北京航空航天大學
BEIHANG UNIVERSITY

2020-2021 春季学期

图像信号处理
实验报告

姓名 祖旭波

学号 18373579

班级 180225

2021 年 6 月 18 日

目录

实验一：图像变换实验

一、实验目的

学会对图像进行傅立叶等变换，在频谱上对图像进行分析，增进对图像频域上的感性认识，并用图像变换进行压缩。

二、实验内容

对 Lena 图像进行傅立叶、离散余弦、哈达玛变换。在频域，对比他们的变换后系数矩阵的频谱情况，进一步，通过逆变换观察不同变换下的图像重建质量情况。

三、实验结果

实验采用获取的图像，为灰度图像，该图像每象素由 8 比特表示。具体要求如下：

（1）对图像进行傅立叶变换、获得变换后的系数矩阵；

（2）将傅立叶变换后系数矩阵的频谱用图像输出，观察频谱；

（3）通过设定门限，将系数矩阵中 90%的（小值）系数置为 0，对图像进行反变换，获得逆变换后图像；

（4）观察逆变换后图像质量，并比较原始图像与逆变后的峰值信噪比（PSNR）。

（5）对输入图像进行离散余弦、哈达玛变换，重复步骤 1-5；

（6）比较三种变换的频谱情况、以及逆变换后图像的质量（PSNR）。

四、实验结果分析

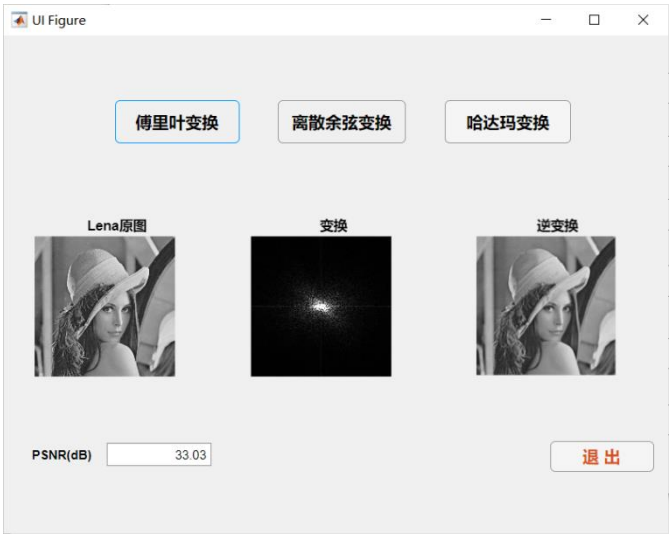

图 1　傅立叶变换及处理后反变换的实验结果

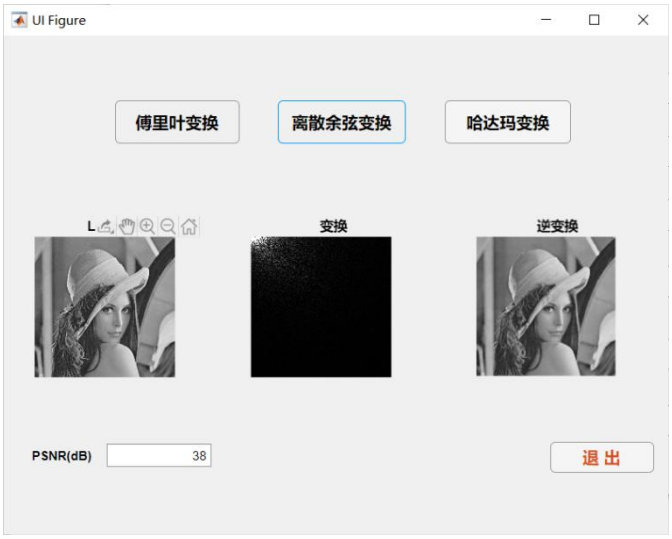

图 2　离散余弦变换及处理后反变换的实验结果

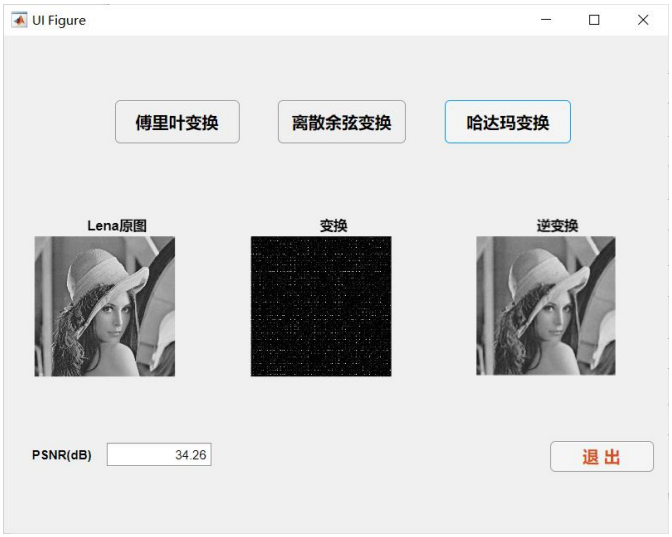

图 3　哈达玛变换及处理后反变换的实验结果

观察实验结果，傅里叶变换和离散余弦变换的频谱均有较明显的能量集中特性，变换域的大值聚集在一小块区域，在图中呈现亮色聚集区域。

经过相应变换并去掉 90%小值后反变换的图像质量有所下降，信噪比降低。但是由于能量集中特性，并没有对原图像效果产生太大影响，总体信噪比较高。从图像反变换的结果看，经离散余弦变换得到的图像质量相对较好，而傅里叶变换相对较差。

实验代码见附录。

实验二：图像复原实验

一、实验目的

利用反向滤波和维纳滤波进行降质图像复原，比较不同参数选择对复原结果的影响。

二、实验内容

（1）利用反向滤波方法进行图像复原；
（2）利用维纳滤波方法进行图像复原。

三、实验结果

（1）输入图像采用实验 1 所获取的图像，对输入图像采用运动降质模型，如下式所示

$$H(u,v)=\frac{T}{\pi(au+bv)}\sin\left[\pi(au+bv)\right]\exp\left\{-j\pi(au+bv)\right\}$$

$$u,v=-N/2,-N/2+1,...,-1,0,1,...,N/2-1$$

与降值图像相关的参数是：$T=5,a=1,b=1$；

（2）对每一种方法通过计算复原出来的图像的峰值信噪比，进行最优参数的选择，包括反向滤波方法中进行复原的区域半径 r_0、维纳方法中的噪声对信号的频谱密度比值 K；

