اجرای برنامه:

برای اجرای برنامه فایل Main.m را در متلب اجرا کنید و در جواب سوال پرسیده شده 'y' یا 'n' را برای قرار دادن یا ندادن نویز روی ورودی وارد کنید پس از اجرا تصاویر ورودی و پس از اعمال فیلتر نمایش داده می شوند.

فایل ها:

Main.m اسکریپت اصلی برنامه است که فراخوانی می شود.

Stat.m توابع آماری در آن قرار دارند.

Fuzzy.m منتطق فازی در آن قرار دارد.

Filter.m فیلتر تصویر در آن قرار دارد.

تصاویر ورودی باید در مسیر همین فایل ها باشند و یا آدرس دهی شوند.

متغیرها:

متغیرها در ابتدای Main مقدار دهی می شوند.

Name: آدرس یا نام تصویر ورودی است مانند 'I2.PNG' (در کنار برنامه 3 تصویر I1-I3.PNG قرار دارند) می توان از هر تصویر مناسب دیگری نیز استفاده کرد(روش این مقاله روی تصاویر با زمینه سیاه عملکرد بهتری دارد.).

Size : اندازه تبدیل تصویر است.برای اینکه خروجی ها هم اندازه و قابل مقایسه باشند تمام تصاویر به ابعاد size * size تبدیل می شوند.

Sigma_sn: میزان نویز اضافه شده به تصویر را نشان می دهد.

W: اندازه پنجره ها در سیستم فازی است.

H_k: اندازه پنجره در فیلتر homogeneous می باشد.

D_k: اندازه پنجره در فیلتر detail می باشد.

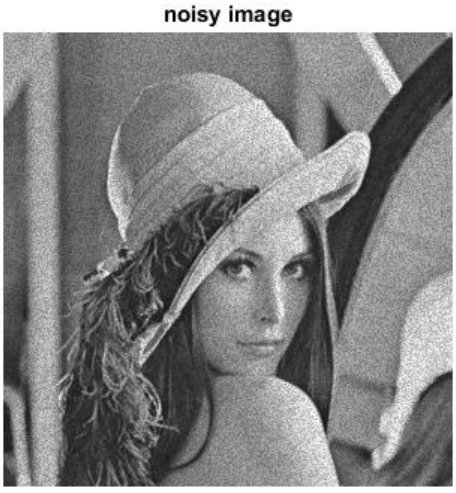
E_k: اندازه پنجره در فیلتر edge می باشد.

C: پارامتر 'tuning' می باشد که حساسیت عملکردی فیلتر egde را مشخص می کند.

خروجی های نمونه:

تصویر اولیه بدون نویز است و نویز به آن اضافه می شود.

تصویر ورودی



تصویر با نویز

تصویر پس از اعمال فیلتر

تصویر اولیه خود حامل نویز است و به آن نویز اضافه نمی شود.

تصویر ورودی با نویز

تصویر پس از اعمال فیلتر

تصویر ورودی به همراه نویز

تصویر پس از اعمال فیلتر

برای هر تصویر باید پارامتر های توضیح داده شده در ابتدا تنظیم شوند در غیر این صورت این روش عملکرد خوبی ندارد مخصوصا زمانی که بافت و روشنایی ارتباط کمی دارند (مانند تصویر lenna که در آن w=14 در نظر گرفته شده بود). و یا ترجیحا از تصاویر فراصوت استفاده کنید.

توضیح کد:

```
% image name
name='I1.PNG';
% image size
size=300;
% speckle noise variance
sigma_sn=0.05;
```

نام تصویر مشخص شده و اندازه آن و واریانس نویز مشخص میشود

```
% window size
```

```
w=6;
% filter parameters
h_k=2;
d_k=2;
e_k=4;
C=1;
```

اندازه پنجره حرکت کننده روی تصویر به همراه پارامترهای لازم برای فیلترها تعریف میشوند.

```
% objects definitions
Stat_obj=Stat;
Fuzzy_obj=Fuzzy;
Filter_obj=Filter;
```

