数字图像处理实验报告四

姓名：鲁国锐　　学号：17020021031　　专业：电子信息科学与技术

2020 年 4 月 24 日

目录

1 实验目标

1. 熟悉图像复原的各种方法；
2. 复习并实现各种相关滤波器（巴特沃斯、高斯等）；
3. 利用滤波器去除周期性噪声；
4. 学习并使用自适应滤波器。

2 实验一

Test 目录下有图像 *windmill_noise.png*，用 *Matlab* 写程序，去除条纹干扰。

2.1 思路

参考 [1] 的思路：

1. 构建一个类，对原图进行傅里叶变换，并将相关的变量都存进成员变量中；
2. 计算沿频谱图行、列方向上的累积分布函数；
3. 显示连个累积分布函数，并在图中标出所有波峰和波谷；
4. 分别在两幅图中找出离中心波峰最近且相对突出的波峰作为噪声的位置，求出噪声在频谱图中到中心的距离；
5. 将求得的距离作为截止频率对原图进行巴特沃斯带阻滤波和高斯带阻滤波器；
6. 显示滤波后的图像及其频谱图。

2.2 代码实现

Listing 1: *remove_waves* 类

```
classdef remove_waves < handle
    % 必须加handle, 不然不能再构造函数以外的函数里改变成员变量
    properties( SetAccess = public, GetAccess = public )
        im_path; % 完整的路径
        path; % 图像所在目录
        im_name; % 图像文件名
        ext; % 图像扩展名
        im_orig; % 原图
        im_double; % 把原图转换成double
        M; % 行数
        N; % 列数
        F; % 原图的傅里叶变换
        logF; % log(1+abs(F))
        spectrum; % 把F的值映射到0~1范围
        SDv;
        SDu;
        SDv_pk; % SDv的峰值
        SDv_pkloc; % SDv峰值的位置
        SDv_valleyloc; % SDv波谷的位置
```

```
        SDu_pk;% SDu的峰值
        SDu_pkloc; % SDu峰值的位置
        SDu_valleyloc; % SDu波谷的位置
    end

    methods
        % 构造函数
        function obj = remove_waves(img_path)
            obj.im_path = img_path;
            [obj.path, obj.im_name, obj.ext]  = fileparts(img_path);
            obj.path = strcat(obj.path, '\');
            obj.im_orig = imread(img_path); % 读取图像
            obj.im_double = im2double(obj.im_orig); % 将图像转为double
            [obj.M, obj.N] = size(obj.im_orig); % 得到图像的行数、列数
            % 计算原图的傅里叶变换
            obj.F = fft2(obj.im_double);
            obj.F = fftshift(obj.F);
            % 把频谱图的值映射到0~1范围
            obj.logF = log(1+abs(obj.F));
            obj.spectrum = obj.logF / max(max(obj.logF));
            % 计算累计分布函数
            obj.projection();
            % 显示原图及频谱
            h = figure
            subplot(2, 1, 1), imshow(obj.im_orig), title('原图')
            subplot(2, 1, 2), imshow(obj.spectrum), title('原图频谱图')
            % 使用print函数保存图像，可以保存为更高的分辨率（600）
            print(h, '-dpng', '-r600', strcat(obj.path, 'spectrum_of_img.png'))
        end
    end



    methods
        % 计算投影图
        function projection(obj)
            % 初始化两个方向的累计函数值
            obj.SDv = zeros(1, obj.N);
            obj.SDu = zeros(1, obj.M);
            % 计算列方向上的累积分布函数
            for v = 1 : obj.N
                for u = 1 : obj.M
                    obj.SDv(v) = obj.SDv(v) + obj.logF(u, v);
                end
            end
            % 找SDv所有峰值及其坐标
            [obj.SDv_pk, obj.SDv_pkloc] = findpeaks(obj.SDv);
            % 取SDu的相反数，可用findpeaks求得所有波谷的坐标
            [temp, obj.SDv_valleyloc] = findpeaks(-obj.SDv);
            % 计算行方向上的累积分布函数
            for u = 1 : obj.M
```

```
                for v = 1 : obj.N
                    obj.SDu(u) = obj.SDu(u) + obj.logF(u, v);
                end
            end
            % 找SDu所有峰值及其坐标
            [obj.SDu_pk, obj.SDu_pkloc] = findpeaks(obj.SDu);
            % 取SDu的相反数，可用findpeaks求得所有波谷的坐标
            [temp, obj.SDu_valleyloc] = findpeaks(-obj.SDu);

            figure
            % 画出原函数
            subplot(2, 1, 1), plot(obj.SDv), hold on
            % 标出所有的峰值
            subplot(2, 1, 1), plot(obj.SDv_pkloc, obj.SDv_pk, 'r.', 'Markersize', 2), hold on
            % 标出所有波谷
            subplot(2, 1, 1), plot(obj.SDv_valleyloc, obj.SDv(obj.SDv_valleyloc), 'g.', 'Markersize'
                , 2)
            title('列方向累积分布函数'), xlabel('v'), ylabel('S_D(v)')
            % 画出原函数
            subplot(2, 1, 2), plot(obj.SDu), hold on
            % 标出所有的峰值
            subplot(2, 1, 2), plot(obj.SDu_pkloc, obj.SDu_pk, 'r.', 'Markersize', 2), hold on
            % 标出所有波谷
            subplot(2, 1, 2), plot(obj.SDu_valleyloc, obj.SDu(obj.SDu_valleyloc), 'g.', 'Markersize'
                , 2)
            title('行方向累积分布函数'), xlabel('u'), ylabel('S_D(u)')
            name = strcat(obj.path, 'SD.png');
            %saveas(gcf, strcat(obj.path, 'SD.png'))
            % 可以保存成更高分辨率的图像，-r600设置分辨率
            print(gcf, '-dpng', '-r600', name);

        end
    end

end
```

