Solutions to Take-Home Exam #1

1. [40 pts.] In this problem, you will create a Matlab script to investigate methods to suppress particular spectral features in an image. The image sensor has 64 pixels/cm and you've been told that a nearby RF source emitting radiation with a wavelength (1 period) of 0.05 cm (+/- 2%) that generates additive periodic image corruption with the same wavelength. In general, the relative position of the RF source and the image sensor is unknown, so we must assume that the signals can come in from any spatial direction. An example image is available on the course web site in exam_image.mat
   a. What kind of filter is desirable be used to suppress this signal, while preserving most other image features? Describe any desirable symmetries in this filter? What are the critical frequencies (in w_x and w_y) for this filter?

      Solution: To suppress a particular frequency, we would want to use a bandstop filter. For this problem, we desire circular symmetry (or in the absence of that, 8-fold symmetry). The sampling spacing is T = 1/64 cm and the RF corruption is has a period of 0.05 cm or a frequency of 20 cm^{-1}. The peak unaliased frequency is $1/2T$ = 32 cm^{-1} (corresponding to +/- p). The stopband should be at $w = (20/32)p$ =1.96 (+/- 2%). Naturally, the transition bands surrounding this stopband will depend on the size of the filter.

   b. Design a 7x7 filter function to suppress this corruption (there is more than one correct answer here, but please justify your chosen method). Use `imagesc` to show the frequency response. Include a `colorbar` and please make sure your axes are labeled and correct.

      Solution: The ideal filter will have response near zero in the region of the stopband and near one at other frequencies. Since a 7x7 support region is rather small, we will have to choose a filter with very large transitions. There are many possible solutions here, but I will give just one (however, many students designed better performing filters than this one). In HW#4, problem 3 we designed a nice 7x7 bandpass filter and we know that bandstop filters can be derived from bandpass filters. We have to do three things differently than we did for the homework: 1) we need to choose a different passband frequency, 2) we need to adjust the amplitude so that it peaks at 1 (rather than 0.5), and 3) we then subtract the filter from from 1 in the Fourier domain or from $d(n,m)$ to produce the bandstop filter. Thus we have

      $$H_{bs}(w) = 1 - H_{bp}(w) = 1 - \left(\frac{1}{2} - \frac{1}{2}\cos\left(2\frac{20}{32}w\right)\right) = \frac{1}{2} + \frac{1}{2}\cos\left(2\frac{20}{32}w\right).$$ See plots below.

      Important: About using $H_{bs}(w) = 1 - H_{bp}(w)$ or $h_{bs}(n,m) = d(n,m) - h_{bp}(n,m)$. This relationship works only if $H_{bp}(w)$ is nearly 1 in the passband. If you design a filter with too narrow of a passband, then it may need to be scaled so that the frequency response is close to 1 prior to applying the above relationships. E.g., you may need to do: $H_{bs}(w) = 1 - aH_{bp}(w)$, where a is chosen to make the bandpass filter peak near 1. This is also true for design of high-pass from low-pass filters, etc.

c. Apply this filter using `conv2` and create an image showing the result. Make sure you label the axes (in cm) and make sure that the filter had zero phase.

See plots below.

d. Create a separable 2D filter in which the 1D component has length 9. Display the frequency response.

Solution: For the 1D filter, we used the same approach we did for the 2D filter, but implemented it in 1D: $H_{bs}(w_x) = \frac{1}{2} + \frac{1}{2}\cos\left(2\frac{20}{32}w_x\right)$. The find the 2D response, we made a separable 2D filter. See plots below.

e. Apply this filter by applying the 1D convolution in two directions and create an image showing the result.

See plots below.

f. Create a 2D frequency domain filter (no restrictions on size) and display this function.

Solution: For this part we designed an ideal stop band filter in the Fourier domain. We needed to carefully scale our axes in order to make this work properly. Finally, we expanded our stopband since there was some bleeding of the artifact signal outside the +/- 2% tolerances. See plots below.

g. Apply the filter (in frequency domain) and create an image showing the result.

