实验一 使用Matlab对图像进行基本操作

1. 实验目的：熟悉Matlab编程工具

二、实验内容：

1）模拟数字图像噪声失真（高斯、椒盐、乘性、泊松 ）

2）模拟数字图像滤波操作（均值滤波、中值滤波）

3）观察分析滤波前后图像统计特征差异（灰度直方图，一阶差分图像的直方图）

1. 实验过程

1、模拟数字图像噪声失真

\* 使用matlab内置噪声污染图像函数：g = imnoise(I,type,parameters)

参数说明：I — 输入图像

g — 被污染后图像

type — 加入噪声的类型

Parameters — 添加噪声的参数

1. 添加高斯噪声：

g=imnoise(f, ’gaussian’,m,var)

m：高斯噪声均值，默认为0，var：高斯噪声方差，越大噪点越明显

1. 添加椒盐噪声：

g=imnoise(f,’salt & pepper’,d)

d：噪声密度（包含噪声值的图像区域的百分比）

1. 添加乘性噪声：

g=imnoise(f,’speckle’,var)

乘性噪声：g=f+n\*f ，n：均值为0，方差为var的均匀分布的随机噪声

（4）添加泊松噪声：

g=imnoise(f,'poisson')

从数据中生成泊松噪声，遵循泊松统计

1. 模拟数字图像滤波操作（均值滤波、中值滤波）
2. 均值滤波

\* 去噪思想：用多个像素灰度的平均值来代替每个像素的灰度。有效抑制加性噪声，但容易引起图像模糊。

\* 实例：3x3 均值滤波

l=ones(3,3);

l=l/9;

apic\_g=conv2(double(pic\_g),l);

\* 基本思路：设矩阵模板大小为n x n，从起始位置开始，循环取出输入矩阵（double(pic\_g)）n行n列元素与模板相乘，对每次的求出的n x n新矩阵求和后求均值，将该值赋给n x n区域的中间位置。

1. 中值滤波

\*去噪思想：基于排序统计理论的一种能有效抑制噪声的非线性平滑滤波信号处理技术。首先确定一个以某个像素为中心点的邻域，然后将邻域中各像素的灰度值排序，取其中间值作为中心像素灰度的新值。利用中值滤波可以对图像进行平滑处理。

\*实例：mpic=medfilt2(pic, ,[m n])

参数说明：pic — 输入的带噪声的二维图像矩阵

mpic — 中值滤波处理后的数字信号

\*基本思路：设矩阵模板大小为m x n，将模板覆盖下各元素的中值赋给模板中心位置的元素，顺次沿二维矩阵个点移动模板。

3、观察分析滤波前后图像统计特征差异（灰度直方图，一阶差分图像的直方图）

（1）计算灰度直方图

\*含义：灰度直方图是将数字图像中的所有像素，按照灰度值的大小，统计其出现的频率。

\*所用函数：imhist

1. 计算一阶差分图像直方图

\*目的：统计相邻像素间灰度差值出现的频率。

\*所用函数：firstdifference(自编写函数），imhist

function dimg = firstdifference(img,direction)

imgPad = padarray(img,[1 1],'symmetric','both');%将原图像的边界扩展

[row,col] = size(imgPad);

dimg = zeros(row,col);

switch (direction)

case 'dx',

dimg(:,1:col-1) = imgPad(:,2:col)-imgPad(:,1:col-1);%x方向差分计算，

case 'dy',

dimg(1:row-1,:) = imgPad(2:row,:)-imgPad(1:row-1,:);

otherwise, disp('Direction is unknown');

end;

dimg = dimg(2:end-1,2:end-1);

