数字图像处理第五次作业

姓名：张璞
班级：自动化 64
学号：2160700034
提交日期：2019 年 4 月 2 日

1. 频域低通滤波器：设计低通滤波器包括 butterworth and Gaussian (选择合适的半径，计算功率谱比),平滑测试图像 test1 和 test2；

1) 问题分析

a) 频率域滤波步骤

①给定一幅大小为 M×N 的输入图像 f（x，y），确定填充参数，典型的选取 P=2M 和 Q=2N；

②对 f（x，y）添加必要数量的 0，形成大小为 P×Q 的填充后的图像 fp（x，y）；

③用（-1）^(x+y)乘以 fp（x，y）移到其变换中心；

④计算来自步骤 3 的图像的 DFT，得到 F(u,v)；

⑤生成一个实的、对称的滤波函数 H(u,v)，其大小为 P×Q，中心在（P/2，Q/2）处，用阵列相乘形成乘积 G(u,v)=H(u,v)F(u,v)；即 G(i,k)=H(i,k)F(i,k)；

⑥得到处理后的图像：

$$g_p(x,y)=\{\text{real}[\Im^{-1}[G(u,v)]]\}(-1)^{x+y}$$

其中，为忽略由于计算不准确导致的寄生复分量，选择了实部，下标 p 指出我们处理的是填充后的阵列。

⑦通过从 gp(x,y)的左上象限提取 M×N 区域，得到最终的处理结果个 g(x,y)。

b) 布特沃斯低通滤波器

截止频率位于距原点 D0 处的 n 阶布特沃斯低通滤波器（BLPF）的传递函数定义为

$$H(u,v)=\frac{1}{1+[D(u,v)/D_0]^{2n}}$$

其中，

$$D(u,v)=[(u-P/2)^2+(v-Q/2)^2]^{1/2}$$

BLPF 传递函数并没有在通过频率和滤除频率之间给出明显截止的尖锐的不连续性。对于具有平滑传递函数的滤波器，可在这样一点定义截止频率，即使得 H(u,v)下降到其最大值的某个百分比点。对于上式，截止频率点是当 D(u,v)=D0 时的点，即 H(u,v)从其最大值 1 下降为 50%。

c) 高斯低通滤波器

$$H(u,v)=e^{-D^2(u,v)/2D_0^2}$$

其中，D(u,v)是距离频率域矩形中心的距离。D0 是截止频率。当 D(u,v)=D0 时，GLPF 下降到其最大值的 0.607 处。

d) 功率谱

建立一组标准截止频率轨迹的一种方法是计算包含规定的总图像功率值 Pt 的圆。该值是通过求每个点(u,v)处填充后图像的功率谱分量的和得到的，其中

u=0,1....P-1，v=0,1....Q-1，

$$P_T = \sum_{u=0}^{P-1}\sum_{v=0}^{Q-1} P(u,v)$$

其中，功率谱定义为

$$P(u,v) = \left|F(u,v)\right|^2 = R^2(u,v) + I^2(u,v)$$

式中，R 和 I 分别是 F(u,v)的实部和虚部，并且所有的计算直接对离散变量 u=0,1...P-1,v=0,1...Q-1。

如果 DFT 已被中心化，那么原点位于频率矩形中心处，半径为 D0 的圆包含 a%的功率，其中 ，

$$\alpha = 100[\sum_{u}\sum_{v} P(u,v) / P_T]$$

2) 实验过程

首先对读入的图像进行填充 0，并对图像乘以(-1)^(x+y)，再对图像进行傅里叶变换可以得到将原点移到中心的傅里叶变换。其次，根据上述 butter worth 滤波器的表达式生成相关的 butter worth 低通滤波器，将滤波器与经傅里叶变换后的图像相乘，相当于在空间域进行卷积。最后对滤波后的图像进行傅里叶反变换后取实部，并对图像乘以(-1)^(x+y)，将其原点移到中心，可得到最终滤波后的图像。

在对图像进行滤波处理过程中需要注意的几点是，

a) 对图像进行 0 填充，得到大小为 P*Q 的图像，主要是为了避免在循环卷积中出现的缠绕错误。当两个矩阵大小相同时，P≥2m-1，Q≥2n-1 时可以避免环绕错误。由于对于偶数尺寸的矩阵计算其傅里叶变换较快，因此 P 取 2m，Q 取 2n。

b) 图像乘以(-1)^(x+y)，再对图像进行傅里叶变换可以得到将原点移到中心的傅里叶变换。这样，对于 F(u,v)来说，中心的频率最低，四周的频率较高。每一点的值表示该频率对于的幅度。在 matlab 中，也可以不乘以(-1)^(x+y)，直接对填充后的图像进行傅里叶变换，之后使用 fftshift 函数对其傅里叶变换进行中心化。得到的结果是一样的。

c) matlab 中读取图片后保存的数据是 uint8 类型，在对图像进行处理时，如不转换，在对 uint8 进行加减时结果会溢出，因此需首先将图像转化为 double 类型（范围为（0，255））。matlab 处理完图像矩阵后，用 imshow()显示图像或用 inwrite()写入图片，imshow()显示图像时，认为 double 型数据位于(0,1),对于数组中大于 1 的元素，会将其归为 1,显示为白色。所以，对于 0~255 范围的 double 图像数组，要想正常显示，应除以 255 进行归一化处理后将其转换成 uint8 类型图像数组显示。

3) Matlab 相关函数

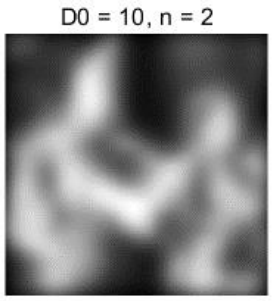
a) fft2 函数用于数字图像的二维傅立叶变换；

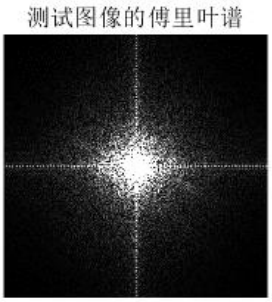
b) ifft2 函数用于数字图像的二维傅立叶反变换；

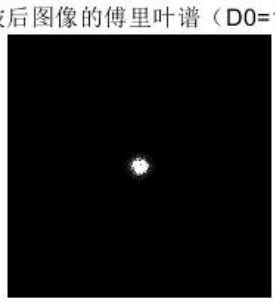
c) fftshift 函数用于将变换后的图象频谱中心从矩阵的原点移到矩阵的中心。