See plots below.

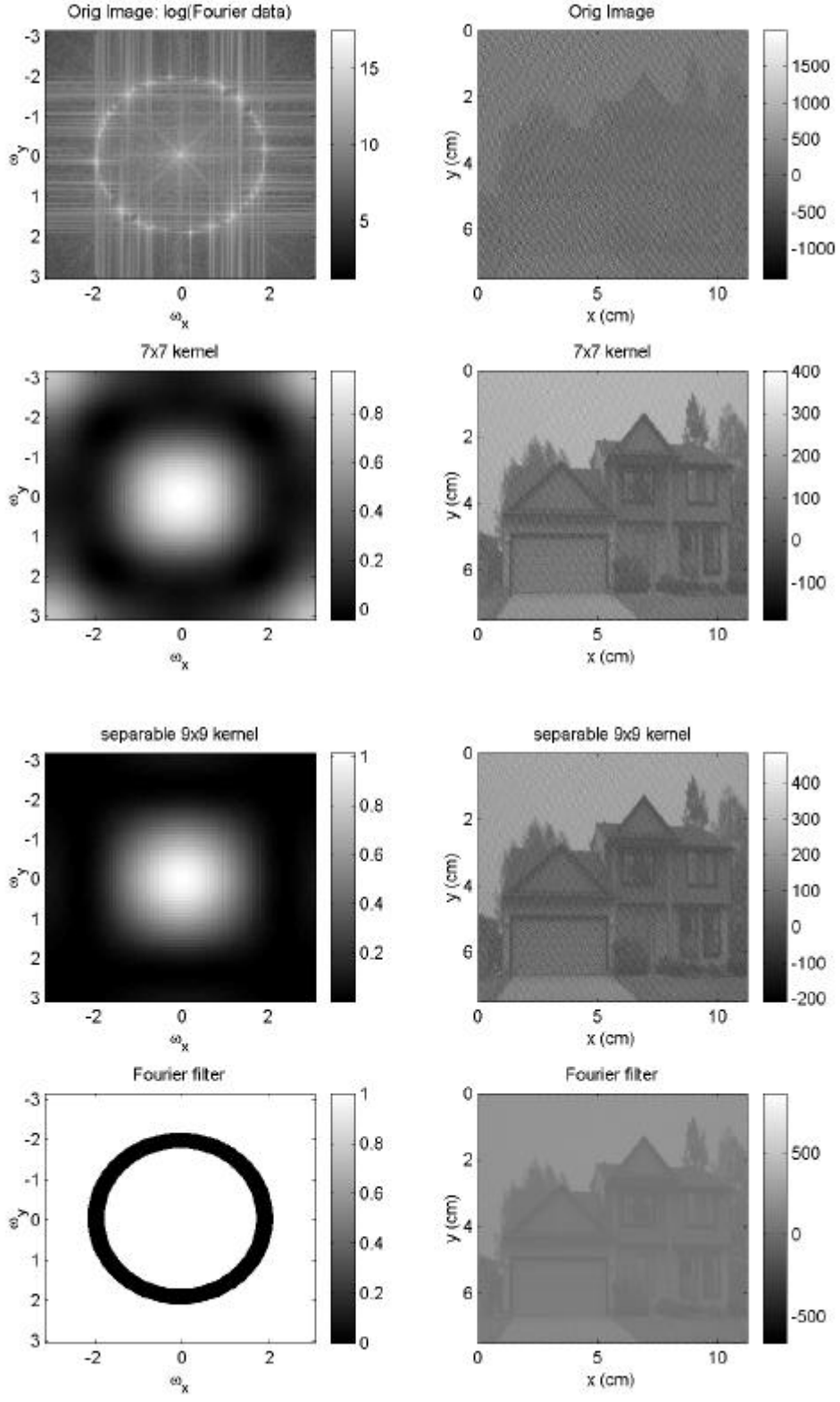
h. Comment on which methods are best and why. Comment on computational issues in addition to performance issues.

