ELEC 6641 Two-dimensional Signal and Image Processing

Project report

Implementation of an Optimized Blockwise Nonlocal Means Denoising Filter for 2-D Magnetic Resonance Images

Name: Xiaowen Ke

Student ID: 27817312

Date: 3 May 2016

1. Abstract

The nonlocal-means filter is a kind of new algorithm which uses the redundancy of the image to remove the noise. The critical disadvantage of the nonlocal-means denoising algorithm is the huge computation burden. The project is aim to optimize the classical nonlocal-means by mainly three ways: 1) an automatic tuning of the smoothing parameter; 2) a selection of the pixels or blocks; 3) a blockwise implementation. Finally the performance of the optimized blockwise nonlocal-means and the comparisons with other denoising methods are test by the BrainWeb database.

Key words: Redundancy, nonlocal-means, Denoising algorithm, Classical nonlocal-means, Blockwise implementation.

2. Introduction

All digital images contain different degree of noise, especially the ultrasound images or magnetic resonance images which always have the barely detectable small structures compared to the noise level. Image denoising algorithms attempt to remove this noise from the image. Ideally, the resulting denoised image will not contain any noise or artifacts. Major denoising methods include Gaussian filtering, Wiener filtering, and wavelet thresholding, restore the intensity value of each image pixel by averaging in some way the intensities of its neighboring pixels simply according to its distance to the pixel to be denoised. Indeed, these data-independent algorithms are very simply and convenient to apply. However, the major drawback is that they always degrade or remove the fine details and edge of the image, which is not acceptable while they are applied to the medical images.

Therefore, the nonlocal-means, which is a kind of data-dependent algorithm, was proposed. Unlike the denoising algorithms above, while calculating the weight of neighboring pixels, the nonlocal-means is not according to the distance but the intensity similarity between the neighbor pixels and the denoising pixels. The idea of this algorithm was came up with the theory that any image has redundancy, and that any pixel of the image has similar pixels that are not necessarily located in a spatial neighborhood. According to this theory, the nonlocal-means can effectively eliminating the blurring effect.

However, because of a more detailed similarity analysis on the nonlocal-means algorithm and a more complete comparison between pixels, the improvement of denoising quality need to be balanced by a dramatic growth of the computation time.
In order to make the nonlocal-means algorithm effective enough to apply in medical image processing area, there are mainly three improvements proposed by this project: 1) an automatic tuning of the smoothing parameter; 2) a selection of the most relevant pixels or blocks for the nonlocal-means computation; 3) a blockwise implementation.

The improvements above can change the classical nonlocal-means filter into a fully automatic according to the noise level of the image and overcome the main drawback of the classical nonlocal-means denoising filter: the huge computation burden.

Finally, in order to test the performance of the optimized nonlocal-means denoising filter, several validations and comparison are set. We add four different levels of noise (3%, 9%, 15% and 21%) to the medical image from the BrainWeb database to test the compare the denoising quality (PSNR value and visual quality) and the computation time of each part of the improvement. Besides, some comparisons with other denoising methods such as Gaussian filtering and Wiener filtering.

3. Method

3.1 Classical Nonlocal-means Filter

In the classical nonlocal-means, each pixel x_i of the non-local means denoised image is computed with the following formula:

$$\mathrm{NL(u)}(x_i) = \sum w(x_i, x_j) u(x_j) \tag{1}$$

Where the $\mathrm{NL(u)}(x_i)$ means the intensity value of the pixel x_i, and each pixel is a weighted average of all the pixels in the search window. The weights are based on the similarity between the neighborhoods of pixels. In order to compute the similarity, a neighborhood must be defined inside the search window first, then the weights can then be computed by calculating the Euclidean distance, using the following formula:

$$\mathrm{w}(x_i, x_j) = \frac{1}{Z_i} e^{-\frac{\left\| u(N_i) - u(N_j) \right\|_2^2}{h^2}} \tag{2}$$

where Z_i is a normalization constant to confirm that the sum of all the weights inside the search window is 1, and h is a smoothing parameter to control the decrease of the exponential function. And $\left\| u(N_i) - u(N_j) \right\|_2^2$ is the classical Euclidean distance between two neighbor blocks $u(N_i)\ and\ u(N_j)$.

