Ben Postlethwaite 76676063

Table of Contents

Assignment 1

Assignment Setup

```
clear all; close all;
loadtools;   % Load the toolboxes
```

Select Image

```
sel = 3;
if sel == 1
    img = 'lena';
elseif sel == 2
    img = 'centaur1';
elseif sel == 3
    img = 'boat';
end
```

Load image and crop if necessary

```
n = 512;
f = load_image(img,[]);
f = rescale(crop(f,n));
```

Question 1 - Intro to Image Processing

To avoid boundary artifacts and estimate really the frequency content of the image (and not of the artifacts!), one needs to multiply M by a smooth windowing function h and compute fft2(M.*h). Use a sine windowing function. Can you interpret the resulting filter ?

Set up Sine Window

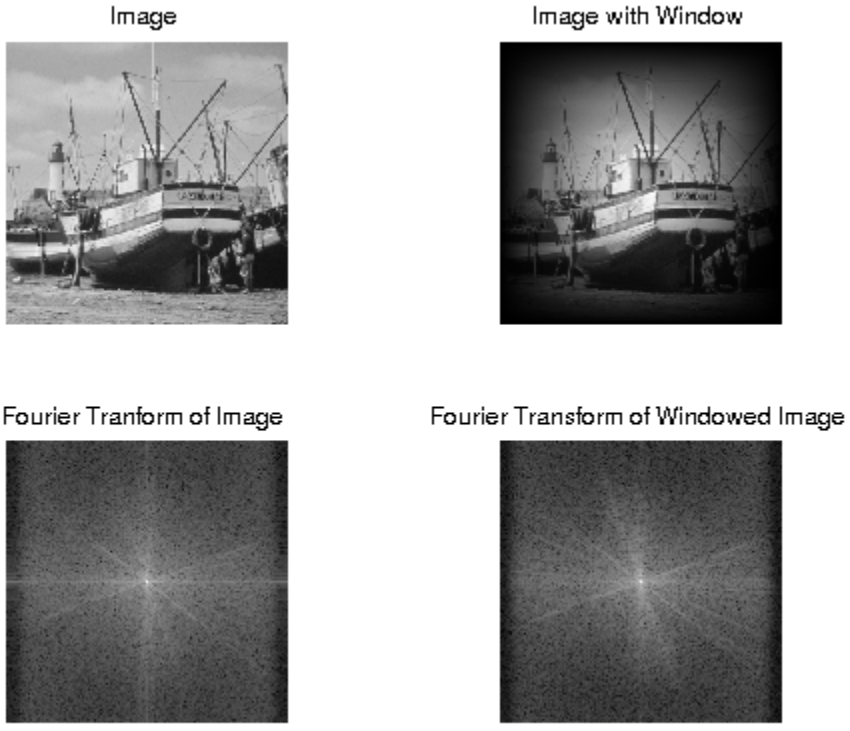
```
x = linspace(0,pi,n);
[X,Y] = meshgrid(x,x);
win = sin(X).*sin(Y);
```

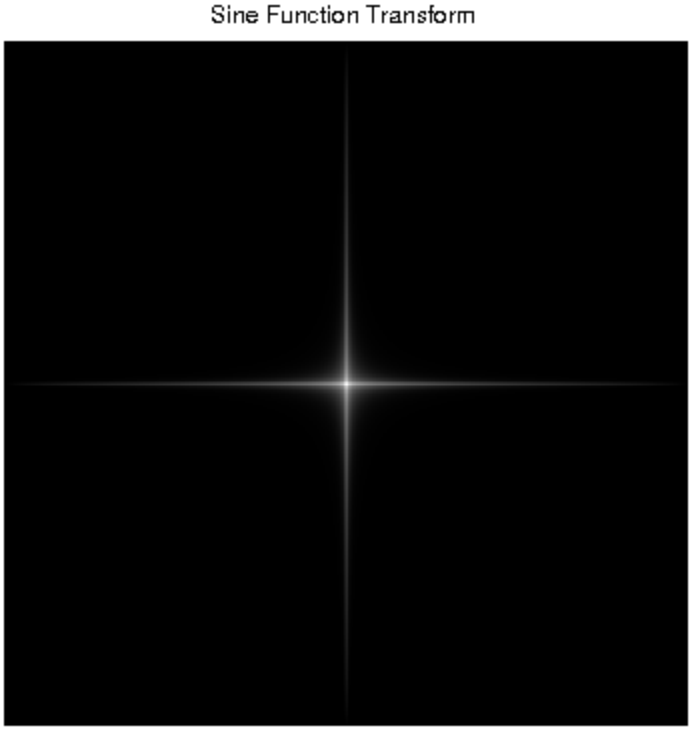
fft image and sine function

```
fw  = f.*win;
Mf  = fft2(f);
Mfw = fft2(fw);
Lf  = fftshift(log( abs(Mf)+1e-1 ));
Lfw = fftshift(log( abs(Mfw)+1e-1 ));
Lw  =fftshift(log( abs(fft2(win))+1e-1 ));
```