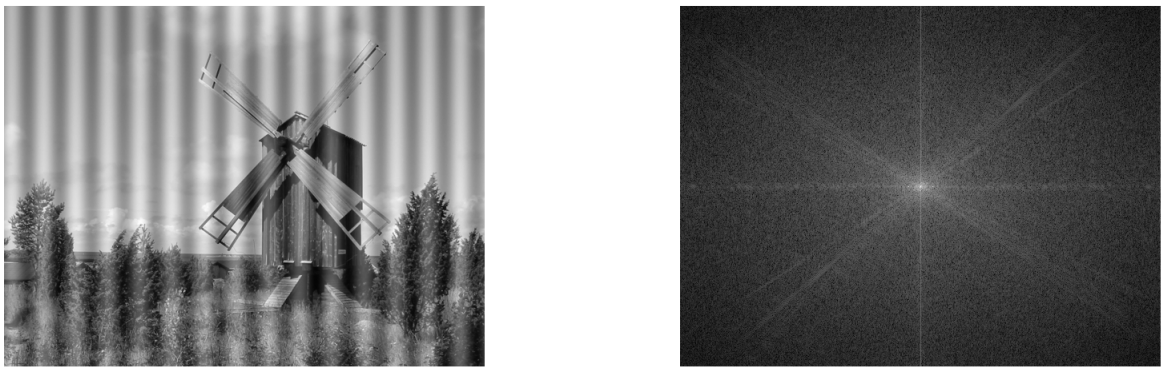
Listing 2: *remove_wave*

```
object = remove_waves(im_path) % 构建remove_waves对象
%找到峰值中心，需要根据具体情况分析出
SDv_pk_cent = floor(length( object.SDv_pk) / 2) + 1;
% 找出噪声频率距离直流分量在频谱图上的水平距离，需要看图来确定
dx = 5;
v = object.SDv_pkloc(SDv_pk_cent) - object.SDv_pkloc(SDv_pk_cent - dx)
% 找出噪声频率距离直流分量在频谱图上的垂直距离，需要看图来确定
dy = 0;
u = 0
% 计算噪声频率距离频谱中心的距离，及截断频率
D0 = sqrt(v^2 + u^2)
% 找出阻带宽度，需要看图来确定
% 找到波谷中心，需要根据具体情况分析
SDv_valley_cent = length(object.SDv_valleyloc) / 2;
```

```
% 计算阻带宽度
w = object.SDv_valleyloc(SDv_valley_cent-3) - object.SDv_pkloc(SDv_pk_cent - 6)

%% 巴特沃斯滤波
n = 3; % 阶数为3
G = object.F; % 复制一份频谱用作滤波
cent = [ object.M/2+1, object.N/2+1 ] % 确定频谱的中心点坐标
for u = 1 : object.M
    for v = 1 : object.N
        d = sqrt( (u-cent(1))^2 + (v-cent(2))^2); % 当前点到中心点的距离
        % 用巴特沃斯滤波器的公式进行滤波
        G(u, v) = 1 / (1 + (w * d / ( d^2 - D0^2 ))^(2*n)) * G(u, v);
    end
end
g=real(ifft2(fftshift(G)));% 对滤波后的频谱进行反变换
h = figure % 指定句柄
% 画出滤波后的空间域图像
subplot(2,1,1),imshow(g,[]);
title('巴特沃斯阻带滤波后的图像')
% 在对滤波后的空间域图像进行傅里叶变换，画出滤波后的频谱图
F=fft2(g);
fc=fftshift(F);
s=log(1+abs(fc));
subplot(2,1,2),imshow(s,[]);
title('巴特沃斯阻带滤波后的频谱图');
print(h, '-dpng', '-r600', strcat(object.path, 'filtered_img.png') )
%% 高斯滤波
G = object.F; % 复制一份频谱用作滤波
for u = 1 : object.M
    for v = 1 : object.N
        d = sqrt( (u-cent(1))^2 + (v-cent(2))^2); % 当前点到中心点的距离
        % 用高斯滤波器的公式进行滤波，1-exp(-1/2*(((d1.^2)-D0^2)./(d1*w)).^2)
        G(u, v) = (1 - exp( -1/2 * ( ( d^2 - D0^2 ) / (d * w) )^2 ) ) * G(u, v);
    end
end
g=real(ifft2(fftshift(G)));% 对滤波后的频谱进行反变换
h = figure % 指定句柄
% 画出滤波后的空间域图像
subplot(2,1,1),imshow(g,[]);
title('高斯阻带滤波后的图像')
% 在对滤波后的空间域图像进行傅里叶变换，画出滤波后的频谱图
F=fft2(g);
fc=fftshift(F);
s=log(1+abs(fc));
subplot(2,1,2),imshow(s,[]);
title('高斯阻带滤波后的频谱图');
print(h, '-dpng', '-r600', strcat(object.path, 'Gaussian_filtered_img.png') )
```