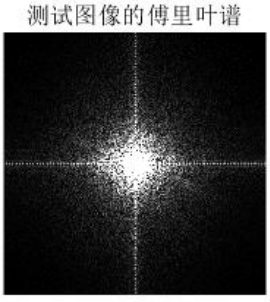
4) 实验结果

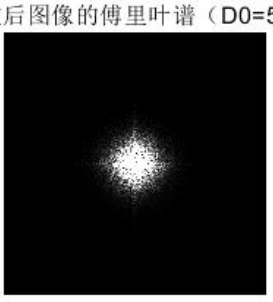
a) Butter worth 滤波器







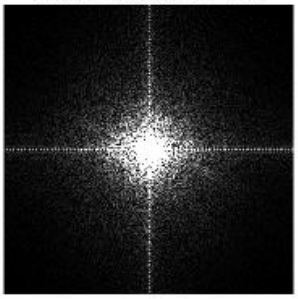




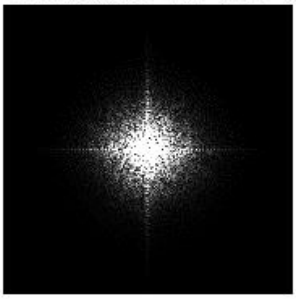
test1 原图像

D0 = 100, n = 2

测试图像的傅里叶谱

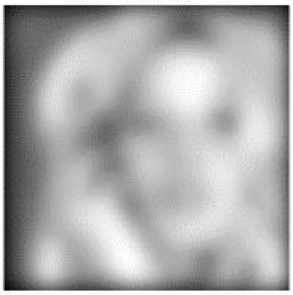


滤波后图像的傅里叶谱（D0=100）

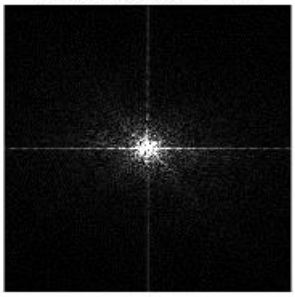


test2 原图像

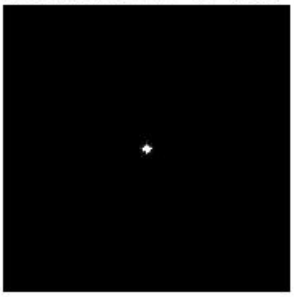
D0 = 10, n = 2



测试图像的傅里叶谱



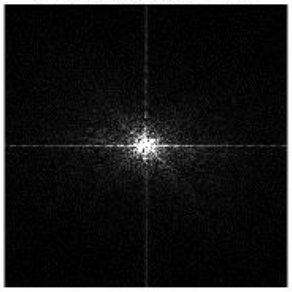
滤波后图像的傅里叶谱（D0=10）



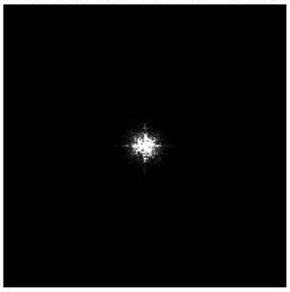
test2 原图像

D0 = 50, n = 2

测试图像的傅里叶谱



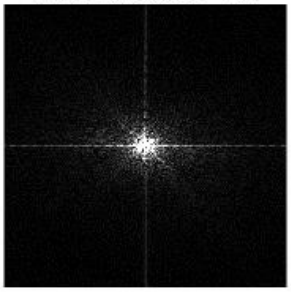
滤波后图像的傅里叶谱（D0=50）



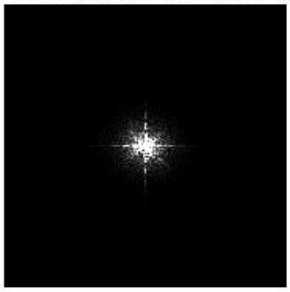
test2 原图像

D0 = 100, n = 2

测试图像的傅里叶谱



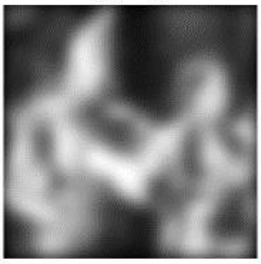
滤波后图像的傅里叶谱（D0=100）



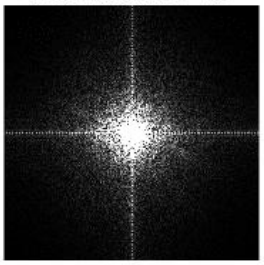
b) Guassian 滤波器

test1 原图像

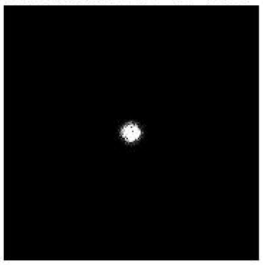
D0 = 10



测试图像的傅里叶谱



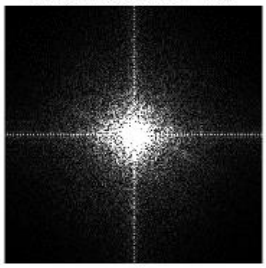
滤波后图像的傅里叶谱（D0=10）



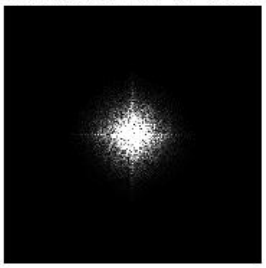
test1 原图像

D0 = 50

测试图像的傅里叶谱



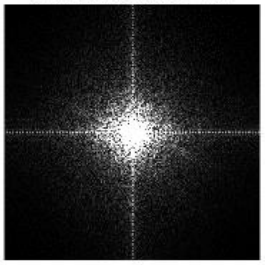
