Take-Home Exam #2 Solutions

Scores: Median = 87, Mean = 88.2, Std. Dev. = 7.9

1.

a. $\hat{s}_w^2 = 109$ vs. the theoretical of 100 – this seems pretty good.

b. Need to first calculate *Py* and from that calculate *Ps* (using $\hat{s}_w^2$).

c. $H(w_x, w_y) = \dfrac{\hat{P}_s(w_x, w_y)}{\hat{P}_s(w_x, w_y) + \hat{s}_w^2}$ and set *H*(0,0) =1 to make it mean preserving..

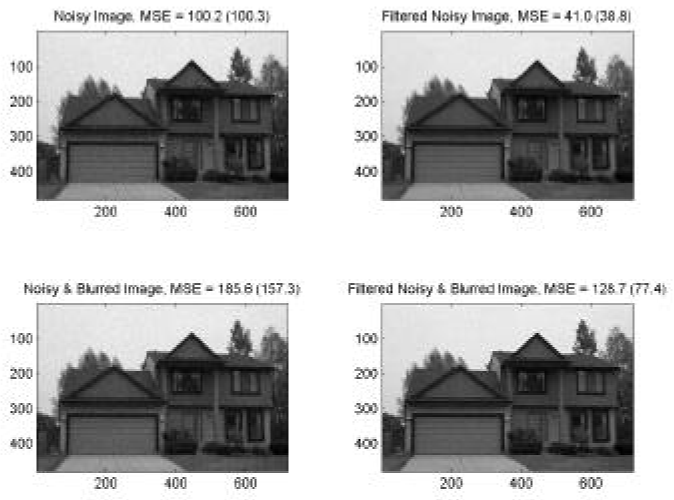
d. Just do a 7x7 ifft, but you need to use ifftshift for odd numbered matrices. This filter has the same symmetries as the periodogram (absolute value of the Fourier transform of a real object) – e.g. it is real and even. It looks 8-fold symmetric, but it is not exactly so.

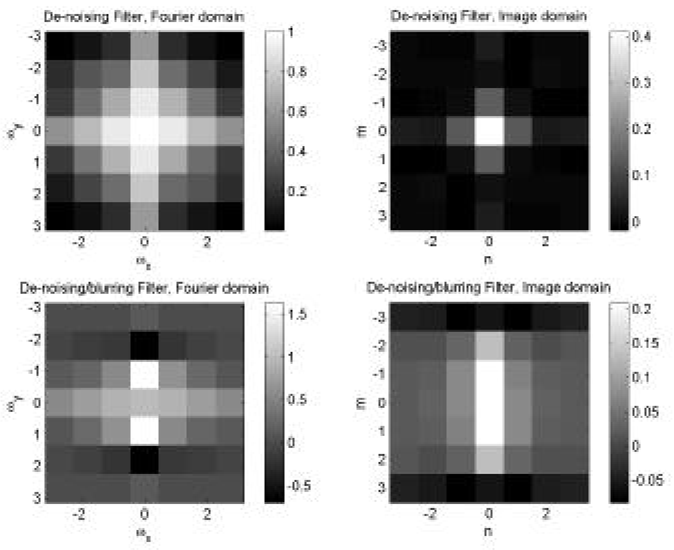
e. See images.

f. Just do a 7x7 ifft, but you need to use ifftshift for odd numbered matrices.

g. $H(w_x, w_y) = \dfrac{\hat{P}_s(w_x, w_y)B(w_x, w_y)}{\hat{P}_s(w_x, w_y)B^2(w_x, w_y) + \hat{s}_w^2}$ (*B* is real). Observe that this filter is trying to deblur in *y* and denoise in *x* – hence the assymetical shape – in fact, it does stronger denoising than the filter from part d. Because the 7x7 filter leaves a width 3 band of edge artifact, I also calculated the MSE's for the "valid" region (in parentheses). Using these figures, we can see that the MSE does get reduced substantially (~50%) by the deblurring/denoising filter.

