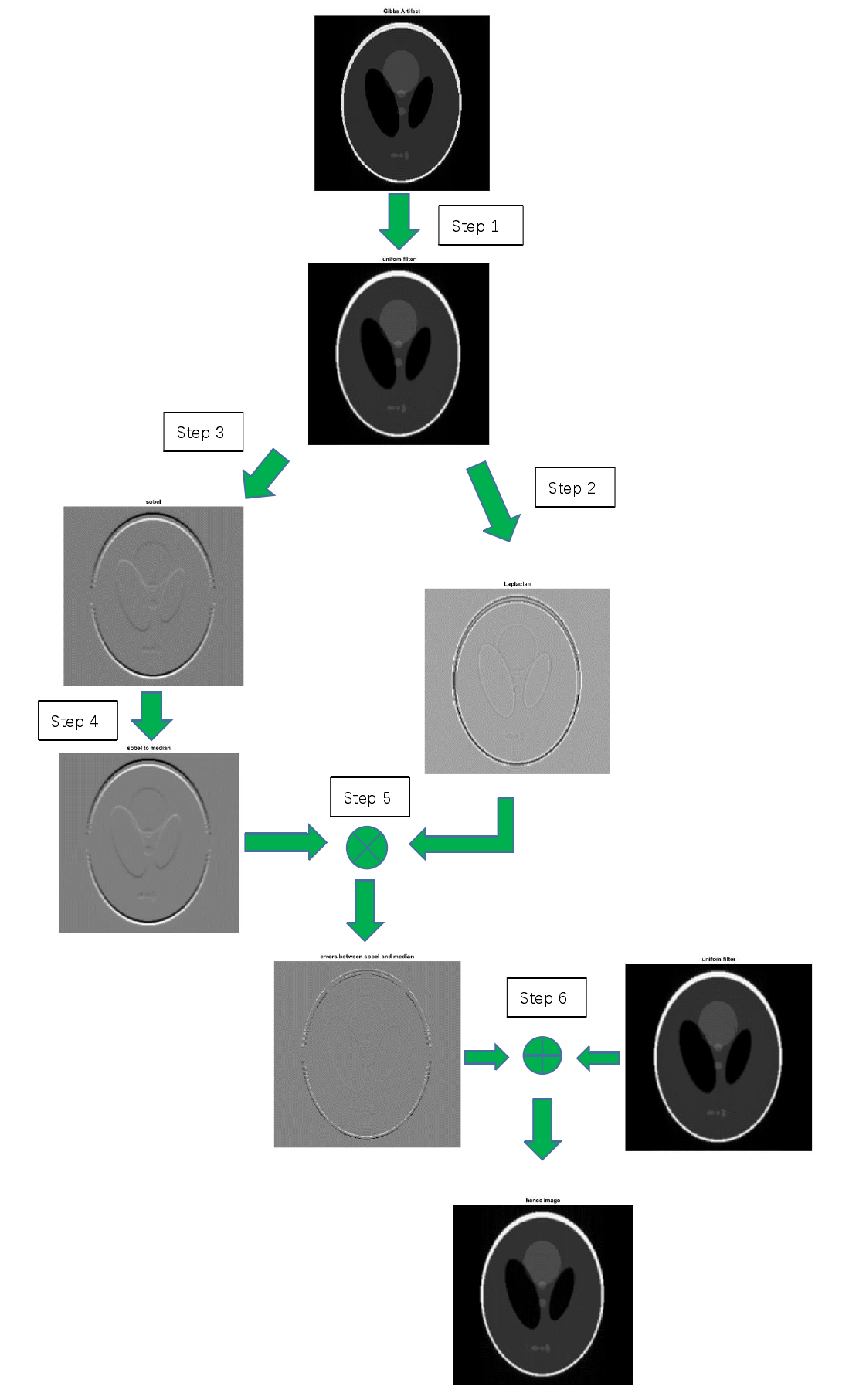
**3148020021**

**李丹阳**

**MRI图像Gibbs伪影去除算法**

本文应用两种算法去除Gibbs伪影。一种为自己提出的空间滤波方法,另一种为基于chebyshev多项式的逆向重建算法（ICPRM）[1,2]。第一种算法基于空间滤波，先对Gibbs伪影进行平滑滤波，损失的空间分辨率由二阶微分算子来锐化，通过叠加细节，来增强图像，改善图像空间分辨率，同时又不引入新的噪声。为降低第二种算法实现难度，假设整幅图都是连续的，故可跳过边缘检测。但由于对算法所涉及的公式理解不到位，最后并没有重建出理想图像。