end

1. 测试数据
2. 模拟数字图像噪声失真

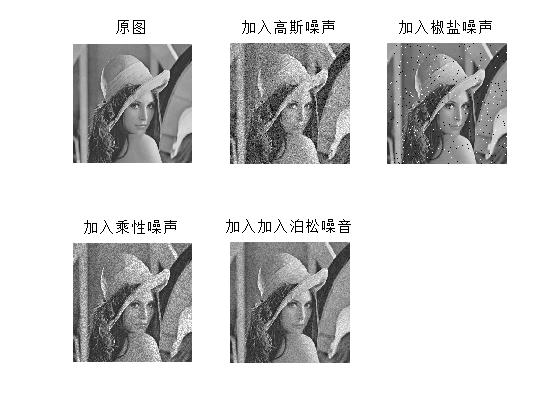


figure 4-1 原图与加入噪声后的图片

1. 模拟数字图像滤波操作



figure 4-2 3x3均值滤波后的图片



figure 4-3 7x7均值滤波后的图片



figure 4-4 中值滤波后的图片

1. 观察分析滤波前后图像统计特征差异（灰度直方图，一阶差分图像的直方图）

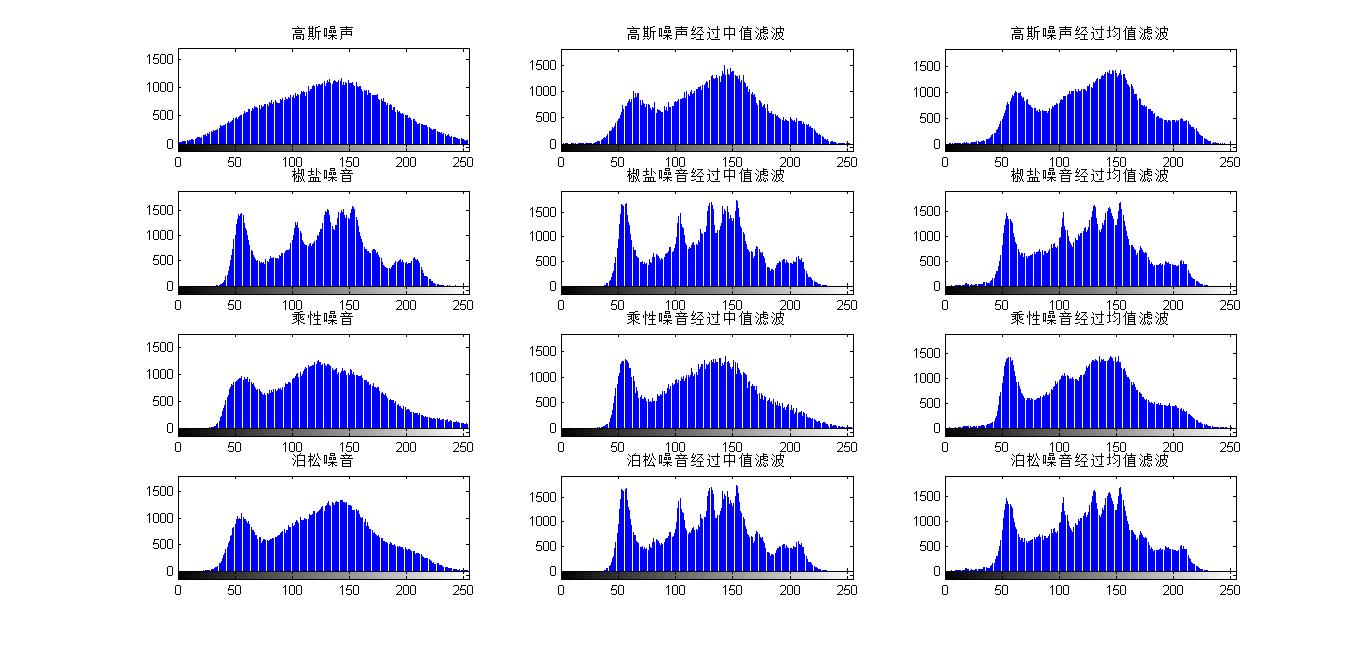


figure 4-5 图像灰度直方图

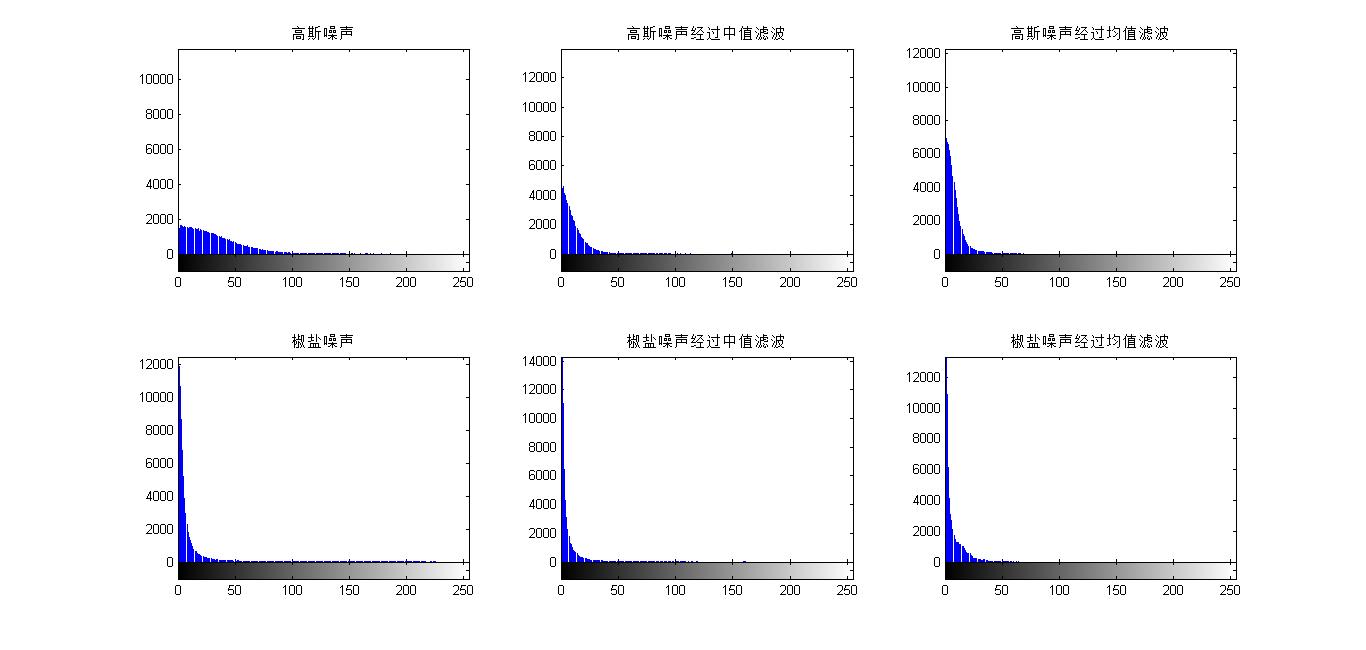


figure 4-6 图像一阶差分图像的直方图

1. 实验结果

高斯噪声出现位置一定，但噪声幅度随机，因此经过均值滤波后图像恢复最好，观察一阶差分图能看出相邻像素间灰度差值小且集中，说明平滑性好。而椒盐噪声出现位置随机，但噪声幅度一定，中值滤波通过数据排序的方法，将图像中被噪声污染的点替换掉，去噪效果最好。观察经过滤波后的图像灰度直方图，在取均值取中值后，图像灰度值更加集中，两种滤波方案均对去噪有一定效果。

1. 附录（实验代码）

Project01.m

%Name: GuoYunting

%Course: 数字内容安全

%aim: 1）模拟数字图像噪声失真（高斯、椒盐…）

% 2）模拟数字图像滤波操作（均值滤波、中值滤波…）

% 3）观察分析滤波前后图像统计特征差异（灰度直方图，一阶差分图像的直方图…）

%

clc

clear all

close all

%% save start time

start\_time=cputime;

%% read in the cover object

file\_name='\_lena\_std\_bw.bmp';

%% 1）模拟数字图像噪声失真（高斯、椒盐…）

figure(1);

pic=imread(file\_name);%原始图像

subplot(2,3,1);

imshow(pic),title('原图');

pic\_g=imnoise(pic,'gaussian',0.02); %加入高斯噪声

subplot(2,3,2);

imshow(pic\_g),title('加入高斯噪声');

pic\_sap=imnoise(pic,'salt & pepper',0.02);%加入椒盐噪声

subplot(2,3,3);

imshow(pic\_sap),title('加入椒盐噪声');

