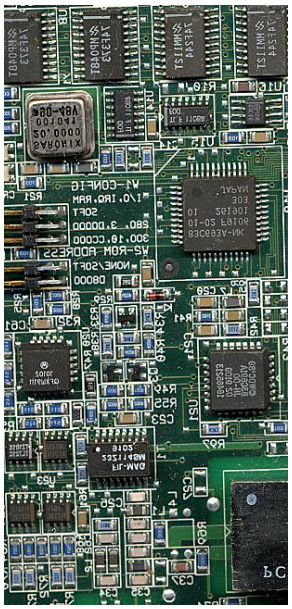
完成自适应局部滤波器以及自适应中值滤波器对图像的处理

一、问题描述

对原始图像分别进行高斯噪声、椒盐噪声添加，分别进行自适应局部滤波器以及自适应中值滤波器降噪处理。



A20.tif

二、解题思路

1.读取图像，将 RGB 图像转化为灰度图像，并作归一化处理。

2.在 matlab 中使用 imnoise 函数给图像添加噪声。分别为图像添加均值为 0, 方差为 0.1 的高斯噪声，和密度为 0.05 的椒盐噪声。

3.为加噪图像设计自适应局部滤波器， 原理：自适应局部滤波器，也叫自适应均值滤波器，作用于局部区域S_{xy}.滤波器在中心化区域中任何点(x,y)上的滤波器响应基于以下 4 个量:

(a) g(x,y)表示噪声图像在点(x,y)上的值;

(b) σ_η^2 干扰 f(x,y)以形成 g(x, y)的噪声方差

(c) m_L在S_{xy}上像素点的局部均值;

(d) σ_L^2 在S_{xy}上像素点的局部方差.

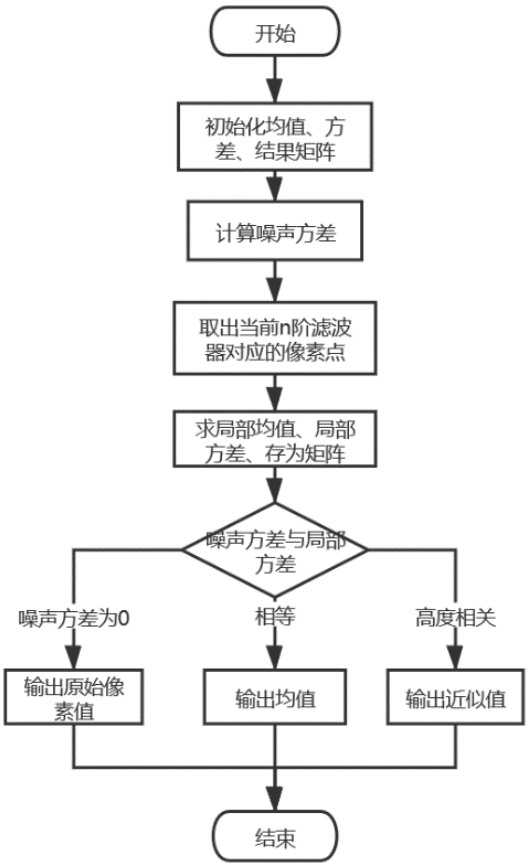
自适应表达式：$\hat{f}(x,y) = g(x,y) - \frac{\sigma_\eta^2}{\sigma_L^2}[g(x,y) - m_L]$

1)如果 $\sigma_\eta^2 = 0$，滤波器应简单地返回 g(x,y)的值。因为 g(x,y)在零噪声情况下等同于 f(x,y)。

2)如果 $\sigma_\eta^2 = \sigma_L^2$, 滤波器返回区域$S_{xy}$上像素的算术均值。这种情况发生在局部面积与全部图像有相同特性的条件下，并且局部噪声简单地用求平均来降低。

3)如果局部方差 σ_η^2 与 σ_L^2 是高度相关的，滤波器返回一个 g(x,y)的近似值。

设计过程如下图所示。



4.为加噪图像设计自适应中值滤波器，原理：

令 S_{xy}：滤波器的作用区域，滤波器窗口所覆盖的区域，该区域中心点为图像中第 y 行第 x 列个像素点;

