1、 掌握各种图像复原算法的思想，推导出维纳滤波算法。

算法原理与推导：

（1）有限长滤波器

对于一列输入信号 x，一般的无限长线性滤波器输出为：

$$y(n)=\Sigma h(m)x(n-m)\ m=0\cdots\infty$$

实际中，滤波器的长度，即阶数是有限长的，设为 M，则有：

$$y(n)=\Sigma h(m)x(n-m)\ m=0\cdots M$$

即滤波器的当前时刻输出为前 M 个时刻的值经过加权之后得到的。

为便于书写与理解，上式可以写为矩阵形式：

$$y(n)=H(m)*X(n)$$

如果期望信号 d 已知，则可以计算输出与期望信号之间的误差：

$$e(n)=d(n)-y(n)=\ d(n)-\ H(m)*X(n)\ m=0\cdots M$$

Wiener 滤波的目标是，如何确定一个长为 M 的系数序列 H，使得上述误差值最小。

（2）最小均方误差滤波

根据目标函数的不同，又可以将滤波算法细分为不同的类别，一般来说有最小均方误差，最小二乘误差等等，这里讨论最小均方误差。

令目标函数为：

$$MinE[e(n)^2]=\ E[(d(n)-\ H(m)*X(n))^2]$$

当滤波器的系数最优时，目标函数对系数的倒数应该为 0，即：

$$dE[e(n)^2]/dH=0$$

$$2\ E[\ (d(n)-H(m)*X(n))]*\ X(n)=0$$

$$E[(d(n)X(n))-\ H(m)E[X(n)X(n)]=0$$

根据随机过程的知识，上式可以表达为：

$$Rxd-H*Rxx=0$$

其中 Rxd 与 Rxx 分别为输入信号与期望信号的相关矩阵与输入信号的自相关矩阵。

从而有：

$$\mathbf{H=Rxx-1*Rxd}$$

综上，得到了 Wiener 滤波的基本原理与公式推导。

2、模拟原始图像受运动模糊、散焦模糊、湍流模型模糊、高斯模糊并加不同类型噪

声降质退化时，编程实现模糊含噪图像的逆滤波复原，维纳滤波复原、有约束最小二乘滤波复原、盲卷积滤波复原、Lucy-Richardson 滤波复原等，并比较分析得到的实验结果。要求创建用户交互界面，能够实现图像读取、显示不同退化过程、复原结果显示等。（自选图像）

（1） 结果分析

逆滤波是无约束滤波，需要根据最优准则，寻找 f 得到最小的范数。

维纳滤波使用的前提是知道信号和噪声的功率谱，但在实际应用中较难得到，只能根据先验知识进行估计。

约束最小二乘滤波算法需要提供点扩散函数和噪声参数，但很多场合下噪声的参数未知，采用不同的约束条件，使用不同的复原技术。

盲卷积滤波复原的复原效果明显。

Lucky-Richardson 属于图像复原中的非线性算法，与维纳滤波这种较为直接的算法不同，该算法使用非线性迭代技术，在计算量、性能方面都有了一定提升，在噪声未知的情况下仍能得到较好的复原结果。该算法用泊松噪声对未知噪声建模，通过迭代求出最可能的复原图像。

