目　录

图像处理试验箱

1 系统要求

1.1 设计要求

（1） 包含 GUI 交互界面

（2） 支持对目标图片直方图进行显示

（3） 支持常见的图像几何变换

（4） 支持不同图像类型转换

（5） 支持空间域滤波

（6） 支持噪声添加

（7） 支持图像线性变换

（8） 支持边缘提取

（9） 能够对变换结果进行实时预览

1.2 方案论证

方案一：采用 MATLAB 提供的图像处理方案，并基于 GUIDE 创建可供用户交互的 GUI 界面。

方案二：采用 Python 的 Pillow 库进行图像处理，并基于 PyQt 创建可供用户交互的 GUI 界面。

1.3 方案选择

本课程设计选用 MATLAB 作为本设计的实现方案。MATLAB 是一个强大的数学软件，广泛应用于图像处理领域。开放性使 MATLAB 广受用户欢迎.除内部函数外,所有 MATLAB 主包文件和各种工具包都是可读可修改的文件,用户通过对源程序的修改或加入自己编写程序构造新的专用工具包。同时 MATLAB 提供更为简洁的操作方法，便于用户快速实现所需功能。

2 软件设计

2.1 设计概述

本程序采用 MATLAB 作为编写平台，因此主要对其进行软件编程。但实现如此功能的程序会较为繁琐复杂，可读性较差。故采用模块化解耦概念设计本程序，降低模块间依赖程度，便于功能添加及删改，提高程序灵活性和可维护性，增强其可扩展性。