Plots

```
figure(1)
    imageplot(f, 'Image', 2,2,1);
    imageplot(fw, 'Image with Window', 2,2,2);
    imageplot(Lf, 'Fourier Tranform of Image', 2,2,3);
    imageplot(Lfw, 'Fourier Transform of Windowed Image', 2,2,4);
figure(2)
    imageplot(Lw, 'Sine Function Transform',1,1,1)
```





Commentary

The fourier transform of the original image shows distinct low frequency or zero frequency (DC) energy. This energy is removed when the original image is multliplied by the Sine Window function. The fourier transform assumes periodicity and therefore each opposing boundary will be mapped together and the resulting jump or step will cause artifacts. **The filter is like a convolution in the frequency domain with a 2D sinc function looking thing.**

Question 2 - Intro to Image Processing

Perform low pass filtering by removing the high frequencies of the spectrum. What do you oberve ?

Design boxcar for zeroing high freqs Note that each frequency 'bin' represents a wavenumber relative to pixels, so the first bin would be features of full image pixel size while the 2nd would show the energy for features represented by 255/256 pixel size, this goes to the Nyquist level of 2 pixels / cycle.

```
M = 64; % Select first __ bins from the transformation
m = round(M/2);
c = n/2 +1; % Centre of ifft2shift image
% Zero out freq's outside of our cutoff range, and transform back to
% spatial dimension
Mlp = fftshift(Mf);              % Swap Vectors
Mlp([1:c-m-1,c+m+1:end],:) = 0;    % Zero outside keeping m coeffs
Mlp(:,[1:c-m-1,c+m+1:end]) = 0;
fM = ifft2(ifftshift(Mlp));
```

Plots

```
figure(3)
    imageplot(fM, 'Low Pass Filtered image',1,1,1)
```

Commentary

The zeroing out of higher frequencies effectively low-passes the image which destroys finer details in the image creating a 'blocky', low resolution image. Also periodic artifacts are introduced, with pixel sizes equal to that of our cutoff point.

Question 1 - Image Processing with Fourier and Wavelets

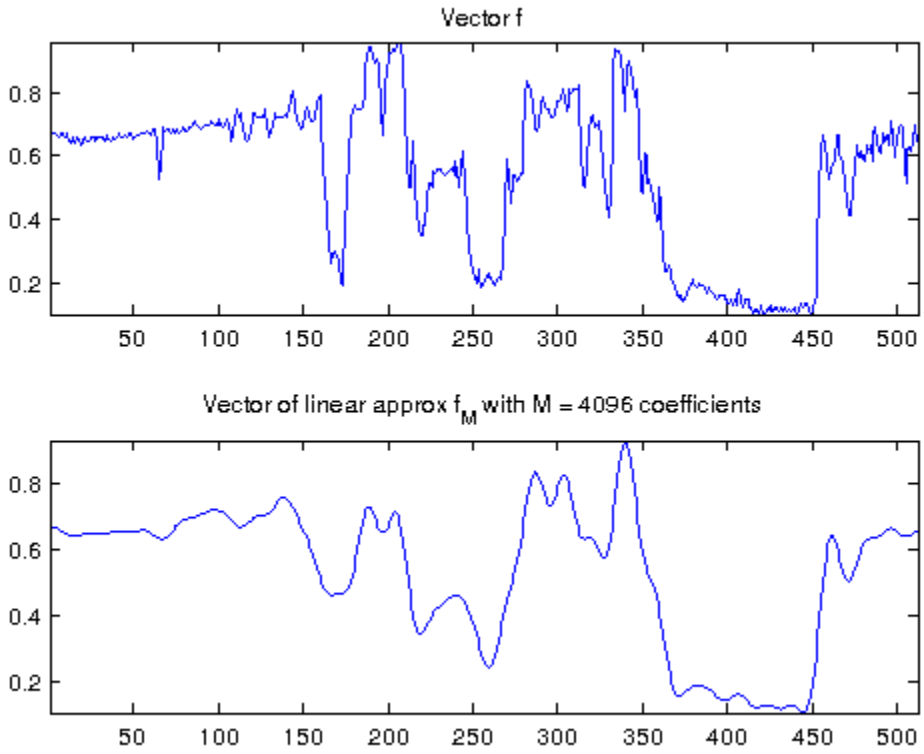
Perform the linear Fourier approximation with M coefficients. Store the result in the variable *fM*.

```
M  = (n^2)/64;
m = round(0.5*sqrt(M));
Mlp = fftshift(Mf);                 % Swap Vectors
Mlp([1:c-m-1,c+m+1:end],:) = 0;     % Zero outside keeping m coeffs
Mlp(:,[1:c-m-1,c+m+1:end]) = 0;
fM = ifft2(ifftshift(Mlp));
```