Solution: For this particular problem and my solutions, there appears to be little benefit to using the 2D filter over the 1D filter – this stems from the rather wide transition regions of the filter that I chose – for other filters, the 2D filter might be better. The computational advantages of the 1D separable filters would make these most desirable. The Fourier filter had the best suppression of the corruption, but was computationally the most expensive. It also had edge effects due to circular convolution. This is especially so given that the image size (480x720) did not lend itself to very fast FFT's.

Matlab code:

```
% Exam 1, Problem 1
load exam_image
[s1 s2] = size(exam_image);
y = [0:s1-1]./64;
x = [0:s2-1]./64;

[nn,mm] = ndgrid([-32:31]);
```

```
rr = abs(nn + i.*mm);
wwx = [-32:31]/32*pi;
wwy = wwx;
wwr = rr/32*pi;

figure(1)
subplot(221); imagesc(wwx,wwy,log(abs(fftshift(fft2(exam_image)))));
colormap gray; colorbar
title('Orig Image: log(Fourier data)');xlabel('\omega_x'); ylabel('\omega_y');

subplot(222); imagesc(x,y,exam_image); colorbar; colormap gray
title('Orig Image');xlabel('x (cm)'); ylabel('y (cm)');


% 2D filter
H2D = 0.5 + 0.5*cos(2*20/32*wwr);
% since H is real and symmetric, h should be real
h = real(fftshift(ifft2(fftshift(H2D))));

%truncate to 7x7
mask = (abs(nn) < 3.5).*(abs(mm) < 3.5);
ht = h.*mask;
HT = real(fftshift(fft2(fftshift(ht))));
subplot(223); imagesc(wwx,wwy,HT); colormap gray; colorbar
title('7x7 kernel');xlabel('\omega_x'); ylabel('\omega_y');

hkern = reshape(h(find(mask)),7,7);

outim = conv2(exam_image,hkern);
% select the right part to make it zero phase
outim = outim(4:3+s1, 4:3+s2);
subplot(224); imagesc(x,y,outim); colorbar; colormap gray
title('7x7 kernel');xlabel('x (cm)'); ylabel('y (cm)');

% 1D filter
H1D = 0.5 + 0.5*cos(2*20/32*wwx);
h1 = real(fftshift(ifft(fftshift(H1D))));
h1d = h1(33-4:33+4);
h1d2 = zeros([64 64]);
h1d2(33-4:33+4,33-4:33+4) = h1d'*h1d;
HT2 = real(fftshift(fft2(fftshift(h1d2))));
figure(2)
subplot(221); imagesc(wwx,wwy,HT2); colormap gray; colorbar
title('separable 9x9 kernel');xlabel('\omega_x'); ylabel('\omega_y');

% use conv2 to perform 2 1D convolutions
outim2 = conv2(h1d,h1d.',exam_image);
% select the right part to make it zero phase
outim2 = outim2(5:4+s1, 5:4+s2);

subplot(222); imagesc(x,y,outim2); colorbar; colormap gray
title('separable 9x9 kernel');xlabel('x (cm)'); ylabel('y (cm)');

% Fourier filter
[nn mm] = ndgrid(-s1/2:s1/2-1,-s2/2:s2/2-1);
wwr = 2*pi*abs(nn/s1 + i.*mm/s2);

tol = 0.05;
ffilt = (wwr < 20/32*pi*(1-tol)) + (wwr > 20/32*pi*(1+tol));
subplot(223); imagesc(wwx,wwy,ffilt); colormap gray; colorbar
title('Fourier filter');xlabel('\omega_x'); ylabel('\omega_y');

ftim = fftshift(fft2(exam_image));
outim3 = real(ifft2(fftshift(ftim.*ffilt)));
subplot(224); imagesc(x,y,outim3); colorbar; colormap gray
title('Fourier filter');xlabel('x (cm)'); ylabel('y (cm)');
```


Orig Image: log(Fourier data)
Orig Image
7x7 kernel
7x7 kernel
separable 9x9 kernel
separable 9x9 kernel
Fourier filter
Fourier filter
ωx
ωy
x (cm)
y (cm)