Images:


Noisy Image, MSE = 100.2 (100.3)
Filtered Noisy Image, MSE = 41.0 (38.8)
Noisy & Blurred Image, MSE = 185.6 (157.3)
Filtered Noisy & Blurred Image, MSE = 128.7 (77.4)
100
200
300
400
200
400
600



Code:

```
% eecs 556, 2003, exam 2, problem 1
load hw7image;
s = hw7image;
w = 10*randn(size(hw7image));
y =  s + w;  % noisy measurement

s2 = conv2([1 1 1 1 1]'/5,s);
s2 = s2(3:end-2,:);
y2 = s2 + w;  % blurred and noisy measurement

%find mean, sd and normalize
subimno = y(1:100,1:300);
varn = var(subimno(:))

% calc periodogram
psz = 7; p2 = (psz-1)/2;
period = zeros([psz psz]);
np = zeros([psz psz]);
cnt = 0;
[l1 l2] = size(y);
for lp = 1:floor(l1/psz);
    for lp2 = 1:floor(l2/psz);
        subim = y((1+(lp-1)*psz):lp*psz,(1+(lp2-1)*psz):lp2*psz);
        period = period + abs(fftshift(fft2(subim))).^2;
        cnt = cnt + 1;
    end
end
Py = period./(cnt*psz*psz);
Pf = (Py - varn);
H1 = Pf./Py;
H1(p2+1,p2+1) = 1;  % fix mean
h1 = real(fftshift(ifft2(ifftshift(H1))))
shat = conv2(y,h1,'same');
% calculate MSE's for images and for 'valid' portion
scrop = s(p2+1:end-p2,p2+1:end-p2);
ycrop = y(p2+1:end-p2,p2+1:end-p2);
shcrop = shat(p2+1:end-p2,p2+1:end-p2);

mse1 = mean((s(:)-y(:)).^2);
```

```
mse1c = mean((scrop(:)-ycrop(:)).^2);
mse2 = mean((s(:)-shat(:)).^2);
mse2c = mean((scrop(:)-shcrop(:)).^2);
figure(1)
subplot(221); imagesc(y,[0 256]); axis('image'); colormap gray
str = sprintf('Noisy Image, MSE = %.1f (%.1f)',mse1,mse1c); title(str);
subplot(222); imagesc(shat,[0 256]); axis('image'); colormap gray
str = sprintf('Filtered Noisy Image, MSE = %.1f (%.1f)',mse2,mse2c); title(str);

% do deblurring too
bb = zeros([7 7]);
bb(2:6,4)=[1 1 1 1 1]'/5;
B = real(fftshift(fft2(ifftshift(bb))));

H2 = Pf.*B./(Pf.*(B.^2) + varn);
H2(p2+1,p2+1) = 1;
h2 = real(fftshift(ifft2(ifftshift(H2))))
shat2 = conv2(y2,h2,'same');
y2crop = y2(p2+1:end-p2,p2+1:end-p2);
sh2crop = shat2(p2+1:end-p2,p2+1:end-p2);

mse3 = mean((s(:)-y2(:)).^2);
mse3c = mean((scrop(:)-y2crop(:)).^2);
mse4 = mean((s(:)-shat2(:)).^2);
mse4c = mean((scrop(:)-sh2crop(:)).^2);

subplot(223); imagesc(y2,[0 256]); axis('image'); colormap gray
str = sprintf('Noisy & Blurred Image, MSE = %.1f (%.1f)',mse3,mse3c); title(str);
subplot(224); imagesc(shat2,[0 256]); axis('image'); colormap gray
str = sprintf('Filtered Noisy & Blurred Image, MSE = %.1f (%.1f)',mse4,mse4c);
title(str);

figure(2)
waxis = [-3:3]/7*2*pi; naxis = [-3:3];
subplot(221); imagesc(waxis,waxis,H1); axis('image'); colormap gray; colorbar
title('De-noising Filter, Fourier domain'); xlabel('\omega_x'); ylabel('\omega_y');
subplot(222); imagesc(naxis,naxis,h1); axis('image'); colormap gray;colorbar
title('De-noising Filter, Image domain');xlabel('n'); ylabel('m');
subplot(223); imagesc(waxis,waxis,H2); axis('image'); colormap gray;colorbar
title('De-noising/blurring Filter, Fourier domain'); xlabel('\omega_x');
ylabel('\omega_y');
subplot(224); imagesc(naxis,naxis,h2); axis('image'); colormap gray;colorbar
title('De-noising/blurring Filter, Image domain');xlabel('n'); ylabel('m');
```

2.

a. `imrotate` pivots around a center location of $(N+1)/2$, so it was necessary to pad out to 33x33 in order to rotate around the desired (17,17) location.