Plot

```
figure(4)
subplot(2,1,1);
    plot(f(:,n/2));
```

```
    axis('tight'); title('Vector f');
subplot(2,1,2);
    plot(fM(:,n/2));
    axis('tight');
  title(sprintf('Vector of linear approx f_M with M = %i coefficients',M));
figure(5)
    imageplot(clamp(fM), strcat(['Linear Fourier Approximation, SNR='...
        num2str(snr(f,fM),3) 'dB  and M = ' num2str(sum(Mlp(:) ~= 0 ))]));
```



Commentary

The high frequencies have been removed in the approximation with only M coefficients kept. Note this is a 'linear' approximation since the selection of M is independent of the particular function f. (In this case we kept only M = sqrt((n^2)/64) where n is pixel count in dimension x. Not only is the resolution more coarse there are periodic artifacts in the image.

Question 2 - Image Processing with Fourier and Wavelets

Compute the value of the threshold T so that the number of coefficients is M. Display the corresponding approximation fM.

Calculate threshold so that we select M coefficients.

```
T = 124;       % Start at threshold at max coeff
m = 0;                          % Initialize m;
dT = 0.1;                       % Get some original value for T
while m < M-1 || m > M+1
m = round(sum ((abs(Mf(:))> T) ~=0));
    if m < M                  %If we need more m's threshold needs to go down
        T = T - dT;
    elseif m > M              %If we need less m's threshold needs to go up
        T = T + dT;
    end
end
fM = ifft2(Mf.*(abs(Mf)>T));    % Non linear approximation
```

Plot

```
figure(6)
    imageplot(clamp(fM), strcat(['non-Linear Fourier Approximation, SNR='...
        num2str(snr(f,fM),3) 'dB  and M = '  num2str(m)]));
```

Commentary

The non-linear approximation is much better than the linear approx. There is some overhead in selecting the Threshold value to match a certain M though the problem may be such that the desired threshold is known and we are not interested in M, in which case there is no computational cost associated with the dependent selection of M. However, selecting only the dominant M frequencies is much more desirable and captures much more of the energy of the original function.

Question 3 - Image Processing with Fourier and Wavelets

```
Perform linear approximation with M wavelet coefficients.
```

```
Jmin = 0;
fw = perform_wavelet_transf(f,Jmin,+1);
fwm = fw;
m = sqrt(M);
fwm(m+1:end,:) = 0;
fwm(:,m+1 : end) = 0;
fM = perform_wavelet_transf(fwm,Jmin,-1);
```

Plot

```
figure(7)
    imageplot(clamp(fM), strcat(['Linear Wavelet Approximation, SNR='...
        num2str(snr(f,fM),3) 'dB  and M = ' num2str(sum(fwm(:) ~= 0 ))]));
```

Commentary

The linear-wavelet approximation with M coefficients exceeds the resolution of the non-linear fourier approx as well as reducing visible artifacts. Taking the first M coefficients is like keeping log2(M) = j where j= 5 for M = 32 scales, or first 5 wavelet resolution spaces.

Question 4 - Image Processing with Fourier and Wavelets

Perform non-linear approximation with M wavelet coefficients by choosing the correct value for T. Store the result in the variable fM.

Calculate threshold so that we select M coefficients.