Z_{min}: S_{xy}中最小的灰度值；

Z_{max}: S_{xy}中最大的灰度值；

Z_{med}: S_{xy}中所有灰度值的中值；

Z_{xy}:表示图像中第 y 行第 x 列个像素点的灰度值

S_{max}: S_{xy}所允许的最大窗口尺寸；

自适应中值滤波器分为 A、B 两个过程：

A：

1.A1 = $Z_{med} - Z_{min}$

2.A2 = $Z_{med} - Z_{max}$

3.如果 A1>0 且 A2<0，则跳转到 B

4.否则，增大窗口的尺寸

5.如果增大后的尺寸≤S_{max}，则重复 A

6.否则，直接输出Z_{med}

B：

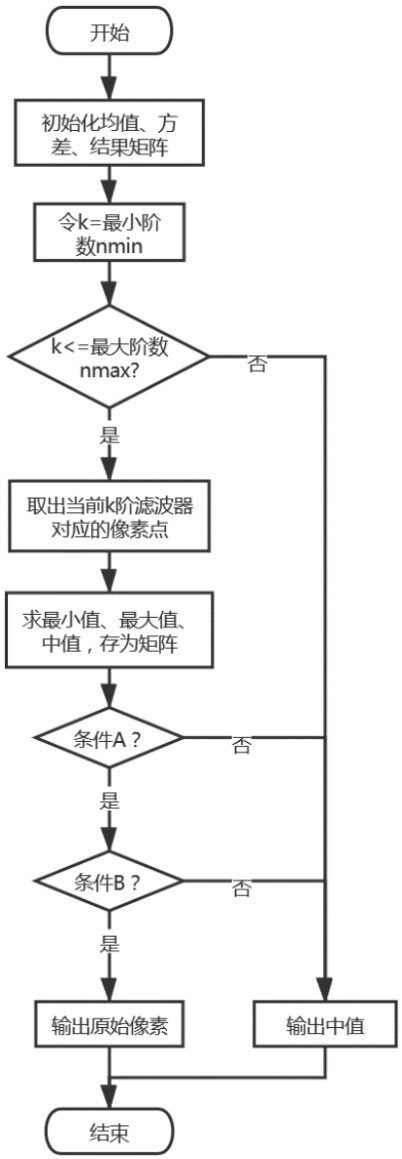
1.B1 = Z_{xy}- Z_{min}

2.B2 = Z_{xy}- Z_{max}

3.如果 B1 >0 且 B2<0，则输出Z_{xy}

4.否则输出Z_{med}

设计过程如下图所示。



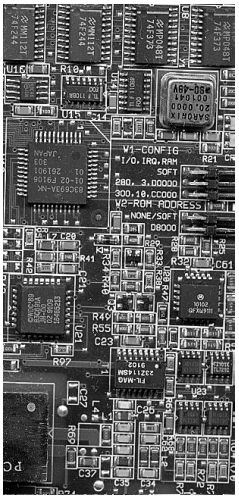
三、程序源代码

```
clc;clear
close all
img=imread('C:\Users\Shoubi007\Desktop\TestA\TestA\A20.tif');
x=rgb2gray(img);
x=im2double(x);
[m,n]=size(x);
%高斯噪声和椒盐噪声
y=imnoise(x,'gaussian',0.1);
z=imnoise(x,'salt & pepper',0.05);
figure(1)
subplot(1,3,1);imshow(x);xlabel('原图像');
subplot(1,3,2);imshow(y);xlabel('加高斯噪声');
subplot(1,3,3);imshow(z);xlabel('加椒盐噪声');

y1= adp_mean(x,y,3);
y2=adp_median(y,20);
z1= adp_mean(x,z,3);
z2=adp_median(z,20);
figure(2)
subplot(1,3,1),imshow(y);xlabel('高斯噪声');
subplot(1,3,2),imshow(y1);xlabel('自适应局部滤波器');
subplot(1,3,3),imshow(y2);xlabel('自适应中值滤波器');
figure(3)
subplot(1,3,1),imshow(z);xlabel('椒盐噪声');
subplot(1,3,2),imshow(z1);xlabel('自适应局部滤波器');
subplot(1,3,3),imshow(z2);xlabel('自适应中值滤波器');

%%自适应局部滤波器
function output = adp_mean(image,imagen,n)
```