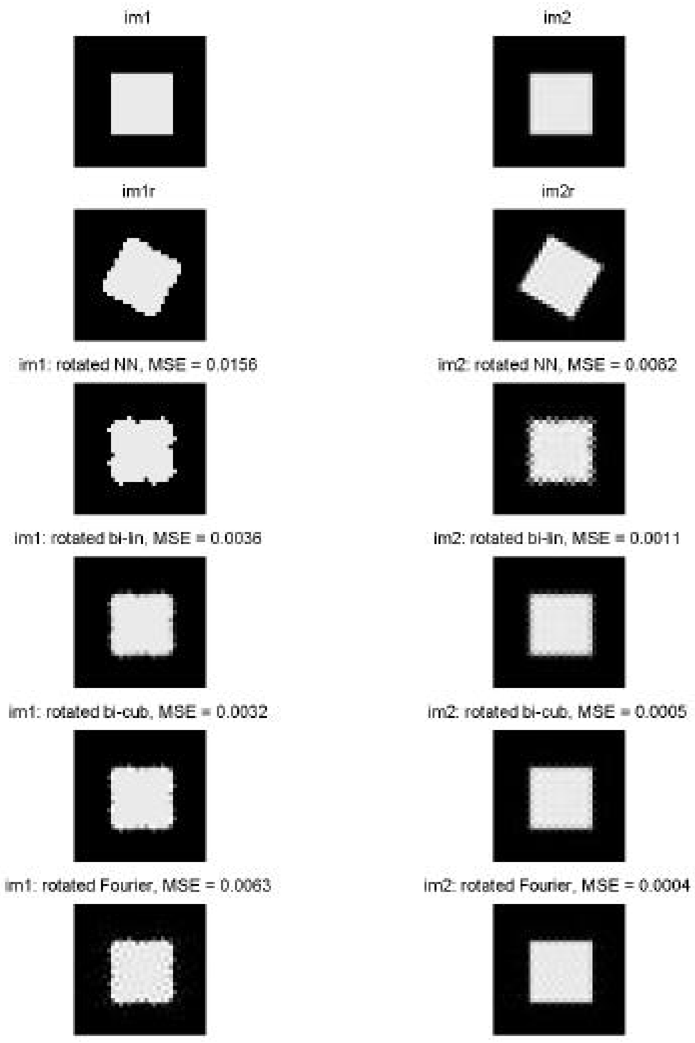
b. Observe that $\begin{bmatrix}1 & c\\0 & 1\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}x+cy\\y\end{bmatrix}$ is a shift of the x coordinate by an amount *cy*. This can be implemented in the Fourier domain by:

$F_{1d,row}^{-1}\left\{F_{1d,row}\{im(n,:)\}\exp\left(-i2p\,\frac{cnm}{N}\right)\right\}$, where *n* is the row index and *m* is the column index. Similarly for the y shear. The rotation is then $\text{Shear}_x\{\text{Shear}_y\{\text{Shear}_x\{\text{im}\}\}\}$.

c. See code.

d. Bi-cubic is best for im1 (and bi-linear is nearly as good) while the Fourier method is best for im2. This is because the im1 is not bandlimited and therefore the Fourier domain representation has substantial aliasing. Manipulating that representation (e.g. via Fourier domain interpolation or sinc interpolation) results in substantial artifacts in the image domain due to the long tails of the sinc interpolator. im2 is bandlimited and thus should work well for the Fourier methods (interestingly, is is now not space-limited, but it falls off quickly and isn't much of a problem. In practice, unless you know that your object is truly bandlimited (e.g. like MRI data), you are usually better with cubic interpolation. If interested, you can read futher in Unser M, *et al*. "Convolution-based interpolation for fast, high-quality rotation of images," *IEEE Trans. Image Processing*, 4:1371-1381, 1995.