（3）将降质图像和利用最优参数恢复后的图像同时显示出来，以便比较。

四、实验结果分析

图 4　降质图像、反向滤波及维纳滤波对比图

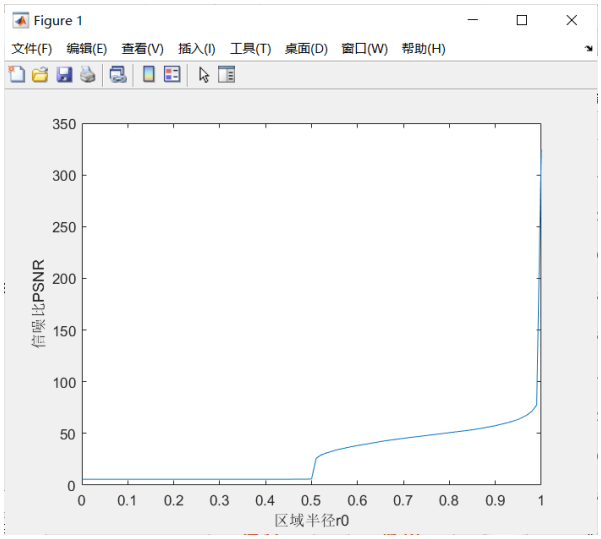

图 5　反向滤波区域半径和信噪比曲线

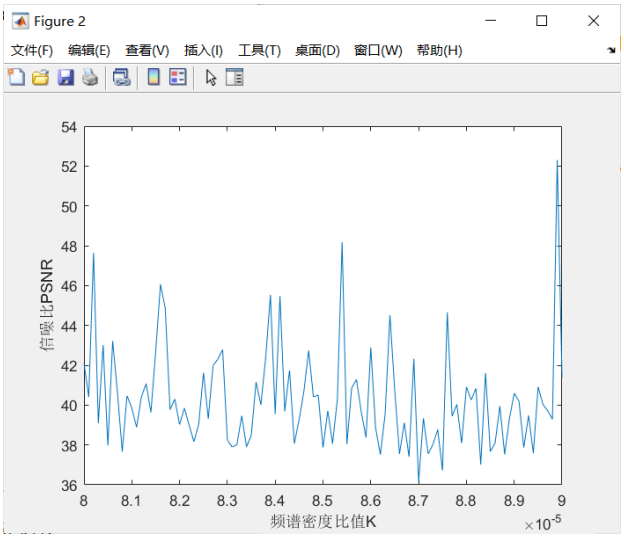

图 6　维纳滤波频谱密度比值 K 和信噪比曲线

对于降质模型，当T=5，a=1，b=1时，滤波复原的效果并不明显，当取T=1，a=0.02，b=0.02时效果较好。

逆滤波时，根据相关曲线可知，r0越接近于1时滤波效果越好，相对标准图像的PSNR值越高。r0等于1时为最佳参数，几乎完全恢复出原图像。

维纳滤波时，手动加入了40dB的AWGN信道噪声，K值和信噪比曲线并没有明显线性关系。考虑到程序运行速度，先手动缩小最小值范围，确定在$8*10^{-5}$~$9*10^{-5}$之间，然后由程序在该范围内搜索。由于噪声的不确定性，每次最佳K值会发生改变。

实验代码见附录。

实验三：图像分割处理实验

一、实验目的

（1）了解图像分割的基本原理，并利用图像分割算法进行图像分割处理；
（2）掌握数学形态学的基本运算。

二、实验内容

（1）利用类间方差阈值算法实现图像的分割处理；
（2）利用形态学处理进行处理结果修正。

三、实验结果

（1）实验用图如图所示；

图 7　实验用图

（2）对输入图像进行平滑处理，以减小噪声对分割处理的影响，比较中值滤波范围取不同值时对图像滤波的效果；

（3）利用类间方差阈值算法对滤波处理后图像进行分割处理，获取分割图像；

（4）利用数学形态学中的腐蚀和膨胀运算处理，剔除分割处理结果中的一些细小的残余误分割点，在进行腐蚀和膨胀运算时可采用半径为 r 的圆形结构元素，注意比较选取不同 r 值时的处理结果（r 分别取 3、5、7）。

四、实验结果分析

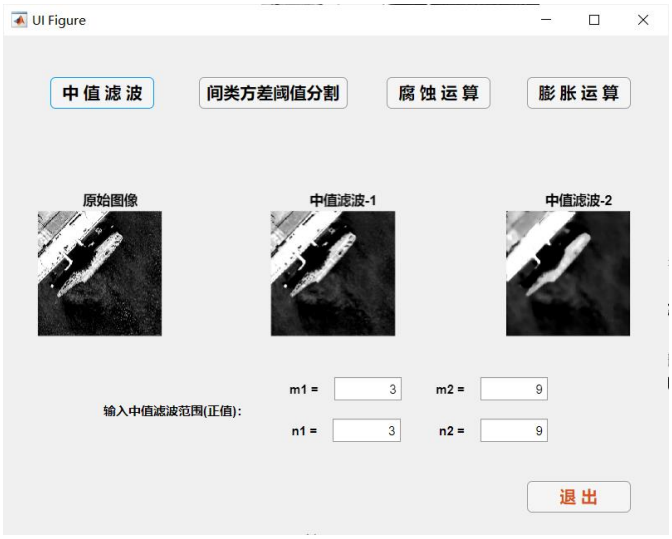

图 8　范围不同时中值滤波对比

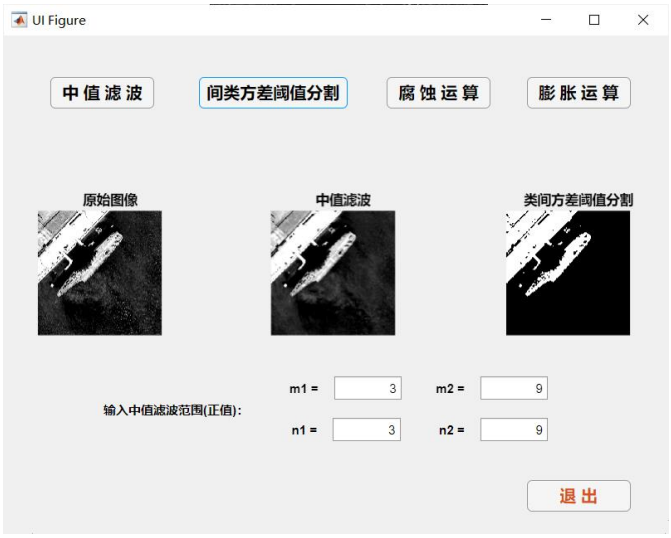

图9　中值滤波范围3*3时的间类方差阈值分割

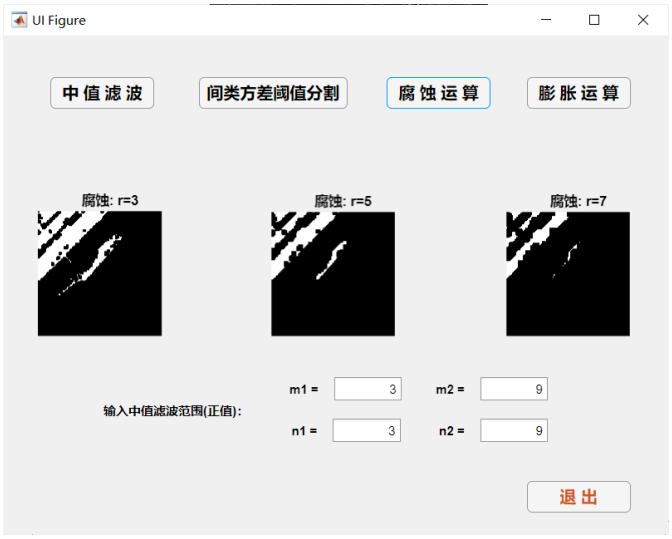

图 10　间类方差阈值分割基础上不同 r 值的腐蚀运算

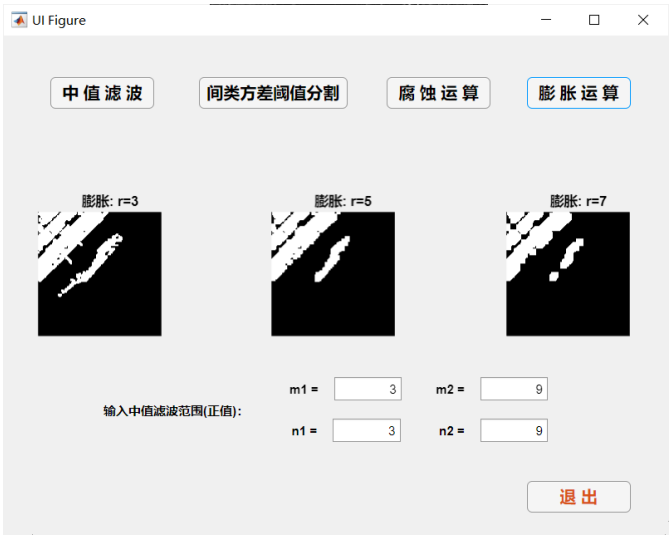

图 11　腐蚀运算基础上同 r 值的膨胀运算

由实验结果可以看出，中值滤波范围越大，图像越平滑，抑制了图像中大部分噪声，但图像也更加模糊，这里为后续实验选择3*3模板。

经过间类方差阈值分割之后，目标的轮廓被提取出来。 腐蚀运算时，目标轮廓缩小，r越大，蚕食部分越大。膨胀运算时，目标轮廓扩大，r越大，扩大部分越大。当r较小时，剔除效果可能较差，当r较大时，复原图像可能越差，所以选择适当的r参数十分重要。

