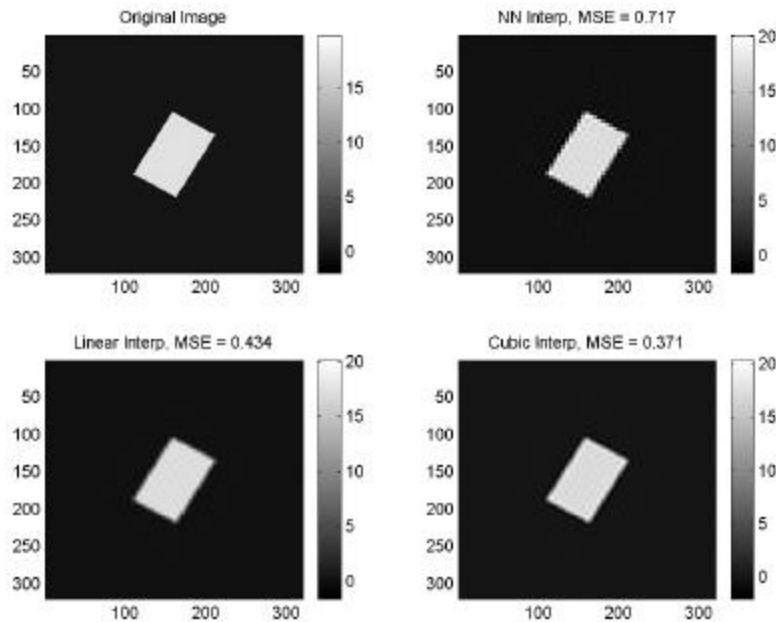


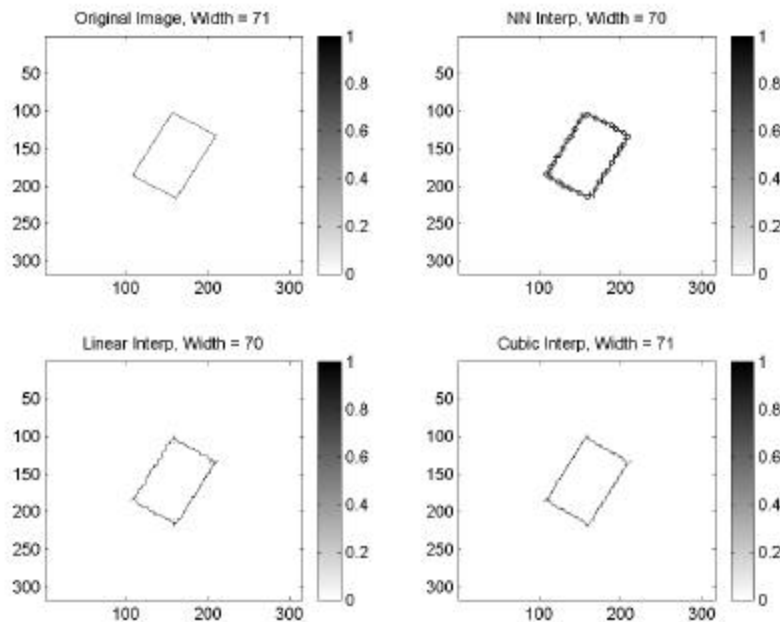
1. Interpolation and edge detection.

a. See fig:



b. Cubic is best in terms of MSE, but most expensive in terms of computational time. Based on a frequency response that is most similar to the rect function, we did expect cubic to be best in terms of MSE. Nearest neighbor is best in terms of computation, but is substantially worse in terms of MSE (and appearance). The linear interpolation might be a good compromise in that it is nearly as good as a cubic, but not as computationally expensive.

c. See figs:



For the edge detection, I lowered the threshold substantially to eliminate false edges in nearly all interpolation methods. I also needed to modify the edge thinning algorithm for the nearest neighbor case since it was finding very few edges – it turns out that the gradient was often maximal in pairs of identically valued point and neither was declared the edge – my modification made them both edge pixels.

In my implementation, the cubic and linear are nearly equivalent. The cubic edge is less jagged, but has some discontinuities.

- d. It was quite hard to find the edge in the nearest neighbor case, but looking at that gradient maps allowed us to select the edge pixels. For the full res, NN, linear, and cubic case, I measured object width as 71, 70, 70, and 71, respectively. The theoretical width was $W = \frac{1}{5.2} \frac{2}{\sqrt{3}} 320 = 71.06$ (the first fraction is the width of the short dimension of the rect, the second is the rotation correction term, the last is the conversion from continuous to discrete. For the interpolation methods tested, the cubic was most accurate for this measure.