Images:


im1
im2
im1r
im2r
im1: rotated NN, MSE = 0.0156
im2: rotated NN, MSE = 0.0062
im1: rotated bi-lin, MSE = 0.0036
im2: rotated bi-lin, MSE = 0.0011
im1: rotated bi-cub, MSE = 0.0032
im2: rotated bi-cub, MSE = 0.0005
im1: rotated Fourier, MSE = 0.0063
im2: rotated Fourier, MSE = 0.0004

Code:
```
% eecs 556, 2003, Exam #2, problem 2
N = 32;
wid = N/2-1;
[nn mm] = ndgrid([-N/2:N/2-1]);
uu = nn./N;
vv = mm./N;

im1 = (abs(nn) < wid/2).*(abs(mm) < wid/2);
im2 = real(fftshift(ifft2(ifftshift(sinc(wid.*uu).*sinc(wid.*vv)*(wid*wid)))));

theta = pi/6;
nr = cos(theta).*nn - sin(theta).*mm;
mr = cos(theta).*mm + sin(theta).*nn;
ur = cos(theta).*uu - sin(theta).*vv;
vr = cos(theta).*vv + sin(theta).*uu;
im1r = (abs(nr) < wid/2).*(abs(mr) < wid/2);
im2r = real(fftshift(ifft2(ifftshift(sinc(wid.*ur).*sinc(wid.*vr)*(wid*wid)))));

% pad with zeros to make center of rotation in imrotate at correct pixel
im1p = zeros([N+1 N+1]);
im1p(1:N,1:N) = im1r;
im2p = zeros([N+1 N+1]);
im2p(1:N,1:N) = im2r;
% imrotate
im1n = imrotate(im1p,theta/pi*180,'nearest','crop');
im2n = imrotate(im2p,theta/pi*180,'nearest','crop');
im1l = imrotate(im1p,theta/pi*180,'bilinear','crop');
im2l = imrotate(im2p,theta/pi*180,'bilinear','crop');
im1c = imrotate(im1p,theta/pi*180,'bicubic','crop');
im2c = imrotate(im2p,theta/pi*180,'bicubic','crop');
% crop back to original size
im1n = im1n(1:N,1:N);
im1l = im1l(1:N,1:N);
im1c = im1c(1:N,1:N);
im2l = im2l(1:N,1:N);
im2n = im2n(1:N,1:N);
im2c = im2c(1:N,1:N);
% do Fourier rotations
im1f = real(frot(im1r,theta));
im2f = real(frot(im2r,theta));
% MSE
mse1f = mean((im1(:)-im1f(:)).^2);
mse1n = mean((im1(:)-im1n(:)).^2);
mse1l = mean((im1(:)-im1l(:)).^2);
mse1c = mean((im1(:)-im1c(:)).^2);
mse2f = mean((im2(:)-im2f(:)).^2);
mse2n = mean((im2(:)-im2n(:)).^2);
mse2l = mean((im2(:)-im2l(:)).^2);
mse2c = mean((im2(:)-im2c(:)).^2);
% plots
figure(1)
subplot(621); imagesc(im1,[0 1.1]); colormap gray; axis('image'); title('im1');
subplot(623); imagesc(im1r,[0 1.1]); colormap gray; axis('image'); title('im1r');
subplot(625); imagesc(im1n,[0 1.1]); colormap gray; axis('image');
title(sprintf('im1: rotated NN, MSE = %.4f',mse1n))
subplot(627); imagesc(im1l,[0 1.1]); colormap gray; axis('image');
title(sprintf('im1: rotated bi-lin, MSE = %.4f',mse1l))
subplot(629); imagesc(im1c,[0 1.1]); colormap gray; axis('image');
title(sprintf('im1: rotated bi-cub, MSE = %.4f',mse1c))
subplot(6,2,11); imagesc(im1f,[0 1.1]); colormap gray; axis('image');
title(sprintf('im1: rotated Fourier, MSE = %.4f',mse1f))
subplot(622); imagesc(im2,[0 1.1]); colormap gray; axis('image'); title('im2');
subplot(624); imagesc(im2r,[0 1.1]); colormap gray; axis('image'); title('im2r');
subplot(626); imagesc(im2n,[0 1.1]); colormap gray; axis('image');
title(sprintf('im2: rotated NN, MSE = %.4f',mse2n))
subplot(628); imagesc(im2l,[0 1.1]); colormap gray; axis('image');
title(sprintf('im2: rotated bi-lin, MSE = %.4f',mse2l))
subplot(6,2,10); imagesc(im2c,[0 1.1]); colormap gray; axis('image');
title(sprintf('im2: rotated bi-cub, MSE = %.4f',mse2c))
subplot(6,2,12); imagesc(im2f,[0 1.1]); colormap gray; axis('image');
title(sprintf('im2: rotated Fourier, MSE = %.4f',mse2f))
```