实验代码见附录。

实验四：用 Hough 变换进行曲线的参数提取

一、实验目的

（1）了解边缘检测算子的原理，并利用边缘算子对图像进行检测；
（2）掌握 Hough 变换的基本原理。

二、实验内容

（1）分别将原始图像及加高斯噪声、椒盐噪声后的图像中圆形边缘检测出来；
（2）用 Hough 变换对边缘进行参数提取。

三、实验结果

（1）实验用图像文件：原始图像（houghorg.bmp）、加高斯噪声后图像（houghgau.bmp）和加椒盐噪声后图像（houghsalt.bmp）；
（2）在含有噪声的背景下，先对图像中值滤波，再进行边缘检测；
（3）将目标的边界提取出来。边缘检测算子可利用 matlab 自带函数实现，使用 Robert、Sobel 和 Laplacian 算子；
（4）利用 Hough 变换提取的参数绘制曲线，并叠加在噪声图像上。

实验要点：

（1）利用算子进行边缘检测：可先将加噪以后的图像进行平滑滤波，如采用 7*7 的掩膜模板进行中值滤波；为了对图像中图形边缘进行线性提取，可通过设置阈值将图像变为二值图像，再利用三种不同的算子（Robert、Sobel 和 Laplacian）来完成边缘的检测；

（2）Hough 变换进行曲线参数提取：在使用三种算子对加噪后图像进行边缘检测以后，使用 Hough 变换对检测后图像进行参数提取，并在提取成功以后，使用提取获得的参数进行图像的重建，最后将重建图像叠加到加噪图像中。注意在进行 Hough 变换时，对比观察获得图像与使用算子进行边缘检测获得图像之间的区别

四、实验结果分析

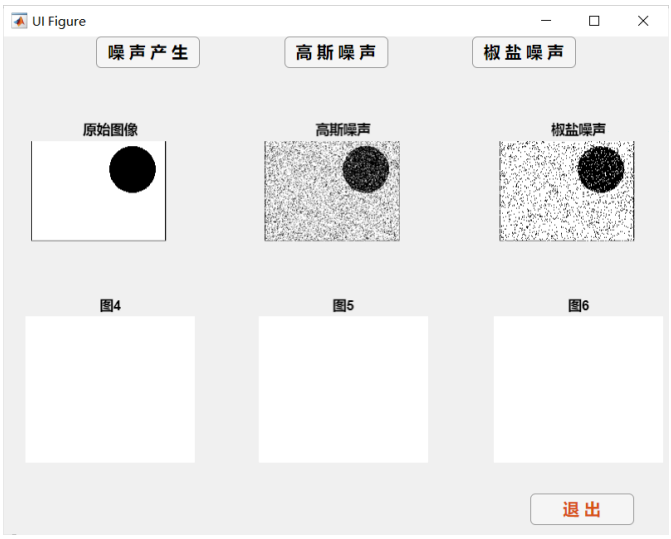

图 12　产生高斯噪声和椒盐噪声（方差均为 0.2）

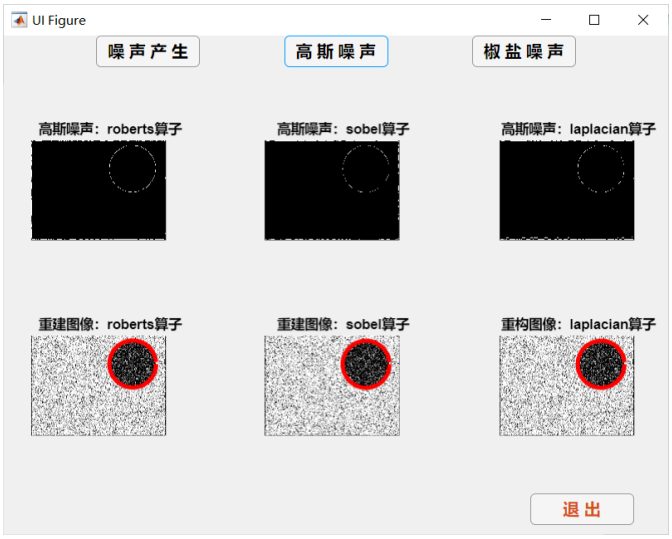

图 13　高斯噪声下提取曲线并重绘

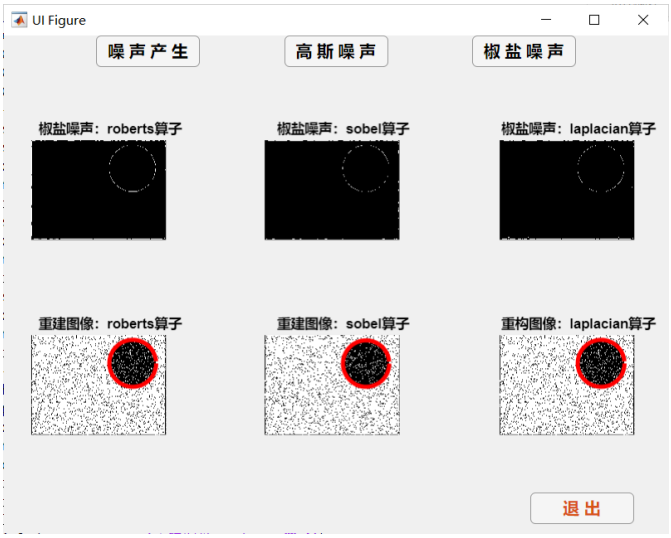

图 14　椒盐噪声下提取曲线并重绘

观察实验结果，三种算子都能有效提取边界，但使用不同的算子提取出来的边缘质量有差别。在噪声的影响下，roberts 算子和 sobel 算子提取边界的效果好于 laplacian 算子，提取结果更精确；laplacian 算子在 Hough 变换（自己定义的函数）后重建的边缘与原图像有一定偏离（图中不太明显，线条较细时更加明显），这可能是由于 laplacia 算子对噪声点有一定的放大作用。 由于 Hough 变换提取的是曲线边界，也就将之前算子检测到的图像边缘的直线型边界滤除了，最终效果较好。

实验代码见附录。

实验五：手写数字识别

一、实验目的

掌握分类、识别问题的实质，了解各种分类问题的机器学习方法，并至少掌握一种，熟悉 Python 编程。

二、实验内容

对实验提供的手写数据库（MNIST）进行训练和测试，最终能够较为准确的识别数据库中的手写体数字。

三、实验结果

编写一完整的 Python 程序，选取一种合适的机器学习方法，对实验提供的手写数据库（MNIST）进行训练和测试，最终能够较为准确的识别数据库中的手写体数字。

数据文件共分为训练集和测试集：

训练数据集：

Training set images: train-images-idx3-ubyte.gz (9.9 MB, 解压后 47 MB, 包含 60,000 个样本)

Training set labels: train-labels-idx1-ubyte.gz (29 KB, 解压后 60 KB, 包含 60,000 个标签)

测试数据集：

Test set images: t10k-images-idx3-ubyte.gz (1.6 MB, 解压后 7.8 MB, 包含 10,000 个样本)