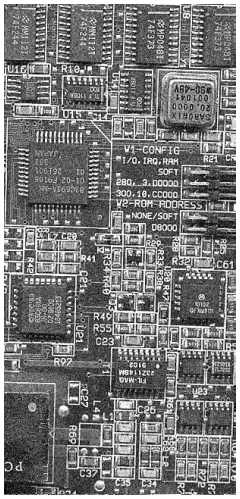
```
[width,height]=size(image);
mu=0;  %均值
imagenn=im2double(imagen);  %imagen 含噪图像
iamgee=im2double(image);
%初始化
imagedd=imagenn;
imagemean=imagenn;
imagevar=imagenn;

sigma=(imagenn-iamgee).^2;
for i=1:width-n+1
        for j=1:height-n+1
            pattern=imagenn(i:i+n-1,j:j+n-1);
            patterns=reshape(pattern,1,length(pattern(:)));
            means=mean(patterns);%求均值
            imagemean(i+(n-1)/2,j+(n-1)/2)=means;
            vars=var(patterns,1);%求方差
            imagevar(i+(n-1)/2,j+(n-1)/2)=vars;
        end
end
%对自适应局部滤波的各项条件作了修改
da=(sigma<1);%噪声方差小于 1 的返回原像素值
dc=~da&(abs(sigma-imagevar)<=100); %噪声方差与局部方差高度相关时，返回一个近似值
db=~dc; %略有调整，剩下的像素位置设置为均值
%da,db,dc 为逻辑值
imagedd(da)=imagenn(da);
imagedd(db)=imagemean(db);
imagedd(dc)=imagenn(dc)-(sigma(dc)./imagevar(dc).*(imagenn(dc)-imagemean(dc)));
output=imagedd;
```

```
end

%%自适应中值滤波器
function II=adp_median(image,Smax)
II=image;
II(:)=0;
alreadyProcessed=false(size(image));
for k=3:2:Smax
zmin=ordfilt2(image,1,ones(k,k),'symmetric');
zmax=ordfilt2(image,k*k,ones(k,k),'symmetric');
zmed=medfilt2(image,[k k],'symmetric');
processUsingLevelB=(zmed>zmin)&(zmax>zmed)&(~alreadyProcessed);%需要转到 B 步骤的像素
zB=(image>zmin)&(zmax>image);
outputZxy=processUsingLevelB&zB;%满足步骤 A，B 的输出原值  对应的像素位置

outputZmed=processUsingLevelB&~zB;%满足 A,不满足 B 的输出中值  对应的像素位置
II(outputZxy)=image(outputZxy);
II(outputZmed)=zmed(outputZmed);
alreadyProcessed=alreadyProcessed|processUsingLevelB;%处理过的像素
if all(alreadyProcessed(:))
    break;
end
end
II(~alreadyProcessed)=image(~alreadyProcessed);%超过窗口大小没被处理的像素位置  输出原值
```