3. In to do this problem, we must assume that w_1 and w_2 are jointly WSS. We can also quickly see that $R_{w1}(n,m) = R_{w2}(n,m) = d(n,m)$ and $R_{w1,w2}(n,m) = R_{w2,w1}(n,m) = 0.5d(n,m)$. We can then determine other correlations and power spectra.

   a.

$$R_f(n,m) = h_1(n,m) ** h_1^*(-n,-m)$$

$$R_v(n,m) = h_2(n,m) ** h_2^*(-n,-m)$$

$$R_{fv}(n,m) = 0.5\left[h_1(n,m) ** h_2^*(-n,-m)\right]$$

$$\begin{aligned} R_g(n,m) &= h_1(n,m) ** h_1^*(-n,-m) + h_2(n,m) ** h_2^*(-n,-m) \\ &\quad + 0.5\left[h_1(n,m) ** h_2^*(-n,-m)\right] + 0.5\left[h_2(n,m) ** h_1^*(-n,-m)\right] \end{aligned}$$

$$R_{fg}(n,m) = h_1(n,m) ** h_1^*(-n,-m) + 0.5\left[h_1(n,m) ** h_2^*(-n,-m)\right]$$

$$P_f(w_x,w_y) = \left|H_1(w_x,w_y)\right|^2$$

$$P_v(w_x,w_y) = \left|H_2(w_x,w_y)\right|^2$$

$$P_{fv}(w_x,w_y) = 0.5H_1(w_x,w_y)H_2^*(w_x,w_y)$$

$$\begin{aligned} P_g(w_x,w_y) &= \left|H_1(w_x,w_y)\right|^2 + \left|H_2(w_x,w_y)\right|^2 \\ &\quad + 0.5H_1(w_x,w_y)H_2^*(w_x,w_y) + 0.5H_1^*(w_x,w_y)H_2(w_x,w_y) \\ &= \left|H_1(w_x,w_y)\right|^2 + \left|H_2(w_x,w_y)\right|^2 + \mathrm{Re}\left\{H_1(w_x,w_y)H_2^*(w_x,w_y)\right\} \end{aligned}$$

$$P_{fg}(w_x,w_y) = \left|H_1(w_x,w_y)\right|^2 + 0.5H_1(w_x,w_y)H_2^*(w_x,w_y)$$

$$s_f^2 = R_f(0,0) = \sum_n \sum_m \left|h_1(n,m)\right|^2$$

$$s_v^2 = R_v(0,0) = \sum_n \sum_m \left|h_2(n,m)\right|^2$$

$$s_g^2 = R_g(0,0) = \sum_n \sum_m \left[\left|h_1(n,m)\right|^2 + \left|h_2(n,m)\right|^2 + h_1(n,m)h_2^*(n,m) + h_1^*(n,m)h_2(n,m)\right]$$

b. Derive the L.S.I. filter, $H(w_x, w_y)$, that you would apply to $g(n,m)$ to produce an estimate of *f*, $\hat{f}(n,m)$ that minimizes the MSE. Since *f* and *v* are not independent, we must use a different for than the one we usually use:

$$H(w_x, w_y) = \frac{P_{fg}(w_x, w_y)}{P_g(w_x, w_y)}$$

$$= \frac{\left|H_1(w_x, w_y)\right|^2 + 0.5 H_1(w_x, w_y) H_2^*(w_x, w_y)}{\left|H_1(w_x, w_y)\right|^2 + \left|H_2(w_x, w_y)\right|^2 + \mathrm{Re}\left\{H_1(w_x, w_y) H_2^*(w_x, w_y)\right\}}$$