3.2 Improvements of the Classical Nonlocal-means Filter

(1) An Automatic Tuning of the Smoothing Parameter h:

The value of the smoothing parameter h is decided by the standard deviation of noise. According to the equation (2), now we want to optimize the parameter h, it also taking the size of the neighbor block $|N_i|$ into consider. Therefore, h is a function of noise level, size of the neighbor blocks and β: $h^2 = f(\sigma^2, |N_i|, \beta)$. After this adjustment, the smoothing parameter h becomes an automatic tuning parameter with the known σ^2 and $|N_i|$. Now the only constant need to be changed is β.

(2) Voxel Selection in the Search Volume:

Since that we need to calculate the weight of all the pixels in the search window. The computational burden will increase dramatically especially with the increasing size of the search window. Besides, it will causes the heavier blurring effect. Fortunately, we found that not every pixels in the search window is related to the pixel under study. In order to avoid useless weight computation, we propose a method to set a range to select the most relevant pixels. The selection part can be expressed as follows:

$$\mathrm{w}(x_i, x_j) = \begin{cases} \frac{1}{Z_i} e^{-\frac{\left\|u(N_i)-u(N_j)\right\|_2^2}{2\beta\sigma^2|N_i|}}, & if\ \mu_1 < \frac{\overline{u(N_i)}}{\overline{u(N_j)}} < \frac{1}{\mu_1}\ and\ \sigma_1^2 < \frac{Var(u(N_i))}{Var(u(N_j))} < \frac{1}{\sigma_1^2} \\ 0, & otherwise \end{cases} \quad (3)$$

Here we need to calculate the mean and the variance of the local neighborhood, respectively.

(3) Blockwise Implementation:

The most important improvement in this project is the blockwise implementation. In pixelwise nonlocal-means, we process one pixel after one movement of the search window. During this time, we need to calculate the weight of all the pixels in the search window, which is main reason why the nonlocal-means has a huge computation burden. Usually, the size of the practical medical images are large, especially refer to the 3D images.

Therefore, the computation time will increase exponentially. In order to speed up the algorithm, the blockwise idea is proposed: 1) dividing the volume into blocks with each other; 2) performing nonlocal-means-like restoration of these blocks; 3) restoring the pixel value based on the restored values of the blocks they belong to. In a word, blockwise means that every movement of the search window restore one block of pixels. So for each block B_{i_k}, a blockwise nonlocal-means restoration is performed as follows:

$$\mathrm{NL(u)}\left(B_{i_k}\right) = \sum w(B_{i_k}, B_j) u(B_j) \quad (4)$$

With

$$\mathrm{w}\left(B_{i_k}, B_j\right) = \frac{1}{Z_{i_k}} e^{-\frac{\left\|u\left(B_{i_k}\right) - u(B_j)\right\|_2^2}{2\beta\sigma^2|N_i|}} \tag{5}$$

Moreover, unlike the pixelwise nonlocal-means, the value of some pixels in some locations is the result of several overlapping blocks. It means that several estimation of the restored intensity $\mathrm{NL(u)}(x_i)$ are obtained in different $\mathrm{NL(u)}(B_{i_k})$. Thus, the final restored intensity of pixel x_i is defined as follows:

$$\mathrm{NL(u)}\left(B_{i_k}\right) = \frac{1}{n}\sum NL(u)(B_{i_k}) \tag{6}$$

Where n is the number of the overlapping blocks.