软件设计主要包括：主程序设计、线性变化模块设计、几何变换模块设计、空间域滤波模块设计、噪声模块设计、边缘提取模块设计、维纳滤波复原模块设计、图像

类型转换模块设计、直方图模块。

2.2 主程序设计

主程序设计主要完成系统界面的初始化，待处理图像导入、用户界面刷新功能，并判断用户触发的对应事件，交给对应模块进行图像处理任务执行。主程序流程图如图 1 所示。

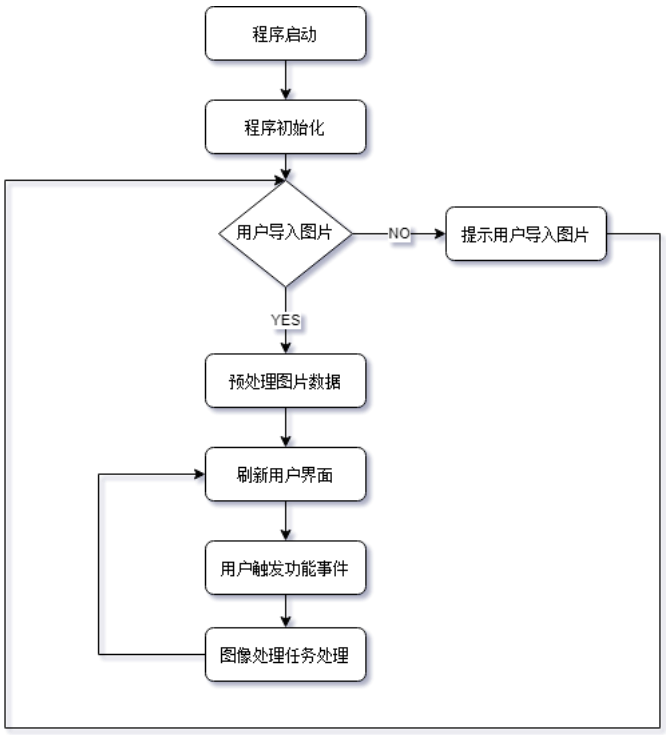

图 1　主程序流程图

初始化代码如下所示

```
% --- Executes just before Image_Processing_LabKit_GUI is made visible.
function Image_Processing_LabKit_GUI_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Image_Processing_LabKit_GUI (see VARARGIN)

% Choose default command line output for Image_Processing_LabKit_GUI
handles.output = hObject;

set(handles.save,'Enable','off');
set(handles.reset,'Enable','off');
set(handles.g1,'Visible','off');
set(handles.g2,'Visible','off');
```

```
set(handles.g3,'Visible','off');
set(handles.g4,'Visible','off');
set(handles.g5,'Visible','off');
set(handles.n1,'Enable','off');
set(handles.n2,'Enable','off');
set(handles.n3,'Enable','off');
set(handles.n4,'Enable','off');
set(handles.n5,'Enable','off');
set(handles.n6,'Enable','off');
set(handles.n7,'Enable','off');
set(handles.n8,'Enable','off');
set(handles.f1,'Enable','off');
set(handles.f2,'Enable','off');
set(handles.f3,'Enable','off');
set(handles.slider1,'Enable','off');
set(handles.slider2,'Enable','off');
set(handles.slider3,'Enable','off');
set(handles.slider4,'Enable','off');
set(handles.slider5,'Enable','off');
set(handles.m1,'Enable','off');
set(handles.m2,'Enable','off');
set(handles.m3,'Enable','off');
set(handles.m4,'Enable','off');
set(handles.m5,'Enable','off');
set(handles.m6,'Enable','off');
set(handles.m7,'Enable','off');
set(handles.m8,'Enable','off');
set(handles.p1,'Enable','off');
set(handles.invc,'Enable','off');
set(handles.g1,'Visible','on');
set(handles.g2,'Visible','on');
set(handles.g3,'Visible','on');
set(handles.g4,'Visible','on');
set(handles.g5,'Visible','on');

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Image_Processing_LabKit_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Executes on button press in load.
function load_Callback(hObject, eventdata, handles)

[file path]=uigetfile({'*.jpg;*.bmp;*.jpeg;*.png'}, '打开文件');
image=[path file];
```

```
handles.file=image;
if (file==0)
    warndlg('您似乎没有打开图片哦...（；´д｀）ゞ') ;
end
[fpath, fname, fext]=fileparts(file);
validex=({'.bmp','.jpg','.jpeg','.png'});
found=0;
for (x=1:length(validex))
if (strcmpi(fext,validex{x}))
       found=1;

set(handles.save,'Enable','on');
set(handles.exit,'Enable','on');
set(handles.reset,'Enable','on');
set(handles.n1,'Enable','on');
set(handles.n2,'Enable','on');
set(handles.n3,'Enable','on');
set(handles.n4,'Enable','on');
set(handles.n5,'Enable','on');
set(handles.n6,'Enable','on');
set(handles.n7,'Enable','on');
set(handles.n8,'Enable','on');
set(handles.f1,'Enable','on');
set(handles.f2,'Enable','on');
set(handles.f3,'Enable','on');
set(handles.slider1,'Enable','on');
set(handles.slider2,'Enable','on');
set(handles.slider3,'Enable','on');
set(handles.slider4,'Enable','on');
set(handles.slider5,'Enable','on');
set(handles.m1,'Enable','on');
set(handles.m2,'Enable','on');
set(handles.m3,'Enable','on');
set(handles.m4,'Enable','on');
set(handles.m5,'Enable','on');
set(handles.m6,'Enable','on');
set(handles.m7,'Enable','on');
set(handles.m8,'Enable','on');
set(handles.p1,'Enable','on');
set(handles.invc,'Enable','on');

handles.img=imread(image);
handles.original=imread(image);
axes(handles.g1);
cla;
```

```
imshow(handles.original);
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);
break;
end
end
if (found==0)
     errordlg('文件扩展名似乎不正确哦，请从可用扩展名[.jpg、.jpeg、.bmp、.png]中选择文件','哎呀，出错了！');
end
% 检查图像是否为彩色，并更新对应直方图
mysize=size(handles.img);
if numel(mysize)>2
    updateg345(handles)
else
    updateg3_1(handles)
end
guidata(hObject,handles);

% hObject    handle to load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2.3 线性变化模块设计

线性变化模块可以对图像的 R/G/B 三个通道及整体亮度进行线性调节，其程序流程图如图 2 所示。

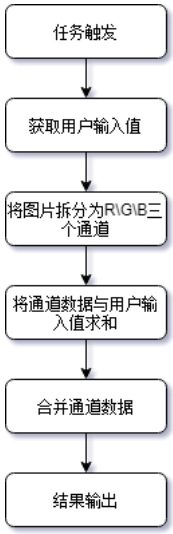

图 2　线性变化模块流程图

用户可以通过滑动滑块实现对图像的实时调节。

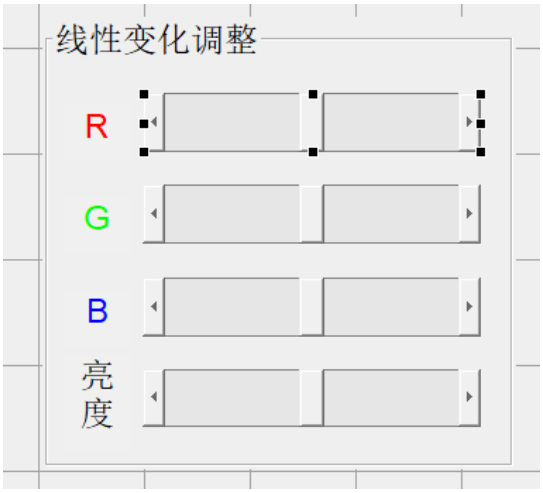

图 3　线性变化模块用户界面

以红色通道线性变换为例的代码如下所示

```
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)

x=get(hObject,'Value');
r=handles.img(:,:,1);
g=handles.img(:,:,2);
b=handles.img(:,:,3);
r1=r+x;
rcon=cat(3,r1,g,b);
axes(handles.g2);
cla;
imshow(rcon)
handles.img=rcon;
updateg345(handles)

% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2.4　几何变换模块设计

几何变换模块可以实现对图像的左右/上下翻转，以及任意角度的旋转变换，其程序流程图如图 4 所示。

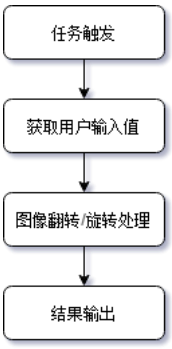

图 4　几何变换模块流程图

用户可以通过按钮和滑块对图像进行几何变换。

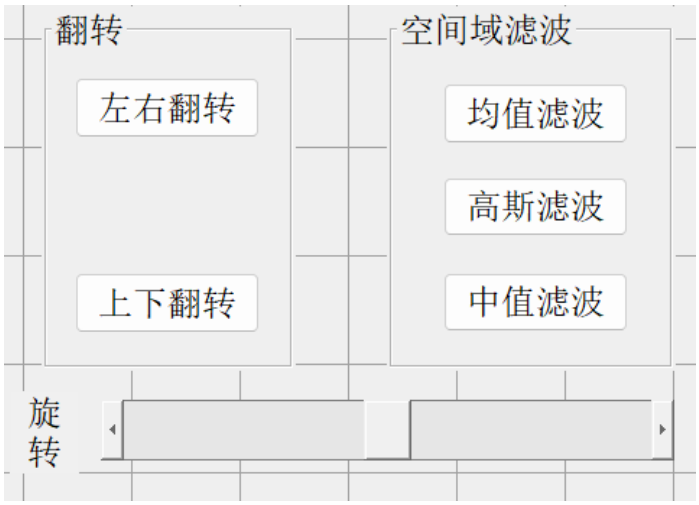

图 5　几何变换模块用户界面

几何变换模块采用 **fliplr**()函数进行左右翻转，**flipud**()进行上下翻转，**imrotate**()函数进行图像旋转变换，代码如下所示

```
% 左右翻转
% --- Executes on button press in m6.
function m6_Callback(hObject, eventdata, handles)

handles.img=fliplr(handles.img);
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);

mysize=size(handles.img);
if numel(mysize)>2
    updateg345(handles)
else
    updateg3_1(handles)
end
```

```
% hObject    handle to m6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% 上下翻转
% --- Executes on button press in m7.
function m7_Callback(hObject, eventdata, handles)

handles.img=flipud(handles.img);
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);

% 检查图像是否为彩色
mysize=size(handles.img);
if numel(mysize)>2
    updateg345(handles)
else
    updateg3_1(handles)
end

% hObject    handle to m7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% 旋转图像
% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)

rrv=(get(hObject,'Value'));
handles.rot=handles.img;
handles.rot=imrotate(handles.rot,rrv);
axes(handles.g2);
cla;
imshow(handles.rot);
guidata(hObject,handles)

% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
```