Test set labels: t10k-labels-idx1-ubyte.gz (5KB, 解压后 10 KB, 包含 10,000 个标签)

数字存储格式：每个数字为 28*28 的灰度图，按行拉伸成一个 784 长的向量以字节形式进行存储。为方便处理，解压后可通过程序读取到 NumPy array 中。

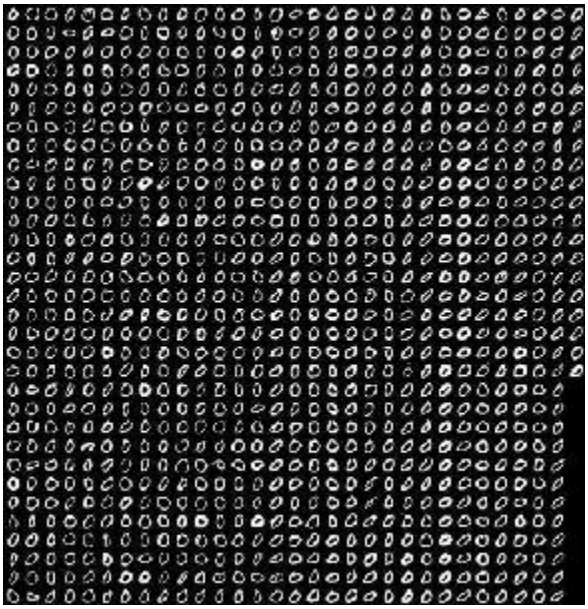

图 15　训练集图像

测试结果及其格式要求：

1、该算法的精确度，并且不能低于 80%；

2、有明确的结果统计形式。例：通过图表显示准确率，Loss 曲线或测试结果可视化等等。（提示：显示方法可以借助 Matlibplot，Tensorboard, Visdom 等等）

提示：可以选取以下几种方法：

（1）利用数字的集合几何形状的特点，计算每幅图的连通域，来进行分类识别；

（2）逻辑回归算法；

（3）支持向量机（SVM）；

现场测评：

根据现场给出的参数，对测试数据集的图片加不同强度的噪声，然后再测试算法在测试集上的精确度。画出算法精确度与噪声强度的关系图。

提示：可以选取以下几种方法：

（1）利用数字的集合几何形状的特点，计算每幅图的连通域，来进行分类识别；

（2）逻辑回归算法；

（3）支持向量机（SVM）；

四、实验结果分析

图 16　加噪和不加噪验证对比

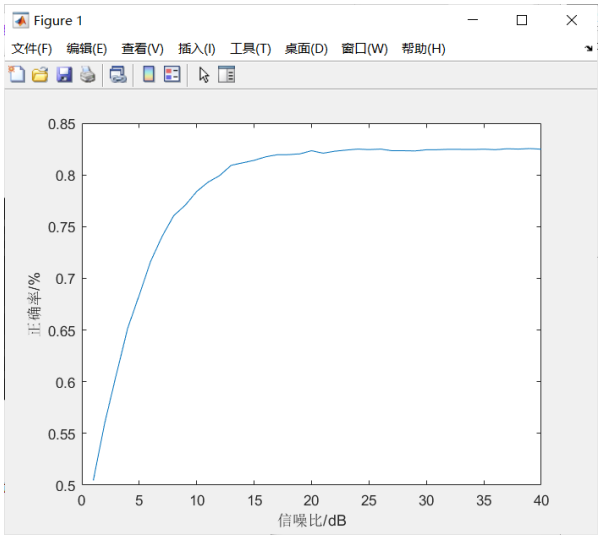

图 17　加噪程度和预测准确率的关系

本实验通过逻辑回归算法设计，用 python 语言编写。该程序没有经过迭代训练，而是通过 minimize 函数寻找局部最优解，返回在当前收敛速度下的最佳 theta 值，存入 dataNew.mat 文件，再被预测函数读取用于最终预测。

本实验最终实现正确率为 82.52%，仍可继续优化，但是考虑到运算速度的问题，并未采用优化方案。由关系曲线可以看到，加入图像的 AWGN 信道噪声的信噪比越小，准确率越低。

实验代码见附录。

附录

一、GUI设计

实验采用 APP Designer 设计图形界面，实验的主函数入口在 EX.mlapp 文件，并在其中分别集成了五个实验的.mlapp 文件（EX_1. mlapp~ EX_1. mlapp）。 GUI 界面如下：

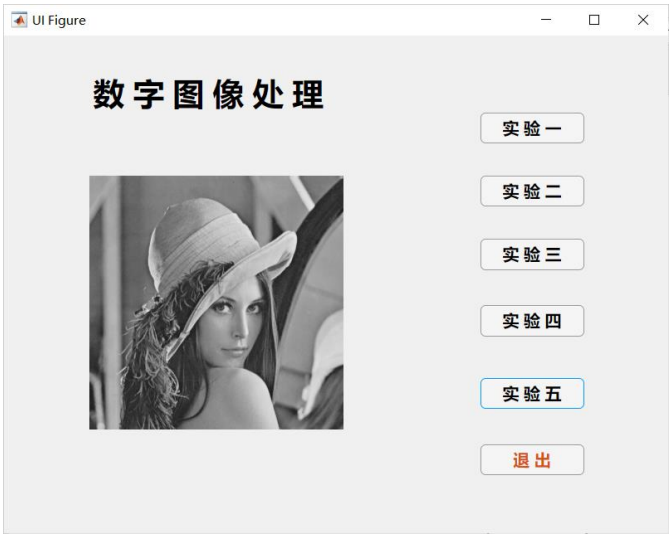

图 18　GUI 界面

注：GUI 部分的代码框架由 APP Designer 自动生成，以下附录中仅展示算法的核心部分，省略 GUI 部分的代码。

二、实验一代码

%函数定义

```
methods (Access = private)
        function [ PSNR ] = PSNR_cal(~,figure1,figure2,bit)
            img=double(figure2);
            imgn=double(figure1);
            Size=size(figure1);
            height=Size(1);
            width=Size(2);
            MAX=2^bit-1;          %图像灰度级数量
            PMSE=sum(sum((img-imgn).^2))/(height*width)/MAX^2; %均方差
            PSNR=-10*log10(PMSE);           %峰值信噪比
        end
end
```

%傅里叶变换按钮回调

```
%------------------------------原图像--------------------------------
        file="lena.bmp";
        figure=imread(file);
        axis(app.UIAxes,'off'); %隐藏坐标轴
```

```
        imshow(figure,'Parent',app.UIAxes); %显示 Lena 原图
        %------------------------------傅立叶变换频谱--------------------
        fourier = fft2(figure);
        shiftf=fftshift(fourier);
        R=real(shiftf); % 取傅立叶变换的实部
        I=imag(shiftf); % 取傅立叶变换的虚部
        spectrum = sqrt(R.^2+I.^2); % 计算频谱幅值
        spectrum_norm =(spectrum-
min(min(spectrum)))/(max(max(spectrum))-min(min(spectrum)))*255;%归一化
        axis(app.UIAxes_2,'off');
        imshow(spectrum_norm,'Parent',app.UIAxes_2); % 显示原图像的频谱
        %------------------------------傅立叶逆变换--------------------
        Size=size(figure);
        height=Size(1);
        width=Size(2);
        spectrum2=sort(reshape(spectrum,[1,width*height])); %排序，取阈值
        threshold=spectrum2(round(width*height*0.90)); %门限
        for p=1:height
            for j=1:width
                if spectrum(p,j)<=threshold
                    R(p,j)=0;
                    I(p,j)=0;
                end
            end
        end
        refigure=ifft2(ifftshift(R+1i*I))/256 ;
        axis(app.UIAxes_3,'off');
        imshow(refigure,'Parent',app.UIAxes_3); %显示逆变化后图象
        PSNR = PSNR_cal(app,figure,256*refigure,8); %计算 PSNR
        app.PSNRdBEditField.Value=PSNR;
```

