CSE166 Assignment #:
Due Date:

Please list all contributors to the assignment below.
Staple this sheet to the front of your submission.

1 Student Name(s):

1. _____
2. _____
3. _____

2 Student ID(s):

1. _____
2. _____
3. _____

3 Email Address(es):

1. _____
2. _____
3. _____

4 Other References (optional):

CSE166 Image Processing Homework 3

Nitay Joffe

10/19/2006

Written exercises

1. *GW, Problem 4.2.*
   Show that the Fourier transform and its inverse are linear processes.

$$
\begin{aligned}
af_1(x) + bf_2(x) &\longleftrightarrow F(u) \\
F(u) &= \frac{1}{M}\sum_{x=0}^{M-1}[af_1(x) + bf_2(x)]e^{-j2\pi ux/M} \\
&= a\frac{1}{M}\sum_{x=0}^{M-1} f_1(x)e^{-j2\pi ux/M} + b\frac{1}{M}\sum_{x=0}^{M-1} f_2(x)e^{-j2\pi ux/M} \\
&= aF_1(u) + bF_2(u)
\end{aligned}
$$

$$
\begin{aligned}
f(x) &\longleftrightarrow aF_1(u) + bF_2(u) \\
f(x) &= \sum_{u=0}^{M-1}[aF_1(u) + bF_2(u)]e^{j2\pi ux/M} \\
&= a\sum_{u=0}^{M-1} F_1(u)e^{j2\pi ux/M} + b\sum_{u=0}^{M-1} F_2(u)e^{j2\pi ux/M} \\
&= af_1(x) + bf_2(x)
\end{aligned}
$$

2. *GW, Problem 4.7.*
   What is the source of the nearly periodic bright points in the horizontal axis of the spectrum in Fig4.11(b)?

   The bright points are the high values of the sinc function, which is the result of running the discrete fourier transform on a box. The original image has many boxes and rectangles, so it is likely caused by them.

3. *GW, Problem 4.8.*
   Each of the spatial filters shown in Fig4.23 has a strong spike at the origin. Explain the source of these spikes.

   The spikes are caused by the $1 - ...$ portion of the equations that all the high pass filters have. The inverse discrete fourier transform of 1 is a delta function.

4. *GW, Problem 4.10.*
Show that if a filter transfer function H(u,v) is real and symmetric, then the corresponding spatial domain filter h(x,y) also is real and symmetric.

$$H(u,v) = H(-u,v) = H(u,-v)$$

$$h(-x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u,v)e^{-j2\pi ux/M}e^{j2\pi vy/M}$$

$$let \;\; a = -u, u = -a$$

$$h(-x,y) = \sum_{a=0}^{M-1}\sum_{v=0}^{N-1} H(-a,v)e^{j2\pi ax/M}e^{j2\pi vy/M}$$

$$h(-x,y) = \sum_{a=0}^{M-1}\sum_{v=0}^{N-1} H(a,v)e^{j2\pi ax/M}e^{j2\pi vy/M}$$

$$h(-x,y) = h(x,y)$$

The argument for the y case is the same, just using different variable.

5. *GW, Problem 4.15.*
The basic approach used to approximate a discrete derivative (Section 3.7) involves taking differences of the form $f(x+1,y) - f(x,y)$.
(a) Obtain the filter transfer function, H(u,v), for performing the equivalent process in the frequency domain.

$$\begin{aligned}
f(x,y) &\longleftrightarrow F(u,v) \\
f(x+1,y) &\longleftrightarrow F(u,v)e^{j2\pi u/M} \\
f(x,y+1) &\longleftrightarrow F(u,v)e^{j2\pi v/N} \\
g(x,y) &= f(x+1,y) - f(x,y) + f(x,y+1) - f(x,y) \\
g(x,y) &\longleftrightarrow G(u,v) \\
G(u,v) &\longleftrightarrow F(u,v)e^{j2\pi u/M} - F(u,v) + F(u,v)e^{j2\pi v/N} - F(u,v) \\
&= \left[e^{j2\pi u/M} - 1\right]F(u,v) + \left[e^{j2\pi v/N} - 1\right]F(u,v)
\end{aligned}$$

(b) Show that H(u,v) is a highpass filter.

When u or v are zero, the respective exponential of that part of the filter goes to one, which cancels out the -1 term. This shows that low frequencies are attenuated by this filter. As u or v increase to infinity, the exponential term grows unbounbed, which makes the overall term also grow since the -1 constant does not change. This shows that high frequencies are passed through. This behavior is exactly that of a high pass filter.

