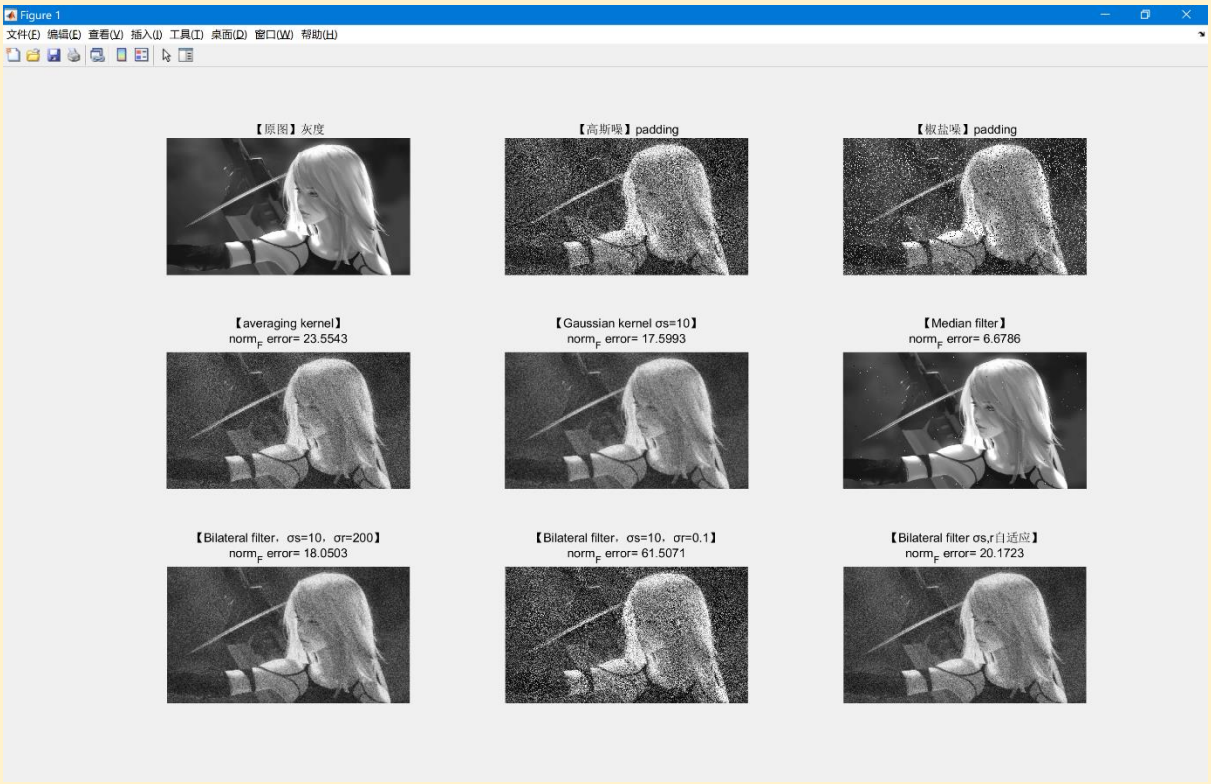
【实验目的】

用 averaging kernel, Gaussian kernel, Bilateral filter, and Midean filter 四种方法对含有高斯噪声和椒盐噪声的图像进行去噪。