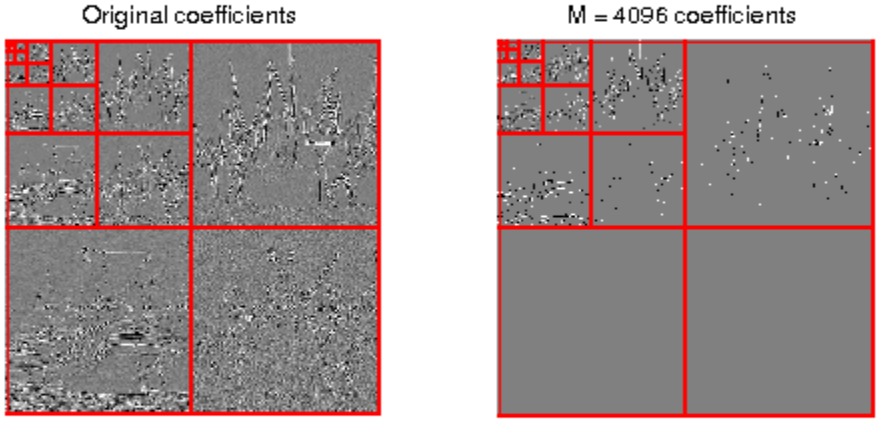
```
T = 100*mean(fw(:));        % Start at threshold at max coeff
m = 0;                          % Initialize m;
dT = 0.001;                       % Get some original value for T
jj = 1;
while m ~= M
```

```
    m = round( sum( (fw(:)>T) ~=0) );
    if m < M                    %If we need more m's threshold needs to go down
        T = T - dT;

    elseif m > M                %If we need less m's threshold needs to go up
        T = T + dT;
    end
    if mod(jj,100) == 0
        dT = dT/10;
    end
    jj = jj + 1;
end
fwT =  fw .* (abs(fw)>T);   % Take only M coefficients
fM = perform_wavelet_transf(fwT,Jmin,-1);   % Transform back
```

Plot

```
figure(8)
subplot(1,2,1);
    plot_wavelet(fw);
    title('Original coefficients');
subplot(1,2,2);
    plot_wavelet(fwT);
    title(sprintf('M = %i coefficients',M))
figure(9)
    imageplot(clamp(fM), strcat(['Non Linear Wavelet Approximation, SNR='...
        num2str(snr(f,fM),3) 'dB  and M = ' num2str(m)]));
```



Non Linear Wavelet Approximation, SNR=25.3dB and M = 4096

Commentary

The non-linear wavelet approximation handles edges, resolution better than all the previous approximations. In effect, the larger coefficients capture more of the original images variance than do the larger coefficients of the fourier approx. Though in the fourier approx, storage is wasted due to the negative complex conjugates being stored... ie, some of the larger coefficients chosen are redundant information in the symmetric hermitian fourier matrix.

Question 1 - Image Processing with Wavelets

Compute the ratio M/N of non-zero coefficients.

Set up New image

```
name = 'cortex';
n = 512;
f = load_image(name,n);
f = rescale( sum(f,3) );
T = .5; % Set threshold
```

Create shortcut funcs

```
Psi = @(f)perform_wavelet_transf(f,Jmin,+1);
PsiS = @(fw)perform_wavelet_transf(fw,Jmin,-1);
Thresh = @(fw,T)fw .* (abs(fw)>T);
```

Compute ratio

```
fw = Psi(f);
fwT = Thresh(fw,T);
display(['The ratio M/N of non-zero coefficients is ', num2str((sum(fwT(:) ~= 0 ))
```

```
The ratio M/N of non-zero coefficients is 0.0048065
```

Commentray

Not much to comment on, my calculation is different than the example, don't know why.

Question 2 - Image Processing with Wavelets

Compute a threshold T to keep only M coefficients.

Calculate threshold so that we select M coefficients.

```
M = n^2/16;
T = 0.0530;       % Start at threshold at max coeff
m = 0;                          % Initialize m;
dT = 0.001;
jj = 0;            % Get some original value for T
while m ~= M
    m = round( sum(  (abs(fw(:))>T) ~=0) );
    if m < M                %If we need more m's threshold needs to go down
        T = T - dT;
    elseif m > M            %If we need less m's threshold needs to go up
        T = T + dT;
    end
    if mod(jj,100) == 0
            dT = dT/10;     % Reduce search vector by 1/10
    end
    jj = jj + 1;
end
fwT = Thresh(fw,T);   % Take only M coefficients
fM = PsiS(fwT); % Transform back
disp(['Threshold T = ', num2str(T), ' used to keep coefficients:'])
disp(strcat(['      M = ' num2str(M)]));
disp(strcat(['|fWT|_0 = ' num2str(sum(fwT(:)~=0))]));
```

```
Threshold T = 0.052944 used to keep coefficients:
      M = 16384
|fWT|_0 = 16384
```

Commentary

Again not much to say, used the same technique for finding appropriate threshold as previous questions, but the method, a simple scoping search, is inefficient, I will think on a way to improve this.