2.5 空间域滤波模块设计

空间域滤波模块可以实现对图像的均值滤波、高斯滤波、中值滤波功能，其流程图如图 6 所示。

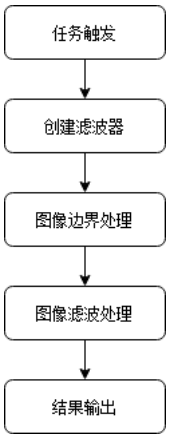

图 6 空间域滤波模块流程图

用户可以通过按钮对图像进行空间域滤波。

图 7 空间域滤波模块用户界面

对于均值滤波和高斯滤波，程序先使用 **fspecial()**函数创建对应的滤波器，再采用 **imfilter()**进行滤波，对于中值滤波，程序采用 **medfilt2()**对三个颜色通道进行分别处理，最后再合并的方式，具体代码如下所示。

```
% 均值滤波
% --- Executes on button press in f1.
function f1_Callback(hObject, eventdata, handles)

h=fspecial('average'); % 创建一个平均滤波器，用于图像平滑处理
handles.img=imfilter(handles.img,h,'replicate'); % 使用创建的平均滤波器对图像进行滤波，边界处理采用复制边缘像素的方法
axes(handles.g2);
```

```
cla;
imshow(handles.img)
guidata(hObject,handles);

% 检查图像是否为彩色
mysize=size(handles.img);
if numel(mysize)>2
    updateg345(handles)
else
    updateg3_1(handles)
end

% hObject    handle to f1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% 高斯滤波
% --- Executes on button press in f2.
function f2_Callback(hObject, eventdata, handles)

hsize=[8 8]; sigma=1.7;
h=fspecial('gaussian',hsize,sigma);
handles.img=imfilter(handles.img,h,'replicate');
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);

% 检查图像是否为彩色
mysize=size(handles.img);
if numel(mysize)>2
    updateg345(handles)
else
    updateg3_1(handles)
end

% hObject    handle to f2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% 中值滤波
% --- Executes on button press in f3.
function f3_Callback(hObject, eventdata, handles)

r=medfilt2(handles.img(:,:,1));
g=medfilt2(handles.img(:,:,2));
```

```
b=medfilt2(handles.img(:,:,3));
handles.img=cat(3,r,g,b);
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);

mysize=size(handles.img);
if numel(mysize)>2
    updateg345(handles)
else
    updateg3_1(handles)
end

% hObject    handle to f3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2.6 噪声模块设计

噪声处理模块支持为图像提供高斯噪声、泊松噪声、椒盐噪声、斑点噪声、瑞利噪声、指数噪声、伽马噪声、均匀噪声共八种噪声。其程序流程图如图 8 所示。

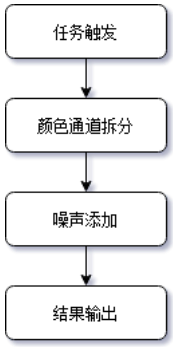

图 8 噪声模块流程图

用户可以通过按钮对图像进行空间域滤波。

图 9 噪声模块用户界面

对于高斯、泊松、椒盐、斑点这四种噪声，本程序采用 **imnoise()**函数实现，实现代码以高斯噪声为例，如下所示。

```
% 高斯噪声
% --- Executes on button press in n1.
function n1_Callback(hObject, eventdata, handles)

handles.img = imnoise(handles.img,'gaussian');
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);

% 检查图像是否为彩色
mysize=size(handles.img);
if numel(mysize)>2
    updateg345(handles)
else
    updateg3_1(handles)
end

% hObject    handle to n1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

对于瑞利、指数、伽马、均匀这四种噪声，本程序采用自定义函数 **add_noise()**函数实现，具体实现流程位于 add_noise.m 文件，实现代码以瑞利噪声为例，如下所示。

```
function IJ = add_noise(I, type, x, y)
% add_noise 函数用以产生前面所述几种噪声的随机序列.
% input:
% I: 输入图像矩阵，为灰度图像
% type:  字符串，取值随噪声种类而定
% 高斯噪声:     gaussian，参数为(x,y)，默认值为(0,10)
```

```
% 瑞利噪声:     rayleigh，参数为x，默认值为30
% 伽马噪声:     gamma，参数为(x,y),默认值为(2,10)
% 指数噪声:     exp，参数为x,默认值为15
% 均匀分布:     uniform，参数为(x,y)，默认值为(-20,20)
% 椒盐噪声:     salt & pepper: 强度为x，默认值为0.02
% output:
% IJ:添加噪声后的图像
% example:
% I=imread('a.bmp');
% IJ=add_noise(I,'salt & pepper',0.1);
% imshow(IJ)

% 预处理
if ndims(I)>=3
    I=rgb2gray(I);
end

[M,N]=size(I);

% 设置默认噪声类型
if  nargin == 1
    type='gaussian';
end

% 开始处理
switch lower(type)
    %高斯噪声的情况
    case 'gaussian'
        if nargin<4
            y=10;
        end
        if nargin <3
            x=0;
        end
        % 产生高斯分布随机数
        R = normrnd(x,y,M,N);
        IJ=double(I)+R;
        IJ=uint8(round(IJ));
        %均匀噪声的情况
    case 'uniform'
        if nargin<4
            y=3;
        end
        if nargin <3
            x=-3;
```

```
        end

        % 产生均匀分布随机数
        R = unifrnd(x,y,M,N);
        IJ=double(I)+R;
        IJ=uint8(round(IJ));

        %椒盐噪声的情况
    case 'salt & pepper'
        if nargin < 3
            x= 0.02;
        end

        % 调用 imnoise 函数
        IJ=imnoise(I,'salt & pepper', x);

        %瑞利噪声的情况
    case 'rayleigh'
        if nargin < 3
            x = 30;
        end

        % 产生瑞利分布随机数
        R = raylrnd(x,M,N);
        IJ=double(I)+R;
        IJ=uint8(round(IJ));

        %指数噪声的情况
    case 'exp'
        if nargin < 3
            x = 15;
        end
        R=exprnd(x,M,N);
        IJ=double(I)+R;
        IJ=uint8(round(IJ));

        %伽马噪声的情况
    case 'gamma'
        if nargin <4
            y=10;
        end
        if nargin<3
            x=2;
        end
```

```
        R=gamrnd(x,y,M,N);
        IJ=double(I)+R;
        IJ=uint8(round(IJ));
    otherwise
        error('Unknown distribution type.')
end
```

```
% 瑞利噪声
% --- Executes on button press in n5.
function n5_Callback(hObject, eventdata, handles)
% 提取三个颜色通道
r=handles.img(:,:,1);
g=handles.img(:,:,2);
b=handles.img(:,:,3);

% 分别添加瑞利噪声并还原图像
r1=add_noise(r, 'rayleigh',30);
g1=add_noise(g, 'rayleigh',30);
b1=add_noise(b, 'rayleigh',30);
rcon=cat(3,r1,g1,b1);

% 显示图像
axes(handles.g2);
cla;
imshow(rcon)
handles.img=rcon;
updateg345(handles)

% hObject    handle to n5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2.7 边缘提取模块设计

边缘提取模块支持 Sobel、Roberts、LOG 三种方法的边缘提取，程序流程图如图 10 所示。

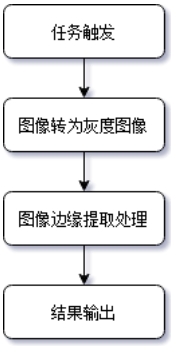

图 10 边缘提取模块流程图

用户可以通过按钮对图像进行边缘提取。

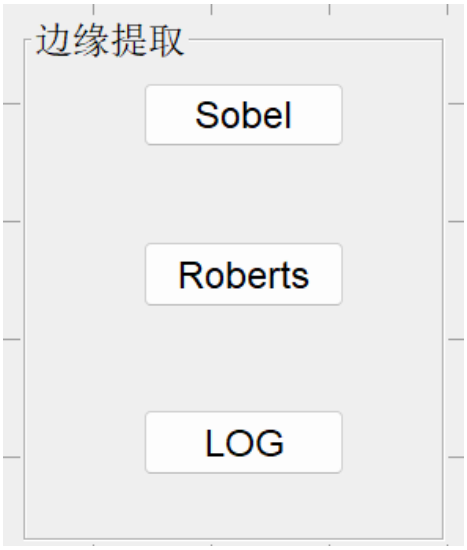

图 11 边缘提取模块用户界面

本程序先对图像类型进行判断，如为彩色图像则转换为灰度图像，然后使用 edge() 函数进行对应类型的边缘提取，具体实现代码如下所示。