（2） 结果图

1） 逆滤波复原

含高斯噪声图像的复原

2） 维纳滤波复原

含高斯噪声图像的复原

3） 有约束最小二乘滤波复原

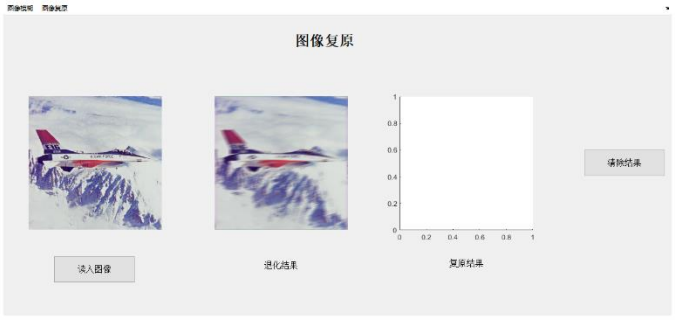
带运动模糊和噪声图像的复原

4） 盲卷积滤波复原

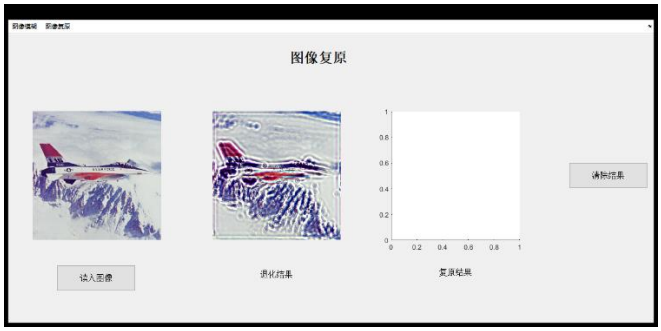
5） Lucy-Richardson 滤波复原

LR100 次迭代

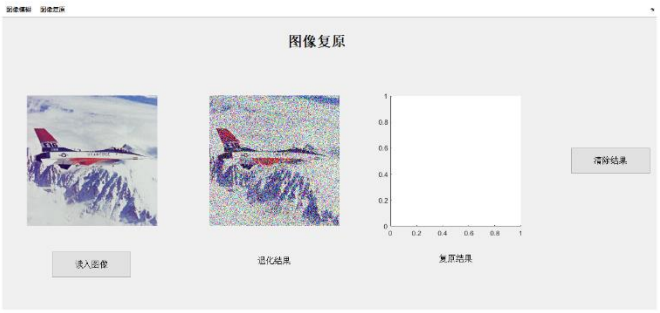
6） 运动模糊



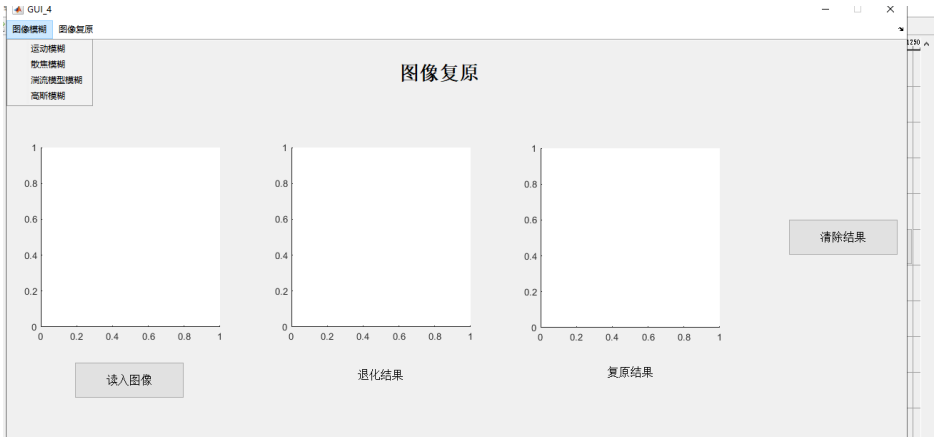
7） 散焦模糊



8） 高斯模糊



(3) 交互界面



(4) 编程代码

```
function varargout = GUI_4(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_4_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_4_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function GUI_4_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = GUI_4_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
% 图像复原菜单栏按钮
function Untitled_1_Callback(hObject, eventdata, handles)

% 逆滤波复原
function Untitled_2_Callback(hObject, eventdata, handles)
```

```
I = getimage(gca);
x1=I(:,:,1);
x1=double(x1);
[r,r1]=size(x1);
y1=fftshift(fft2(x1));
[r,r1]=size(y1);
m=1:r;
m1=1:r1;
[m,m1]=meshgrid(m,m1);
noise=20.*imnoise(zeros(r,r1),'gaussian',0,0.008);
a=double(21/100);
b=double(21/100);
t=double(88/100);
f=ones(r,r1);
g=(m-r/2-1).*a+(m1-r1/2-1).*b+eps;
f=t.*sin(pi.*g).*exp(-j.*pi.*g)./(pi.*g);
h=f'.*y1;
tu=ifft2(h);
tu=abs(tu)+noise;
axes(handles.axes2);
imshow(tu,[]);
handles.img=tu;
guidata(hObject,handles);
y1=h./f';
axes(handles.axes3);
imshow(abs(ifft2(y1)),[]);
handles.img=abs(ifft2(y1));
guidata(hObject,handles);

% 维纳滤波复原
function Untitled_3_Callback(hObject, eventdata, handles)
I = getimage(gca);
%I=rgb2gray((I);
x1=I(:,:,1);
x1=double(x1);
[r,r1]=size(x1);
y1=fftshift(fft2(x1));
[r,r1]=size(y1);
m=1:r;
m1=1:r1;
[m,m1]=meshgrid(m,m1);
noise=20.*imnoise(zeros(r,r1),'gaussian',0,0.008);
a=double(21/100);
b=double(21/100);
```

```
t=double(88/100);
f=ones(r,r1);
g=(m-r/2-1).*a+(m1-r1/2-1).*b+eps;
f=t.*sin(pi.*g).*exp(-j.*pi.*g)./(pi.*g);
h=f'.*y1;
tu=ifft2(h);
tu=abs(tu)+noise;
axes(handles.axes2);
imshow(tu,[]);
handles.img=tu;
guidata(hObject,handles);
h=fftshift(fft2(tu));
x=fftshift(fft2(noise));
K=x.*conj(x)./(y1.*conj(y1));
w=(f.*conj(f))'.*h./(f.*(f.*conj(f)+K'))';
weina=abs(ifft2(w));
