%% Noises Types

clc;

close all;

clear;

X=imread('yjsp.jpg');

I=rgb2gray(X); %I is an Gray image

[numRows,numCols,Layers]=size(I);

figure(1);

imshow(I);

imwrite(I,'noise result\original picture.png');

title('original picture');

%% Impulse Noise

%脉冲噪声：信号因短时间内具有非常大的负值或正值的峰值而失真，并且可能出现解码错误等情况

%会出现白色或黑色的点，被称为噪点。

p=0.1;%p1 is the probability of noise

I\_impulse=imnoise(I,'salt & pepper',p);

figure(2);

imshow(I\_impulse);

imwrite(I\_impulse,'noise result\Impulse Noise picture.png');

title('Impulse Noise Picture');

%% Additive Noise

%加性噪声：噪声影响产生于三个方面，分别是"人为噪声"、"自然噪声"、"内部噪声"

%而这些噪声基本符合某种分布，并且对原图的影响是直接相加，因此称为加性噪声

%我们这里以符合高斯分布的噪声为例

a=0;b=200;% a is mean of noise,b is variance

Gaussian\_noise=a+sqrt(b)\*randn(numRows,numCols);%create noise of Gaussian distribution

for m=1:1:numRows

for n=1:1:numCols

I\_additive(m,n)=I(m,n)+Gaussian\_noise(m,n);%add the original pixel with noise

end

end

figure(3);

imshow(I\_additive);

imwrite(I\_additive,'noise result\Additive Noise picture.png');

title('Additive Noise Picture');

%% Multiplicative Noise

%乘性噪声：噪声产生与加性噪声类似，只是对原图影响为相乘

I\_multiply=imnoise(I,'speckle');

figure(4);

imshow(I\_multiply);

imwrite(I\_multiply,'noise result\Multiplicative Noise picture.png');

title('Multiplicative Noise Picture');

%% Gaussian (Normal) Noise

%高斯（正态）噪声：正态分布描述略，原理是向原图中添加高斯分布的白噪声

I\_Gaussian=imnoise(I,'gaussian',0.001);

figure(5);

imshow(I\_Gaussian);

imwrite(I\_Gaussian,'noise result\Gaussian (Normal) Noise picture.png');

title('Gaussian (Normal) Noise Picture');

%% Quantization Noise

% 量化噪声：导致物体周围出现虚假轮廓或删除图像中对比度低的细节。

% 量化噪声可以近似地用泊松分布来描述

I\_Quantization=imnoise(I,'poisson');

figure(6);

imshow(I\_Quantization);

imwrite(I\_Quantization,'noise result\Quantization Noise picture.png');

title('Quantization Noise Picture');

%% Images Filtering

%图像滤波:在尽量保留图像细节特征的条件下对目标图像的噪声进行抑制

clc;

close all;

clear;

X=imread('yjsp.jpg');

I=rgb2gray(X); %I is an Gray image

[numRows,numCols]=size(I);

I\_Impulse=imread('noise result\Impulse Noise picture.png');%Impulse noise picture as example

I\_Additive=imread('noise result\Additive Noise picture.png');%Additive noise picture as example

I\_Multiplicative=imread('noise result\Multiplicative Noise picture.png');%Multiplicative noise picture as example

I\_Normal=imread('noise result\Gaussian (Normal) Noise picture.png');%Gaussian (Normal) noise picture as example

I\_Quantization=imread('noise result\Quantization Noise picture.png');%Quantization noise picture as example

%run this after running p3.m

I\_imp\_salt=imread('Nonlinear filtering\Adaptive Median Filtering picture.png');

%% Low-pass Filtering

%低通滤波：衰减高通分量(强度变化较大的区域)，保持图像的低通分量不变。用于降低噪声

%有两个主要特点：

% 1. 掩码系数非负

% 2. 所有系数加和为1

%% Arithmetic Mean Filter

%算数均值滤波器

%滤波器将原像素周围的像素进行均值化，计算结果接近原图像。

%由于在几何学中，二维化的滤波器掩码看起来像平行六面体（长方体），因此得名"box-filter"

mask1=fspecial('average',3);