%离散余弦变换按钮回调

```
  %------------------------------原图像--------------------------------
        file="lena.bmp";
        figure=imread(file);
        axis(app.UIAxes,'off'); %隐藏坐标轴
        imshow(figure,'Parent',app.UIAxes); %显示 Lena 原图
        %-------------------------离散余弦变换频谱----------------------
        DCT=dct2(figure); %离散余弦变换
        axis(app.UIAxes_2,'off');
        imshow(abs(DCT),[0 255],'Parent',app.UIAxes_2); % 显示原图像的频谱
        %-------------------------离散余弦逆变换----------------------
        Size=size(DCT);
        height=Size(1);
        width=Size(2);
```

```
        DCT_sort=sort(reshape(DCT,1,width*height));%排序，取阈值
        threhold=DCT_sort(round(width*height*0.90)); %门限
        DCT(abs(DCT)<threhold)=0;
        refigure=idct2(DCT);
        imshow(refigure,[0 255],'Parent',app.UIAxes_3);
        PSNR = PSNR_cal(app,figure,refigure,8); %计算 PSNR
            app.PSNRdBEditField.Value=PSNR;
```

%哈达玛变换按钮回调

```
%----------------------------原图像-----------------------------------
        file="lena.bmp";
        figure=imread(file);
        imshow(figure,'Parent',app.UIAxes);
        %-----------------------------哈达玛变换-------------------------
------
        H=hadamard(512);
        figure=double(figure);
        DHT=H*figure*H./(512^2);
        imshow(DHT*2^10,[0,255],'Parent',app.UIAxes_2);
        %--------------------------------哈达玛逆变换--------------------
------
        Size=size(figure);
        height=Size(1);
        width=Size(2);
        DHT_sort=sort(reshape(DHT,1,width*height)); %排序，取阈值
        threshold=abs(DHT_sort(round(width*height*0.90))); %门限
        DHT(abs(DHT)<=threshold)=0;
        refigure=H*DHT*H;
        imshow(refigure,[0,255],'Parent',app.UIAxes_3);
        PSNR = PSNR_cal(app,figure,refigure,8); %计算 PSNR
        app.PSNRdBEditField.Value=PSNR;
```

三、实验二代码

%函数定义

```
methods (Access = private)
        function [PSNR] = PSNR_cal(~,figure1,figure2,bit)
            img=double(figure2);
            imgn=double(figure1);
            Size=size(figure1);
            height=Size(1);
            width=Size(2);
            MAX=2^bit-1;          %图像灰度级数量
            MES=sum(sum((img-imgn).^2))/(height*width); %均方差
            PSNR=20*log10(MAX/sqrt(MES));           %峰值信噪比
```

```
        end
end
```

%运行按钮回调

```
%------------------------------原图像--------------------------------
            file="lena.bmp";
            lena=imread(file);
            axis(app.UIAxes,'off'); %隐藏坐标轴
            imshow(lena,'Parent',app.UIAxes); %显示 Lena 原图
            title(app.UIAxes,'Lena 原图');
             %------------------------------降质模型-------------------
            [x,y]=size(lena);
            v=[-x/2:x/2-1];
            u=v';
            T=app.TEditField.Value;
            a=app.aEditField.Value;
            b=app.bEditField.Value;
            A=repmat(a.*u,1,x)+repmat(b.*v,x,1);
            H1=T/pi./A.*sin(pi.*A).*exp(-1i*pi.*A);
            H1(A==0)=T;
             %------------------------------降质变换-------------------
            fftlena=ifftshift(fft2(lena));
            Q0=fftlena.*H1;
            q0=ifft2(ifftshift(Q0));
            axis(app.UIAxes_2,'off'); %隐藏坐标轴

imshow(uint8(255.*mat2gray(real(q0))),'Parent',app.UIAxes_2); %显示
            peaksnr=PSNR_cal(app,lena,q0,8);     % 信噪比
            title(app.UIAxes_2,sprintf('降质图像，PSNR=%.4fdB',peaksnr));
             %------------------------------反向滤波-1-----------------
            r1=app.r1EditField.Value; %r 值读取
            [peaksnr1,q0_inv1]=inverse_filter(app,r1,x,y,H1,Q0,lena);
            axis(app.UIAxes_3,'off'); %隐藏坐标轴

imshow(uint8(255.*mat2gray(real(q0_inv1))),'Parent',app.UIAxes_3);
            title(app.UIAxes_3,sprintf('r_0=%.1f,PSNR=%.4f
dB',r1,peaksnr1));
             %------------------------------反向滤波-2-----------------
            r2=app.r2EditField.Value; %r 值读取
            [peaksnr2,q0_inv2]=inverse_filter(app,r2,x,y,H1,Q0,lena);
            axis(app.UIAxes_4,'off'); %隐藏坐标轴

imshow(uint8(255.*mat2gray(real(q0_inv2))),'Parent',app.UIAxes_4);
            title(app.UIAxes_4,sprintf('r_0=%.1f,PSNR=%.4f
dB',r2,peaksnr2));
```

```
            %------------------------------反向滤波最优参数 r 选取--------
            r0=0;
            peaksnr01=0;
            peaksnr01_0 = ones(1,101);
            for r=0:0.01:1
                [peaksnr11,~]=inverse_filter(app,r,x,y,H1,Q0,lena);
                peaksnr01_0(1,int8(r*100+1)) = peaksnr11;
                if peaksnr11>peaksnr01
                    r0=r;
                    peaksnr01=peaksnr11;
                end
            end
            app.rEditField.Value=r0;
            figure(1);
            r=0:0.01:1;
            plot(r, peaksnr01_0);
            xlabel('区域半径 r0');
            ylabel('信噪比 PSNR');
            %------------------------------反向滤波函数-----------------
            function [peaksnr,q0_inv] =
inverse_filter(~,r,x,y,H1,Q0,lena)
                x_r=round(x*r);
                y_r=round(y*r);
                H_r=ones(x,y).*100000;
                H_r(1:x_r,1:y_r)=H1(1:x_r,1:y_r);
                Q0_inv=Q0./H_r;
                q0_inv=ifft2(ifftshift(Q0_inv));
                peaksnr=PSNR_cal(app,lena,q0_inv,8); % PSNR
            end
            %------------------------------维纳滤波-1-----------------
            K1=app.K1EditField.Value; %K 值读取
            HM=abs(H1.*H1);
            W1=HM./(H1.*(HM+K1));
            [peaksnr3,q_k1]=wiener_filter(app,W1,Q0,lena);
            axis(app.UIAxes_5,'off'); %隐藏坐标轴

imshow(uint8(255.*mat2gray(real(q_k1))),'Parent',app.UIAxes_5);
            title(app.UIAxes_5,sprintf('K=%.5f,PSNR=%.4f
dB',K1,peaksnr3));
            %------------------------------维纳滤波-2-----------------
            K2=app.K2EditField.Value; %K 值读取
            W2=HM./(H1.*(HM+K2));
            [peaksnr4,q_k2]=wiener_filter(app,W2,Q0,lena);
            axis(app.UIAxes_6,'off'); %隐藏坐标轴
```

```
imshow(uint8(255.*mat2gray(real(q_k2))),'Parent',app.UIAxes_6);
            title(app.UIAxes_6,sprintf('K=%.5f,PSNR=%.4f
dB',K2,peaksnr4));
             %------------------------------维纳滤波最优参数 K 选取--------
            K0=0;
            peaksnr02=0;
            peaksnr02_0 = ones(1,101);
            for K=0.00008:0.0000001:0.00009
                W=HM./(H1.*(HM+K));
                [peaksnr22,~]=wiener_filter(app,W,Q0,lena);
                peaksnr02_0(1,int8(K*10000000-800+1)) = peaksnr22;
                if peaksnr22>peaksnr02
                    K0=K;
                    peaksnr02=peaksnr22;
                end
            end
            app.KEditField.Value=K0;
            figure(2);
            K=0.00008:0.0000001:0.00009;
            plot(K, peaksnr02_0);
            xlabel('频谱密度比值 K');
            ylabel('信噪比 PSNR');
             %------------------------------维纳滤波函数-----------------
            function [peaksnr,q_k0] = wiener_filter(~,W_k,Q0,lena)
                Q0_k=ifft2(ifftshift(Q0));
                Q0_kn=awgn(Q0_k,40,'measured');
                Q0_n=ifftshift(fft2(Q0_kn));
                Q_k=Q0_n.*W_k;
                q_k0=ifft2(ifftshift(Q_k));
                peaksnr=PSNR_cal(app,lena,q_k0,8);          % PSNR
            end
```