axes(handles.axes3);
imshow(weina,[]);
handles.img=weina;
guidata(hObject,handles);

% 有约束最小二乘滤波复原
function Untitled_4_Callback(hObject, eventdata, handles)
I = im2double(getimage(gca));
[hei,wid,~] = size(I);
LEN = 21;
THETA = 11;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(I, PSF, 'conv', 'circular');
Pf = psf2otf(PSF,[hei,wid]);
noise_mean = 0;
noise_var = 0.00001;
blurred_noisy = imnoise(blurred, 'gaussian',noise_mean, noise_var);
axes(handles.axes2);
imshow(blurred_noisy);
handles.img=blurred_noisy;
guidata(hObject,handles);
p = [0 -1 0;-1 4 -1;0 -1 0];
P = psf2otf(p,[hei,wid]);
gama = 0.001;
If = fft2(blurred_noisy);
numerator = conj(Pf);
denominator = Pf.^2 + gama*(P.^2);
deblurred2 = ifft2( numerator.*If./ denominator );
```

```
axes(handles.axes3);
imshow(deblurred2);
handles.img=deblurred2;
guidata(hObject,handles);

%盲卷积滤波复原
function Untitled_5_Callback(hObject, eventdata, handles)
I=getimage(gca);
A=rgb2gray(I);
f = im2double(A);
F = fftshift(fft2(f));
[M, N] = size(F);
[u, v] = meshgrid(1:N, 1:M);
k = 0.0025;
H = exp(-k*((v-M/2).^2+(u-N/2).^2).^(5/6));
G = F.*H;
g = ifft2(ifftshift(G));
g = uint8(abs(g)*255);
axes(handles.axes2);
imshow(g);
handles.img=g;
guidata(hObject,handles);
I = deconv(g, H, 110);
axes(handles.axes3);
imshow(I);
handles.img=I;
guidata(hObject,handles);
function I_new = deconv(I, H, thresh)
if size(I, 3) == 3
    I = rgb2gray(I);
end
I = im2double(I);
G = fftshift(fft2(I));
[M, N] = size(G);
F = G;
[x, y] = meshgrid(1:N, 1:M);
if thresh > M/2
    F = G./(H+eps);
else
    idx = (x-N/2).^2 + (y-M/2).^2 < thresh^2;
    F(idx) = G(idx)./(H(idx)+eps);
end
I_new = ifft2(ifftshift(F));
I_new = uint8(abs(I_new)*255);
```

```
%Lucy-Richardson滤波复原
function Untitled_6_Callback(hObject, eventdata, handles)
I=getimage(gca);
f=rgb2gray(I);
f = im2double(f);
PSF = fspecial('motion', 7, 45);
gb = imfilter( f, PSF, 'circular' );
noise = imnoise( zeros(size(f)), 'gaussian', 0, 0.001 );
g = gb + noise;
axes(handles.axes2);
imshow(gb);
handles.img=gb;
guidata(hObject,handles);

I=getimage(gca);
g = im2double(I);
ori = g;