```
% Sobel 边缘提取
% --- Executes on button press in m1.
function m1_Callback(hObject, eventdata, handles)

mysize=size(handles.img);
if numel(mysize)>2
    handles.img=rgb2gray(handles.img);
end
handles.img=edge(handles.img,'sobel');
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);
updateg3_1(handles);

% hObject    handle to m1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Roberts 边缘提取
% --- Executes on button press in m2.
function m2_Callback(hObject, eventdata, handles)

% 检查图像是否为彩色
mysize=size(handles.img);
if numel(mysize)>2
    handles.img=rgb2gray(handles.img);
end
handles.img=edge(handles.img,'roberts');
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);
updateg3_1(handles);

% hObject    handle to m2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% LOG 边缘提取
% --- Executes on button press in m3.
function m3_Callback(hObject, eventdata, handles)

% 检查图像是否为彩色
mysize=size(handles.img);
if numel(mysize)>2
    handles.img=rgb2gray(handles.img);
end
handles.img=edge(handles.img,'log');
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);
updateg3_1(handles);

% hObject    handle to m3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2.8 维纳滤波复原模块设计

维纳滤波复原模块先对图像添加了运动模糊和噪声，然后使用维纳滤波去卷积。具体流程图如图 12 所示。

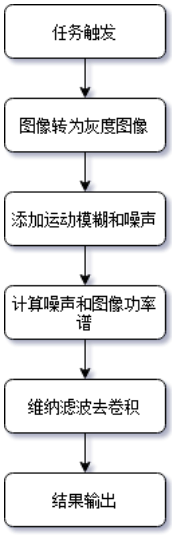

图 12 维纳滤波复原模块流程图

本程序设置运动模糊的长度 LEN 为 20，设置运动模糊的角度 THETA 为 10 度。使用 **fspecial**()函数创建一个运动模糊的脉冲响应函数（PSF），使用 **imfilter**()函数将运动模糊的 PSF 应用到图像 I 上，并使用’conv’和’circular’选项进行卷积，生成与图像大小相同的随机噪声，其标准差为 0.03。使用 **imadd**()函数将噪声添加到运动模糊的图像上，得到含噪声的图像 K。计算噪声的快速傅里叶变换（FFT）的绝对值平方，并存储在 NP 中。计算噪声功率 NPower，通过将 NP 的所有元素相加然后除以噪声元素的总数。计算 NP 的逆 FFT，并使用 **fftshift**()和 **real**()函数调整其位置并获取实部，得到噪声的自相关 NCORR。对原始图像 I 执行类似的操作以计算图像的功率 IPower 和自相关 ICORR。使用 **deconvwnr**()函数对含噪声的图像 K 进行维纳滤波去卷积，使用前面计算得到的 PSF、噪声功率和图像功率。

```
% --- Executes on button press in p1.
function p1_Callback(hObject, eventdata, handles)
%维纳滤波复原，复原含有噪声和运动模糊的图片

% 检查图像是否为彩色
mysize=size(handles.img);
if numel(mysize)>2
    handles.img = rgb2gray(handles.img);
end

I=handles.img;
I=im2double(I);
LEN=20; % 参数设置
```

```
THETA=10;
PSF=fspecial('motion',LEN,THETA); % 产生 PSF
J=imfilter(I,PSF,'conv','circular');% 运动模糊
noise=0.03*randn(size(I));
K=imadd(J,noise); % 添加噪声
NP=abs(fft2(noise)).^2;
NPower=sum(NP(:))/prod(size(noise));
NCORR=fftshift(real(ifft2(NP)));
IP=abs(fft2(I).^2);
IPower=sum(IP(:))/prod(size(I));
ICORR=fftshift(real(ifft2(IP)));
L=deconvwnr(K,PSF,NCORR,ICORR);
handles.img=L;

axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);

updateg3_1(handles)

% hObject    handle to p1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2.9 图像类型转换模块设计

图像类型转换模块支持将图像进行二值化、灰度、索引等操作，流程图如图 13 所示。

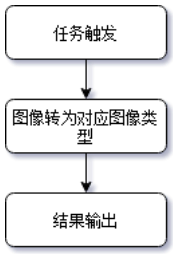

图 13 图像类型转换模块流程图

本程序使用常见的图像类型转换函数实现，具体代码如下所示。

```
% 二值化
% --- Executes on button press in m5.
function m5_Callback(hObject, eventdata, handles)

thresh = graythresh(handles.img);      %自动确定二值化阈值
handles.img = im2bw(handles.img,thresh);
axes(handles.g2);
```