2.3 结果展示



(a) 原图　　(b) 频谱图

图 1: 测试结果

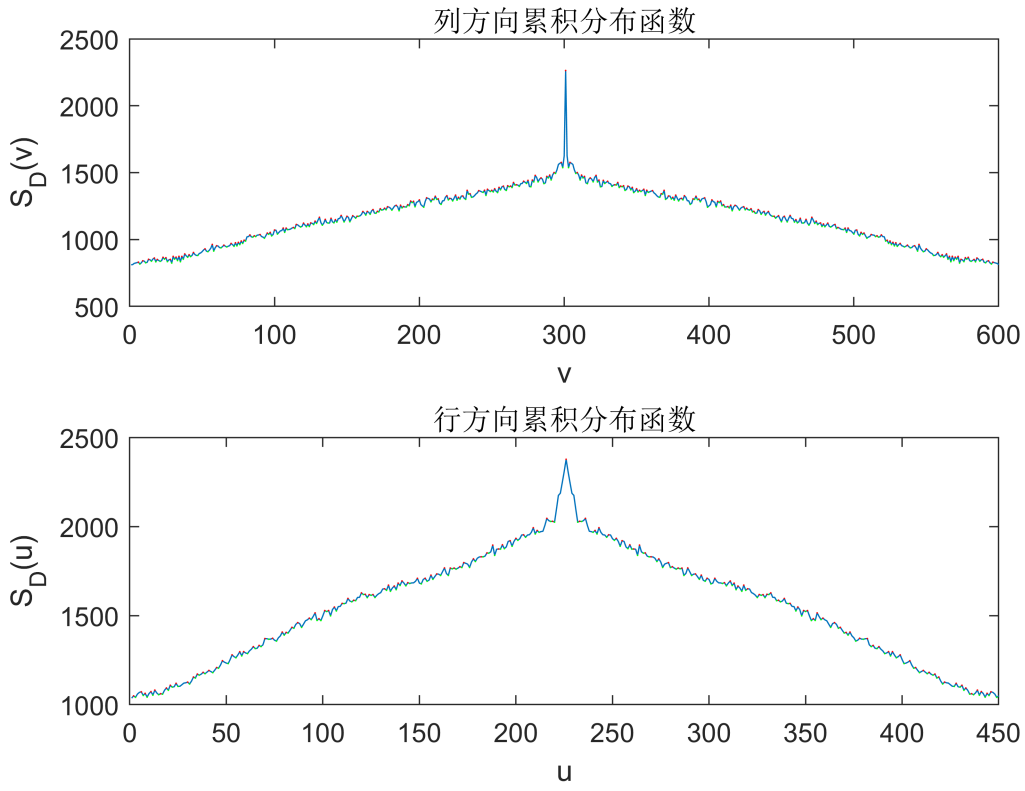
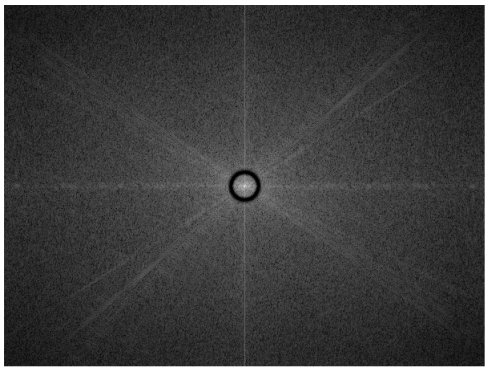