滤波后图像的傅里叶谱（D0=50）



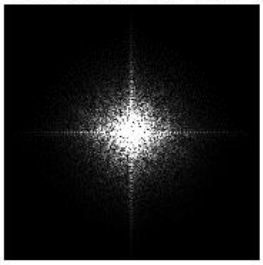
test1 原图像

D0 = 100

测试图像的傅里叶谱

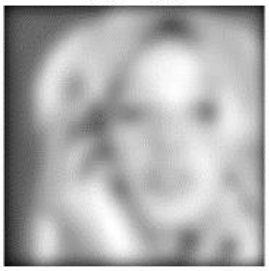


滤波后图像的傅里叶谱（D0=100）

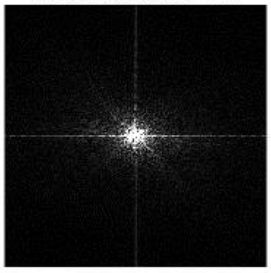


test2 原图像

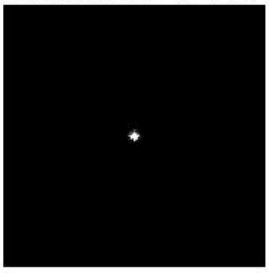
D0 = 10，

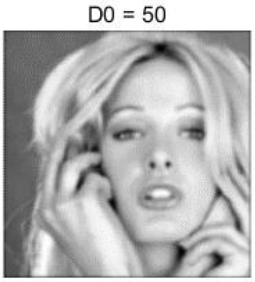


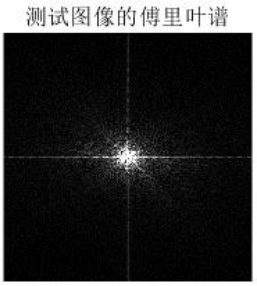
测试图像的傅里叶谱

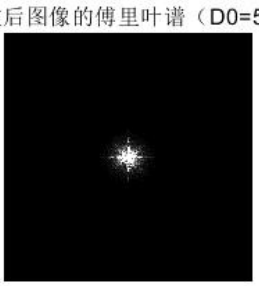


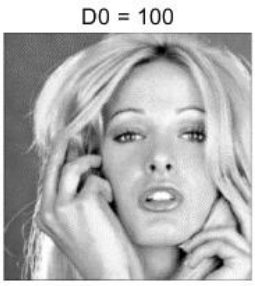
滤波后图像的傅里叶谱（D0=10）

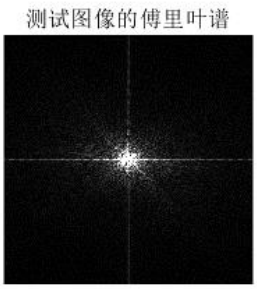


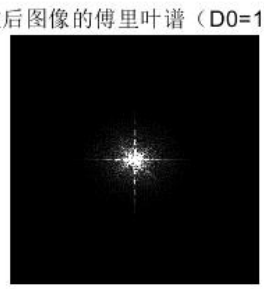












5) 结果分析

a) 对比每组图像处理结果中的原始图像和低通滤波后的图像，可以清晰看到低通滤波器的平滑效果（模糊效果）；对比每组图像中原始图像的傅里叶谱以及滤波后图像的傅里叶谱，可以看到滤波在空间域是卷积关系和在频率域是相乘关系。低通滤波器对于低频分量可以通过，而对于高频分量则不能通过。通过三幅图的对比，可以清晰的看到滤波器的截断效果。

b) 对于 test1 和 test2 分别选取 D0=10、50、100 的二阶布特沃斯低通滤波器进行低通滤波。对比不同的 D0 值得到的结果知，随着截止频率 D0 的减小，滤波后的图像越来越模糊，功率谱比越来越小，即滤波后包含的低频分量越来越少。

c) 对于 test1 和 test2 分别选取 D0=10、50、100 的高斯低通滤波器进行低通滤波。对比不同的 D0 值得到的结果知，随着截止频率 D0 的减小，滤波后的图像越来越模糊，功率谱比越来越小，即滤波后包含的低频分量越来越少。

d) 最后，对比二阶布特沃斯低通滤波器和高斯低通滤波器的效果知，两种滤波器达到的基本效果是一致的，即平滑图像，滤除高频分量，保留低频分量。但两者在相同截止频率 D0 时，得到的功率谱比却不同，主要原因是两个滤波器在过渡带处的差异。