【运行结果】



实验中我用到的误差统计方法是，PSNR 也统计过，但是可视化不明显，所以用范数进行统计。

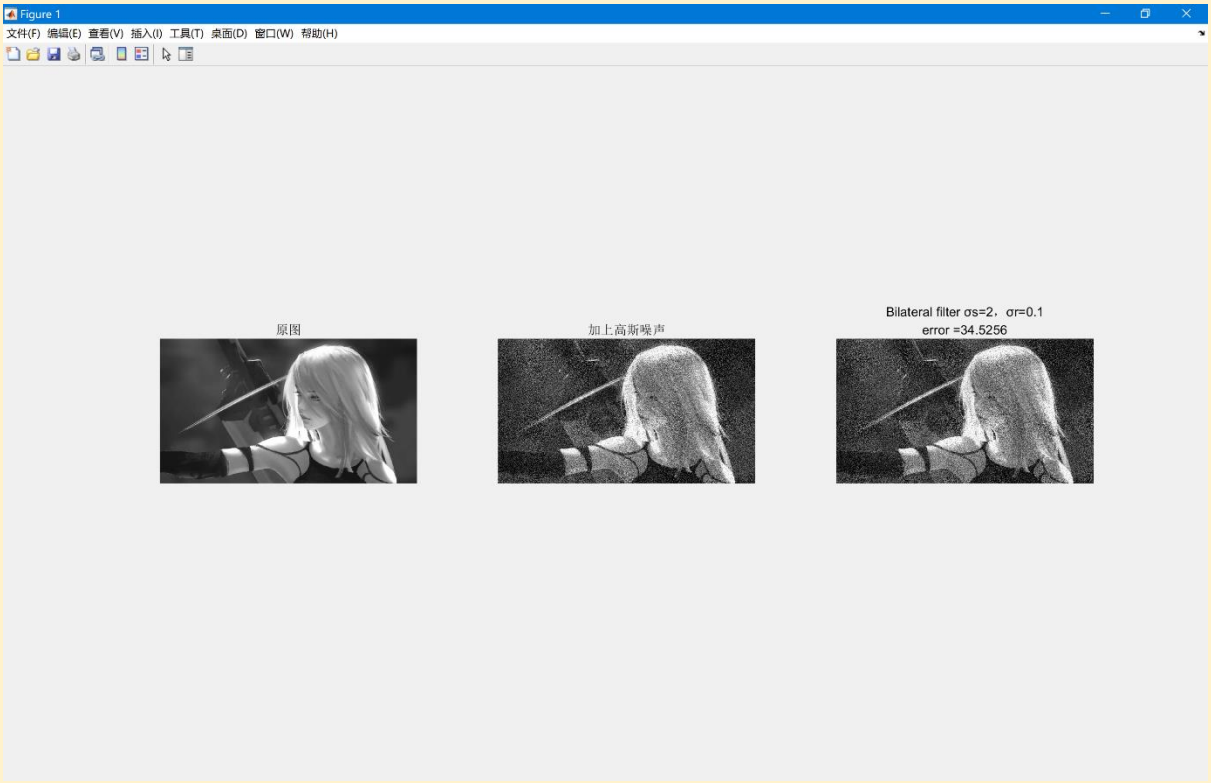
$$error = 100 \times \frac{||I' - I||_F}{||I||_F}$$

在这里统计一下数据（效果最佳从上至下）：

【Method】	【Error】
Median filter	6.6786
Gaussian kernel σs=10	17.5993
Bilateral filter σs=10、σr=200	18.0503
Bilateral filter σs,r 自适应	20.1723
Averaging kernel	23.5543
Bilateral filter σs=10、σr=0.1	61.5071

【分析一波】

1. 中值滤波对椒盐噪声的原图像处理要明显好于其他几个方法，猜测原图像保留的像素点较多，再一个在中值选择像素点的过程中，留下了更多原图像像素点，所以效果非常好。
2. 对于第二中的高斯核、第三与最后的双边滤波，采取的σs 都是 10，对于 Gaussian kernel，可以看作是σr=∞，故随着σr 的不断减小，误差值是越来越大的，为了确保我说的并不带有主观性，我找到了 CSDN 上其他人写的代码，并加以实现，得到以下结果：



这里显示的σr 依旧是一个很大的值，并且直观效果也并不好。

3. 对于第四个双边滤波，σs 与σr 自适应，采取的方法是：每次核化都用当前卷积核对应图像区块的方差作为σs 与σr 的取值，所以得到了一个数值与直观上都不错的结果。
4. 在效果比较好的这些方法中，Averaging kernel 效果也不错误差值也是在 20 左右，只是从精确度上来说，就没有前几者的高了。

【疑问】

对于 Bilateral kernel 来说，若按照课件上给的σs 与σr 给一个特定值，σs=16，σr=0.1，实验结果运行并不是很好，反而给σr 一个十分大的值，即几乎可以去掉 Bilateral kernel 的权值的后半项（I'-I 项），是高斯核的近似，那么我们得到一个比较好的效果图，所以σs 与σr 到底应该怎么取值？本实验中采取二者自适应的取法效果比较好。