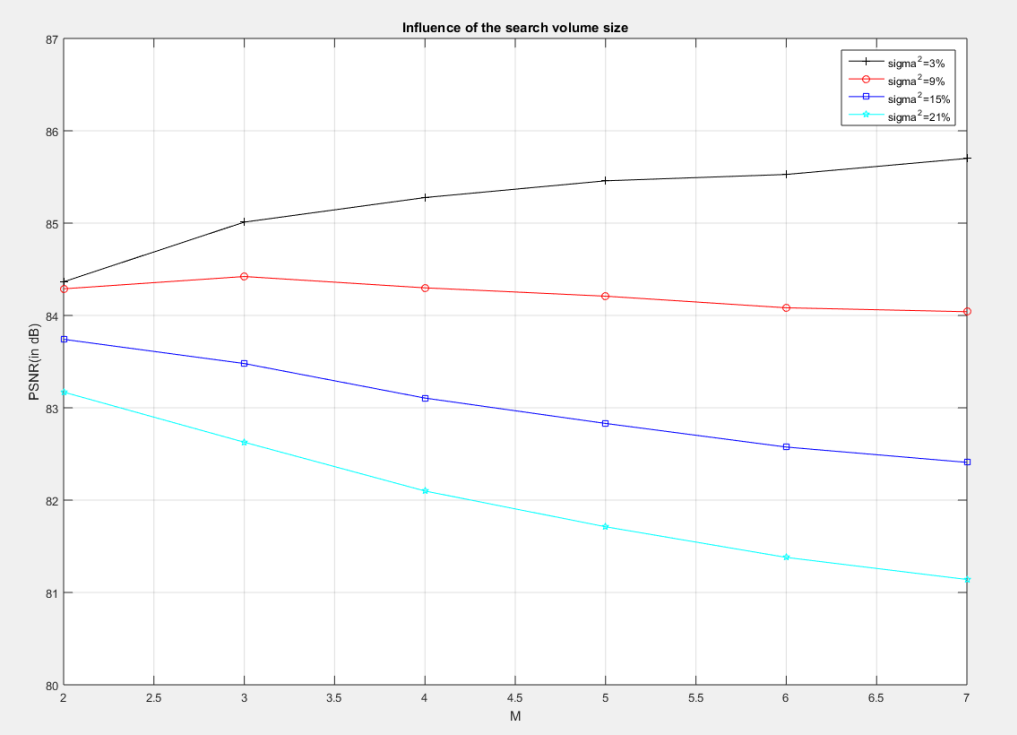
After applying the block selection part, the blockwise nonlocal-means can be expressed as follows:

$$\mathrm{w}\left(B_{i_k}, B_j\right) = \begin{cases} \frac{1}{Z_{i_k}} e^{-\frac{\left\|u\left(B_{i_k}\right) - u\left(B_j\right)\right\|_2^2}{2\beta\sigma^2|N_i|}}, & if\, \mu_1 < \frac{\overline{u\left(B_{i_k}\right)}}{\overline{u\left(B_j\right)}} < \frac{1}{\mu_1} \;\; and \;\; \sigma_1^2 < \frac{Var\left(u\left(B_{i_k}\right)\right)}{Var\left(u\left(B_j\right)\right)} < \frac{1}{\sigma_1^2} \\ 0, & otherwise \end{cases} \tag{7}$$