6. *GW, Problem 4.18.*
Can you think of a way to use the Fourier transform to compute (or partially compute) the magnitude of the gradient for use in image differentiation (see Section 3.7.3)? If your answer is yes, give a method to do it. If your answer is no, explain why.

   No. Computing the magnitude of the gradient of an image requires calculating squares and square roots. These calculations are not linear, and therefore a Fourier transform, which is a linear process, cannot be used to compute them.

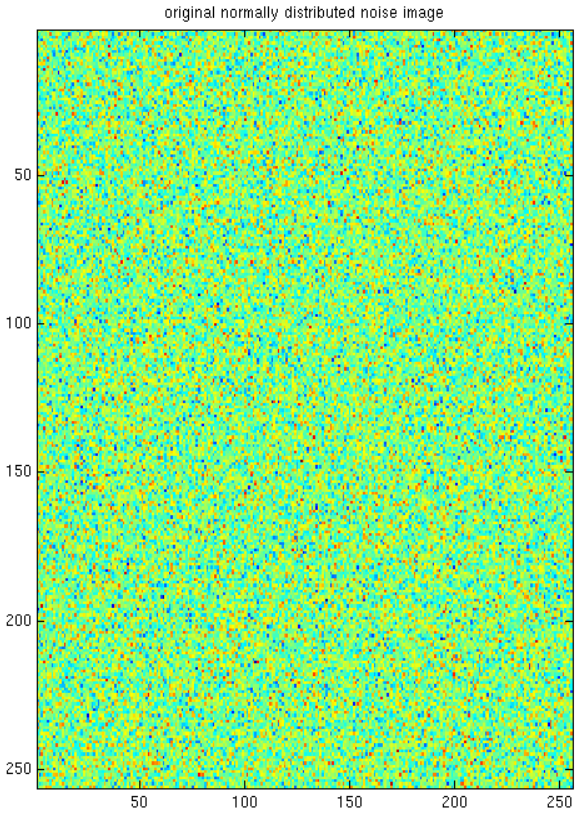
Matlab exercises

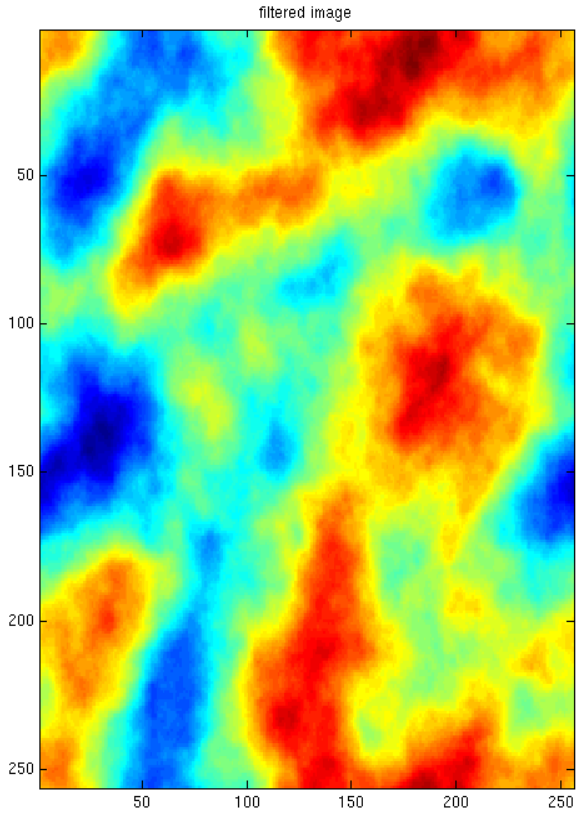
1. *Filtered Noise*

   (b) *Display the filtered image along with the original noise image. Quite remarkably, the filtered image should look like a "natural" texture, such as clouds or terrain. What does this suggest about the statistics of natural images vs. that of images of manmade objects?*

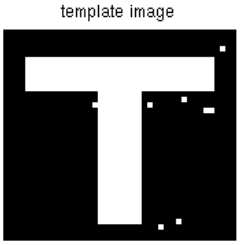
   The fact that the filtered normally distributed noise images look like suggests that the objects and behavior humans see and experience in our surroundings is caused by random processes, while the structures that people create tend to have very regular structure. Images of buildings have and cars have a defined shape, while images of natural things have much more random variation.