2. 频域高通滤波器：设计高通滤波器包括 butterworth and Gaussian，在频域增强边缘。选择半径和计算功率谱比，测试图像 test3,4。

1) 问题分析

a) 布特沃斯高通滤波器：

截止频率位于距原点 D0 处的 n 阶布特沃斯高通滤波器（BHPF）的传递函数定义为

$$H(u,v)=\frac{1}{1+[D_0/D(u,v)]^{2n}}$$

其中

$$D(u,v)=[(u-P/2)^2+(v-Q/2)^2]^{1/2}$$

b) 高斯高通滤波器：

$$H(u,v)=1-e^{-D^2(u,v)/2D_0^2}$$

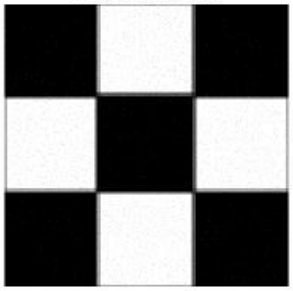
2) 实验过程

与问题一中的低通滤波器差异在于滤波器的表达式不同，其余相同。

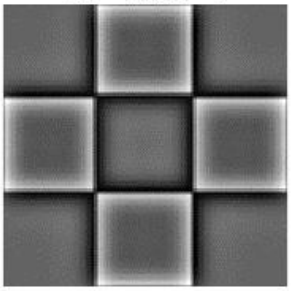
3) 实验结果

a) Butterworth 滤波器

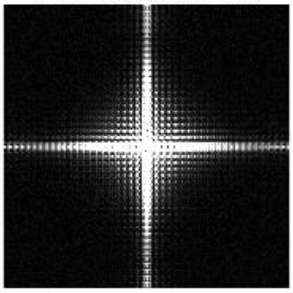
test3 原图像



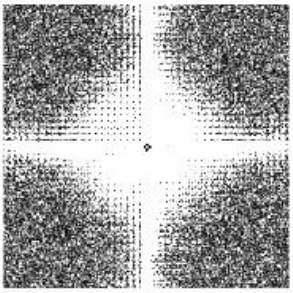
D0 = 10, n = 2



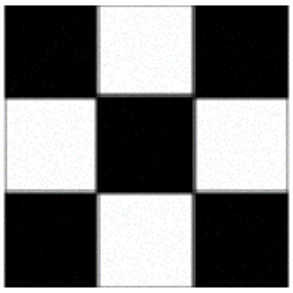
测试图像的傅里叶谱



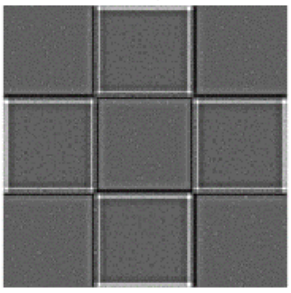
滤波后图像的傅里叶谱（D0=10）



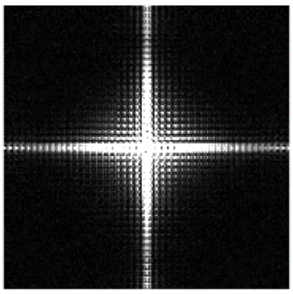
test3 原图像



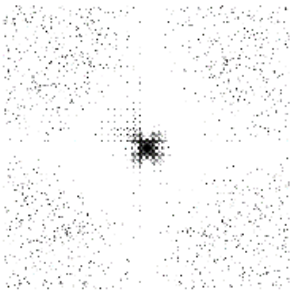
D0 = 50, n = 2



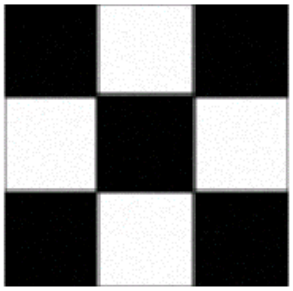
测试图像的傅里叶谱



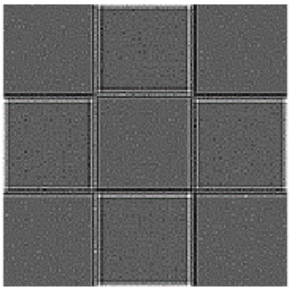
滤波后图像的傅里叶谱（D0=50）



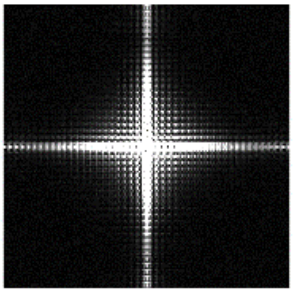
test3 原图像



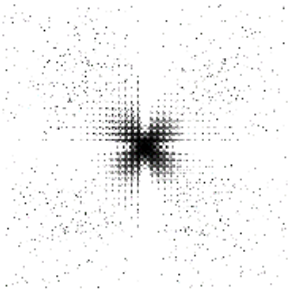
D0 = 100, n = 2



测试图像的傅里叶谱



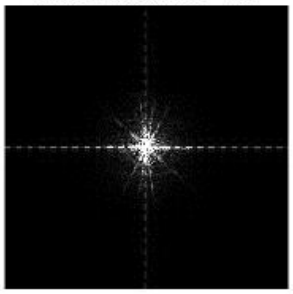
滤波后图像的傅里叶谱（D0=100）



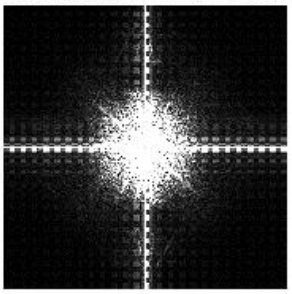
test4 原图像

D0 = 10, n = 2

测试图像的傅里叶谱



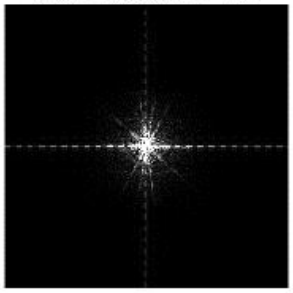
滤波后图像的傅里叶谱（D0=10）



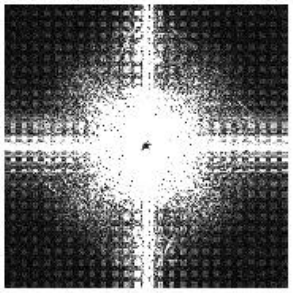
test4 原图像

D0 = 50, n = 2

测试图像的傅里叶谱

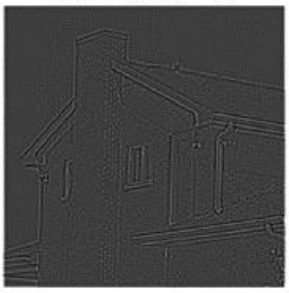


滤波后图像的傅里叶谱（D0=50）

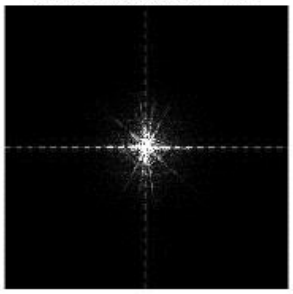