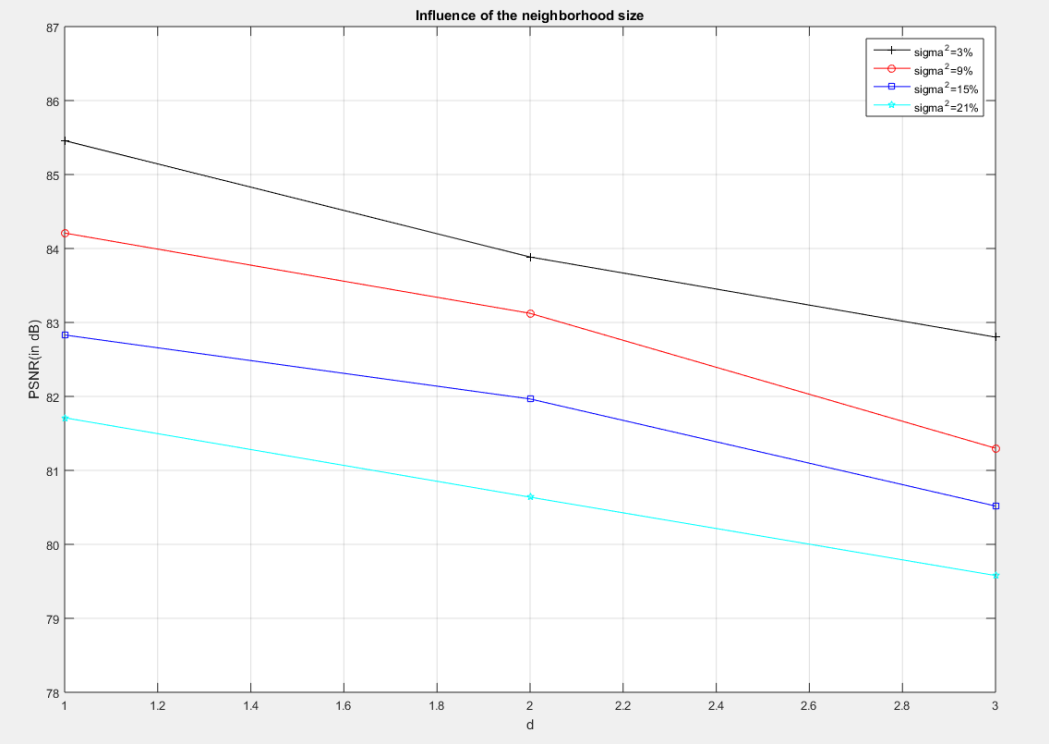
4. Results

Based on the theories above, Matlab programs are used to achieve the classical nonlocal-means and the improvements. In this project, the validation experiment is divided into four parts: 1) pixelwise nonlocal-means; 2) optimized pixelwise nonlocal-means; 3) blockwise nonlocal-means; 4) optimized blockwise nonlocal-means. All the nonlocal-means mentioned above are both implemented with the automatic tuning smoothing parameter h. And the 'optimized' one is implemented with the pixel or block selection part. In order to test and compare the performance and the computation time of four kinds of the nonlocal-means algorithms, medical image T2 from the BrainWeb database with four noise levels (3%, 9%, 15% and 21%) are used. By doing so, we can evaluate the contribution of every optimization. Finally, some comparisons with orther denoising such as Wiener filtering and Gaussian filtering.

(1) The first part of the test is the influence of the size of the search window and the neighbor blocks.



(a)



(b)

Fig.1. Influence of the size $|V_i| = (2M+1)^2$ and $|N_i| = (2d+1)^2$ for denoising: influence of the size of the search window and the size of the neighborhood on the PSNR, for the four levels noise. (a) Variation of the size M of the search window for d=1. (b) Variation of the size d of the neighborhood N_i for M=5.

Fig.1(a) shows that with the increasing size of the search window, the value of the PSNR decreases slightly. In other words, the more irrelevant pixels taken into account, the small PSNR we get. Fig.1(b) shows that increasing d degrades the denoising quality.

(2) The second part of the experiment is the Influence of the automatic tuning of smoothing parameter h:

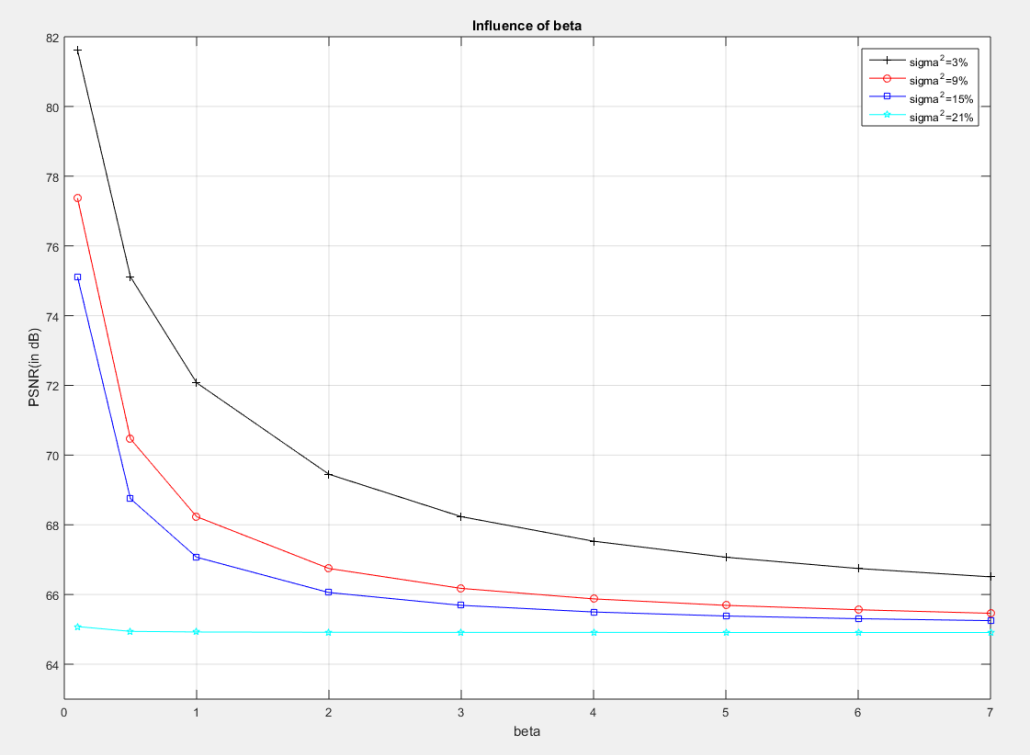


Fig.2. Influence of the smoothing parameter h.

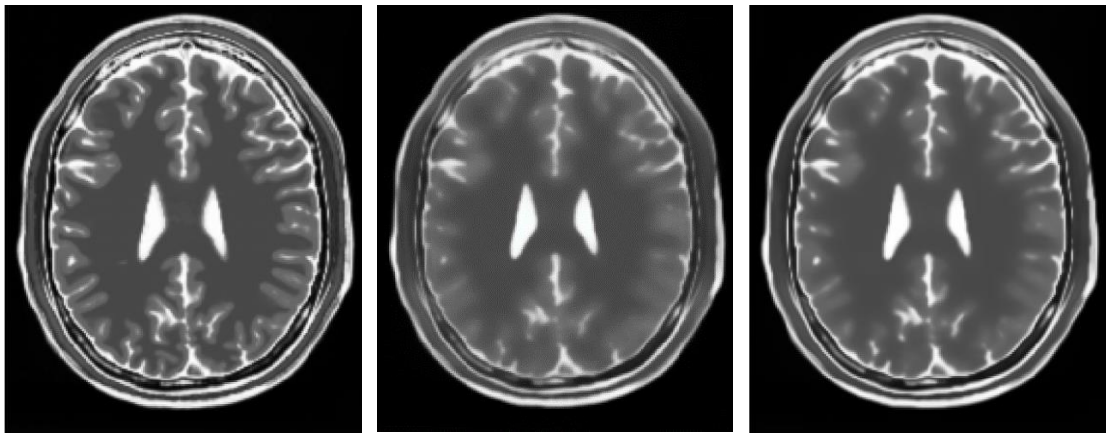
Fig.2 shows that when the size of the neighborhood and the noise extent is known, β is the only variable can be adjusted, and with increasing value of β, the PSNR drops dramatically for the low and middle level of noise. When it comes to high level of noise, the PSNR keeps constant. And for low and middle levels of noise, the best value of β is close to 0.1.