Question 3 - Image Processing with Wavelets

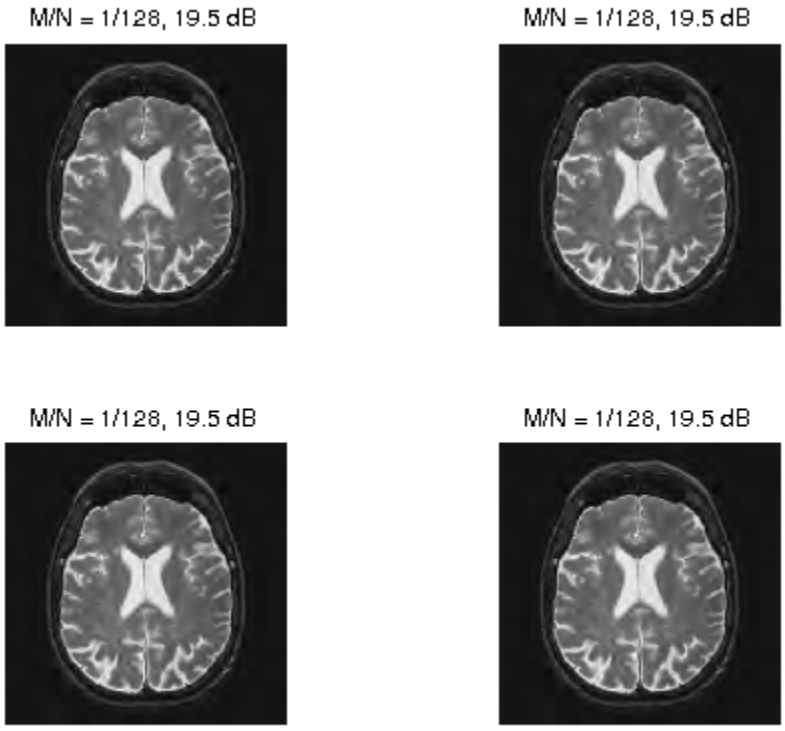
Compute an approximation with an decreasing number of coefficients.

```
T = 0.0530; % Start at some initial threshold
MM = n^2./[16,32,64,128] ;
for ii = 1:length(MM)
    M = MM(ii);
```

```
    m = 0;                              % Initialize m;
    dT = 0.001; % Get some original value for T
    jj = 1;
    while m ~= M
        m = round( sum(  (abs(fw(:))>T) ~=0) );
        if m < M                %If we need more m's threshold needs to go down
            T = T - dT;
        elseif m > M            %If we need less m's threshold needs to go up
            T = T + dT;
        end
        if mod(jj,400) == 0
            dT = dT/10;
        end
        jj = jj + 1;
    end
    fwT = Thresh(fw,T);   % Take only M coefficients
    fM(:,:) = PsiS(fwT);
end
```

Plot

```
    figure(101)
    for ii = 1:4
    imageplot(fM,sprintf('M/N = 1/%i, %2.1f dB',n^2/sum(fwT(:)~=0),snr(f,fM)), 2,2
    end
```



Commentary

The wavelet approximations enjoy fast decay, that is even though we drop many more coefficients the variance stored in the largest coefficients is considerable. I will investigate the dominant eigenvalues of

this matrix, it is my guess that they will decay steeply.

Question 4 - Image Processing with Wavelets.