1、空间滤波算法



算法流程如上图所示。具体操作为，

**Step1：**对含有Gibbs伪影的图像进行空间平滑滤波，得到较少伪影的图像。平滑滤波器选取3×3均值滤波器。由于平滑滤波带来图像空间分辨率及细节信息的丢失，下面进行空间细节增强操作；

**Step2：**对平滑图像进行二阶微分滤波，提取图像细节，同时图像增加的噪声由下面步骤去除。滤波器采用Laplacian算子；

**Step3：**对平滑图像进行一阶边缘检测，提取图像轮廓。滤波器采用3×3 Sobel滤波算子；

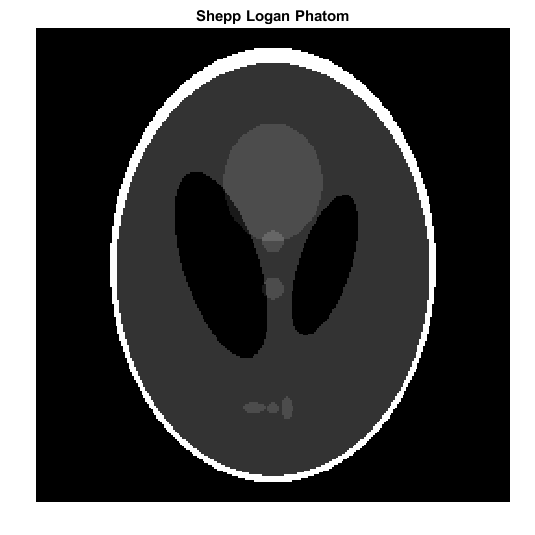
**Step4：**由于一阶锐化滤波对图像小噪声点不敏感，如此再进行中值滤波，会有效滤除图像中的噪声；

**Step5：**将二阶锐化图像与中值滤波图像进行相乘。二阶锐化图像含有大量噪声，如果直接叠加到平滑滤波图像上会带来图像噪声的增加。中值滤波图像可看作模板，进行相乘操作，二阶锐化图像中平滑区域的噪声点会被去除，而灰度变化强烈的细节区域会被加强；

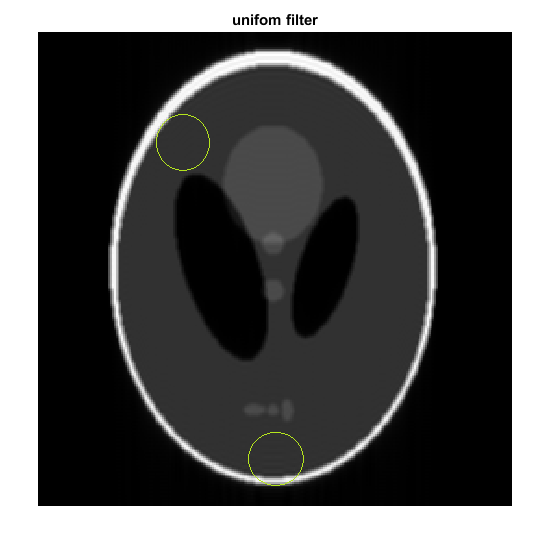
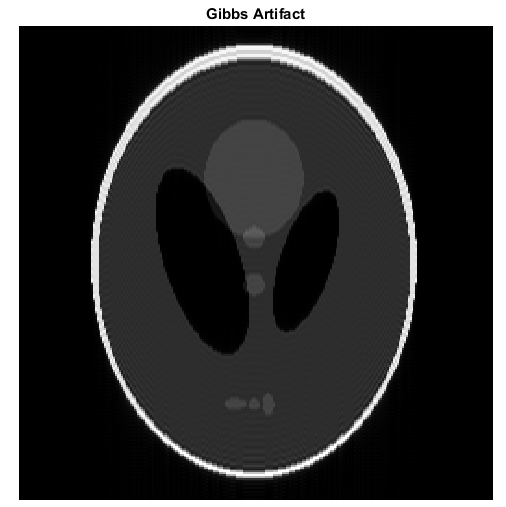
**Step6：**将乘积图像进行灰度标定后，再与平滑滤波图像进行相加操作，如此，增强平滑图像的细节，得到最终的增强图像。