(3) Impact of the blockwise implementation and pixels or blocks selection

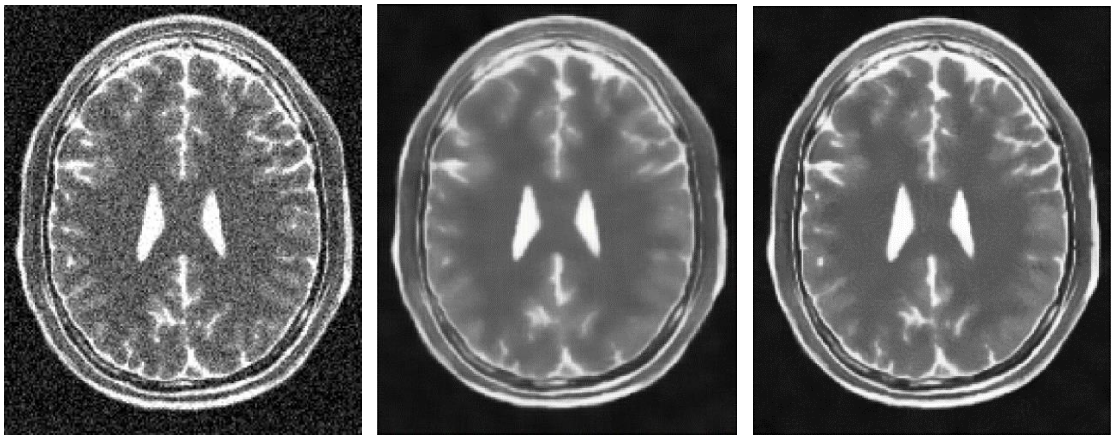
The third part is the comparison of four kinds of nonlocal-means. In this test, the denoising performance and the computation time after applying the selection part and the blockwise idea will be shown.

	Computation time (in s) Core i5-5200U 2.2GHz	PSNR (in dB)
Pixelwise Nonlocal-means	109s	72.5504
Optimized Pixelwise Nonlocal-means	538s	77.3767
Blockwise Nonlocal-means	21s	67.3379
Optimized Blockwise Nonlocal-means	118s	68.3072

Table .1. Comparison of different nonlocal-means in term of computation time and denoising quality.



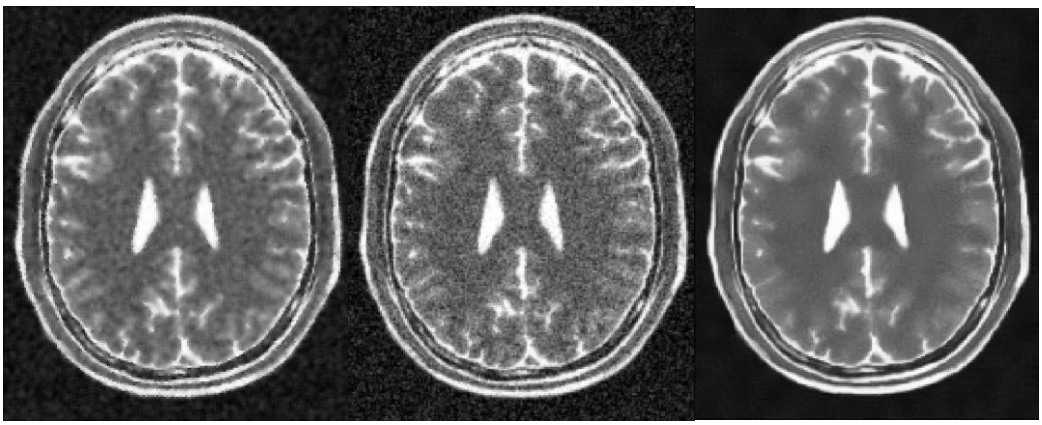
(a)“Ground truth” (b) Pixelwise NL-means (c) Optimized pixelwise NL-means



(d)Noisy image 9% (e) Blockwise NL-means (f) Optimized blockwise NL-means

Fig.3. Comparison of different denoised images of different implements of the nonlocal-means in term of the visual quality.