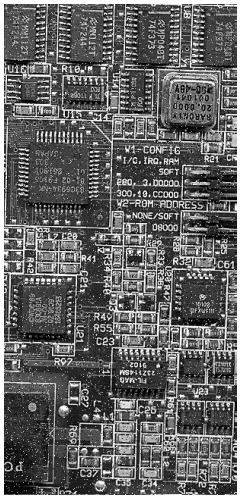
四、运行结果及说明



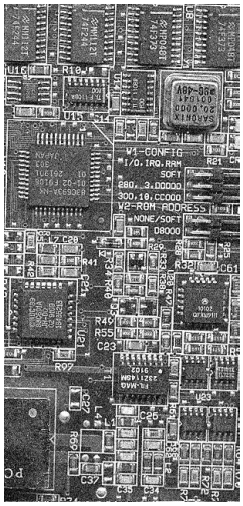
原图像



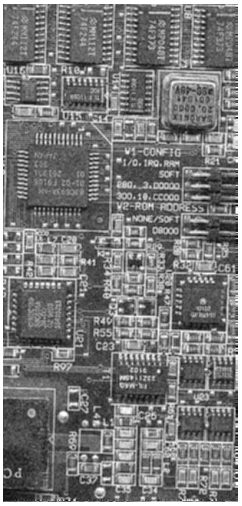
加高斯噪声



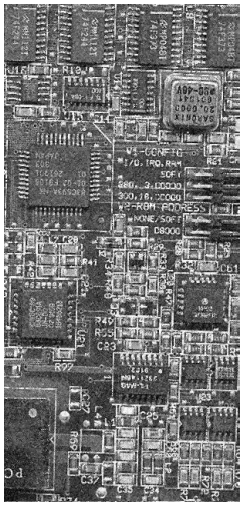
加椒盐噪声



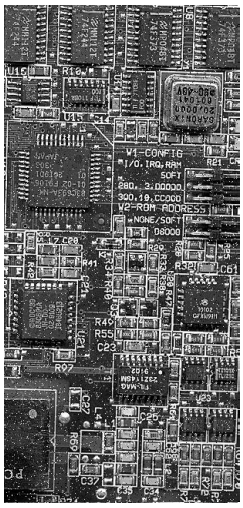
高斯噪声



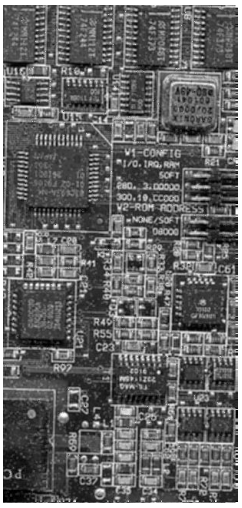
自适应局部滤波器



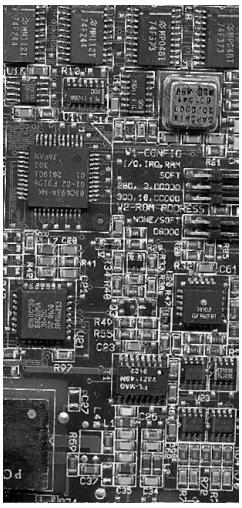
自适应中值滤波器



椒盐噪声



自适应局部滤波器



自适应中值滤波器

根据运行结果对比可知，用自适应局部滤波器处理高斯噪声，自适应中值滤波器处理椒盐噪声，效果更显著。对于椒盐噪声，只影响了图片的部分像素点而不是全部像素点，使用中值滤波方法正好排除了被噪声影响的像素点（被椒盐噪声影响的像素点表现为灰度值 255 的盐噪点和灰度值 0 的胡椒噪点），而局部滤波方法采用了包含噪声像素点在内的所有像素点的平均值，效果自然不如中值滤波器。对于高斯噪声，高斯噪声影响了图片的全部像素点，如果只用像素中的中间值来替代所有像素点，会损失掉其他同样受噪声影响的像素点信息，效果不如采用所有像素点平均值的自适应局部滤波器。

实现对失真图像的逆变换滤波、维纳滤波以及约束最小二乘方滤波

一、问题描述

完成对原始图像运动模糊、噪声+运动模糊失真处理，分别进行逆滤波、维纳滤波、约束最小二乘方滤波，注意约束最小二乘方滤波的实现方法。

A28.jpg

二、解题思路

1.读取图像，将 RGB 图像作归一化处理

2.图像模糊化。构建位移为 10，角度为 20 的运动模糊的点扩散函数 PSF，使用 imfilter 函数对图像添加线性运动模糊，得到运动模糊图像 Blurred.

3.模糊图像加噪。使用 imnoise 函数在模糊图像上添加均值为 0，方差为 0.002 的高斯噪声，得到噪声+运动模糊失真处理图像 BlurredNoisy.

4.直接逆滤波，就是用退化函数除退化图像的傅里叶变换，得到退化前图像的傅里叶变换的估计，$\hat{F}(u,v)$为 F(u,v)的估计，则$\hat{F}(u,v)=\frac{G(u,v)}{H(u,v)}$，同时$G(u,v)=H(u,v)F(u,v)+N(u,v)$，其中 N(u,v)为噪声的傅里叶变换，得$\hat{F}(u,v)=F(u,v)+\frac{N(u,v)}{H(u,v)}$. 根据上式运用 fft2 函数对模糊图像，模糊加噪图像，以及点函数图像进行傅里叶变化，矩阵运算得图像的逆滤波处理。