**结果说明：**

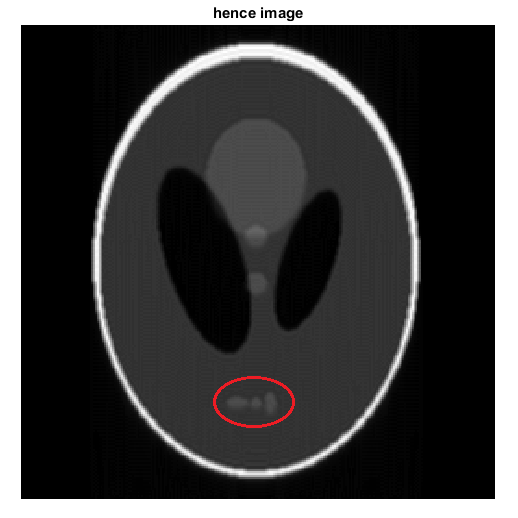
原始体模与含有Gibbs伪影的图像进行对比如下，在黄色圈中可以看到灰度变化剧烈的附近伪影明显。



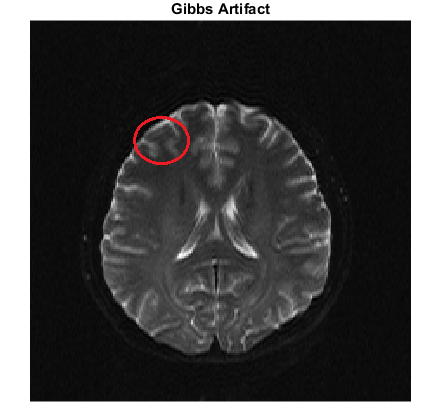
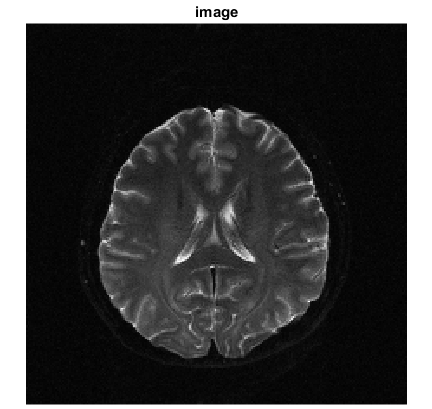
平滑滤波后的图像浅绿色圈中的伪影被显著去除，但图像的空间分辨率下降，如下，