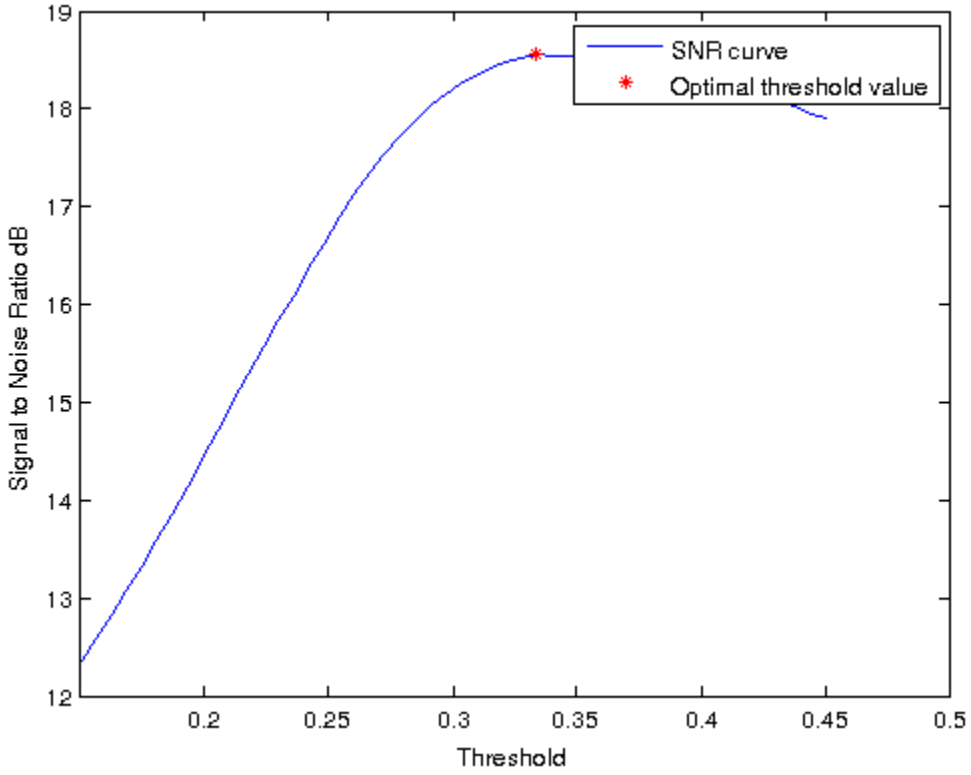
Try to optimize the value of the threshold T to get the best possible denoising result.

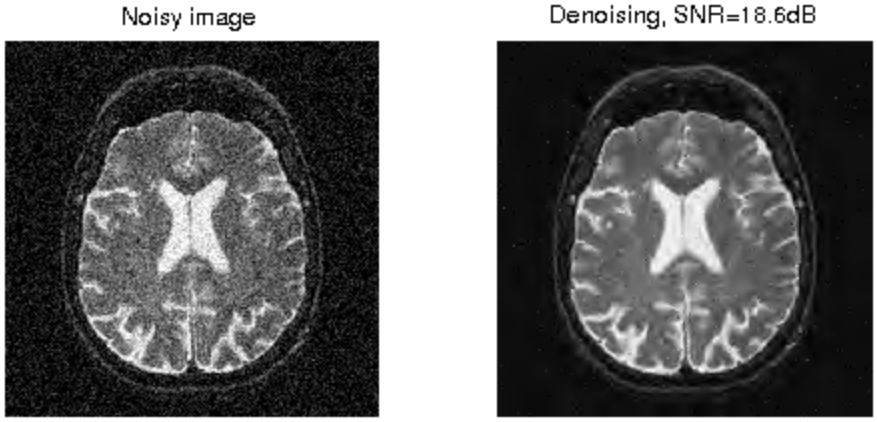
Set threshold to T = 3*sigma, take threshold and invert back

```
sigma = .1;
y = f + randn(n,n)*sigma;
fW = Psi(y);
T = 3*sigma;
Tv = linspace(0.5*T,1.5*T,50);
SNR = zeros(1,length(Tv));
for ii = 1:length(Tv)
    fWT = Thresh(fW,Tv(ii));
    f1 = PsiS(fWT);
    SNR(ii) = snr(f,f1);
end
Tvbest = Tv(max(SNR) == SNR);
fWT = Thresh(fW,Tvbest);
f1 = PsiS(fWT);
```

Plots

```
figure(37)
plot(Tv,SNR,Tvbest,max(SNR),'r*')
xlabel('Threshold'); ylabel('Signal to Noise Ratio dB')
legend('SNR curve','Optimal threshold value')
figure(365)
imageplot(clamp(y), 'Noisy image', 1,2,1);
imageplot(clamp(f1), strcat(['Denoising, SNR='...
    num2str(snr(f,f1),3) 'dB']), 1,2,2);
```


