为了显示得更清楚有些地方也会做处理。

此外，在实验过程中遇到一个没法解释得问题，记录一下。当我运行下面得代码时：

```
img = imread('Fig4.11(a).jpg');
disp(img)
subplot(121);
imshow(img);
title('Fig4.18(a)原图');

subplot(122);
m = ones([500,500]).*229;
imshow(m);
title('test')
```


```
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 228 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 230 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 228 230 229 229
229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 229 254
```

从结果可以看到`Fig4.18(a)`右下角的部分灰度值普遍在200以上，但是显示出来却是灰色，我自己创建了一个每个灰度值都为229的矩阵，显示出来是白色的！不知道是不是跟图片格式有关，不是很清楚。