(4) The third part of the experiment is the comparison with other denoising filterings: In this part, the blockwise nonlocal-means is compared to the Gaussian and Wiener filtering.



(a) Wiener filtering (b) Gaussian filtering (c) Blockwise NLM

Fig.4. comparison with Wiener filtering and Gaussian filtering.

As we can see from the figure, the denoising effect of nonlocal-means is much better than the Wiener filtering and Gaussian filtering. The nonlocal-means protects more edge details.

5. Conclusion

The results above shows that the blockwise implement obviously improves the efficiency of the algorithm. However, since the selection part is deep inside the “for loop” in the Matlab program, it increases the computation time but it indeed improve the denoising quality and protect more edge details. Compared to other denoising filters such as Wiener and Gaussian filters, the nonlocal-means denoising effect of NLM is much better.

6. Reference

[1] An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images P Coupé, P Yger, S Prima, P Hellier, C Kervrann, C Barillot Medical Imaging, IEEE Transactions on 27 (4), 425-441

Appendix

```
function DenoisedImg_P=NLmeans(I,ds,Ds,beta,sigma_sq)
%I: Noisy image
%ds: Size of the neighbor block
%Ds: Size of the search window
%h: Gaussian smoothing parameter
%DenoisedImg£°Denoised image
I=double(I);
[m,n]=size(I);
DenoisedImg=zeros(m,n);
DenoisedImg_P=zeros(m+2,n+2);
PaddedImg = padarray(I,[ds,ds],'symmetric','both');

Ni=(2*ds+1)^2;
h2=2*beta*sigma_sq*Ni;

for i=1:ceil(m/2)
    for j=1:ceil(n/2)
        i1=2*i;
        j1=2*j;
        W1=PaddedImg(i1-ds:i1+ds,j1-ds:j1+ds);
%       uNi_mean=mean2(W1);
%       VarNi_mean=sum((W1-mean2(W1)).^2)/length(W1);
        wmax=0;
        average=zeros(3,3);
        sweight=0;
        rmin = max(i1-Ds,ds+1);
        rmax = min(i1+Ds,m+ds);
        smin = max(j1-Ds,ds+1);
        smax = min(j1+Ds,n+ds);
        for r=rmin:rmax
            for s=smin:smax
                if(r==i1&&s==j1)
                continue;
                end
                W2=PaddedImg(r-ds:r+ds,s-ds:s+ds);
%               uNj_mean=mean2(W2);
%               VarNj_mean=sum((W2-mean2(W2)).^2)/length(W2);
%               if
(0.99<(uNi_mean/uNj_mean)<(1/0.99))&&(0.9<(VarNi_mean/Var
Nj_mean)<(1/0.95))
                Dist2=sum(sum((W1-W2).*(W1-W2)));
                w=exp(-Dist2/h2);
```

```
%             else
%                   w=0;
%              end
               if(w>wmax)
                   wmax=w;
               end
               sweight=sweight+w;
               average=average+w*W2;
           end
       end
       average=average+wmax*W1;
       sweight=sweight+wmax;
       Overlap=[4,2,4;2,1,2;4,2,4];
       DenoisedImg_P(i1-ds:i1+ds,j1-
ds:j1+ds)=DenoisedImg_P(i1-ds:i1+ds,j1-
ds:j1+ds)+(average/sweight)./Overlap;
   end
end
////////////////////////////////////////////////////////

function DenoisedImg=NLmeans2(I,ds,Ds,beta,sigma_2)
%I: Noisy image
%ds: Size of the neighbor block
%Ds: Size of the search window
%h: Gaussian smoothing parameter