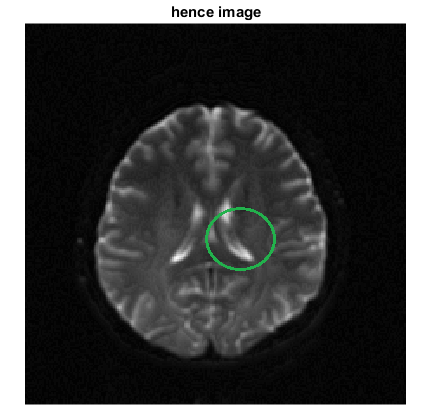


最终增强图像与滤波图像如下，图像的空间分辨率增强，在红色圈中表现明显

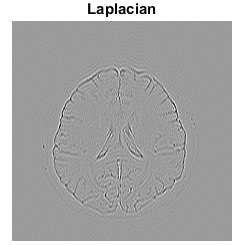


对实际临床图像进行Gibbs伪影去除结果如下，



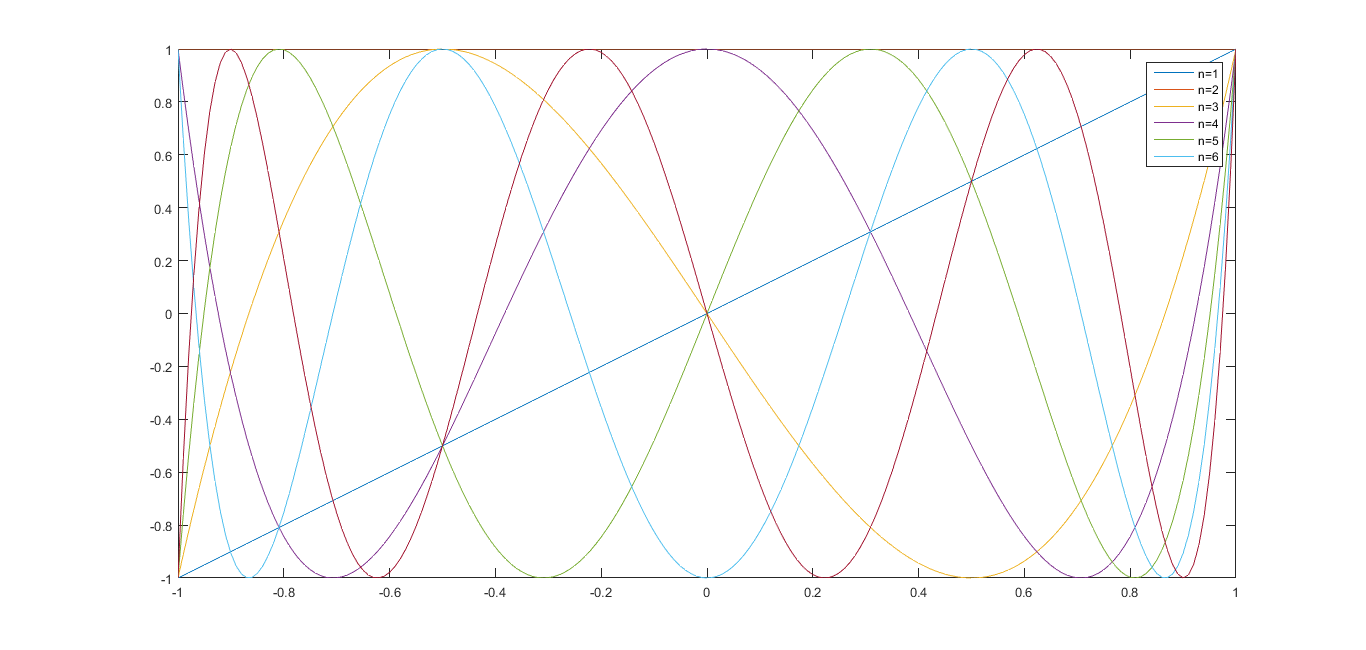


增强的细节如图，

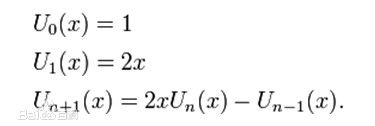


2、基于chebyshev多项式的逆向重建算法（ICPRM）

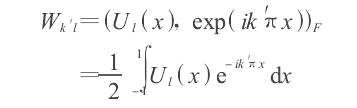
Chebyshev多项式分一类和二类，对于一类Chebyshev多项式，可作出在区间[-1,1]上，n=6时的曲线组，如下，



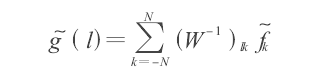
ICPRM算法基于第二类多项式，递推公式[ 3 ]为：



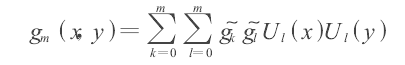
变换矩阵[ 1 2 ]为：



系数为：



最终图像的成像方程为：



**算法实现：**

对于输入的向量x，其二类Chebyshev多项式T计算如下，

[X,Y]=size(x);

T0=ones(1,Y);

T1=2\*x;

Tn=T1;

Tn1=T0;

if n>=2

for i=1:n

T=2\*x.\*Tn-Tn1;

Tn1=Tn;