test4 原图像

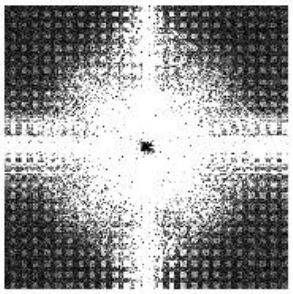
D0 = 100, n = 2



测试图像的傅里叶谱

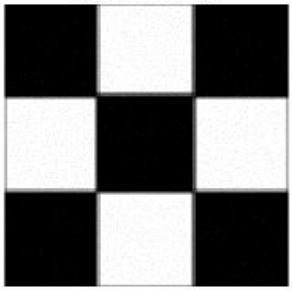


滤波后图像的傅里叶谱（D0=100）

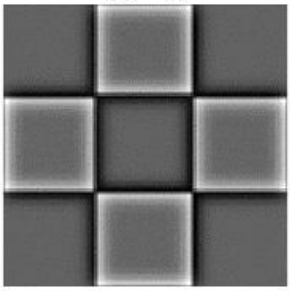


b) Guassian 滤波器

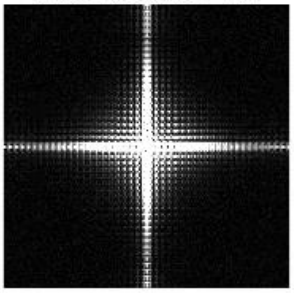
test3 原图像



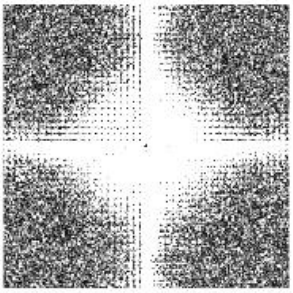
D0 = 10



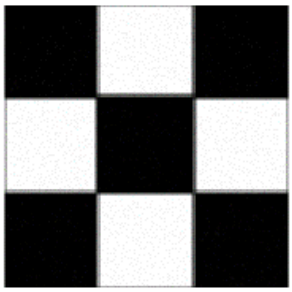
测试图像的傅里叶谱



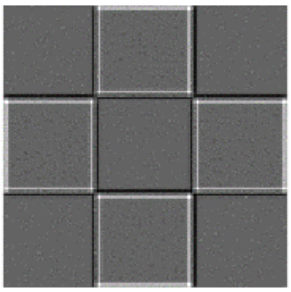
滤波后图像的傅里叶谱（D0=10）



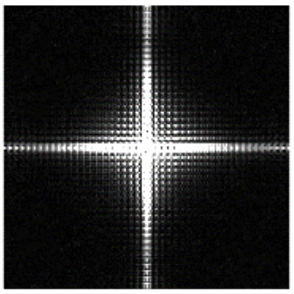
test3 原图像



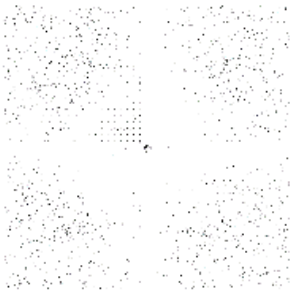
D0 = 50



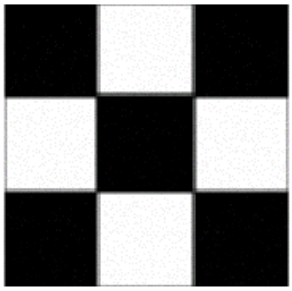
测试图像的傅里叶谱



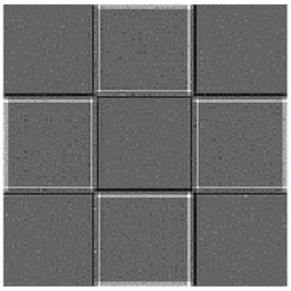
滤波后图像的傅里叶谱（D0=50）



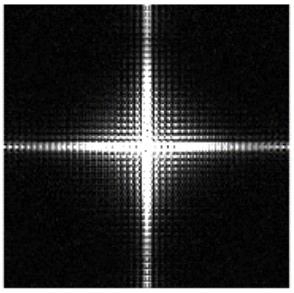
test3 原图像



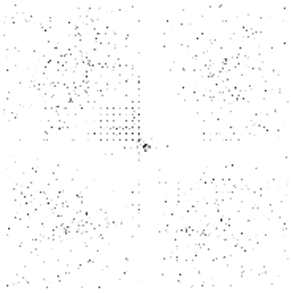
D0 = 100



测试图像的傅里叶谱



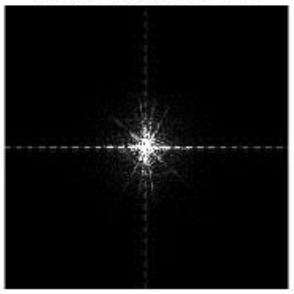
滤波后图像的傅里叶谱（D0=100）



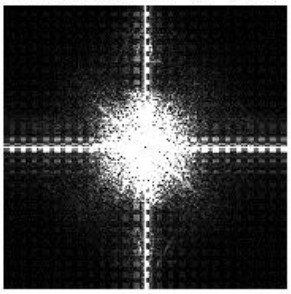
test4 原图像

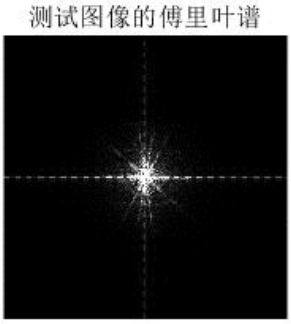
D0 = 10,

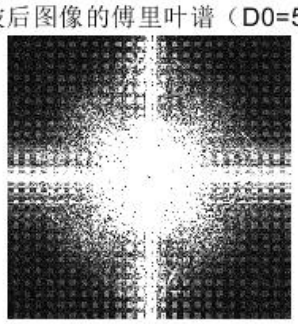
测试图像的傅里叶谱

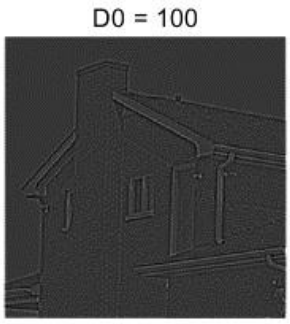


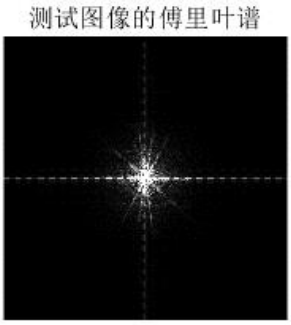
滤波后图像的傅里叶谱（D0=10）

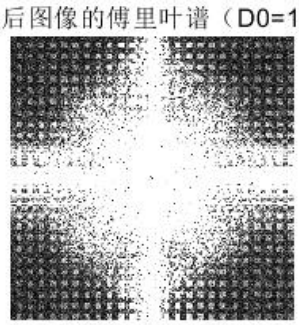












4) 结果分析

a) 对比每组图像处理结果中的原始图像和高通滤波后的图像，可以清晰看到高通滤波器的边缘增强效果；对比每组图像中原始图像的傅里叶谱以

及滤波后图像的傅里叶谱，可以看到滤波在空间域是卷积关系和在频率域是相乘关系。高通滤波器对于低频分量的滤除和对于高频分量的保留作用。通过三幅图的对比，可以清晰的看到滤波器的截断效果。