توابع محاسبه کننده ضرایب تغییرات، تابع عضویت و فیلترهای لازم در مرحله حذف نویز تعریف میشوند.

```
% image resize and grayscale conversion
Io=imresize(rgb2gray(imread(name)),[size,size]);
% request to add noise or not
nm=input(['add noise to image? ','y',' or ','n',' : ']);
if(nm=='y')
    Ii=imnoise(Io,'speckle',sigma_sn);
else
    Ii=Io;
end
```

تصویر ورودی تغییر اندازه و به خاکستری تبدیل میشود. بنا به درخواست کاربر نویز به آن اضافه میشود.

```
% log transform on image
I=log(double(Ii));
```

لگاریتم تصویر محاسبه میشوند. این کار نویز ضرب شونده را به جمع شونده تبدیل میکند.

```
% coefficient of variation of image calculation
[ cv,v ]=Stat_obj.CV( I,size,w );
```

ضرایب تغییرات تصویر مرحله قبل محاسبه میشود و ماکزیمم آن برابر پارامتر a قرار میگیرد.

```
[ gv,~ ]=Stat_obj.CV( gradient(I),size,w );
% max value of above CV
c=max(gv);
```

مرحله قبل برای تصویر گرادیان تکرار میشود و پارامتر c محاسبه میشود.

```
% b value
b=(a+c)/2;
```

مقدار b که میانگین دو مقدار قبلی است محاسبه میشود.

```
% a,b,c update for specific condition
if( a<1 && b<1 && c<1 )
    mmv=max(max(I));
    nmv=min(min(I));
    a=(a*(mmv-nmv))+nmv;
    b=(b*(mmv-nmv))+nmv;
    c=(c*(mmv-nmv))+nmv;
end
```

اگر مقادیر قبلی همگی کمتر از 1 باشند با توجه به ماکزیمم و مینیمم تصویر به روز میشوند. به عبارتی تغییر مقیاس داده شده اند.

```
% fuzzy membership function calculation
mu=Fuzzy_obj.MF( I,w,size,v,a,b,c );
```

توابع عضویت محاسبه میشوند

```
% fuzzy classification result
cl=Fuzzy_obj.FCA( mu,size );
```

کلاس پیکسلها بر اساس توابع عضویت تعیین میشوند.

```
% filter calculation for denoising stage
hfi=Filter_obj.HF( I,size,h_k );
dfi=Filter_obj.DF( I,size,d_k );
efi=Filter_obj.EF( I,size,e_k,C );
```

فیلترهای لازم هر کلاس تعریف میشوند. (سه کلاس یکپارچه، جزییات و لبه)

```
% filter normalization
fi=hfi./max(max(hfi));
dfi=dfi./max(max(dfi));
efi=efi./max(max(efi));
```

فیلترها نرمالیزه میشوند.

```
% denoising according to pixel class and appropriate filter
ofi=[];
for i=1:size
    for j=1:size
        if (cl(i,j)==1)
            ofi(i,j)=efi(i,j);
        elseif (cl(i,j)==2)
            ofi(i,j)=dfi(i,j);
        else
            ofi(i,j)=hfi(i,j);
        end
    end
end
```

مرحله حذف نویز با کمک فیلترها و بر اساس کلاس هر پیکسل انجام میشود.