I\_Arithmetic=uint8(filter2(mask1,I\_Normal));

figure(2);

subplot(1,2,1);

imshow(I\_Normal);

title('before filtering Picture');

subplot(1,2,2);

imshow(I\_Arithmetic);

title('Arithmetic Mean Filter Picture');

imwrite(I\_Arithmetic,'Low-pass Filtering\Arithmetic Mean Filter picture.png');

saveas(2,'Low-pass Filtering\compare Arithmetic Mean Filter picture.png');

%% Geometric Mean Filter

%几何均值滤波器

%与算数均值滤波器类似，更适用于失真较轻的图片，抑制高通加性噪声时有更好的性能。

I\_Geometric=zeros(numRows,numCols);

%本部分使用的图片预处理函数ext()代码部分在最下面

I\_extra1=ext(I\_Normal,2);

for x=1:1:numRows

for y=1:1:numCols

I\_Geometric(x,y)=double(I\_extra1(x,y)\*I\_extra1(x+1,y)\*I\_extra1(x,y+2)...

\*I\_extra1(x+2,y)\*I\_extra1(x,y+1)\*I\_extra1(x+1,y+1)\*I\_extra1(x+2,y+1)...

\*I\_extra1(x+1,y+2)\*I\_extra1(x+2,y+2))^(1/9);

end

end

I\_Geometric=uint8(I\_Geometric);

figure(3);

subplot(1,2,1);

imshow(I\_Normal);

title('before filtering Picture');

subplot(1,2,2);

imshow(I\_Geometric);

title('Geometric Mean Filter Picture');

imwrite(I\_Geometric,'Low-pass Filtering\Geometric Mean Filter picture.png');

saveas(3,'Low-pass Filtering\compare Geometric Mean Filter picture.png');

%% Harmonic Mean Filter

%谐波均值滤波器

I\_extra2=ext(I\_imp\_salt,2);% if you didn't run p3.m,you can try other noise pictures

onematrix=ones(3,3);

for x=1:1:numRows

for y=1:1:numCols

I\_Harmonic(x,y)=9/(sum(sum(onematrix./I\_extra2(x:x+2,y:y+2))));

end

end

I\_Harmonic=uint8(I\_Harmonic);

figure(4);

subplot(1,2,1);

imshow(I\_imp\_salt);

title('before filtering Picture');

subplot(1,2,2);

imshow(I\_Harmonic);

title('Harmonic Mean Filter Picture');

imwrite(I\_Harmonic,'Low-pass Filtering\Harmonic Mean Filter picture.png');

saveas(4,'Low-pass Filtering\compare Harmonic Mean Filter picture.png');

%% Counterharmonic Mean Filter

%反谐波均值滤波器

I\_extra3=ext(I\_Normal,2);

Q=0;

%Q > 0 suppresses noises like «pepper»,

%and if Q < 0 — noises like «salt», however, it is not possible to remove

%white and black points at the same time. If Q = 0 the filter turns into

%an arithmetic one, and if Q = −1 — to harmonic.

for Q=-1:1:1

for x=1:1:numRows

for y=1:1:numCols

dividend(x,y)=(sum(sum(I\_extra3(x:x+2,y:y+2).^(Q+1))));

divisor(x,y)=(sum(sum(I\_extra3(x:x+2,y:y+2).^Q)));

I\_Counterharmonic(x,y,Q+2)=dividend(x,y)/divisor(x,y);

end

end

end

I\_Counterharmonic=uint8(I\_Counterharmonic);

figure(5);

subplot(2,2,1);

imshow(I\_Normal);

title('before filtering Picture');

subplot(2,2,2);

imshow(I\_Counterharmonic(:,:,1));

title('Counterharmonic Mean Filter Picture(Q=-1)');

subplot(2,2,3);

imshow(I\_Counterharmonic(:,:,2));

title('Counterharmonic Mean Filter Picture(Q=0)');

subplot(2,2,4);

imshow(I\_Counterharmonic(:,:,3));

title('Counterharmonic Mean Filter Picture(Q=1)');

imwrite(I\_Counterharmonic(:,:,1),'Low-pass Filtering\Counterharmonic Mean Filter picture(Q=-1).png');

imwrite(I\_Counterharmonic(:,:,2),'Low-pass Filtering\Counterharmonic Mean Filter picture(Q=0).png')

imwrite(I\_Counterharmonic(:,:,3),'Low-pass Filtering\Counterharmonic Mean Filter picture(Q=1).png')

saveas(5,'Low-pass Filtering\compare Counterharmonic Mean Filter picture.png');