b) 对于 test3 和 test4 分别选取 D0=10、50、100 的二阶布特沃斯高通滤波器进行高通滤波。对比不同的 D0 值得到的结果知，随着截止频率 D0 的增加，滤波后的图像边缘应该越来越清晰，功率谱比越来越小，即滤波后包含的高频分量越来越少。但当 D0 增大到一定程度时，边缘将不见，主要是因为滤除的能量过多，图像几乎全部变成了黑色。

c) 对于 test3 和 test4 分别选取 D0=10、50、100 的高斯高通滤波器进行高通滤波。对比不同的 D0 值得到的结果知，随着截止频率 D0 的增加，滤波后的图像边缘应该越来越清晰，功率谱比越来越小，即滤波后包含的高频分量越来越少。但当 D0 增大到一定程度时，边缘将不见，主要是因为滤除的能量过多，图像几乎全部变成了黑色。

d) 对比二阶布特沃斯高通滤波器和高斯高通滤波器的效果知，两种滤波器达到的基本效果是一致的，即增强图像边缘，滤除低频分量，保留高频分量。但两者在相同截止频率 D0 时，得到的功率谱比却不同，主要原因是两个滤波器在过渡带处的差异。

e) 对比高通滤波器和低通滤波器知，高通滤波器在滤波的时候会将直流分量也一同滤除，导致图像变暗。造成当 D0 增大到一定程度时，边缘将不见，整个图像变为黑色。

3. 其他高通滤波器：拉普拉斯和 Unmask，对测试图像 test3,4 滤波；

1) 问题分析

a) 频率域的拉普拉斯算子：

拉普拉斯算子可使用如下滤波器在频率域实现：

$$H(\mathrm{u},\mathrm{v}) = -4\pi^2[(\mathrm{u}-\mathrm{P}/2)^2 + (\mathrm{v}-\mathrm{Q}/2)^2] = -4\pi^2 D^2(\mathrm{u},\mathrm{v})$$

拉普拉斯图像由下式得到：

$$\nabla^2 f(\mathrm{x},\mathrm{y}) = \Im^{-1}\{\mathrm{H}(\mathrm{u},\mathrm{v})\,\mathrm{F}(\mathrm{u},\mathrm{v})\}$$

增强可使用下式实现：

$$g(\mathrm{x},\mathrm{y}) = f(\mathrm{x},\mathrm{y}) + c\nabla^2 f(\mathrm{x},\mathrm{y})$$

在频率域：

$$g(\mathrm{x},\mathrm{y}) = \Im^{-1}\{\mathrm{F}(\mathrm{u},\mathrm{v}) - \mathrm{H}(\mathrm{u},\mathrm{v})\,\mathrm{F}(\mathrm{u},\mathrm{v})\} = \Im^{-1}\{[1-\mathrm{H}(\mathrm{u},\mathrm{v})]\mathrm{F}(\mathrm{u},\mathrm{v})\} = \Im^{-1}\{[1-4\pi^2\,\mathrm{D}^2(\mathrm{u},\mathrm{v})]\mathrm{F}(\mathrm{u},\mathrm{v})\}$$

b) 钝化模板（unmask）：

钝化模板由下式给出：

$$g_{mask}(\mathrm{x},\mathrm{y}) = f(\mathrm{x},\mathrm{y}) - f_{LP}(\mathrm{x},\mathrm{y})$$