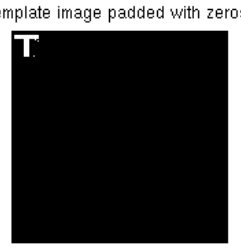
Problem 1 - Filtered Noise
Part A

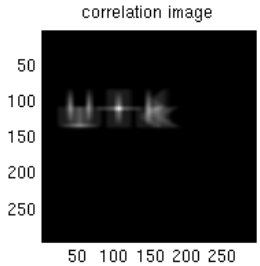


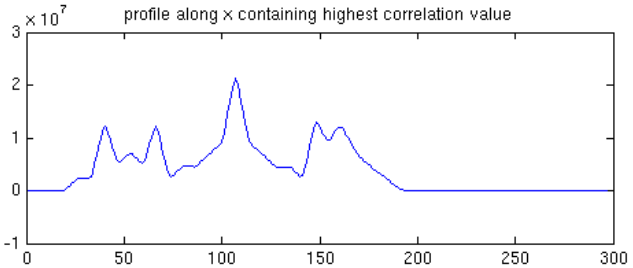


Problem 2 - Filtering in the Frequency Domain
Part A









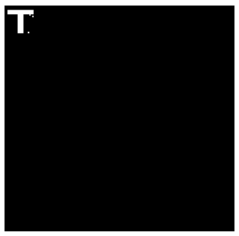
Problem 2 - Filtering in the Frequency Domain
Part B