【源代码】

```
clear
clc

f=imread('riven.jpg');
g=rgb2gray(f);
g1=imnoise(g,'gaussian',0,0.1);
g2=imnoise(g,'salt & pepper',0.2);
subplot(331);imshow(g),title('【原图】灰度');
[m,n]=size(g);
normg=norm(double(g),'fro');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X2=zeros(m+2,n+2);
for i=1:m
    for j=1:n
        X2(i+1,j+1)=g1(i,j);
    end
end
subplot(332);imshow(uint8(X2)),title('【高斯噪】padding');
X3=zeros(m+4,n+4);
for i=1:m
    for j=1:n
        X3(i+2,j+2)=g1(i,j);
    end
end%用了两种大小的高斯噪声 padding
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X4=zeros(m+4,n+4);
for i=1:m
    for j=1:n
        X4(i+2,j+2)=g2(i,j);
    end
end
X=zeros(m+2,n+2);
for i=1:m
    for j=1:n
        X(i+1,j+1)=g2(i,j);
    end
end%用了两种大小的椒盐噪声 padding
subplot(333);imshow(uint8(X)),title('【椒盐噪】padding');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
averaging=[1/9,1/9,1/9;1/9,1/9,1/9;1/9,1/9,1/9];%averaging filter，1/9 各元素
```

```
Y=zeros(m,n);
for i=1:m
    for j=1:n
        sum=0;
        for k=1:3
            for l=1:3
                sum=sum+averaging(k,l)*X2(i+k-1,j+l-1);%X2 是（m+2，n+2）大小
            end
        end
        Y(i,j)=sum;
    end
end
error=100*norm(Y-double(g),'fro')/normg;
subplot(334);imshow(uint8(Y)),title({' 【averaging kernel】', ['norm_F error= ', num2str(error)] })
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Gaussian=zeros(5,5);%σ 取 2，高斯核
sumGaussian=0;%系数的和
for i=1:5
    for j=1:5
        Gaussian(i,j)=(1/sqrt(2*pi)*exp(-(((i-3)^2+(j-3)^2)/2)));
    end
end
for i=1:5
    for j=1:5
        sumGaussian=sumGaussian+Gaussian(i,j);
    end
end
Gaussian=Gaussian/sumGaussian;%给高斯核的系数做一个归一化处理

Z=zeros(m,n);
for i=1:m
    for j=1:n
        sum=0;
        for k=1:5
            for l=1:5
                sum=sum+Gaussian(k,l)*X3(i+k-1,j+l-1);
            end
        end
        Z(i,j)=sum;
    end
end
error=100*norm(Z-double(g),'fro')/normg;
subplot(335);imshow(uint8(Z)),title({' 【Gaussian kernel】', ['norm_F error= ', num2str(error)] })
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
B1=zeros(5,5);%这是双边滤波系数的第一项
s1=zeros(5,5);%开始求一个距离权值的方差
for i=1:5
        for j=1:5
                s1(i,j)=sqrt((i-2)^2+(j-2)^2);
        end
end
s=(std2(s1))^2;
for i=1:5
        for j=1:5
                B1(i,j)=exp(-((i-3)^2+(j-3)^2)/(2*s^2));
        end
end
Z=zeros(m,n);
for i=1:m
        for j=1:n
                sum=0;%一次卷积的加权平均
                sumB=0;%给系数做归一化处理
                var1=zeros(5,5);
                var1(1:5,1:5)=X3(i-1+1:i-1+5,j-1+1:j-1+5)-X3(i-1+3,j-1+3);
                r=(std2(var1))^2;
                for k=1:5
                        for l=1:5
                                sumB=sumB+B1(k,l)*exp(-(((X3(i-1+k,j-1+l)-X3(i-1+3,j-1+3))^2)/(2*(r^2))));
                        end
                end

                for k=1:5
                        for l=1:5
                                sum=sum+(B1(k,l)/sumB)*X3(i+k-1,j+l-1)*exp(-(((X3(i-1+k,j-1+l)-X3(i-1+3,j-
1+3))^2)/(2*(r^2))));
                        end
                end
                Z(i,j)=sum;
        end
end
error=100*norm(Z-double(g),'fro')/normg;
subplot(339);imshow(uint8(Z)),title({'【Bilateral filter σs,r 自适应】', ['norm_F error= ', num2str(error)] })
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B1=zeros(5,5);%这是双边滤波系数的第一项
s=10;r=200;
for i=1:5
        for j=1:5
                B1(i,j)=exp(-((i-3)^2+(j-3)^2)/(2*s^2));
```

```
    end
end%没有问题的值
Z=zeros(m,n);
for i=1:m
    for j=1:n
        sum=0;%卷积运算的和
        sumB=0;%权值求和，归一化处理
        for k=1:5
            for l=1:5
                sumB=sumB+B1(k,l)*exp(-(((X3(i-1+k,j-1+l)-X3(i-1+3,j-1+3))^2)/(2*(r^2))));
            end
        end

        for k=1:5
            for l=1:5
                sum=sum+(B1(k,l)/sumB)*X3(i+k-1,j+l-1)*exp(-(((X3(i-1+k,j-1+l)-X3(i-1+3,j-
1+3))^2)/(2*(r^2))));
            end
        end
        Z(i,j)=sum;
    end
end
error=100*norm(Z-double(g),'fro')/normg;
subplot(337);imshow(uint8(Z)),title({' 【Bilateral filter， σs=10， σr=200】', ['norm_F error= ',
num2str(error)] })
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B1=zeros(5,5);%这是双边滤波系数的第一项
s=16;r=0.1;
for i=1:5
    for j=1:5
        B1(i,j)=exp(-((i-3)^2+(j-3)^2)/(2*s^2));
    end
end%没有问题的值
Z=zeros(m,n);
for i=1:m
    for j=1:n
        sum=0;%卷积运算的和
        sumB=0;%权值求和，归一化处理
        for k=1:5
            for l=1:5
                sumB=sumB+B1(k,l)*exp(-(((X3(i-1+k,j-1+l)-X3(i-1+3,j-1+3))^2)/(2*(r^2))));
            end
        end
```

```
            for k=1:5
                for l=1:5
                    sum=sum+(B1(k,l)/sumB)*X3(i+k-1,j+l-1)*exp(-(((X3(i-1+k,j-1+l)-X3(i-1+3,j-
1+3))^2)/(2*(r^2))));
                end
            end
            Z(i,j)=sum;
        end
end
error=100*norm(Z-double(g),'fro')/normg;
subplot(338);imshow(uint8(Z)),title({' 【Bilateral filter， σs=16， σr=0.1】', ['norm_F error= ',
num2str(error)] })
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Z=zeros(m,n);
for i=1:m
        for j=1:n
            kernel=zeros(3,3);
            for k=1:3
                for l=1:3
                    kernel(k,l)=X(i-1+k,j-1+l);
                end
            end
            kernelmedian=median(kernel,'all');
            Z(i,j)=kernelmedian;
        end
end
error=100*norm(Z-double(g),'fro')/normg;
subplot(336);imshow(uint8(Z)),title({' 【Median filter】', ['norm_F error= ', num2str(error)] })
```