5.用真实的 PSF 函数采用维纳滤波方法复原图像。维纳滤波表达式为 $F'(u,v)=\frac{H^*(u,v)D(u,v)}{|H(u,v)|^2+NSR}$，其中$F'(u,v)$为复原图像频谱，$D(u,v)$为模糊图像频谱，

$H(u,v)$为传输函数，NSR 为信噪比的倒数。可在 matlab 中使用 deconvwnr 函数，根据真 PSF 函数分别对运动模糊图像 Blurred 和噪声+运动模糊失真图像 BlurredNoisy 进行维纳滤波处理。其中 BlurredNoisy 的 NSR 过大，进行维纳滤波处理时，将 NSR 缩小 600 倍。

6.用真实的 PSF 函数和噪声强度作为参数进行约束最小二乘法图像复原。约束最小二乘复原必须对原始图像、PSF 和噪声特性由先验知识。假设Q为f的线性算子，$\hat{f}$为f的最优估计而使下列函数为最小：$\left\|Q\hat{f}\right\|^2+\alpha\left(\left\|g-H\hat{f}\right\|^2-\|n\|^2\right)$

$\hat{f}$可表示为：$\hat{f}=(H^TH+\gamma Q^TQ)^{-1}H^Tg$ 其中$\gamma=a^{-1}$。a为拉格朗日算子。约束最小二乘法复原式反复迭代常数a直至$\left\|g-H\hat{f}\right\|^2=\|n\|^2$。

在 matlab 中使用 deconvreg 函数，根据真实的 PSF 函数分别对运动模糊图像 Blurred 和噪声+运动模糊失真图像 BlurredNoisy 进行最小二乘法图像复原处理。

三、程序源代码

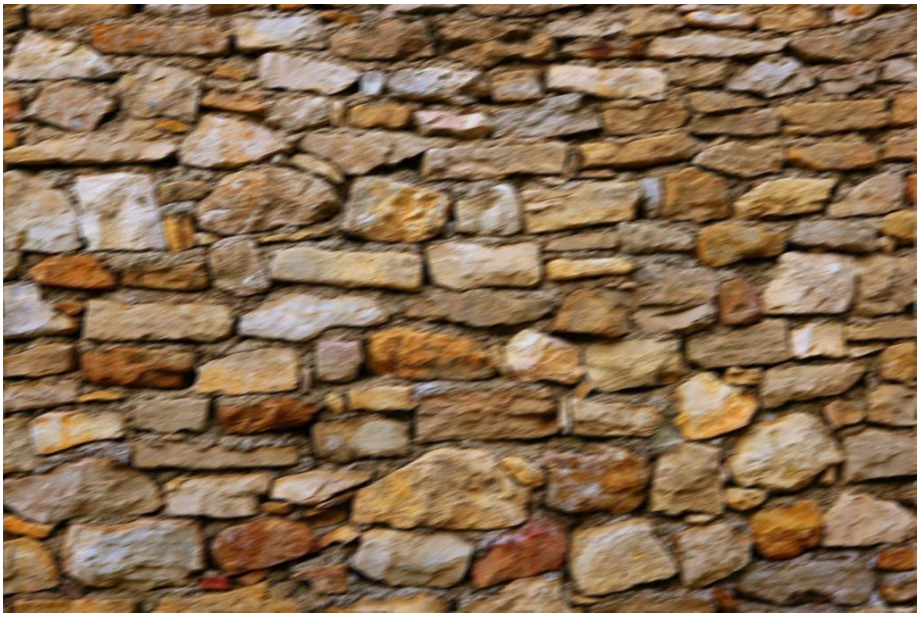
```
clc;clear
close all;
I = imread('C:\Users\Shoubi007\Desktop\TestA\TestA\A28.jpg');
I=im2double(I);
[m,n,~] = size(I);
%图像模糊化
LEN = 10;
THETA = 20;
PSF = fspecial('motion',LEN,THETA);
Blurred = imfilter(I,PSF,'circular','conv');
%模糊化图像加噪
V = .002;
BlurredNoisy = imnoise(Blurred,'gaussian',0,V);
figure,imshow(I);xlabel('原图像');
figure,imshow(Blurred);xlabel('运动模糊处理');
figure,imshow(BlurredNoisy);xlabel('噪声+运动模糊失真处理');
```

```
%用真实的 PSF 函数采用维纳滤波方法复原图像
wnr = deconvwnr(Blurred,PSF);
NSR = sum((V*prod(size(I))).^2) / sum(im2double(I(:)).^2);%信噪比的倒数
wnr1 = deconvwnr(BlurredNoisy,PSF,NSR/600);
%用真实的 PSF 函数和噪声强度作为参数进行约束最小二乘复原
NP = V*prod(size(I)); % noise power
Edged = edgetaper(Blurred,PSF);
Edged1 = edgetaper(BlurredNoisy,PSF);
[~,LAGRA] = deconvreg(Blurred,PSF);
[~,LAGRA1] = deconvreg(BlurredNoisy,PSF,NP);
reg = deconvreg(Edged,PSF,[],LAGRA);
reg1 = deconvreg(Edged1,PSF,[],LAGRA1);
%逆滤波
If= fft2(Blurred);
Pf = fft2(PSF,m,n);
deblurred = ifft2(If./Pf);
If = fft2(BlurredNoisy);
Pf = fft2(PSF,m,n);
Noisy = BlurredNoisy - Blurred;
Nf = fft2(Noisy);
deblurred1= ifft2(If./Pf - Nf./Pf);

figure;
subplot(2,2,1),imshow(Blurred);xlabel('运动模糊处理');
subplot(2,2,2),imshow(deblurred);xlabel('逆滤波处理');
subplot(2,2,3),imshow(wnr);xlabel('维纳滤波处理');
subplot(2,2,4),imshow(reg);xlabel('约束最小二乘法');
figure;
subplot(2,2,1),imshow(BlurredNoisy);xlabel('噪声+运动模糊失真处理');
subplot(2,2,2),imshow(deblurred1);xlabel('逆滤波处理');
```