图 2: 累积分布函数

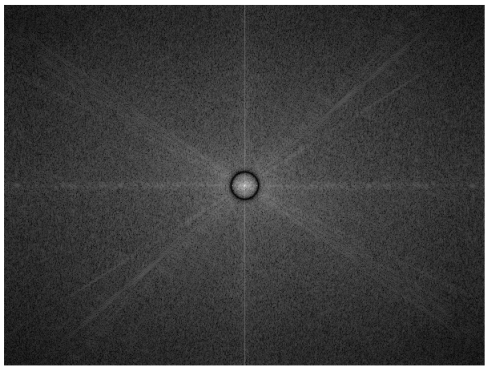
(a) 巴特沃斯带阻滤波后的图像



(b) 巴特沃斯带阻滤波后的频谱图

图 3: 测试结果

(a) 高斯带阻滤波后的图像



(b) 高斯带阻滤波后的频谱图

图 4: 测试结果

3 实验二

Test 目录下有图像 *boardWithNoise.jpg*，用 *Matlab* 写程序，采用自适应中值滤波器去除噪声干扰。

3.1 思路

参考 [2]

4 总结

本次实验分为两个部分：第一个部分要求我们用所学的方法去除一副图像中的条纹干扰；第二个部分是要求我们学习并使用滤波器对另一幅图像进行处理。

第一个部分最大的难点在于如何确定噪声的频率，即准确算出噪声在频谱图中到中心点的距离。关于这一点在实验提供的文献中有一个方案，即分别在行、列方向上计算并画出它的累积分布函数。由于有噪声的存在，在中心峰值的附近应当会有一个相对突出的小波峰。分别在行、列方向的累积分布函数中找到这两个波分的位置，就能确定其行、列坐标。

理论上这样是可行的，从论文中给的图片来看也确实有用。但在本次实验过程中，待处理图片的噪声在频谱中的能量并不突出，因而导致难以从两个累计分布函数中找到对应的小波峰。

此外，如何确定小波峰的坐标也是一个问题。这里我所采用的方法是分别对两个累积分布函数使用 $findpeaks$ 函数。该函数可以返回每一个极大值点的值及其坐标。我们用它找到在累计函数分布图中用红点标出所有小波峰，然后对原函数的相反数在用一遍 $findpeaks$，便可找出所有小波谷的位置，并在图中用绿点标记出来（如图2所示）。之后便需要由人来数出所求小波峰到中心波峰之间的小波峰数目 m，然后再在 $findpeaks$ 返回的波峰坐标中确定中心波峰的位置，再往前推 $m+1$ 个位置即可求得小波峰在原频谱图中的坐标。波谷的位置可用类似的方法求出。

按照上述办法求出来的截止频率误差比较大，主要是由于小波峰不突出，难以确定其坐标，所以最后又稍微调了一下参数。得到如图??和图??所示的结果。可以看出，巴特沃斯带阻滤波的去除效果还不错，而高斯带阻滤波则还有比较明显的噪声残余。

参考文献

[1] 邹园园. 基于频域滤波的 thz 图像条纹噪声处理. 计算机工程与应用, 45(17):241–243, 2009. 2

[2] 齐春亮，马义德. 自适应中值滤波器的实现与在图像降噪中的应用. 中国科技论文在线, 2005. 7