【CSDN 代码】

```
close all;
clc;

img=imread('riven.jpg');
img=rgb2gray(img);
[m,n]=size(img);
subplot(131);imshow(img);title('原图');
img1=imnoise(g,'Gaussian',0,0.0246);
subplot(132);imshow(img1);title('加上高斯噪声');
r=10;%模板半径
imgn=zeros(m+2*r+1,n+2*r+1);
imgn(r+1:m+r,r+1:n+r)=img1;
imgn(1:r,r+1:n+r)=img1(1:r,1:n);%扩展上边界
imgn(1:m+r,n+r+1:n+2*r+1)=imgn(1:m+r,n:n+r);%扩展右边界
imgn(m+r+1:m+2*r+1,r+1:n+2*r+1)=imgn(m:m+r,r+1:n+2*r+1);%扩展下边界
imgn(1:m+2*r+1,1:r)=imgn(1:m+2*r+1,r+1:2*r);%扩展左边界

sigma_d=2;
sigma_r=0.1;
[x,y] = meshgrid(-r:r,-r:r);
w1=exp(-(x.^2+y.^2)/(2*sigma_d^2));%以距离作为自变量高斯滤波器

%h=waitbar(0,'wait...');%初始化 waitbar
for i=r+1:m+r
    for j=r+1:n+r
        w2=exp(-(imgn(i-r:i+r,j-r:j+r)-imgn(i,j)).^2/(2*sigma_r^2));
        %以周围和当前像素灰度差值作为自变量的高斯滤波器
        w=w1.*w2;
        s=imgn(i-r:i+r,j-r:j+r).*w;
        sumw=0;
        for k=1:2*r+1
            for l=1:2*r+1
                sumw=sumw+w(k,l);
            end
        end
        sums=0;
        for k=1:2*r+1
            for l=1:2*r+1
                sums=sums+s(k,l);
            end
```

```
            end
            imgn(i,j)=sums/sumw;
        end
        %waitbar(i/m);
end
%close(h)
img2=imgn(r+1:m+r,r+1:n+r);
%figure;
subplot(133);imshow(uint8(img2));title('网上代码得到的双边滤波结果');
```