2. [20 pts.] You've been hired by NASA to work on data from a defective space telescope. The digital image sensor on this telescope suffered an unfortunate accident with an asteroid, which knocked out the decoding circuitry for ½ of all detector elements. You've discovered that a design engineer anticipated this possibility and created the sensor array in a particular way. The original sampling pattern was rectilinear with spacing T in both x and y directions. The sample locations that remain are those for which (n,m) are both even or for which (n,m) are both odd. The new sampling function can be written as:

$$p(x,y) = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} d(x-2nT, y-2mT) + \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} d(x-2nT-T, y-2mT-T)$$

a. Describe the spectrum of object sampled by the <u>original</u> array, given some object spectrum $X(u,v)$ (or $X(w_x,w_y)$, if you prefer). What are the conditions for prevention of aliasing?

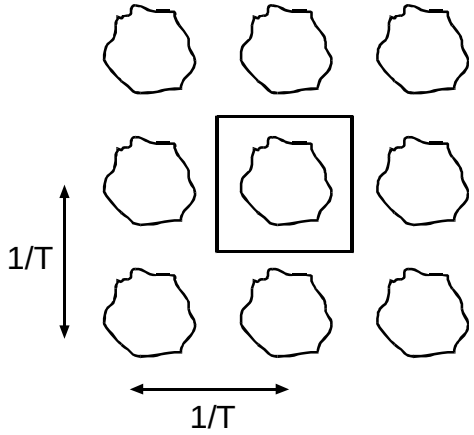
<u>Solution:</u> The original sampling function was:

$p(x,y) = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} d(x-nT, y-mT) = \frac{1}{T^2} comb\left(\frac{x}{T}, \frac{y}{T}\right)$. The sampled spectrum is:

$$F\{x(x,y)p(x,y)\} = X(u,v) ** comb(Tu,Tv) = X(u,v) ** \frac{1}{T^2}\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} d\left(u-\frac{n}{T}, v-\frac{n}{T}\right).$$

$$= \frac{1}{T^2}\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} X\left(u-\frac{n}{T}, v-\frac{n}{T}\right)$$

This is a replication of $X(u,v)$ in the u and v directions with spacing 1/T.



The conditions to prevent aliasing are that the original spectrum, $X(u,v)$, must be bandlimited to the region $|u| < \frac{1}{2T}$ and $|v| < \frac{1}{2T}$, corresponding to the square in the above figure.

b. Describe the spectrum of object sampled by the <u>defective</u> array. What are the conditions for prevention of aliasing?

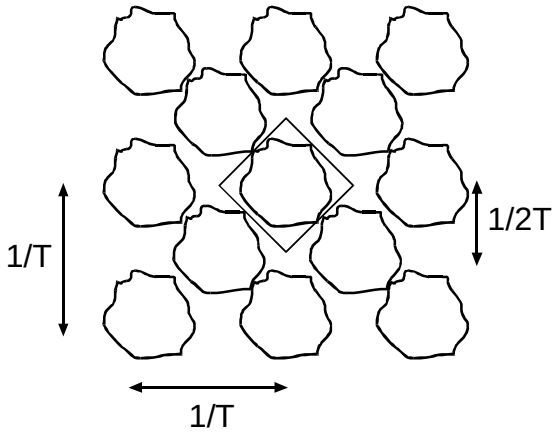
<u>Solution:</u> We can write the new sampling function as:
$p(x,y) = \frac{1}{4T^2}\left(comb\left(\frac{x}{2T},\frac{y}{2T}\right) + comb\left(\frac{x-T}{2T},\frac{y-T}{2T}\right)\right)$. The sampled spectrum is then:

$$F\{x(x,y)\,p(x,y)\} = X(u,v) ** \left(comb(2Tu,2Tv) + comb(2Tu,2Tv)\exp\left(-i2p\left(uT+vT\right)\right)\right)$$

$$= X(u,v) ** \frac{1}{4T^2}\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} d\left(u-\frac{n}{2T},v-\frac{n}{2T}\right)\left(1+(-1)^{n+m}\right)$$

$$= \frac{1}{4T^2}\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} X\left(u-\frac{n}{4T},v-\frac{n}{4T}\right)\left(1+(-1)^{n+m}\right)$$

This last statement says that an spectral island exists at spacing $1/2T$, but only when $n+m$ is even.