original image

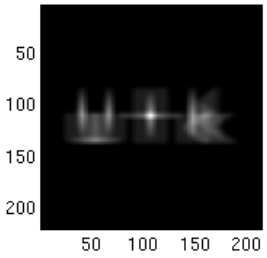
template image

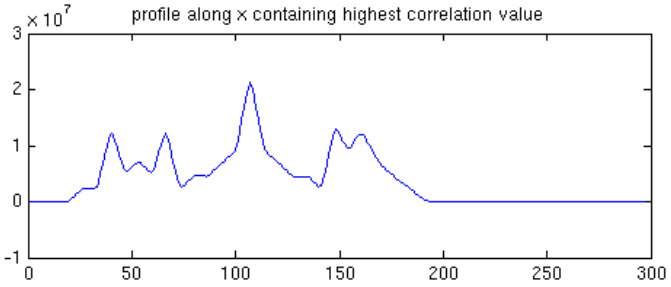
original image padded with zeros

template image padded with zeros



correlation image





```
% Author: <njoffe@ucsd.edu> Nitay Joffe
% Date: 10/19/2006
% Class: CSE 166 - Image Processing
% Homework: 3
% Problem: 1 - Filtered Noise
clear;

% (a) Consider the filter with Fourier transform H(u,v) = 1/(u^2+v^2) on the
%     interval [-127, 128] x [-127, 128]. This is known as a 1/f^2 transfer
%     function. Since it blows up at the origin, replace that value with zero.
%     Apply this filter to a 256x256 image of normally distributed random
%     noise (use _randn_). For practical reasons, it is best to perform this
%     operation in the frequency domain. Hint: you will need to use _meshgrid_,
%     _fft2_, _ifft2_, and _fftshift_. Also, due to numerical error, you will
%     need to use _real_ to look at the real part of the filtered image in the
%     spatial domain.
u_range = -127:128;
v_range = u_range;

% Evaluate filter function over a grid of values.
[u,v] = meshgrid(u_range,v_range);
filter_kernel_frequency = 1 ./ (u.^2 + v.^2);
filter_kernel_frequency(filter_kernel_frequency == inf) = 0;

% Generate image of normally distributed random noise.
noise_image = randn(256,256);
noise_image_frequency = fft2(noise_image);

% Run filter on noise image.
filtered_noise_image_frequency = filter_kernel_frequency.*noise_image_frequency;
filtered_noise_image_frequency_shifted=fftshift(filtered_noise_image_frequency);

% Use inverse fourier transform to get filtered noise image in spatial domain.
filtered_noise_image = ifft2(filtered_noise_image_frequency_shifted);
filtered_noise_image_real = real(filtered_noise_image);

% (b) Display the filtered image along with the original noise image. Quite
%     remarkably, the filtered image should look like a "natural" texture, such
%     as clouds or terrain. What does this suggest about the statistics of
%     natural images vs. that of images of manmade objects?
figure;
subplot(1,2,1);
imagesc(noise_image);
title('original normally distributed noise image');
subplot(1,2,2);
imagesc(filtered_noise_image_real);
title('filtered image');
```

```
% Author: <njoffe@ucsd.edu> Nitay Joffe
% Date: 10/19/2006
% Class: CSE 166 - Image Processing
% Homework: 3
% Problem: 2 - Filtering in the Frequency Domain
clear;

% (a) Write an m-file to reproduce Figure 4.41(a-f) using Frequency domain
%     filtering. Note: the Matlab command for the complex conjugate is conj.
original_image = double(imread('Fig4.41(a).jpg'));
template_image = double(imread('Fig4.41(b).jpg'));

% Compute minimum size of images padded with zeros.
[image_x_size, image_y_size] = size(original_image);
[template_x_size, template_y_size] = size(template_image);
padded_image_x_size = image_x_size + template_x_size - 1;
padded_image_y_size = image_y_size + template_y_size - 1;

% Pad images with zeros so they are the same size and frequencies don't corrupt
% the fourier transform computations.
image_padded = zeros(padded_image_x_size, padded_image_y_size);
image_padded(1:image_x_size,1:image_y_size) = original_image;
template_padded = zeros(padded_image_x_size, padded_image_y_size);
template_padded(1:template_x_size,1:template_y_size) = template_image;

% Run discrete fourier transform on images to get frequencies.
image_padded_frequency = fft2(image_padded);
template_padded_frequency = fft2(template_padded);

% Compute conjugate of template. Use it to get correlation in frequency domain.
template_padded_frequency_conjugate = conj(template_padded_frequency);
correlation_image_frequency = template_padded_frequency_conjugate.*image_padded_frequency;
correlation_image_real = real(ifft2(correlation_image_frequency));

% Find row containing highest value in correlation image, get profile of it.
[max_per_column, max_per_column_indices] = max(correlation_image_real);
[max_overall, max_overall_index_in_columns] = max(max_per_column);
highest_value_row = max_per_column_indices(max_overall_index_in_columns);
profile_highest_value_along_x = correlation_image_real(highest_value_row,:);

% Display results.
figure;
subplot(3,2,1);
imshow(original_image);
title('original image');
subplot(3,2,2);
imshow(template_image);
title('template image');
subplot(3,2,3);
imshow(image_padded);
title('original image padded with zeros');
subplot(3,2,4);
imshow(template_padded);
title('template image padded with zeros');
subplot(3,2,5);
imagesc(correlation_image_real);
% Set aspect ratio so that x and y increments are equal, and make plot box fit
% tightly around the data.
axis image;
title('correlation image');
subplot(3,2,6);
plot(profile_highest_value_along_x);
title('profile along x containing highest correlation value');

% (b) Repeat the previous step using operations in the spatial domain and show
%     that the results are the same. Hint: use conv2 and rot90.

%----- IMPORTANT -----
% NOTE: The code for reading and padding the images in the spatial domain is
%       _exactly_ the same, so rather than copying it I just used the results.

% Compute conjugate of original image for spatial correlation computation.
```

```
image_conjugate = conj(original_image);

% Need to rotate the template image 180 degrees since correlation does not flip
% the image. This is to undo the 180 degrees flip that convolution does.
template_rotated = rot90(rot90(template_image));

% Compute spatial correlation between images using 2D convolution.
correlation_image = conv2(image_conjugate, template_rotated, 'valid');

% Find row containing highest value in correlation image, get profile of it.
[max_per_column, max_per_column_indices] = max(correlation_image);
[max_overall, max_overall_index_in_columns] = max(max_per_column);
highest_value_row = max_per_column_indices(max_overall_index_in_columns);
profile_highest_value_along_x = correlation_image_real(highest_value_row,:);

% Display results.
figure;
subplot(3,2,1);
imshow(original_image);
title('original image');
subplot(3,2,2);
imshow(template_image);
title('template image');
subplot(3,2,3);
imshow(image_padded);
title('original image padded with zeros');
subplot(3,2,4);
imshow(template_padded);
title('template image padded with zeros');
subplot(3,2,5);
imagesc(correlation_image);
% Set aspect ratio so that x and y increments are equal, and make plot box fit
% tightly around the data.
axis image;
title('correlation image');
subplot(3,2,6);
plot(profile_highest_value_along_x);
title('profile along x containing highest correlation value');
```