四、实验三代码

%函数定义

```
methods (Access = private)
       function best = ostu (~,I)  %用类间方差阈值求最佳阈值 best
            s=prod(size(I));
            N=zeros(256,1);
            T=zeros(256,1);

            for k=0:255
                num=size(find(I==k),1);
                N(k+1,1)=num;
```

```
            end

            for t=1:256
                a0=0; a1=0; b0=0; b1=0;
                % 一部分图像
                for m=1:t
                    a0=a0+N(m,1);
                    b0=b0+(m-1)*N(m,1);
                end
                % 另一部分图像
                for m=t+1:256
                    a1=a1+N(m,1);
                    b1=b1+(m-1)*N(m,1);
                end
                w0=a0/s;
                w1=a1/s;
                % u0 u1
                u0=0; u1=0;
                for m=1:t
                    u0=u0+(m-1)*N(m,1)/s;
                end
                for m=t+1:256
                    u1=u1+(m-1)*N(m,1)/s;
                end
                u0=u0/w0;
                u1=u1/w1;
                T(t,1)=w0*w1*(u0-u1)*(u0-u1);
            end

            best=find(T==max(max(T)))-1;
        end

        function im = erode_exp3(~,r,dock3)    % 腐蚀函数
                ele=strel('disk',r,8);
                im=imerode(dock3,ele);
        end

        function im = dilate_exp3(~,r,dock3)  %膨胀函数
            ele=strel('disk',r,8);
            im=imdilate(dock3,ele);
        end
end
```

%中值滤波按钮回调

```
%------------------------------原图像--------------------------------
```

```
            file="shiyan3.bmp";
            dock1=rgb2gray(imread(file));
            axis(app.UIAxes,'off'); %隐藏坐标轴
            imshow(dock1,'Parent',app.UIAxes); %显示原图
            title(app.UIAxes,'原始图像');
             %------------------------------中值滤波-1-----------------
            m1=app.m1EditField.Value;
            n1=app.n1EditField.Value;
            dock2=medfilt2(dock1,[m1,n1]);
            axis(app.UIAxes_2,'off'); %隐藏坐标轴
            imshow(dock2,'Parent',app.UIAxes_2); %显示
            title(app.UIAxes_2,'中值滤波-1');
             %------------------------------中值滤波-2-----------------
            m2=app.m2EditField.Value;
            n2=app.n2EditField.Value;
            dock3=medfilt2(dock1,[m2,n2]);
            axis(app.UIAxes_3,'off'); %隐藏坐标轴
            imshow(dock3,'Parent',app.UIAxes_3); %显示
            title(app.UIAxes_3,'中值滤波-2');
```

%间类方差阈值分割按钮回调

```
%------------------------------原图像-------------------------------
            file="shiyan3.bmp";
            dock1=rgb2gray(imread(file));
            axis(app.UIAxes,'off'); %隐藏坐标轴
            imshow(dock1,'Parent',app.UIAxes); %显示原图
            title(app.UIAxes,'原始图像');
             %------------------------------中值滤波-------------------
            dock2=medfilt2(dock1,[3,3]);
            axis(app.UIAxes_2,'off'); %隐藏坐标轴
            imshow(dock2,'Parent',app.UIAxes_2); %显示
            title(app.UIAxes_2,'中值滤波');
             %------------------------------类间方差阈值分割-------------
            t=ostu(app,dock2);
            dock3=dock2;
            dock3(dock3<t)=0;
            dock3(dock3>=t)=255;
            axis(app.UIAxes_3,'off'); %隐藏坐标轴
            imshow(dock3,'Parent',app.UIAxes_3); %显示
            title(app.UIAxes_3,'类间方差阈值分割');
```

%腐蚀运算按钮回调

```
%------------------------------原图像-------------------------------
            file="shiyan3.bmp";
            dock1=rgb2gray(imread(file));
```

```
 %------------------------------中值滤波-------------------
dock2=medfilt2(dock1,[3,3]);
 %------------------------------类间方差阈值分割------------
t=ostu(app,dock2);
dock3=dock2;
dock3(dock3<t)=0;
dock3(dock3>=t)=255;
%------------------------------腐蚀运算--------------------
r1=3;
im1=erode_exp3(app,r1,dock3);
imshow(im1,'Parent',app.UIAxes);
title(app.UIAxes,sprintf('腐蚀: r=%d',r1));
r2=5;
im2=erode_exp3(app,r2,dock3);
imshow(im2,'Parent',app.UIAxes_2);
title(app.UIAxes_2,sprintf('腐蚀: r=%d',r2));
r3=7;
im3=erode_exp3(app,r3,dock3);
imshow(im3,'Parent',app.UIAxes_3);
title(app.UIAxes_3,sprintf('腐蚀: r=%d',r3));
```

%膨胀运算按钮回调

```
%------------------------------原图像-------------------------------
file="shiyan3.bmp";
dock1=rgb2gray(imread(file));
%------------------------------中值滤波--------------------
dock2=medfilt2(dock1,[3,3]);
%------------------------------类间方差阈值分割-------------
t=ostu(app,dock2);
dock3=dock2;
dock3(dock3<t)=0;
dock3(dock3>=t)=255;
%------------------------------腐蚀后膨胀运算---------------
r1=3;
im1=erode_exp3(app,r1,dock3);
im4=dilate_exp3(app,r1,im1);
imshow(im4,'Parent',app.UIAxes);
title(app.UIAxes,sprintf('膨胀: r=%d',r1));
r2=5;
im2=erode_exp3(app,r2,dock3);
im5=dilate_exp3(app,r2,im2);
imshow(im5,'Parent',app.UIAxes_2);
title(app.UIAxes_2,sprintf('膨胀: r=%d',r2));
r3=7;
im3=erode_exp3(app,r3,dock3);
```

```
            im6=dilate_exp3(app,r3,im3);
            imshow(im6,'Parent',app.UIAxes_3);
            title(app.UIAxes_3,sprintf('膨胀: r=%d',r3));
```