Tn=T;

end

elseif n==0

T=T0;

else

T=T1;

end

变换矩阵Wkl为，

qu=Chebyshev(x,l).\*exp(-1i\*k\*pi\*x);

Wkl=1/2\*sum(qu);

展开式系数g由函数countg计算，具体语句为

x=linspace(-1,1,size(f,1))';

for k=-N:N

g=(transWkl(x,k,l)).^(-1)\*fk(x,f,k);

end

其中fk为，

che=f.\*exp(1i\*k\*pi\*x);

fk=(1/2)\*sum(che);

最终重建图像gmxy由下面两个for循环计算，先进行「列」，在进行「行」

for i=1:256

f=kpart(:,i);

for l=1:m

gmc=countg(f,l,N)\*Chebyshev(x,l);

gm=gm+gmc;

end% m次叠加

gmxy(:,i)=gm;

end % 256列执行完毕

for i=1:256

f=kpart(i,:)';% 转成列向量

for l=1:m

gmc=countg(f,1,N)\*Chebyshev(x,l);

gm=gm+gmc;

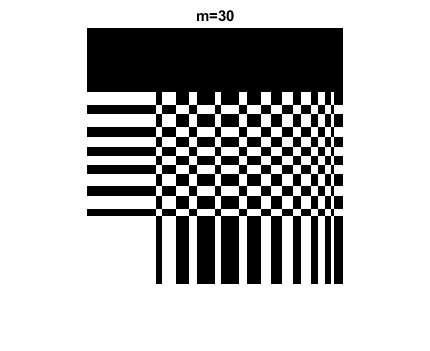
end

gmxy(i,:)=gm';

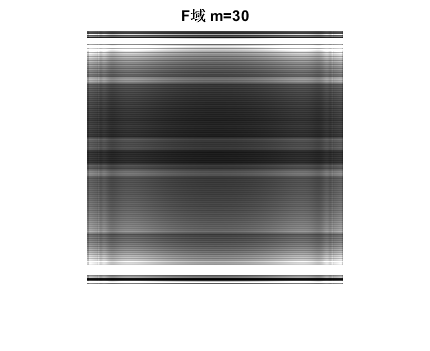
end

**结果：**

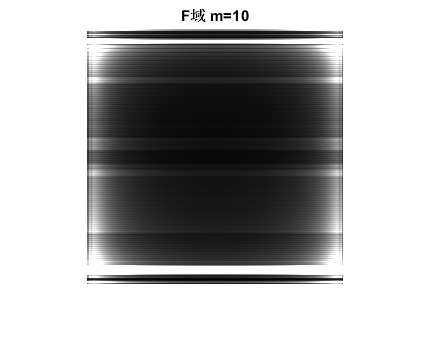
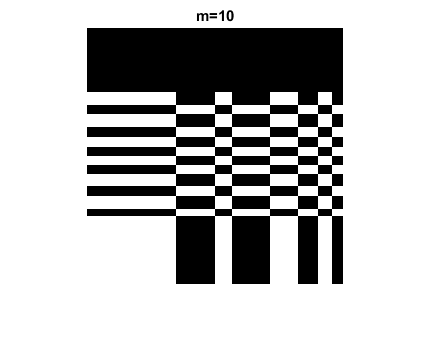
文献[2]中指出,若用Shepp-Logon体模时，N=128，此时m=30，误差最小。最终图像以二值图显示为

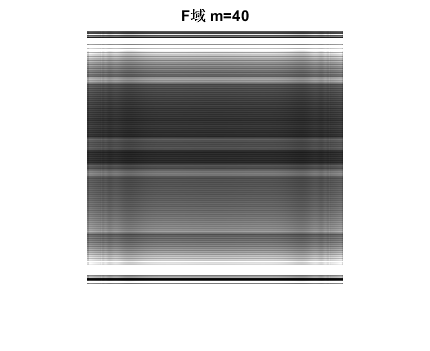
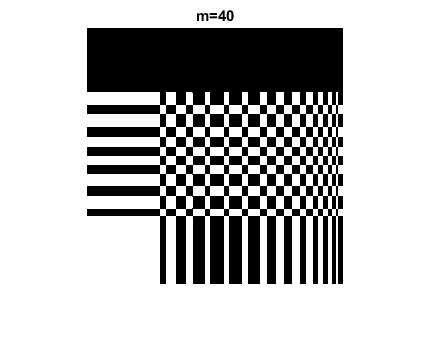