pic\_spe=imnoise(pic,'speckle',0.02);%加入乘性噪声

subplot(2,3,4);

imshow(pic\_spe),title('加入乘性噪声');

pic\_poi=imnoise(pic,'poisson');%加入泊松噪音

subplot(2,3,5);

imshow(pic\_poi),title('加入加入泊松噪音');

%% 2）模拟数字图像滤波操作（均值滤波、中值滤波…）

%中值滤波

figure(2);

mpic\_g=medfilt2(pic\_g,[3 3]); %对加入高斯噪声图像进行中值滤波

subplot(2,2,1);

imshow(mpic\_g),title('高斯噪声');

mpic\_sap=medfilt2(pic\_sap,[3 3]); %对加入椒盐噪声图像进行中值滤波

subplot(2,2,2);

imshow(mpic\_sap),title('椒盐噪声');

mpic\_spe=medfilt2(pic\_spe,[3 3]); %对加入乘性噪声图像进行中值滤波

subplot(2,2,3);

imshow(mpic\_spe),title('乘性噪声');

mpic\_poi=medfilt2(pic\_poi,[3 3]); %对加入泊松噪音图像进行中值滤波

subplot(2,2,4);

imshow(mpic\_poi),title('泊松噪声');

%3x3 均值滤波

figure(3);

l=ones(3,3);

l=l/9;

apic\_g=conv2(double(pic\_g),l);

subplot(2,2,1);

imshow(apic\_g,[]),title('高斯噪声'); %对加入高斯噪声图像进行均值滤波

apic\_sap=conv2(double(pic\_sap),l); %对加入椒盐噪声图像进行均值滤波

subplot(2,2,2);

imshow(apic\_sap,[]),title('椒盐噪声');

apic\_spe=conv2(double(pic\_spe),l); %对加入乘性噪声图像进行均值滤波

subplot(2,2,3);

imshow(apic\_spe,[]),title('乘性噪声');

apic\_poi=conv2(double(pic\_poi),l); %对加入泊松噪音图像进行均值滤波

subplot(2,2,4);

imshow(apic\_poi,[]),title('泊松噪声');