与

$$f_{LP}(x,y) = \Im^{-1}[H_{LP}(u,v)F(u,v)]$$

最后的图像由下式给出：

$$g(x,y) = f(x,y) + k * g_{mask}(x,y)$$

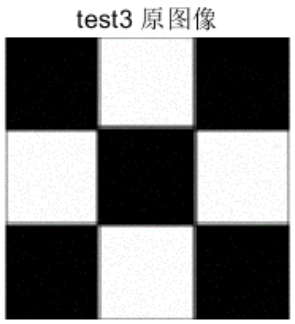
最后的频域滤波公式：

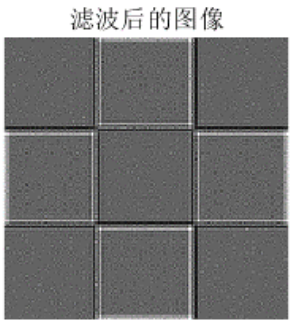
2) 实验过程

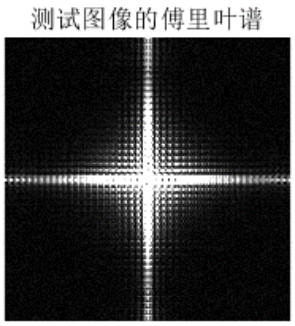
与问题一中的低通滤波器差异在于滤波器的表达式不同，其余相同。

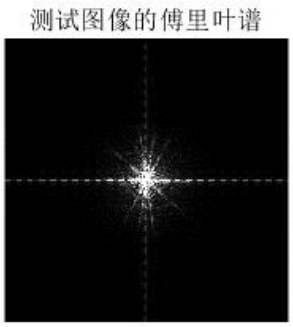
3) 实验结果

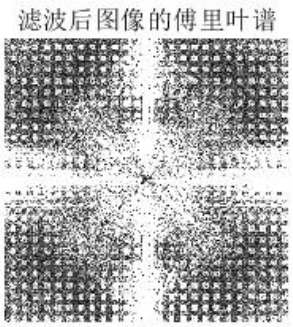
a) Laplace 算子滤波



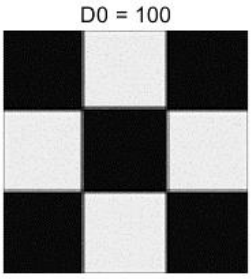


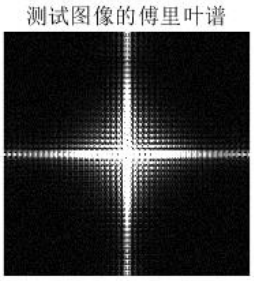


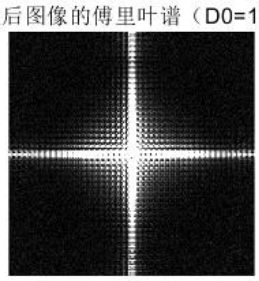


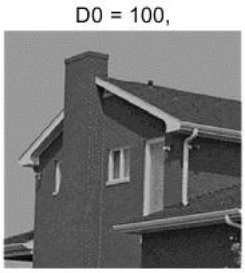


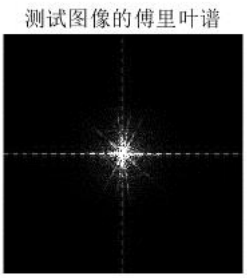
b) Unmark 滤波（选择 D0=100 的 Guassian 高通滤波器）

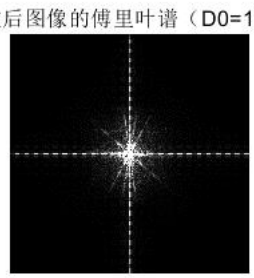












4) 结果分析

对比每组图像处理结果中的原始图像和滤波后的图像，可以隐约看到滤波器的边缘增强效果；由于最后得到的高频图像要加到原始图像上构成新的图像，所以视觉上原始图像的傅里叶谱和滤波后图像的傅里叶谱基本一致。对比拉普拉斯算子和 unmask 滤波，两者达到的基本效果是一致的。

4. 比较并讨论空域低通高通滤波（Project3）与频域低通和高通的关系。

空间域和频域滤波间的纽带是卷积定理。空间域中的滤波定义为滤波函数 h（x，y）与输入图像 f（x，y）进行卷积；频率域中的滤波定义为滤波函数 H（u，v）与输入图像的傅里叶变换 F（u，v）进行相乘。空间域的滤波器和频率域的滤波器互为傅里叶变换。

频域增强技术与空域增强技术有密切的联系。一方面，许多空域增强技术可借助频域概念来分析和帮助设计；另一方面，许多空域增强技术可转化到频域实现，而许多频域增强技术可转化到空域实现。空域滤波主要包括平滑滤波和锐化滤波。平滑滤波是要滤除不规则的噪声或干扰的影响，从频域的角度看，不规则的噪声具有较高的频率，所以可用具有低通能力的频域滤波器来滤除。由此可见空域的平滑滤波对应频域的低通滤波。锐化滤波是要增强边缘和轮廓处的强度，从频域的角度看，边缘和轮廓处都具有较高的频率，所以可用具有高通能力的频域滤波器来增强。由此可见，空域的锐化滤波对应频域的高通滤波。频域里低通滤波器的转移函数应该对应空域里平滑滤波器的模板函数的傅里叶变换。频域里高通滤波器的转移函数应该对应空域里锐化滤波器的模板函数的傅里叶变换。即空域和频域的滤波器组成傅里叶变换对。给定一个域内的滤波器，通过傅里叶变换或反变换得到在另一个域内对应的滤波器。 空域的锐化滤波或频域的高通滤

波可用两个空域的平滑滤波器或两个频域的低通滤波器实现。在频域中分析图像的频率成分与图像的视觉效果间的对应关系比较直观。空域滤波在具体实现上和硬件设计上有一定优点。区别：例如，空域技术中无论使用点操作还是模板操作，每次都只是基于部分像素的性质，而频域技术每次都利用图像中所有像素的数据，具有全局性，有可能更好地体现图像的整体特性，如整体对比度和平均灰度值等。

参考文献

[1] 冈萨雷斯.数字图像处理（第三版）北京：电子工业出版社，2011
[2] 周品.MATLAB 数字图像处理 北京：清华大学出版社，2012
[3] 杨杰.数字图像处理及 MATLAB 实现 北京：电子工业出版社，2010

附录

1. butter worth 低通滤波器

```
%%butterworth 低通滤波器函数
function [image_out,F1,BW,G] = Butterworth(image_in,D0,N)
  [m, n] = size(image_in);
P = 2 * m;
Q = 2 * n;
fp = zeros(P,Q);
for i = 1 : m
      for j= 1 : n
            fp(i,j) = double(image_in(i,j)) ;
            %对数据类型进行变换，防止运算时溢出
      end
end
%对图像进行填充
  F1 = fftshift(fft2(fp));
%对图像进行傅里叶变换，并对其傅里叶进行中心化
BW = zeros(P,Q);
a = D0 ^(2 * N);
for u = 1 : P
      for v= 1 : Q
            temp = (u-(m+1.0))^2 + (v-(n+1.0))^2;
```

```
            BW(u,v) = 1/(1 + (temp)^N / a) ;
        end
    end
    G = F1.* BW;
    gp = real(ifft2(G));
    image_out = zeros(m, n, 'uint8');
    g = zeros(m, n);
     for i = 1 : m
        for j = 1 : n
            g(i,j) = gp(i,j) * (-1) ^(i+j);
        end
    end
    mmax = max(g(:));
    mmin = min(g(:));
    range = mmax-mmin;
    for i = 1 : m
        for j = 1 : n
            image_out(i,j) = uint8(255 * (g(i, j)-mmin) / range); %归一化
        end
    end
    end
```