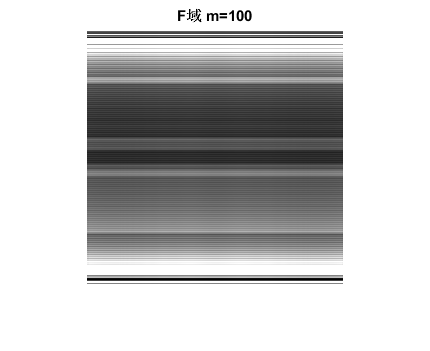
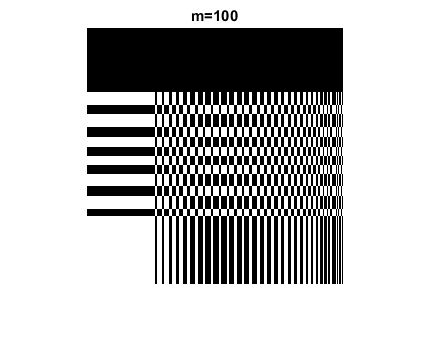
在对其进行二维Fourial变换，显示为



同时，做出m=10,40,100时的图像，即二维傅里叶变换的幅度灰度值，如图







**文献**

[1] Jae-Hun Jung and Bernie D. Shizgal. Inverse Polynomial Reconstruction of Two Dimensional

Fourier Images [J]. Journal of Scientiﬁc Computing, Vol. 25, No. 3, December 2005, DOI: 10.1007/s10915-004-4795-3.

[2] 冯前进,黄鑫,冯衍秋.基于Chebyshev多项式的消除Gibbs伪影的快速算法[J].中国图象图形学报,2006,11(8):1132-1138.

[3] baidu (2017). 切比雪夫多项式 [OL]. http://baike.baidu.com/link?url=0RUu704c\_-O5ePfq7rhMdagiIPJXSI1MekAV3tR02fnSO2Sl\_h2ud5whSs0mMppCwWWAhPFCxmvOoxPt3ABTQwGMUPxrd0TgGcogeouihu8VUxPGlXR\_cduOXuAVk0q70qo2Oncozz8jHfJVsjIsLcmu6PSNSVqQXkWBXdyynPK. Quoted on Jun.8, 2017.

**代码**

1、基于空间滤波算法

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%涉及的函数%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ henceim ] = cutGibbs1( partKspace,n )

% 空间滤波+二阶微分图像增强

% 输入部分k空间数据，输入增强细节权重n

% 2017/6/4 by李丹阳

abs\_kpart=partKspace;

%平滑滤波

w1=fspecial('average',3);

filim=imfilter(abs\_kpart,w1,'replicate');

figure,imshow(filim,[]),title('unifom filter')

%锐化

%二阶滤波

wl=fspecial('laplacian');%对称

% lap\_filim=imfilter(filim,wl,'replicate');

lap\_filim=imfilter(abs\_kpart,wl,'replicate');

figure,imshow(lap\_filim,[]),title('Laplacian')

% 一阶边缘检测

ws=fspecial('sobel');%不对称

ws=rot90(ws,2);

sob\_filim=imfilter(filim,ws,'replicate');

figure,imshow(sob\_filim,[]),title('sobel')

med\_sob\_filim=medfilt2(sob\_filim);

figure,imshow(med\_sob\_filim,[]),title('sobel to median')

figure,imshow(med\_sob\_filim-sob\_filim,[]),title('errors between sobel and median')

% 相乘图像

lapmulmedsob=lap\_filim.\*med\_sob\_filim;

tab\_lapmul=imadjust(lapmulmedsob,[0.5,1],[0,1],0.8);

figure,imshow(tab\_lapmul),title('detial image')