The conditions to prevent aliasing are that the original spectrum, $X(u,v)$, must be bandlimited to the region $|u+v| < \frac{1}{2\sqrt{2}T}$ and $|u-v| < \frac{1}{2\sqrt{2}T}$, corresponding to the square in the above figure. For a circular spectral pattern (which would is typical in optical telescopy), the maximum spectral pattern is reduced by $1/\sqrt{2}$.

c. Given the alternatives of losing, say, the odd elements in x or the odd elements in y, did the design engineer make a good decision?

<u>Solution:</u> Losing every other element with the lost elements along diagonal lines rather than horizontal or vertical lines appears to be a better solution. The peak frequency that aliases is reduced by $1/\sqrt{2}$ rather than 1/2 in a particular direction for the alternatives with no degradation in the other direction. The latter case would only be advantageous if we knew the object to be detected had a preferred direction of high frequency content. [Many people didn't understand my question, so unless you said something obviously wrong, you got full credit for this part.]

3. [20 pts.] Consider the following 2D discrete domain filters. For each, determine if the filter is linear and/or shift invariant. If the filter is linear, give its impulse response.

   a. $y(n,m) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} x(k-n, l-m)$

   Solution: This function is **linear**, but the response is reflected about the origin and is thus **space variant**. The impulse response is:

   $h(n,m;n',m') = S[d(n-n', m-m')] = \sum_{k=-1}^{1} \sum_{l=-1}^{1} d(k-n-n', l-m-m')$. Observe again, that this is not a function of $(n-n', m-m')$ and is thus not space invariant.

   b. $y(n,m) = \max_{k,l \in (-1,0,1)} x(n+k, m+l)$

   Solution: This function chooses a local maximum. It is **not linear** (consider the case where we scale the input by a where $a < 0$). This system is, however, **space invariant** – clearly shifting the input by (n', m') will lead to a similarly shifted output.

   c. $y(n,m) = x(n,m) + x(2n,m) + x(n,2m) + x(2n,2m)$

   Solution: This system is **linear**, but **space variant**. The impulse response is:

   $h(n,m;n',m') = S[d(n-n', m-m')]$

   $= d(n-n', m-m') + d(2n-n', m-m') + d(n-n', 2m-m') + d(2n-n', 2m-m')$.

   Observe again, that this is not a function of $(n-n', m-m')$ and is thus not space invariant.

   d. $y(n,m) = x(n,m) + x(n-2,m) + x(n,m-2) + x(n-2,m-2)$

   Solution: This function is both **linear** and **space invariant**. The impulse response is:

   $$h(n,m) = d(n,m) + d(n-2,m) + d(n,m-2) + d(n-2,m-2) = \begin{bmatrix} \underline{1} & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

   e. $y(n,m) = x(m,n)$

   Solution: This function is **linear**, but the response is transposed and is thus **space variant**. The impulse response is:

   $h(n,m;n',m') = S[d(n-n', m-m')] = d(n-m', m-n')$.

   f. $y(n,m) = x^2(n,m)$

   Solution: This function is **non-linear**, but **space invariant**.

4. [20 pts.] Sitting in Starbuck's, you overhead a conversation in which someone claimed that you could determine the 2D Fourier transform by using just a single 1D Fourier transform. Flabbergasted, you set out prove or disprove this assertion. You set up the a length *NM* 1D function $y(p)$ from a *N*x*M* 2D function $x(n,m)$:

   $y(p) = y(n+mN) = x(n,m)$ (column-wise stacking, if thinking in Matlab coordinates)

   Letting $Y(q)$ be the 1D DFT of $y(p)$ and $X(k,l)$ be the 2D DFT of $x(n,m)$, you define:

   $X'(k,l) = Y(kM+l) = Y(q)$ (row-wise unstacking).

   a. Determine the relationship between $X'(k,l)$ and $X(k,l)$. Assume the standard definitions of the 1D and 2D DFT's.