%% Gaussian Filter

%高斯滤波器

I\_Gaussian=imgaussfilt(I\_Normal);

figure(6);

subplot(1,2,1);

imshow(I\_Normal);

title('before filtering Picture');

subplot(1,2,2);

imshow(I\_Gaussian);

title('Gaussian Filter Picture');

imwrite(I\_Gaussian,'Low-pass Filtering\Gaussian Filter picture.png')

saveas(6,'Low-pass Filtering\compare Gaussian Filter picture.png');

%% Expand the matrix function

function ext=ext(I\_ori,num)%I\_ori: original picture;num:extra circle numbers

[numRows,numCols]=size(I\_ori);

ext=I\_ori;

for x=1:1:num

[numRows,numCols]=size(ext);

ext=cat(1,ext,ext(numRows,:));

ext=cat(2,ext,ext(:,numCols));

end

ext=double(ext);

for x=1:1:numRows

for y=1:1:numCols

if ext(x,y)==0

ext(x,y)=1;

end

end

end

end

%% Nonlinear Filtering

%非线性滤波

%低通滤波器是线性的，当数字图像中存在正态分布的噪声时，低通滤波器是最优的。

%为了消除脉冲噪声，最好使用非线性滤波器，例如中值滤波器。

clc;

close all;

clear;

X=imread('yjsp.jpg');

I=rgb2gray(X); %I is an Gray image

I\_Impulse=imread('noise result\Impulse Noise picture.png');%Impulse noise picture as example

I\_Additive=imread('noise result\Additive Noise picture.png');%Additive noise picture as example

I\_Multiplicative=imread('noise result\Multiplicative Noise picture.png');%Multiplicative noise picture as example

I\_Normal=imread('noise result\Gaussian (Normal) Noise picture.png');%Gaussian (Normal) noise picture as example

I\_Quantization=imread('noise result\Quantization Noise picture.png');%Quantization noise picture as example

[numRows,numCols]=size(I);

%% Median Filtering

%中值滤波器

I\_ori1=I\_Impulse;

I\_Median=medfilt2(I\_ori1);

figure(2);

subplot(1,2,1);

imshow(I\_ori1);

title('before filtering Picture');

subplot(1,2,2);

imshow(I\_Median);

title('Median Filtering Picture');

imwrite(I\_Median,'Nonlinear filtering\Median Filtering picture.png');

saveas(2,'Nonlinear filtering\compare Median Filtering picture.png');

%% Weighted Median Filtering

%加权中值滤波器

m=2;n=3;

I\_ori2=I\_Impulse;

I\_Weighted=medfilt2(I\_ori2,[m,n]);

figure(3);

subplot(1,2,1);

imshow(I\_ori2);

title('before filtering Picture');

subplot(1,2,2);

imshow(I\_Weighted);

title('Weighted Median Filtering Picture');

imwrite(I\_Weighted,'Nonlinear filtering\Weighted Median Filtering picture.png');

saveas(3,'Nonlinear filtering\compare Weighted Median Filtering picture.png');

%% Adaptive Median Filtering

%自适应中值滤波

S\_ori=2;%initial size of window(from 0)

S\_max=10;

%the maxium size of window could be determined by yourself,which I choose 10.