```
cla;
imshow(handles.img);
guidata(hObject,handles);
updateg3_1(handles);

% hObject    handle to m5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% 反色
% --- Executes on button press in invc.
function invc_Callback(hObject, eventdata, handles)

x=handles.img;
r=x(:,:,1); r=256-r;
g=x(:,:,2); g=256-g;
b=x(:,:,3); b=256-b;
handles.img=cat(3,r,g,b);
axes(handles.g2);
cla;
imshow(handles.img);
updateg345(handles);
guidata(hObject,handles);

% hObject    handle to invc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% 灰度
% --- Executes on button press in m4.
function m4_Callback(hObject, eventdata, handles)

handles.img = rgb2gray(handles.img);
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);
updateg3_1(handles);


% hObject    handle to m4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% 索引
```

```
% --- Executes on button press in m8.
function m8_Callback(hObject, eventdata, handles)

map3= colorcube(256);
handles.img=rgb2ind(handles.img,map3);
axes(handles.g2);
cla;
imshow(handles.img);
guidata(hObject,handles);

% 检查图像是否为彩色
mysize=size(handles.img);
if numel(mysize)>2
    updateg345(handles)
else
    updateg3_1(handles)
end

% hObject    handle to m8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2.10 直方图模块设计

直方图模块采用三个坐标系控件显示，支持 RGB 和灰度双模式显示，可以自动根据预览图像绘制直方图。具体流程图如图 14 所示。

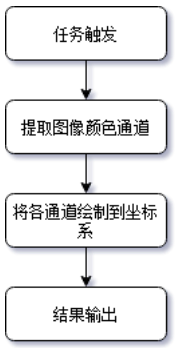

图 14 图像类型转换模块流程图

程序代码如下所示。

```
%彩色直方图更新函数
function updateg345(handles)
ImageData1 = reshape(handles.img(:,:,1), [size(handles.img, 1) * size(handles.img,
2) 1]);
ImageData2 = reshape(handles.img(:,:,2), [size(handles.img, 1) * size(handles.img,
2) 1]);
```

```
ImageData3 = reshape(handles.img(:,:,3), [size(handles.img, 1) * size(handles.img,
2) 1]);
[H1, X1] = hist(ImageData1, 1:5:256);
[H2, X2] = hist(ImageData2, 1:5:256);
[H3, X3] = hist(ImageData3, 1:5:256);
axes(handles.g3);
cla;
hold on;
plot(X1, H1, 'r');
axis([0 256 0 max(H1)]);
axes(handles.g4);
cla;
hold on;
plot(X2, H2, 'g');
axis([0 256 0 max(H2)]);
axes(handles.g5);
cla;
hold on;
plot(X3, H3, 'b');
axis([0 256 0 max(H3)]);

% 灰度直方图更新函数
function updateg3_1(handles)
% 清除其他颜色直方图
axes(handles.g4);
cla;
axes(handles.g5);
cla;

% 绘制直方图
ImageData = reshape(handles.img, [size(handles.img, 1) * size(handles.img, 2) 1]);
[H, X] = hist(ImageData, 1:5:256);
axes(handles.g3);
cla;
hold on;
plot(X, H, 'k');
axis([0 256 0 max(H)]);
```