   Solution: Starting with $X(k,l) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)\exp\left(-i2p\left(\frac{nk}{N}+\frac{ml}{M}\right)\right)$ and

   $$Y(q) = \sum_{p=0}^{NM-1} y(p)\exp\left(-i2p\left(\frac{pq}{NM}\right)\right) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} y(n+mN)\exp\left(-i2p\left(\frac{(n+mN)q}{NM}\right)\right).$$

   Substituting for *X'* and *x* we get:

   $$X'(k,l) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)\exp\left(-i2p\left(\frac{(n+mN)(kM+l)}{NM}\right)\right)$$

   $$= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)\exp\left(-i2p\left(\frac{nk}{N}+\frac{ml}{M}+\frac{nl}{NM}+mk\right)\right)$$

   Observe that the *mk* term is always an integer and thus exp(-*i*2*pmk*) = 1, so this term can be discarded. Also observe that *nl* term represents a linear phase term in *n* (and has no effect on the FT in *m*). This results in a shift of the of the output image by an amount -*l*/*M* in the *k* direction. So:

   $$X'(k,l) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)\exp\left(-i2p\left(\frac{ml}{M}\right)\right)\exp\left(-i2p\left(\frac{n(k+l/M)}{N}\right)\right)$$

   $$= X(k+l/M,l).$$

   This appears as a shearing (*l* dependent shift in *k*) of the true Fourier data.

   b. For a 32x32 image size and image data defined by $x(n,m) = \mathrm{sinc}(n/4)\mathrm{sinc}(m/2)$, write a Matlab script to compare $X'(k,l)$ and $X(k,l)$. For both of these let the origin of the input and output system at the (17,17) location. Display the images and include real and imaginary parts if appropriate. Please attach the Matlab script and all images.

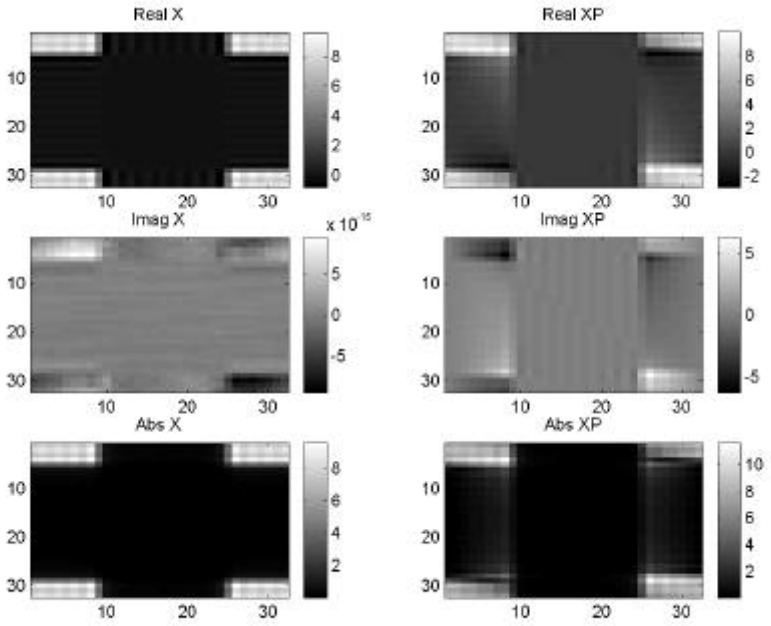
   Solution: We create a Matlab function:

```
function xp=new2dft(x)
% function to perform 2D fft by a 1D fft
[sx,sy]=size(x);
y = x(:);
Y=fft(y);
xp = reshape(Y,sy,sx).';
```

and then test it with:

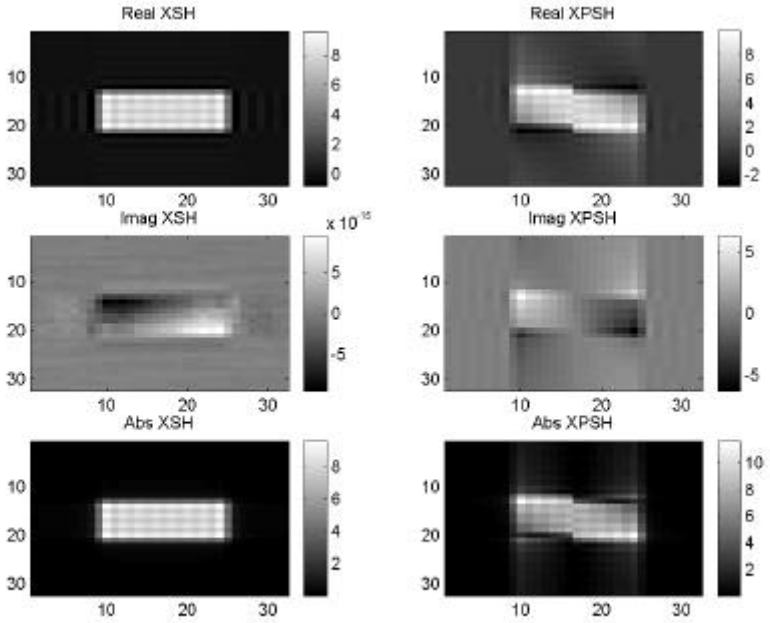
```
% testing new 2DFT
[nn,mm] = ndgrid([-16:15]);
x = sinc(nn./4).*sinc(mm./2);
% shift center
xsh = fftshift(x);
% look at 2d ft's without fftshift
X = fft2(xsh);
XP = new2dft(xsh);

figure(1)
subplot(321); imagesc(real(X)); colormap gray; colorbar; title 'Real X'
subplot(322); imagesc(real(XP)); colormap gray; colorbar; title 'Real XP'
subplot(323); imagesc(imag(X)); colormap gray; colorbar; title 'Imag X'
subplot(324); imagesc(imag(XP)); colormap gray; colorbar; title 'Imag XP'
subplot(325); imagesc(abs(X)); colormap gray; colorbar; title 'Abs X'
subplot(326); imagesc(abs(XP)); colormap gray; colorbar; title 'Abs XP'

% now add the post-ft fftshift
XSH = fftshift(X);
XPSH = fftshift(XP);

figure(2)
subplot(321); imagesc(real(XSH)); colormap gray; colorbar; title 'Real XSH'
subplot(322); imagesc(real(XPSH)); colormap gray; colorbar; title 'Real XPSH'
subplot(323); imagesc(imag(XSH)); colormap gray; colorbar; title 'Imag XSH'
subplot(324); imagesc(imag(XPSH)); colormap gray; colorbar; title 'Imag XPSH'
subplot(325); imagesc(abs(XSH)); colormap gray; colorbar; title 'Abs XSH'
subplot(326); imagesc(abs(XPSH)); colormap gray; colorbar; title 'Abs XPSH'
```

Resultant figures are (without fftshift):



Here we can see that as we move in the *l* direction (horizontal), we become shifted in *k* (vertically). (Matlab indices are transposed form what you expect.)

And with fftshift:



In this case the `fftshift` moves the shift discontinuity to the center of the image. The most common mistake made in this part was implementation of the `fftshift` on the 1D data vs. the 2D data (the correct approach).

c. How close to being correct was the coffee shop assertion?

Solution: The 1D equivalent is quite similar to the 2D FFT. All shifts are less than 1 pixel in size. When using with `fftshift`, however, the discontinuity in the shift from N-1 to 0 is move to center of the image creating a discontinuity in the image. We could create an alternate `fftshift` that would produced the expected image with no discontinuity at the origin and then the only effects would be the skewing and the non-realness of the new method.