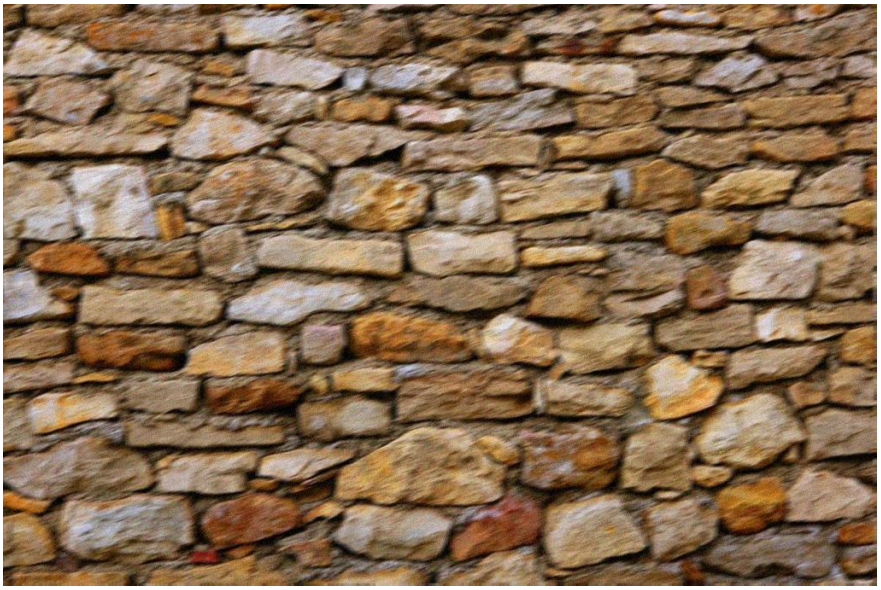
```
subplot(2,2,3),imshow(wnr1);xlabel('维纳滤波处理');
subplot(2,2,4),imshow(reg1);xlabel('约束最小二乘法');
```

四、运行结果及说明

原图像



运动模糊处理



噪声+运动模糊失真处理

运动模糊处理

逆滤波处理

维纳滤波处理

约束最小二乘法

噪声+运动模糊失真处理

逆滤波处理

维纳滤波处理

约束最小二乘法

对于本题中的运动模糊图像，维纳滤波或约束最小二乘的效果更好，而对于噪声+运动模糊失真图像，已知噪声的逆滤波效果更好。