五、实验四代码

```
%函数定义
methods (Access = private)
        function [par1, par3] = Hough(~,BW)
            r_max=100;
            r_min=40;
            step_r=1;
            step_angle=pi/20;
            p=0.5;
            [m,n] = size(BW);
            size_r = round((r_max-r_min)/step_r)+1;
            size_angle = round(2*pi/step_angle);
            hough_space = zeros(m,n,size_r);
            [rows,cols] = find(BW);
            ecount = size(rows);
            % Hough 变换
            % 将图像空间(x,y)对应到参数空间(a,b,r)
            % a = x-r*cos(angle)
            % b = y-r*sin(angle)
            for i=1:ecount
                for r=1:size_r
                    for k=1:size_angle
                        a = round(rows(i)-(r_min+(r-
1)*step_r)*cos(k*step_angle));
                        b = round(cols(i)-(r_min+(r-
1)*step_r)*sin(k*step_angle));
                        if(a>0 & a<=m & b>0 & b<=n)
                            hough_space(a,b,r) = hough_space(a,b,r)+1;
                        end
                    end
                end
            end
            % 搜索超过阈值的聚集点
            max_para = max(max(max(hough_space)));
            index = find(hough_space>=max_para*p);
            length = size(index);
            hough_circle = false(m,n);
            for i=1:ecount
                for k=1:length
                    par3 = floor(index(k)/(m*n))+1;
```

```
                    par2 = floor((index(k)-(par3-1)*(m*n))/m)+1;
                    par1 = index(k)-(par3-1)*(m*n)-(par2-1)*m;
                    if((rows(i)-par1)^2+(cols(i)-par2)^2<(r_min+(par3-
1)*step_r)^2+5&...
                            (rows(i)-par1)^2+(cols(i)-
par2)^2>(r_min+(par3-1)*step_r)^2-5)
                        hough_circle(rows(i),cols(i)) = true;
                    end
                end
            end
            % 打印检测结果
            for k=1:length
                par3 = floor(index(k)/(m*n))+1;
                par2 = floor((index(k)-(par3-1)*(m*n))/m)+1;
                par1 = index(k)-(par3-1)*(m*n)-(par2-1)*m;
                par3 = r_min+(par3-1)*step_r;
            end
            % viscircles([par2 par1],par3);
            par1=[par2 par1];
        end
end
```

%噪声产生按钮回调

```
            shape=rgb2gray(imread('houghorg.bmp'));
            gauss=imnoise(shape,'gaussian',0,0.2);
            pepper=imnoise(shape,'salt & pepper',0.2);
            %-----------------------------------原图像-----------------
            axis(app.UIAxes,'off'); %隐藏坐标轴
            imshow(shape,'Parent',app.UIAxes); %显示
            title(app.UIAxes,'原始图像');
            %-----------------------------------高斯噪声---------------
            axis(app.UIAxes_2,'off'); %隐藏坐标轴
            imshow(gauss,'Parent',app.UIAxes_2); %显示
            title(app.UIAxes_2,'高斯噪声');
            %-----------------------------------椒盐噪声---------------
            axis(app.UIAxes_3,'off'); %隐藏坐标轴
            imshow(pepper,'Parent',app.UIAxes_3); %显示
            title(app.UIAxes_3,'椒盐噪声');
```

%高斯噪声按钮回调

```
            shape=rgb2gray(imread('houghorg.bmp'));
            gauss=imnoise(shape,'gaussian',0,0.2);
            gs0=medfilt2(gauss,[9,9]);       %滤波
            gs0(gs0>127)=255;               %线性提取
            gs0(gs0<128)=0;
```

```
%-----------------------------------求算子-----------------
ps=cell(1,3);
ps{1}=edge(gs0,'roberts');            % roberts 算子
axis(app.UIAxes,'off'); %隐藏坐标轴
imshow(ps{1},'Parent',app.UIAxes);
title(app.UIAxes,'高斯噪声：roberts 算子');
ps{2}=edge(gs0,'sobel');              % sobel 算子
axis(app.UIAxes_2,'off'); %隐藏坐标轴
imshow(ps{2},'Parent',app.UIAxes_2);
title(app.UIAxes_2,'高斯噪声：sobel 算子');
ps{3}=edge(gs0,'log');                % laplacian 算子
axis(app.UIAxes_3,'off'); %隐藏坐标轴
imshow(ps{3},'Parent',app.UIAxes_3);
title(app.UIAxes_3,'高斯噪声：laplacian 算子');
%-----------------------------------重建图像---------------
theta=0:0.1:2*pi;
[par1,par3]=Hough(app,ps{1});
axis(app.UIAxes_4,'off'); %隐藏坐标轴
imshow(gauss,'Parent',app.UIAxes_4);
hold(app.UIAxes_4,'on');
Circle1=par1(1)+par3*cos(theta);
Circle2=par1(2)+par3*sin(theta);
plot(app.UIAxes_4,Circle1,Circle2,'r','LineWidth',3);
title(app.UIAxes_4,'重建图像：roberts 算子');
[par1,par3]=Hough(app,ps{2});
axis(app.UIAxes_5,'off'); %隐藏坐标轴
imshow(gauss,'Parent',app.UIAxes_5);
hold(app.UIAxes_5,'on');
Circle1=par1(1)+par3*cos(theta);
Circle2=par1(2)+par3*sin(theta);
plot(app.UIAxes_5,Circle1,Circle2,'r','LineWidth',3);
title(app.UIAxes_5,'重建图像：sobel 算子');
[par1,par3]=Hough(app,ps{3});
axis(app.UIAxes_6,'off'); %隐藏坐标轴
imshow(gauss,'Parent',app.UIAxes_6);
hold(app.UIAxes_6,'on');
Circle1=par1(1)+par3*cos(theta);
Circle2=par1(2)+par3*sin(theta);
plot(app.UIAxes_6,Circle1,Circle2,'r','LineWidth',3);
title(app.UIAxes_6,'重构图像：laplacian 算子');
```

%椒盐噪声按钮回调

```
shape=rgb2gray(imread('houghorg.bmp'));
pepper=imnoise(shape,'salt & pepper',0.2);
p0=medfilt2(pepper,[9,9]);       %滤波
```

```
p0(p0>127)=255;                    %线性提取
p0(p0<128)=0;
%-----------------------------------求算子-----------------
ps=cell(1,3);
ps{1}=edge(p0,'roberts');            % roberts 算子
axis(app.UIAxes,'off'); %隐藏坐标轴
imshow(ps{1},'Parent',app.UIAxes);
title(app.UIAxes,'椒盐噪声：roberts 算子');
ps{2}=edge(p0,'sobel');              % sobel 算子
axis(app.UIAxes_2,'off'); %隐藏坐标轴
imshow(ps{2},'Parent',app.UIAxes_2);
title(app.UIAxes_2,'椒盐噪声：sobel 算子');
ps{3}=edge(p0,'log');                % laplacian 算子
axis(app.UIAxes_3,'off'); %隐藏坐标轴
imshow(ps{3},'Parent',app.UIAxes_3);
title(app.UIAxes_3,'椒盐噪声：laplacian 算子');
%-----------------------------------重建图像---------------
theta=0:0.1:2*pi;
[par1,par3]=Hough(app,ps{1});
axis(app.UIAxes_4,'off'); %隐藏坐标轴
imshow(pepper,'Parent',app.UIAxes_4);
hold(app.UIAxes_4,'on');
Circle1=par1(1)+par3*cos(theta);
Circle2=par1(2)+par3*sin(theta);
plot(app.UIAxes_4,Circle1,Circle2,'r','LineWidth',3);
title(app.UIAxes_4,'重建图像：roberts 算子');
[par1,par3]=Hough(app,ps{2});
axis(app.UIAxes_5,'off'); %隐藏坐标轴
imshow(pepper,'Parent',app.UIAxes_5);
hold(app.UIAxes_5,'on');
Circle1=par1(1)+par3*cos(theta);
Circle2=par1(2)+par3*sin(theta);
plot(app.UIAxes_5,Circle1,Circle2,'r','LineWidth',3);
title(app.UIAxes_5,'重建图像：sobel 算子');
[par1,par3]=Hough(app,ps{3});
axis(app.UIAxes_6,'off'); %隐藏坐标轴
imshow(pepper,'Parent',app.UIAxes_6);
hold(app.UIAxes_6,'on');
Circle1=par1(1)+par3*cos(theta);
Circle2=par1(2)+par3*sin(theta);
plot(app.UIAxes_6,Circle1,Circle2,'r','LineWidth',3);
title(app.UIAxes_6,'重构图像：laplacian 算子');
```