% 增强图像

henceim=tab\_lapmul\*10^n+filim;

figure,imshow(henceim,[]),title('hence image')

end

%%%%%%%%%%%%%%%%%%%%%%%%M文件%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%去除Shepp-logan图像伪影

load('Shepp\_Logan\_Phantom.mat');

figure,imshow(img,[]),title('Shepp Logan Phatom')

kpart=fftshift(fft2(img));

kpart(1:64,:)=0;

kpart(194:end,:)=0;

abs\_kpart=rot90(abs(fft2(kpart)),2);

figure,imshow(abs\_kpart,[]),title('Gibbs Artifact')

henceim=cutGibbs1(abs\_kpart,3);

%去除头部扫描图像伪影

load('DWIB0.mat');

img=b0imfinal(:,:,5);%取第5帧图像

figure,imshow(img,[]),title('image')

kpart1=fftshift(fft2(img));

kpart1(1:55,:)=0;

kpart1(167:end,:)=0;

abs\_kpart1=rot90(abs(fft2(kpart1)),2);

figure,imshow(abs\_kpart1,[]),title('Gibbs Artifact')

henceim1=cutGibbs1(abs\_kpart1,3.5);

2、ICPRM算法

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%涉及的函数%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ T ] = Chebyshev( x,n )

%用递推公式计算chebyshev多项式

% 第二类切比雪夫多项式

% 2017/5/25 by李丹阳

[X,Y]=size(x);

T0=ones(1,Y);

T1=2\*x;

Tn=T1;

Tn1=T0;

if n>=2

for i=1:n

T=2\*x.\*Tn-Tn1;

Tn1=Tn;

Tn=T;

end

elseif n==0

T=T0;

else

T=T1;

end

%第一类切比雪夫多项式

% [X,Y]=size(x);

% T0=ones(1,Y);

% T1=x;

%

% Tn=T1;

% Tn1=T0;

% if n>=2

% for i=1:n

% T=2\*x.\*Tn-Tn1;

% Tn1=Tn;

% Tn=T;

% end

% elseif n==0

% T=T0;

% else

% T=T1;

% end

end

%%%%%%%%%%%%%%%%%%%

function [ g ] = countg( f,l,N )

%计算Chebyshev系数 g

% 原始数据f在此函数中已归一化到在-1到1间，如此才能使用Chebyshev重建

% 输入的f必须转换成列向量，N表示所采的部分傅立叶空间中的行数

% k为-N到N，l为chebyshev的逼近阶数

% f=mapminmax(f);%归一化到（-1,1）

x=linspace(-1,1,size(f,1))';

for k=-N:N

g=(transWkl(x,k,l)).^(-1)\*fk(x,f,k);

end

% gn=((1-x.^2).^(1/2)).\*f.\*Chebyshev(x,l);

% g=2/pi\*sum(gn);

end

%%%%%%%%%%%%%%%%%%%%%

function [ Wkl ] = transWkl(x,k,l)

%UNTITLED3 Summary of this function goes here

% 变换矩阵Wkl

qu=Chebyshev(x,l).\*exp(-1i\*k\*pi\*x);

Wkl=1/2\*sum(qu);%积分在-1到1上

end

%%%%%%%%%%%%%%%%%%

function [ fk ] = fk(x,f,k)

%UNTITLED6 Summary of this function goes here

% Detailed explanation goes here

che=f.\*exp(1i\*k\*pi\*x);

fk=(1/2)\*sum(che);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%M文件%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load('Shepp\_Logan\_Phantom.mat');

kpart=fftshift(fft2(img));

kpart(1:64,:)=0;

kpart(194:end,:)=0;

% imshow(kpart,[])

N=128;

m=30;

x=linspace(-1,1,size(kpart,1))';

gm=linspace(0,0,size(kpart,1))';

%最终图像

gmxy=zeros(size(kpart));

for i=1:256

f=kpart(:,i);

for l=1:m

gmc=countg(f,l,N)\*Chebyshev(x,l);

gm=gm+gmc;

end%m次叠加

gmxy(:,i)=gm;

end %256列执行完毕

for i=1:256

f=kpart(i,:)';%转成列向量

for l=1:m

gmc=countg(f,1,N)\*Chebyshev(x,l);

gm=gm+gmc;

end

gmxy(i,:)=gm';

end