2. guassian 滤波器

```
    %%guassian 低通滤波器函数
    function [image_out,F1,GS,G] = Gaussian(image_in,D0)
      [m, n] = size(image_in);
    P = 2 * m;
    Q = 2 * n;
    fp = zeros(P,Q);
    for i = 1 : m
        for j= 1 : n
        end
    end
    F1 = fftshift(fft2(fp));
    GS = zeros(P,Q);
    for u = 1 : P
        for v= 1 : Q

            D(u,v)=sqrt((u-fix(P/2))^2+(v-fix(Q/2))^2);
            GS(u,v)=exp(-D(u,v)^2/(2*D0^2));

        end
    end
    G = F1.* GS;
    gp = real(ifft2(G));
    image_out = zeros(m, n, 'uint8');
```

```
g = zeros(m, n);

for i = 1 : m
    for j = 1 : n
        g(i,j) = gp(i,j) * (-1) ^(i+j);
    end
end
mmax = max(g(:));
mmin = min(g(:));
range = mmax-mmin;
for i = 1 : m
    for j = 1 : n
        image_out(i,j) = uint8(255 * (g(i, j)-mmin) / range);
    end
end
end
```

3. butterworth 高通滤波器

```
function [image_out,F1,BW,G] = Butterworth(image_in,D0,N)
  [m, n] = size(image_in);
P = 2 * m;
Q = 2 * n;
fp = zeros(P,Q);
for i = 1 : m
    for j= 1 : n
        fp(i,j) = double(image_in(i,j)) ;
    end
end
F1 = fftshift(fft2(fp));
BW = zeros(P,Q);
a = D0 ^(2 * N);
for u = 1 : P
    for v= 1 : Q
        temp = (u-(m+1.0))^2 + (v-(n+1.0))^2;
        BW(u,v) = 1/(1 + a/(temp)^N    ) ;
    end
end
G = F1.* BW;
gp = real(ifft2(G));
image_out = zeros(m, n, 'uint8');
g = zeros(m, n);
for i = 1 : m
    for j = 1 : n
        g(i,j) = gp(i,j) * (-1) ^(i+j);
    end
end
mmax = max(g(:));
```

```
mmin = min(g(:));
range = mmax-mmin;
for i = 1 : m
    for j = 1 : n
        image_out(i,j) = uint8(255 * (g(i, j)-mmin) / range);
    end
end
end
```

4. guassian 高通滤波器

```
function [image_out,F1,GS,G] = Gaussian(image_in,D0)
 [m, n] = size(image_in);
P = 2 * m;
Q = 2 * n;
fp = zeros(P,Q);
for i = 1 : m
    for j= 1 : n
        fp(i,j) = double(image_in(i,j)) ;
     end
end
F1 = fftshift(fft2(fp));
GS = zeros(P,Q);
for u = 1 : P
    for v= 1 : Q

            D(u,v)=sqrt((u-fix(P/2))^2+(v-fix(Q/2))^2);
            GS(u,v)=1 - exp(-D(u,v)^2/(2*D0^2));


    end
end
G = F1.* GS;
gp = real(ifft2(G));
image_out = zeros(m, n, 'uint8');
g = zeros(m, n);

for i = 1 : m
    for j = 1 : n
        g(i,j) = gp(i,j) * (-1) ^(i+j);
    end
end
mmax = max(g(:));
mmin = min(g(:));
range = mmax-mmin;
for i = 1 : m
    for j = 1 : n
        image_out(i,j) = uint8(255 * (g(i, j)-mmin) / range);
    end
```

```
end
end
```

5. Laplace 滤波器

```
function [image_out,F1,LP,G] = Laplace(image_in)
  [m, n] = size(image_in);
P = 2 * m;
Q = 2 * n;
fp = zeros(P,Q);
for i = 1 : m
      for j= 1 : n
            fp(i,j) = double(image_in(i,j)) ;
       end
end
F1 = fftshift(fft2(fp));
LP = zeros(P,Q);
for u = 1 : P
      for v= 1 : Q
          D(u,v)=sqrt((u-fix(P/2))^2+(v-fix(Q/2))^2);
           LP(u,v)=1+4*pi^2*D(u,v)^2;
      end
end
G = F1.* LP;
gp = real(ifft2(G));
image_out = zeros(m, n, 'uint8');
g = zeros(m, n);
for i = 1 : m
      for j = 1 : n
            g(i,j) = gp(i,j) * (-1) ^(i+j);
      end
end
mmax = max(g(:));
mmin = min(g(:));
range = mmax-mmin;
for i = 1 : m
      for j = 1 : n
            image_out(i,j) = uint8(255 * (g(i, j)-mmin) / range);
      end
end
end
```

6. Unmask 滤波器

```
function [image_out,F1,US,G] = Unmask(image_in,D0)
  [m, n] = size(image_in);
P = 2 * m;
Q = 2 * n;
fp = zeros(P,Q);
```

```
for i = 1 : m
    for j= 1 : n
        fp(i,j) = double(image_in(i,j)) ;
    end
end
F1 = fftshift(fft2(fp));
US = zeros(P,Q);
for u = 1 : P
    for v= 1 : Q
        D(u,v)=sqrt((u-fix(P/2))^2+(v-fix(Q/2))^2);
        H(u,v)=1-exp(-D(u,v)^2/(2*D0^2));
        US(u,v)=1+H(u,v);
    end
end
G = F1.* US;
gp = real(ifft2(G));
image_out = zeros(m, n, 'uint8');
g = zeros(m, n);
 for i = 1 : m
    for j = 1 : n
        g(i,j) = gp(i,j) * (-1) ^(i+j);
    end
end
mmax = max(g(:));
mmin = min(g(:));
range = mmax-mmin;
for i = 1 : m
    for j = 1 : n
        image_out(i,j) = uint8(255 * (g(i, j)-mmin) / range);
    end
end
end
```