%DenoisedImg£°Denoised image
I=double(I);
[m,n]=size(I);
DenoisedImg=zeros(m,n);
PaddedImg = padarray(I,[ds,ds],'symmetric','both');

Ni=(2*ds+1)^2;
h2=2*beta*sigma_2*Ni;

for i=1:m
   for j=1:n
       i1=i+ds;
       j1=j+ds;
       W1=PaddedImg(i1-ds:i1+ds,j1-ds:j1+ds);
%      uNi_mean=mean2(W1);
%      VarNi_mean=sum((W1-mean2(W1)).^2)/length(W1);
       wmax=0;
       average=0;
       sweight=0;
```

```
        rmin = max(i1-Ds,ds+1);
        rmax = min(i1+Ds,m+ds);
        smin = max(j1-Ds,ds+1);
        smax = min(j1+Ds,n+ds);
        for r=rmin:rmax
            for s=smin:smax
                if(r==i1&&s==j1)
                continue;
                end
                W2=PaddedImg(r-ds:r+ds,s-ds:s+ds);
%               uNj_mean=mean2(W2);
%               VarNj_mean=sum((W2-
mean2(W2)).^2)/length(W2);
%               if
(0.99<(uNi_mean/uNj_mean)<(1/0.985))&&(1<(VarNi_mean/VarN
j_mean)<(1/0.8))
                Dist2=sum(sum((W1-W2).*(W1-W2)));
                w=exp(-Dist2/h2);
%               else
%                    w=0
%               end
                if(w>wmax)
                   wmax=w;
                end
                sweight=sweight+w;
                average=average+w*PaddedImg(r,s);
            end
        end
        average=average+wmax*PaddedImg(i1,j1);
        sweight=sweight+wmax;
        DenoisedImg(i,j)=average/sweight;
    end
end
////////////////////////////////////////////////////////

close all;
clear all;
clc

load t2_1mm_noise0_RF0
I=vol(:,:,100);
I=double(I);
I=I-min(I(:));
I=I/max(I(:));
```

```
level=0.09;
sigma_2=level*max(I(:));
I1=imnoise(I,'gaussian',sigma_2);

tic
O1=NLmeans2(I,1,5,0.3,sigma_2);
toc

figure;
imshow(rot90(I),[]);
title('Ground truth');
figure;
imshow(rot90(I1),[]);
title('Noisy image 9%');
figure;
imshow(rot90(O1),[]);
title('Optimized Pixelwise NL-means');

rmse=sqrt((sum(sum((I-O1).^2)))/(181*217));
range=255;
psnrl=20*log10(range/rmse)
////////////////////////////////////////////////////////

load t2_1mm_noise0_RF0
I=vol(:,:,100);
I=double(I);
I=I-min(I(:));
I=I/max(I(:));

level=0.09;
sigma_2=level*max(I(:));
I1=imnoise(I,'gaussian',sigma_2);

PaddedImg = padarray(I1,[1,1],'symmetric','both');
I2 = padarray(I,[1,1],'symmetric','both');

tic
O1=NLmeans(I1,1,5,0.1,sigma_2);
toc

figure;
imshow(rot90(I2),[]);
title('Ground truth');
```

```
figure;
imshow(rot90(PaddedImg),[]);
title('Noisy image 9%');
figure;
imshow(rot90(O1),[]);
title('Blockwise NL-means');

rmse=sqrt((sum(sum((I2-O1).^2)))/(181*217));
range=255;
psnrl=20*log10(range/rmse)
////////////////////////////////////////////////////////

load t2_1mm_noise0_RF0
I=vol(:,:,100);
I=double(I);
I=I-min(I(:));
I=I/max(I(:));
level=0.09;
sigma_2=level*max(I(:));
I1=imnoise(I,'gaussian',sigma_2);

% I2=wiener2(I1,[3 3]);
I2=filter2(fspecial('gaussian',3),I1)/255;

figure;
imshow(rot90(I),[]);
title('Ground truth');
figure;
imshow(rot90(I1),[]);
title('Noisy image 9%');
figure;
imshow(rot90(I2),[]);
title('');

rmse=sqrt((sum(sum((I2-I).^2)))/(181*217));
range=255;
psnrl=20*log10(range/rmse)
```