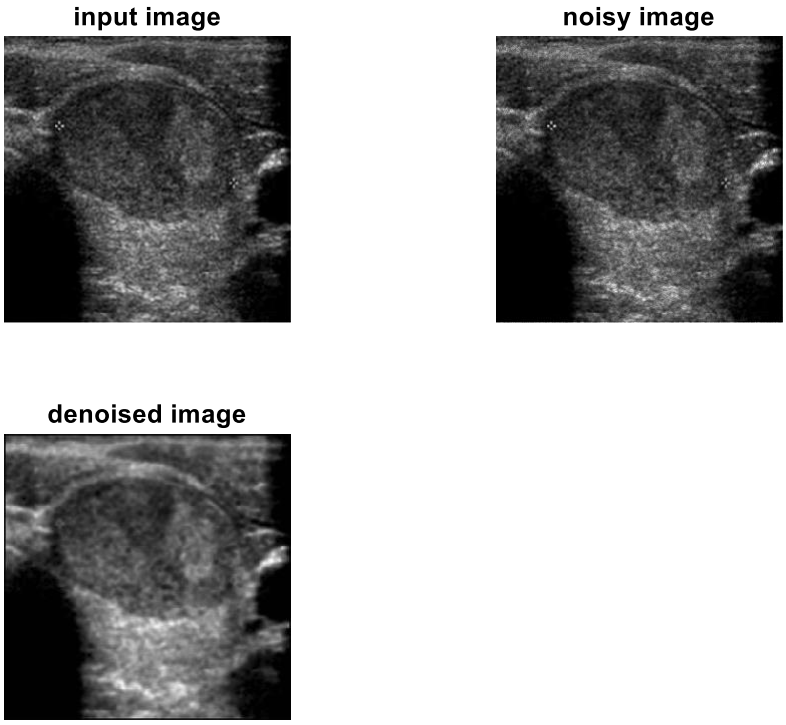
```
% showing results
imshow(Io);
title('input image image');
if(nm=='y')
```

```
    figure();
    I=exp(I);
    Is=I./max(max(I));
    imshow(Is);
    title('noisy image');
end
figure();
ofi=exp(ofi);
ofi=ofi./max(max(ofi));
imshow(ofi);
title('denoised image');
figure;
subplot(221);
imshow(Io);
title('input image');
subplot(222);
imshow(Ii);
title('noisy image');
subplot(223);
imshow(ofi);
title('denoised image');
% ssim calculation
[ ssimval, ssimmap ]=ssim(ofi,double(Io));
disp(['SSIM value is: ',num2str(ssimval)]);
```

نتایج نمایش داده میشود. و فاکتور شباهت SSIM بین تصویر حذف نویز شده و تصویر بدون نویز اصلی محاسبه میشود.

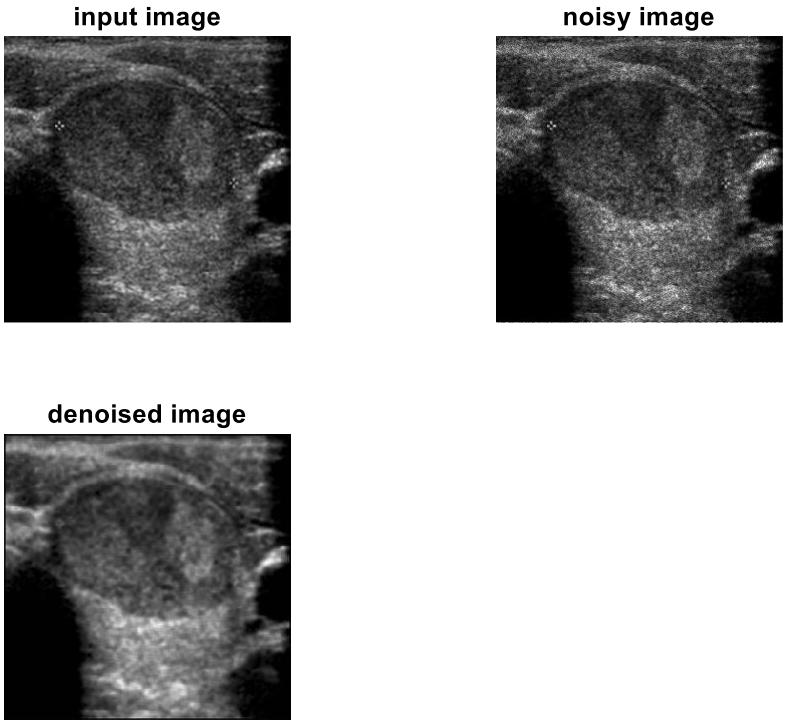
نتایج:

برای واریانس نویز 0.01 داریم



مقدار SSIM=0.064153 بدست آمده است.

برای واریانس نویز 0.05 داریم



مقدار SSIM=0.064349 است که نسبت به قبلی بیشتر است که منطقی است. چون واریانس نویز افزایش یافته است.