SNR curve
Optimal threshold value
Signal to Noise Ratio dB
Threshold
19
18
17
16
15
14
13
12
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Noisy image
Denoising, SNR=18.6dB

Commentary

After optimization the image has a marginally better SNR, with an increase of 0.3 dB. (From 18.3 -> 18.6)

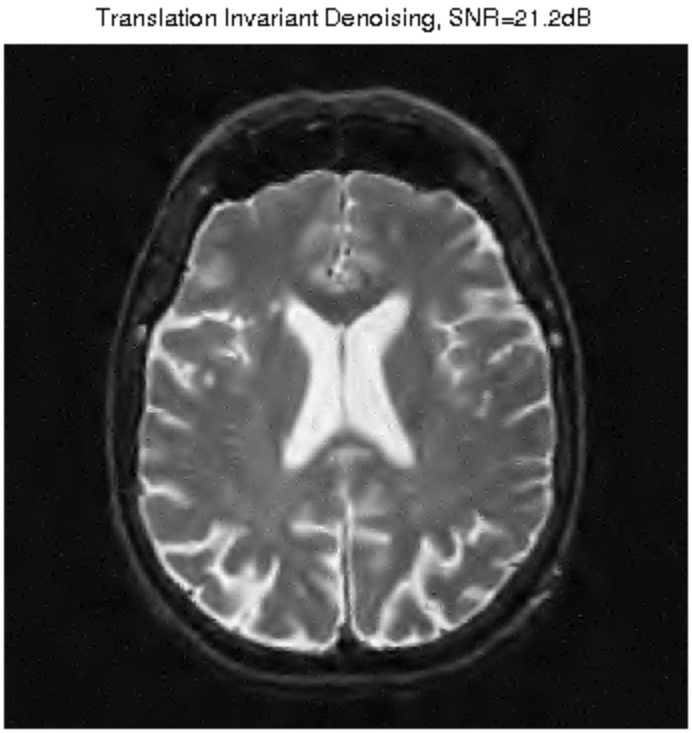
Question 5 - Image Processing with Wavelets.

```
Compute the full denoising by cycling through the i indices.
```

```
T = 3*sigma;
tau_max = 8;
[Y,X] = meshgrid(0:tau_max-1,0:tau_max-1); % Generate a set of trans vectors
f1 = zeros(n,n); % Initialize the denoised image f~ as 0.
for i=1:tau_max^2
    fTrans = circshift(y,[X(i) Y(i)]);
    fTrans = PsiS( Thresh( Psi(fTrans) ,T) );
    fTrans = circshift(fTrans,-[X(i) Y(i)]);
    f1 = (i-1)/i*f1 + fTrans/i;
end
```

Plot

```
figure(346)
imageplot(clamp(f1), strcat(['Translation Invariant Denoising, SNR='...
    num2str(snr(f,f1),3) 'dB']), 1,1,1);
```



Commentary

The translation denoising process increases the signal to noise ratio by basically stacking the image, it translates the image, shifts it under the assumtion of periodic b/c's and then performs the transform thresholding and inverse transform, it is then stacked by reweighting the stack and the new thresholded image so that the weights are unity and they scale like 1/i to weight the stack more heavily with further iterations. Cool idea.

Question 6 - Image Processing with Wavelets

Determine the optimal threshold T for this translation invariant denoising.

```
Tv = linspace(0.8*T,1.8*T,30);
SNR = zeros(1,length(Tv));
tau_max = 4; %Smaller tau_max for optimization
% Loop builds out line search for maximum SNR
for ii = 1:length(Tv)
    f1 = zeros(n,n); % Initialize the denoised image f~ as 0.
    for i=1:tau_max^2
        fTrans = circshift(y,[X(i) Y(i)]);
        fTrans = PsiS( Thresh( Psi(fTrans) ,Tv(ii)) );
        fTrans = circshift(fTrans,-[X(i) Y(i)]);
        f1 = (i-1)/i*f1 + fTrans/i;
    end
    SNR(ii) = snr(f,f1);
end
% Choose Threshold best that gives highest SNR
Tvbest = Tv(max(SNR) == SNR);
```

Now iterate with higher tau_max using optimal threshold

```
tau_max = 8; %Increase Taumax for final
for i=1:tau_max^2
    fTrans = circshift(y,[X(i) Y(i)]);
    fTrans = PsiS( Thresh( Psi(fTrans) ,Tvbest) );
    fTrans = circshift(fTrans,-[X(i) Y(i)]);
    f1 = (i-1)/i*f1 + fTrans/i;
end
```

Plot

```
figure(123)
imageplot(clamp(f1), strcat(['Translation Invariant Denoising with '...
    'Optimal Threshold, SNR=' num2str(snr(f,f1),3) 'dB']), 1,1,1);
```

Commentary

For optimizing the translation invariant method I found that the optimal threshold vs SNR plot maximum was close enought to the initial guess 3*sigma that it had no real positive impact on the SNR of the final image.