六、实验五代码

% **EX_5_train.py**

```
    # -*- coding: utf-8 -*-
    import numpy as np
    import scipy.optimize as op
    import scipy.io as scio
    import os
    import gzip


    def load_data(data_folder):
        files   =   ['train-labels-idx1-ubyte.gz',   'train-images-idx3-
    ubyte.gz']
        paths = []
        for fname in files:
            paths.append(os.path.join(data_folder, fname))
        with gzip.open(paths[0], 'rb') as lbpath:
            y_train = np.frombuffer(lbpath.read(), np.uint8, offset=8)
        with gzip.open(paths[1], 'rb') as imgpath:
            x_train     =     np.frombuffer(imgpath.read(),     np.uint8,
    offset=16).reshape(len(y_train), 28, 28)
        return x_train, y_train


    def sigmoid(x):
        return 1 / (1 + np.exp(-x))


    def cost_reg(theta_0, train_images_0, train_labels_0, l0_0):
        m, _ = train_images_0.shape
        hx = sigmoid(np.dot(train_images_0, theta_0.T))
        ln_h = np.log(hx)
        part1 = -np.dot(ln_h.T, train_labels_0.T) / m
        ln_1h = np.log(1 - hx)
        part2 = -np.dot(ln_1h.T, 1 - train_labels_0.T) / m
        reg = l0_0 * np.dot(theta_0, theta_0.T) / (2 * m)
        return (part1 + part2 + reg).flatten()


    def grad_reg(theta_0, train_images_0, train_labels_0, l0_0):
        m, _ = train_images_0.shape
        theta_tempt = theta_0.copy()
        hx = sigmoid(np.dot(train_images_0, theta_tempt.T))
        part1 = np.dot(train_images_0.T, hx - train_labels_0.T)
        part2 = l0_0 * theta_tempt
```

```
        return ((part1 + part2) / m).flatten()


    def fmincg(theta_0, train_images_0, train_labels_0, l0_0,
    num_labels_0):
        for i in range(num_labels_0):
            y_tempt = train_labels_0.copy()
            pos = np.where(train_labels_0 == i)
            neg = np.where(train_labels_0 != i)
            y_tempt[pos] = 1
            y_tempt[neg] = 0
            result_0 = op.minimize(cost_reg, theta_0[i],
    args=(train_images_0, y_tempt, l0_0),
                                   method="TNC", jac=grad_reg)
            theta_0[i] = result_0.x
        return theta_0


    def train():
        #         ==================         导         入         数         据
    ============================================================
        train_images, train_labels = load_data('MNIST_data/')
        train_images = np.array([im.reshape(784) for im in train_images])
        num_labels = 10
        #       ============================       训       练       模       型
    ==========================================
        l0 = 0.001
        num = 100
        m, n = train_images.shape
        theta = np.zeros((num_labels, n))
        for i in range(num):
            print(i)
            left = i * m / num
            right = (i + 1) * m / num
            theta = fmincg(theta, train_images[int(left):int(right)],
    train_labels[int(left):int(right)], l0, num_labels)
        scio.savemat('dataNew.mat', {'theta': theta})
    return 0
```

% **EX_5_test.py**

```
    # -*- coding: utf-8 -*-
    import numpy as np
    import scipy.io as scio
    import os
    import gzip
```

```
def load_data(data_folder):
    files    =    ['t10k-labels-idx1-ubyte.gz',    't10k-images-idx3-
ubyte.gz']
    paths = []
    for fname in files:
        paths.append(os.path.join(data_folder, fname))
    with gzip.open(paths[0], 'rb') as lbpath:
        y_test = np.frombuffer(lbpath.read(), np.uint8, offset=8)
    with gzip.open(paths[1], 'rb') as imgpath:
        x_test     =     np.frombuffer(imgpath.read(),     np.uint8,
offset=16).reshape(len(y_test), 28, 28)
    return x_test, y_test


def sigmoid(x):
    return 1 / (1 + np.exp(-x))


def pred_lr(theta_0, train_images_0):
    train_images_0 = train_images_0.reshape((1, -1))
    result_0 = sigmoid(train_images_0 @ theta_0.T)
    predict = np.argmax(result_0.flatten())
    return predict


def pred_accuracy(theta_0, train_images_0, train_labels_0):
    right_num = 0
    m, _ = train_images_0.shape
    for i in range(m):
        pred = pred_lr(theta_0, train_images_0[i])
        if pred == train_labels_0[i]:
            right_num += 1
    return right_num / m


def test(num):
    #        ==================        导        入        数        据
============================================================
    test_images, test_labels = load_data('MNIST_data/')
    test_images = np.array([im.reshape(784) for im in test_images])
    #        ==================        识        别        预        测
============================================================
```

```
        num = num-1
        data = scio.loadmat('dataNew.mat')
        theta = data['theta']
        result = pred_lr(theta, test_images[int(num)])
        accuracy = pred_accuracy(theta, test_images, test_labels)
        test_image = np.array(test_images[int(num)].reshape((28, 28)))
        test_images = np.array([im.reshape((28, 28)) for im in
    test_images])
        scio.savemat('test.mat', {'test_images': test_images,
    'test_image': test_image,
                                  'result': result, 'accuracy': accuracy,
    'label': test_labels[int(num)]})
        return 0


    def test_n(num):
        #         ==================         导         入         数         据
    ================================================================
        _, test_labels = load_data('MNIST_data/')
        num = num-1
        data = scio.loadmat('dataNew.mat')
        theta = data['theta']
        data_n = scio.loadmat('dataNew_n.mat')
        test_images = data_n['test_image_n']
        test_images = np.array([im.reshape(784) for im in test_images])
        result = pred_lr(theta, test_images[int(num)])
        accuracy = pred_accuracy(theta, test_images, test_labels)
        test_image = np.array(test_images[int(num)].reshape((28, 28)))
        scio.savemat('test_n.mat', {'test_image': test_image, 'result':
    result, 'accuracy': accuracy})
    return 0
```

%训练模型按钮回调

```
py.EX_5_train.test();
```

%预测验证模型按钮回调

```
            num = app.EditField.Value;
            py.EX_5_test.test(num);
            a = load('test.mat');
            axis(app.UIAxes,'off'); %隐藏坐标轴
            imshow(a.test_image,'Parent',app.UIAxes); %显示
            title(app.UIAxes,sprintf('实际值：%d    预测值：%d', a.label,
a.result));
            app.EditField_2.Value = a.accuracy;
            %------------------------------加噪------------------------
```

```
            w = app.dBEditField.Value;
            test_image_n = ones(10000,28,28);
            for i=1:10000
                str = reshape(a.test_images(i,:,:),[28 28]);
                str = awgn(double(str),w,'measured');
                test_image_n(i,:,:) = uint8(str);
            end
            save ('dataNew_n.mat','test_image_n')
            py.EX_5_test.test_n(num);
            b = load('test_n.mat');
            axis(app.UIAxes_2,'off'); %隐藏坐标轴
            imshow(b.test_image,'Parent',app.UIAxes_2); %显示
            title(app.UIAxes_2,sprintf('加噪后/n 实际值：%d    预测值：%d',
a.label, b.result));
            app.EditField_3.Value = b.accuracy;
            %------------------------------绘图------------------------
            accur = ones(1,40);
            for w=1:40
                for i=1:10000
                    str = reshape(a.test_images(i,:,:),[28 28]);
                    str = awgn(double(str),w,'measured');
                    test_image_n(i,:,:) = uint8(str);
                end
                save ('dataNew_n.mat','test_image_n')
                py.EX_5_test.test_n(num);
                b = load('test_n.mat');
                accur(1,w) = b.accuracy;
            end
            t = 1:40;
            plot(t,accur);
            xlabel('信噪比/dB');
            ylabel('正确率/%');
```