3　仿真效果图

3.1　功能测试

运行程序，程序界面如图所示。

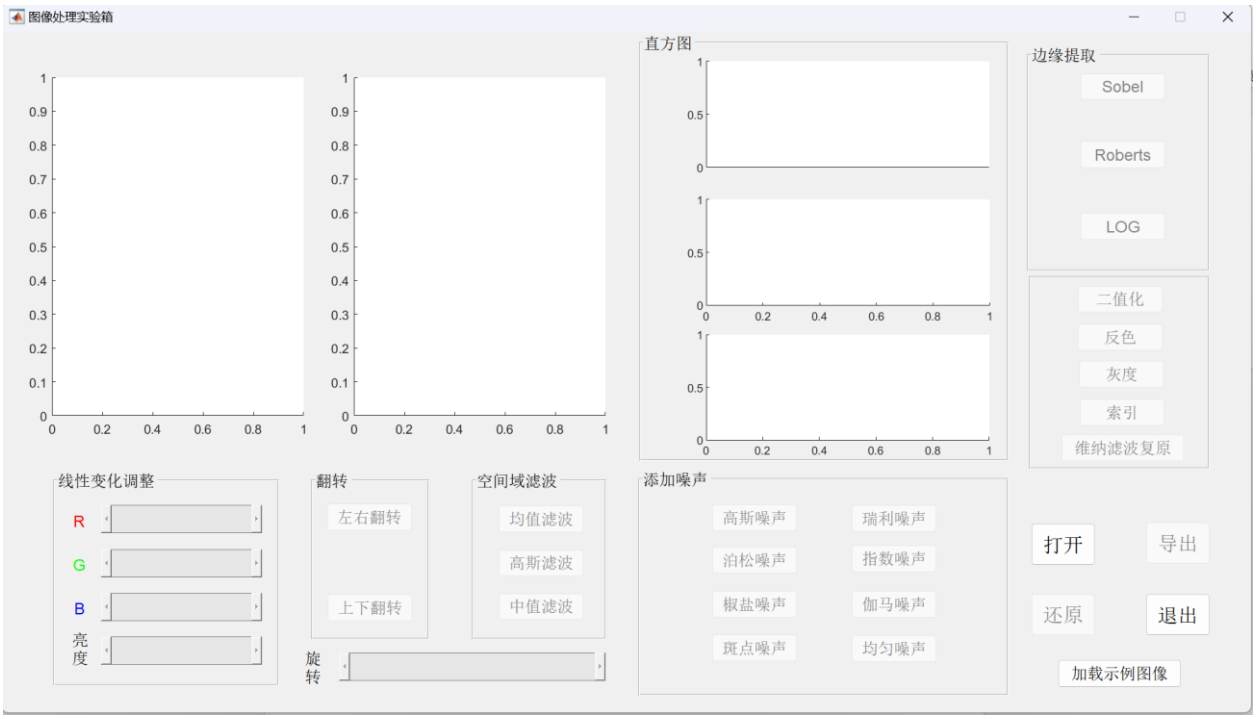

图 15　程序界面

点击【加载示例图片】即可导入演示图片，用户也可以自行使用【打开】导入。

图 16　导入图片

拖动【R】滑块可改变图像红色通道。

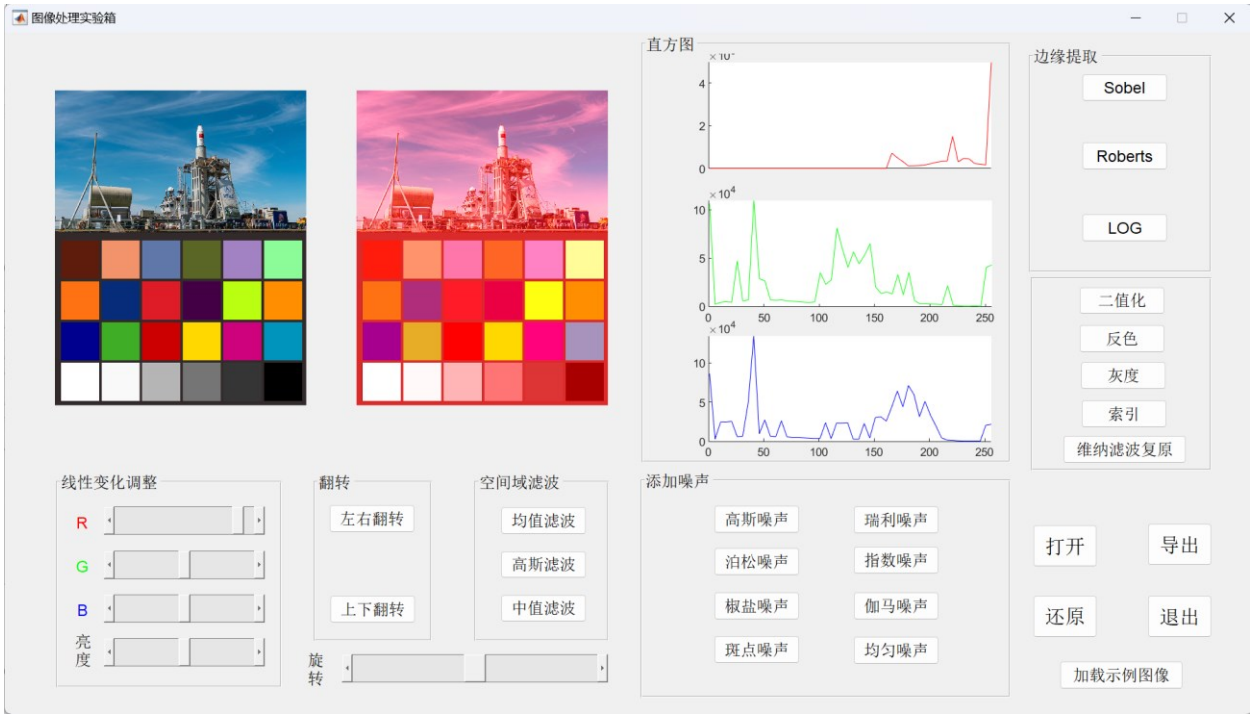

图 17 线性变化演示

拖动【旋转】滑块，可以旋转图像。

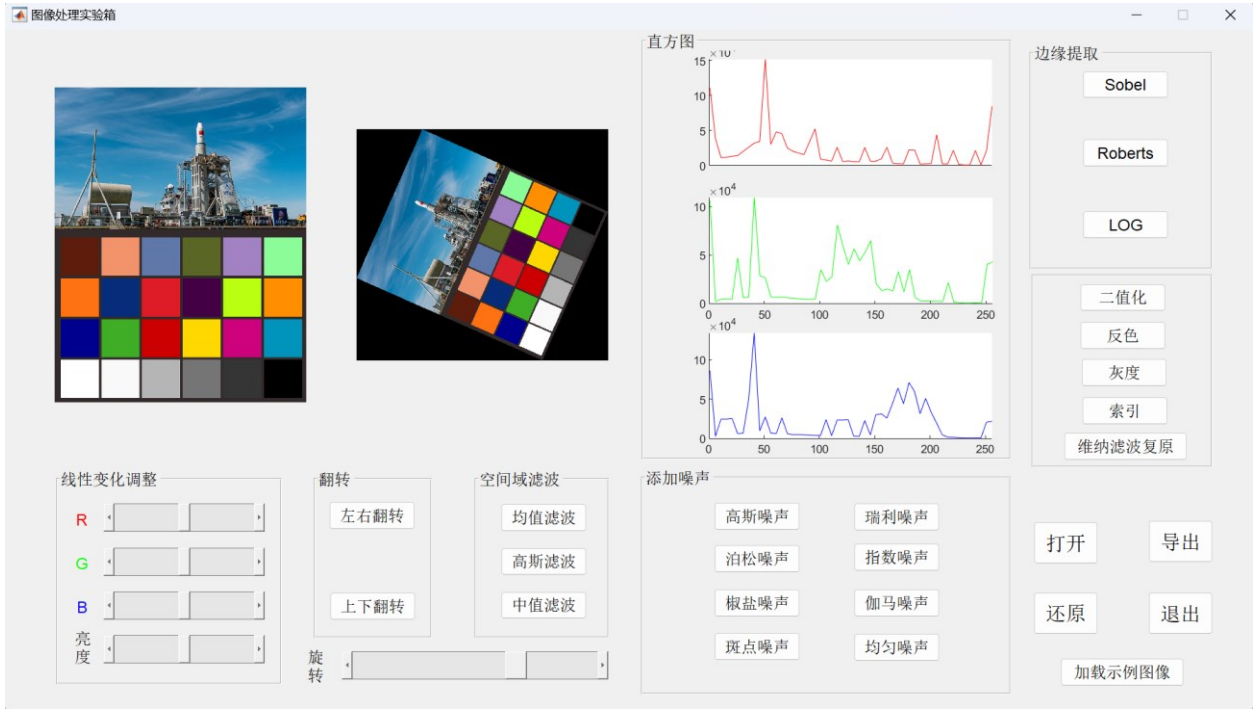

图 18 几何变换演示

点击【瑞利噪声】为图像添加噪声。

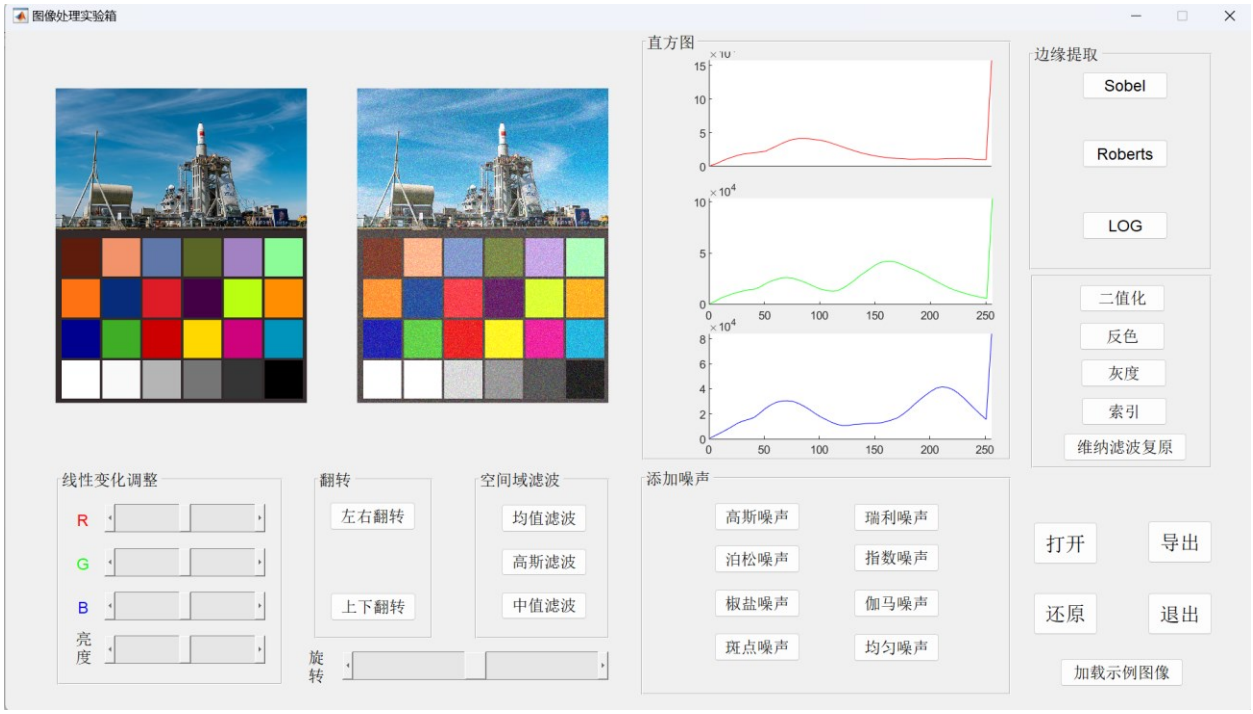