Question 7 - Image Processing with Wavelets

Test other images

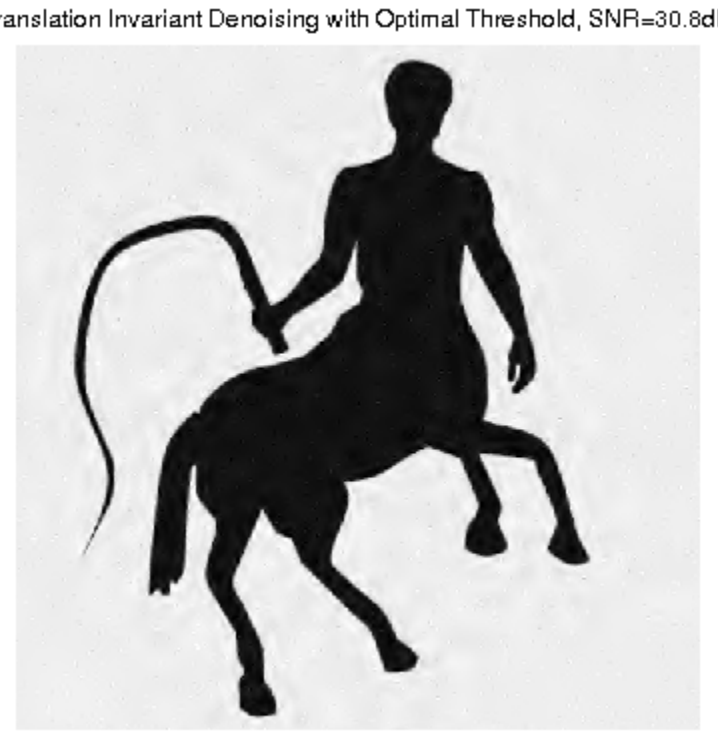
```
sigma = .1;
T = 3*sigma;

for qq = 1:2
if qq == 1
    img = 'lena';
elseif qq == 2
    img = 'centaur1';
end
% Load image and crop if necessary


f = load_image(img,n);
f = rescale( sum(f,3) );

y = f + randn(n,n)*sigma;
Tv = linspace(0.5*T,1.5*T,30);
SNR = zeros(1,length(Tv));
tau_max = 4; %Smaller tau_max for optimization
% Loop builds out line search for maximum SNR
```

```
for ii = 1:length(Tv)
    f1 = zeros(n,n); % Initialize the denoised image f~ as 0.
    for i=1:tau_max^2
        fTrans = circshift(y,[X(i) Y(i)]);
        fTrans = PsiS( Thresh( Psi(fTrans) ,Tv(ii)) );
        fTrans = circshift(fTrans,-[X(i) Y(i)]);
        f1 = (i-1)/i*f1 + fTrans/i;
    end
    SNR(ii) = snr(f,f1);
end
% Choose Threshold best that gives highest SNR
Tvbest = Tv(max(SNR) == SNR);
% Now iterate with higher tau_max using optimal threshold
tau_max = 8; %Increase Taumax for final
for i=1:tau_max^2
    fTrans = circshift(y,[X(i) Y(i)]);
    fTrans = PsiS( Thresh( Psi(fTrans) ,Tvbest) );
    fTrans = circshift(fTrans,-[X(i) Y(i)]);
    f1 = (i-1)/i*f1 + fTrans/i;
end
% *Plot*
figure(123+qq)
imageplot(clamp(f1), strcat(['Translation Invariant Denoising with '...
    'Optimal Threshold, SNR=' num2str(snr(f,f1),3) 'dB']), 1,1,1);
end
```



Commentary

Other images, the deblurring looks pretty good, even on the sharp edges of the centaur.

Published with MATLAB® 7.11