PSF = fspecial( 'gaussian', 7, 10 );
SD = 0.01;
g = imnoise( imfilter(g, PSF), 'gaussian', 0, SD^2 );
damper = 10*SD;
lim = ceil( size(PSF, 1) / 2 );
weight = zeros( size(g) );
weight( lim+1:end-lim, lim+1:end-lim ) = 1;
numit = 5;
f5 = deconvlucy( g, PSF, numit, damper, weight );
numit = 20;
f20 = deconvlucy( g, PSF, numit, damper, weight );
numit = 50;
f50 = deconvlucy( g, PSF, numit, damper, weight );
numit = 100;
f100 = deconvlucy( g, PSF, numit, damper, weight );
axes(handles.axes3);
imshow(f100, []);
handles.img=f100,[];
guidata(hObject,handles);

% 图像读入按钮
function pushbutton1_Callback(hObject, eventdata, handles)
[filename,pathname,filterindex]=...
uigetfile({'*.*';'*.bmp';'*.tif';'*.png';'*.jpg';'*.jpeg'},'select picture');
str=[pathname filename];
s=str;
```

```
handles.filebig=filterindex;
if filterindex==0
return
else
im1=imread(str);
end
axes(handles.axes1);
imshow(im1);
handles.img=im1;
guidata(hObject,handles);

%页面清除按钮
function pushbutton2_Callback(hObject, eventdata, handles)
axes(handles.axes1);
cla reset;
axes(handles.axes2);
cla reset;
cla reset;

%图像模糊菜单栏按钮
function Untitled_7_Callback(hObject, eventdata, handles)
%运动模糊
function Untitled_8_Callback(hObject, eventdata, handles)
I = im2double(getimage(gca));
[hei,wid,~] = size(I);
LEN = 21;
THETA = 11;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(I, PSF, 'conv', 'circular');
Pf = psf2otf(PSF,[hei,wid]);
noise_mean = 0;
noise_var = 0.00001;
blurred_noisy = imnoise(blurred, 'gaussian',noise_mean, noise_var);
axes(handles.axes2);
imshow(blurred_noisy);
handles.img=blurred_noisy;
guidata(hObject,handles);

%散焦模糊
function Untitled_9_Callback(hObject, eventdata, handles)
I = getimage(gca);
I=im2double(I);
psf=fspecial('disk',10);
res1=deconvblind(I,psf);
```

```
axes(handles.axes2);
imshow(res1);

%高斯模糊
function Untitled_11_Callback(hObject, eventdata, handles)
I = getimage(gca);
m=1;
if m==1
    J=imnoise(I,'gaussian',0,0.1);
    axes(handles.axes2);
    imshow(J);
else if m==2
        J=imnoise(I,'salt & pepper',0.1);
        axes(handles.axes2);
        imshow(J);
end
```