图 19 噪声添加演示

点击【Roberts】为图像进行边缘提取。

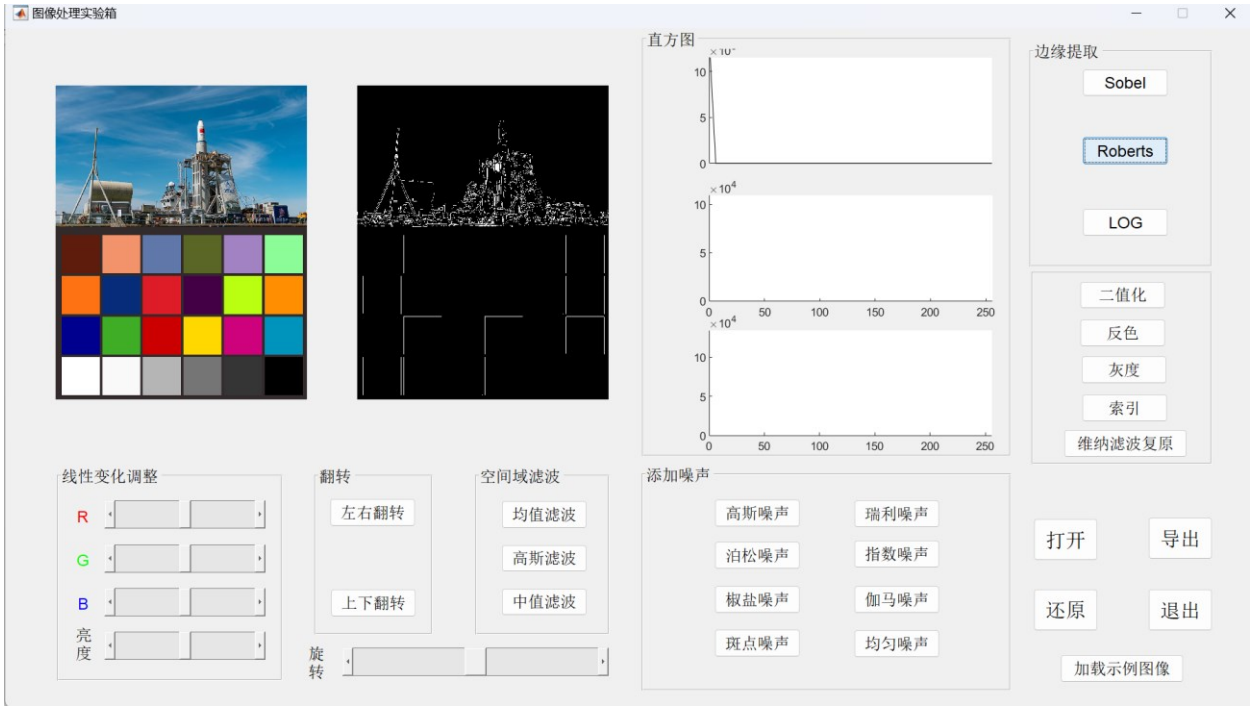

图 20 边缘提取演示

点击【维纳滤波复原】进行维纳滤波实验。

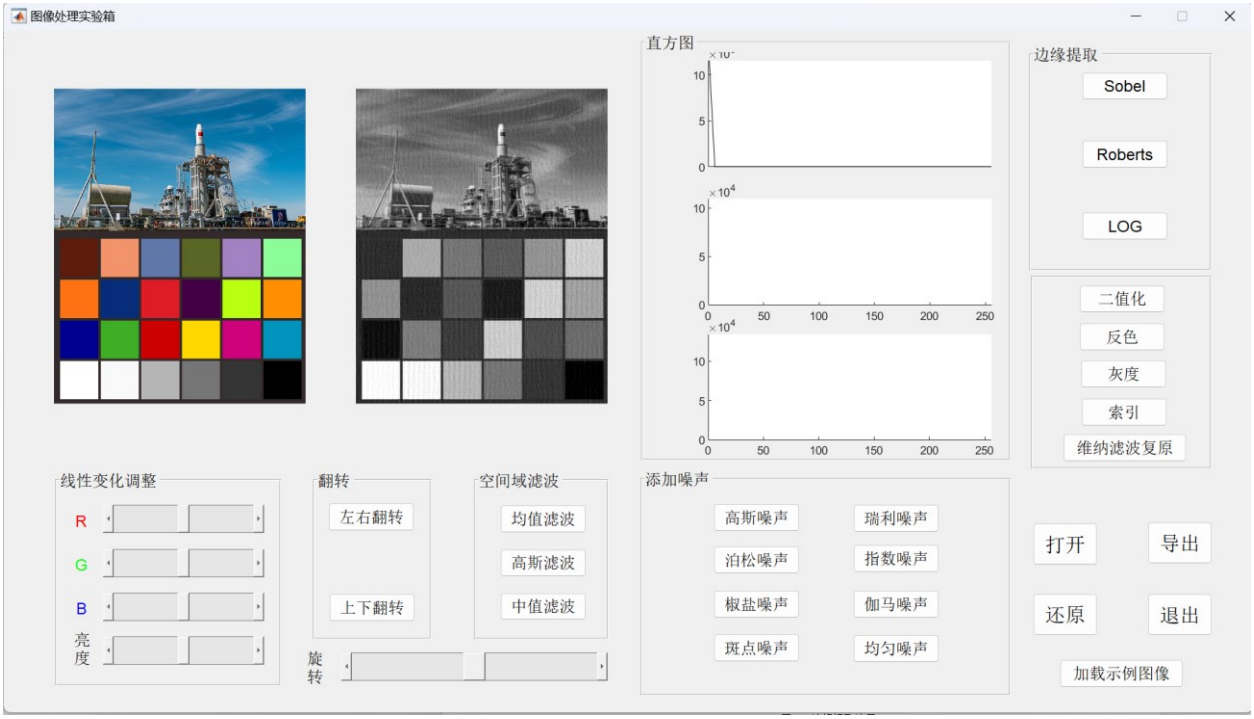

图 21 维纳滤波复原演示

点击【二值化】对图像进行二值化转换。

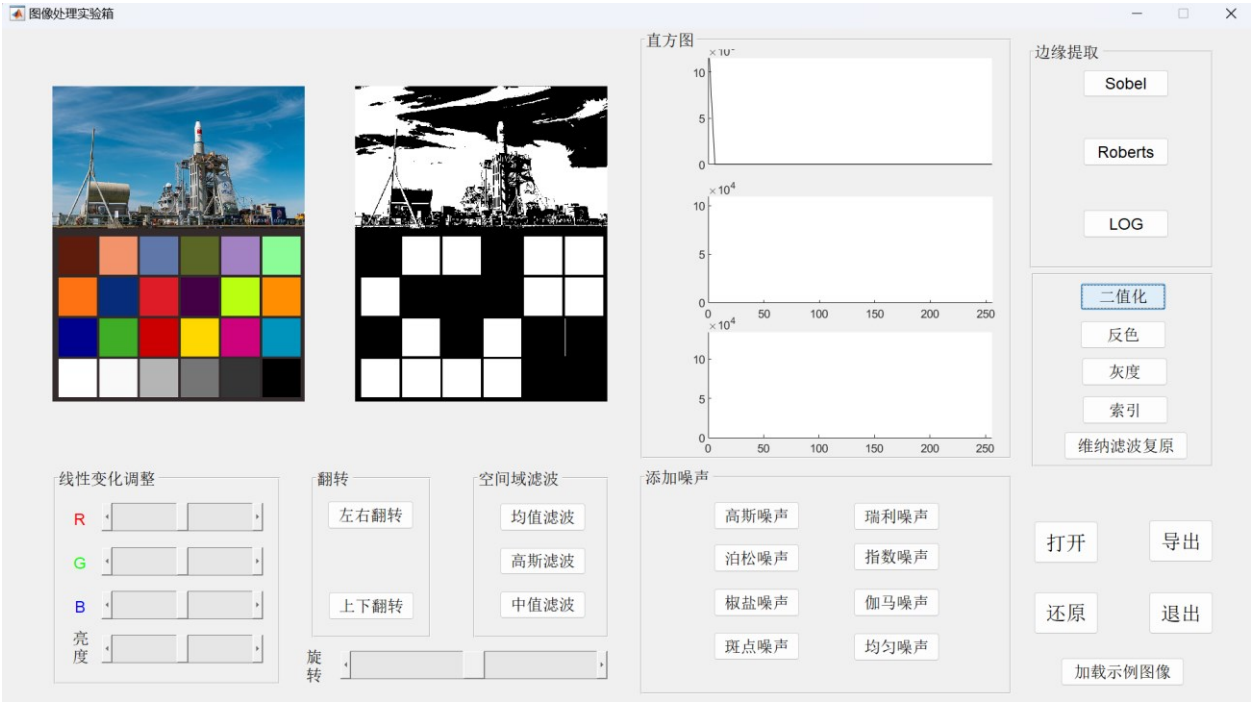

图 22 图像类型转换演示

4 课程总结

首先，通过本次的课程设计我又进一步学习和掌握图像处理的有关知识，加深了对图像处理原理的理解。同时初步掌握简单 MATLAB 图像处理应用的设计、制作、调试的方法。提高了自己动手实践能力和科学的思维能力。

其次，在课程设计过程中，能够不断地发现问题，并想办法解决，如此提高了我自己解决问题的能力。在编写程序方面，我对 MATLAB 编程结构和技巧也有了深刻的理解和领会。此次课程设计还让我明白了流程图的重要性，以前在编程的时候，我从不写流程图，直接开始写程序，这样出现了不该出现的问题。但这次课程设计时，我试着先写出流程图然后按照流程图编写程序，结果错误少了很多，即使有错误只要根据流程图一查就知道错在哪里，这让我节省了大量的时间和精力。所以我认识到，以后要编写程序时，先写流程图是很有必要的。

本次图像处理课程设计让我受益匪浅。在今后的学习和工作中，我将继续努力，不断提高自己的专业技术水平。