%the bigger constant you choose,the longer time you run

I\_ori3=I\_Impulse;

I\_extra=ext(I\_ori3,S\_ori);

I\_adaptive=zeros(numRows,numCols);

A1=0;A2=0;

c=0;d=0;e=0;%debug counters

for x=1:1:numRows

for y=1:1:numCols

s=S\_ori;%initialize size of window

while 1

Z=double(I\_extra(x:x+s,y:y+s));

z\_med=median(Z,"all");

z\_max=max(Z,[],"all");

z\_min=min(Z,[],"all");

A1=z\_med-z\_min;

A2=z\_med-z\_max;

s\_max=min([numRows-x,numCols-y,S\_max]);

c=c+1;

if A1>0 && A2<0

B1=Z(1,1)-z\_min;

B2=Z(1,1)-z\_max;

if B1>0 && B2<0

I\_adaptive(x,y)=Z(1,1);

break

else

I\_adaptive(x,y)=z\_med;

d=d+1;

break

end

else

if s<s\_max

s=s+1;

else

I\_adaptive(x,y)=Z(1,1);

e=e+1;

break

end

end

end

end

end

I\_adaptive=uint8(I\_adaptive);

figure(4);

subplot(1,2,1);

imshow(I\_ori3);

title('before filtering Picture');

subplot(1,2,2);

imshow(I\_adaptive);

title('adaptive Filtering Picture');

imwrite(I\_adaptive,'Nonlinear filtering\Adaptive Median Filtering picture.png');

saveas(4,'Nonlinear filtering\compare Adaptive Median Filtering picture.png');

%% Rank Filtering

I\_ori4 = I\_Impulse;

I\_Rank1 = ordfilt2(I\_ori4,1,ones(3,3));

I\_Rank2 = ordfilt2(I\_ori4,5,ones(3,3));

I\_Rank3 = ordfilt2(I\_ori4,9,ones(3,3));

figure(5);

subplot(2,2,1);

imshow(I\_ori4);

title('before filtering Picture');

subplot(2,2,2);

imshow(I\_Rank1);

title(' Rank Filter(filter rank is 0%)');

subplot(2,2,3);

imshow(I\_Rank2);

title(' Rank Filter(filter rank is 50%)');

subplot(2,2,4);

imshow(I\_Rank3);

title(' Rank Filter(filter rank is 100%)');

saveas(5,'Nonlinear filtering\compare Rank Filter result.png')

imwrite(I\_Rank1,'Nonlinear filtering\Rank Filter(filter rank is 0%) picture.png');

imwrite(I\_Rank2,'Nonlinear filtering\Rank Filter(filter rank is 50%) picture.png');

imwrite(I\_Rank3,'Nonlinear filtering\Rank Filter(filter rank is 100%) picture.png');

%% Wiener Filtering

I\_ori5 = I\_Impulse;

m=8;n=8;

I\_Wiener = wiener2(I\_ori5,[m n]);

figure(6);

subplot(1,2,1);

imshow(I\_ori5);

title('before filtering Picture');

subplot(1,2,2);

imshow(I\_Wiener);

title('Wiener Filtering');

imwrite(I\_Wiener,'Nonlinear filtering\Wiener Filtering picture.png');

saveas(6,'Nonlinear filtering\compare Wiener Filtering picture.png');

%% Expand the matrix function

function ext=ext(I\_ori,num)%I\_ori: original picture;num:extra circle numbers

[numRows,numCols]=size(I\_ori);

ext=I\_ori;

for x=1:1:num

[numRows,numCols]=size(ext);

ext=cat(1,ext,ext(numRows,:));

ext=cat(2,ext,ext(:,numCols));

end

ext=double(ext);

for x=1:1:numRows

for y=1:1:numCols

if ext(x,y)==0

ext(x,y)=1;

end

end

end

end

clc

clear

I = imread('yjsp.jpg');

I = rgb2gray(I);

figure

subplot(2,3,1);

imshow(I);

title('Original image');

Iroberts = edge(I,'Roberts');

subplot(2,3,2);

%figure

imshow(~Iroberts);

title('Roberts');

Ipr = edge(I,'Prewitt');

subplot(2,3,3);

%figure

imshow(~Ipr);

title('Prewitt');

Isobel = edge(I,'Sobel');

subplot(2,3,4);

%figure

imshow(~Isobel);

title('Sobel');

Ilog = edge(I,'log');

subplot(2,3,5);

%figure

imshow(~Ilog);

title('L o G');

Iedge = edge(I,'Canny');

subplot(2,3,6);

%figure

imshow(~Iedge);

title('Canny');