%7x7 均值滤波

figure(4);

l2=ones(7,7);

l2=l2/49;

apic\_g=conv2(double(pic\_g),l);

subplot(2,2,1);

imshow(apic\_g,[]),title('高斯噪声'); %对加入高斯噪声图像进行均值滤波

apic\_sap=conv2(double(pic\_sap),l); %对加入椒盐噪声图像进行均值滤波

subplot(2,2,2);

imshow(apic\_sap,[]),title('椒盐噪声');

apic\_spe=conv2(double(pic\_spe),l); %对加入乘性噪声图像进行均值滤波

subplot(2,2,3);

imshow(apic\_spe,[]),title('乘性噪声');

apic\_poi=conv2(double(pic\_poi),l); %对加入泊松噪音图像进行均值滤波

subplot(2,2,4);

imshow(apic\_poi,[]),title('泊松噪声');

%% 3）观察分析滤波前后图像统计特征差异（灰度直方图，一阶差分图像的直方图…）

%统计灰度直方图

figure(5);

subplot(4,3,1);

imhist(pic\_g); %高斯噪音

title('高斯噪声');

subplot(4,3,2);

imhist(mpic\_g);%中值滤波后

title('高斯噪声经过中值滤波');

subplot(4,3,3);

imhist(uint8(apic\_g));%均值滤波后

title('高斯噪声经过均值滤波');

subplot(4,3,4);

imhist(pic\_sap); %椒盐噪音

title('椒盐噪音');

subplot(4,3,5);

imhist(mpic\_sap);%中值滤波后

title('椒盐噪音经过中值滤波');

subplot(4,3,6);

imhist(uint8(apic\_sap));%均值滤波后

title('椒盐噪音经过均值滤波');

subplot(4,3,7);

imhist(pic\_spe); %乘性噪音

title('乘性噪音');

subplot(4,3,8);

imhist(mpic\_spe);%中值滤波后

title('乘性噪音经过中值滤波');

subplot(4,3,9);

imhist(uint8(apic\_spe));%均值滤波后

title('乘性噪音经过均值滤波');

subplot(4,3,10);

imhist(pic\_poi); %泊松噪音

title('泊松噪音');

subplot(4,3,11);

imhist(mpic\_sap);%中值滤波后

title('泊松噪音经过中值滤波');

subplot(4,3,12);

imhist(uint8(apic\_sap));%均值滤波后

title('泊松噪音经过均值滤波');

%一阶差分图像的直方图

figure(6);

subplot(2,3,1);

diffpic\_g= firstdifference(pic\_g,'dx');

imhist(uint8(diffpic\_g));

title('高斯噪声');

subplot(2,3,2);

diffmpic\_g= firstdifference(mpic\_g,'dx');

imhist(uint8(diffmpic\_g));

title('高斯噪声经过中值滤波');

subplot(2,3,3);

diffapic\_g= firstdifference(apic\_g,'dx');

imhist(uint8(diffapic\_g));

title('高斯噪声经过均值滤波');

subplot(2,3,4);

diffpic\_sap= firstdifference(pic\_sap,'dx');

imhist(uint8(diffpic\_sap));

title('椒盐噪声');

subplot(2,3,5);

diffmpic\_sap= firstdifference(mpic\_sap,'dx');

imhist(uint8(diffmpic\_sap));

title('椒盐噪声经过中值滤波');

subplot(2,3,6);

diffapic\_sap= firstdifference(apic\_sap,'dx');

imhist(uint8(diffapic\_sap));

title('椒盐噪声经过均值滤波');

% display processing time

elapsed\_time=cputime-start\_time; display(strcat('Runing\_time=',num2str(elapsed\_time),'s;'))

Firstdifference.m

function dimg = firstdifference(img,direction)

imgPad = padarray(img,[1 1],'symmetric','both');%将原图像的边界扩展

[row,col] = size(imgPad);

dimg = zeros(row,col);

switch (direction)

case 'dx',

dimg(:,1:col-1) = imgPad(:,2:col)-imgPad(:,1:col-1);%x方向差分计算，

case 'dy',

dimg(1:row-1,:) = imgPad(2:row,:)-imgPad(1:row-1,:);

otherwise, disp('Direction is unknown');

end;

dimg = dimg(2:end-1,2:end-1);

end