Matlab code (main):

```
% pubpixel edge detection
[wx wy] = ndgrid([-160:159]);
thet = pi/3;
wvx = wx*cos(thet)+wy*sin(thet);
wvy = -wx*sin(thet)+wy*cos(thet);
X = sinc(wvx./5.2).*sinc(wvy./3.3);
xmini = real(fftshift(fft2(fftshift(X(129:192,129:192)))));
xfull = real(fftshift(fft2(fftshift(X))));

upfact = 5;
nsx = 64*upfact; nsy = 64*upfact;
im0 = zeros([nsx nsy]);
im0(1:5:nsx,1:5:nsy) = xmini;

% nearest neighbor
k1 = ones([5 5]);
im1 = conv2(im0,k1,'same');
mse1 = mean((im1(:)-xfull(:)).^2);
% linear
k21 = [ (1:5) (4:-1:1)]/5;
k2 = k21'*k21;
im2 = conv2(im0,k2,'same');
mse2 = mean((im2(:)-xfull(:)).^2);
% cubic
xx = [-1.8:.2:1.8];
k31 = (abs(xx) < 1).*(1-2.5.*xx.^2+1.5.*abs(xx).^3) + ...
      (1-(abs(xx) < 1)).*(2-4.*abs(xx)+2.5.*xx.^2-0.5.*abs(xx).^3);
k3 = k31'*k31;
im3 = conv2(im0,k3,'same');
mse3 = mean((im3(:)-xfull(:)).^2);

% display
figure(1)
subplot(221); imagesc(xfull); colormap gray; colorbar
title('Original Image');
subplot(222); imagesc(im1); colormap gray; colorbar
str = sprintf('NN Interp, MSE = %.3f',mse1); title(str);
subplot(223); imagesc(im2); colormap gray; colorbar
str = sprintf('Linear Interp, MSE = %.3f',mse2); title(str);
subplot(224); imagesc(im3); colormap gray; colorbar
```

```

str = sprintf('Cubic Interp, MSE = %.3f',mse3); title(str);

% edge map
efull = d_edge(xfull,0.3);
e1 = d_edge(im1,0.3);
e2 = d_edge(im2,0.3);
e3 = d_edge(im3,0.3);

% display
figure(2)
subplot(221); imagesc(efull); colormap(1-gray); colorbar
title('Original Image, Width = 71');
subplot(222); imagesc(e1); colormap(1-gray); colorbar
str = sprintf('NN Interp, Width = 70'); title(str);
subplot(223); imagesc(e2); colormap(1-gray); colorbar
str = sprintf('Linear Interp, Width = 70',mse2); title(str);
subplot(224); imagesc(e3); colormap(1-gray); colorbar
str = sprintf('Cubic Interp, Width = 71',mse3); title(str);

```

Matlab code (edge detection function):

```

function edthmap = d_edge(im,thresh);
% doug's sobel edge detector

soby = [-1 -2 -1; 0 0 0 ; 1 2 1];
sobx = soby';
gx = conv2(im,sobx,'valid');
gy = conv2(im,soby,'valid');
asg = abs(gx + i.*gy);
edmap = asg > thresh*(max(max(asg)));
% thinning - find pixels that are a maximum along x or a maximum along y
m1 = ((asg(2:end-1,:) >= asg(1:end-2,:)) .* (asg(2:end-1,:) >= asg(3:end,:)));
m2 = ((asg(:,2:end-1) >= asg(:,1:end-2)) .* (asg(:,2:end-1) >= asg(:,3:end)));
% now keep only the edges that are also a max
edthmap = ((m1(:,2:end-1) + m2(2:end-1,:))>0) .* edmap(2:end-1,2:end-1);

asgcrop = asg(2:end-1,2:end-1);
edloc = find(edthmap(152,:))
asgcrop(152,edloc)

```

2. Empirical and Adaptive Wiener Filtering. Specific parameter estimates and MSE values will depend your specific noise.

- $R_y(n, m) = R_s(n, m) + R_w(n, m) = \mathbf{s}_s^2 a \sqrt{n^2 + m^2} + 100 \mathbf{d}(n, m)$
- Empirical: $\hat{\mathbf{s}}_w^2 = 109.4$, theoretical: $\mathbf{s}_w^2 = 100$. Pretty close, but a bit over.
- After subtracting the mean from the image, we get

$$\hat{R}_y(0,0) = \hat{\mathbf{s}}_s^2 + \hat{\mathbf{s}}_w^2 = 6,295$$

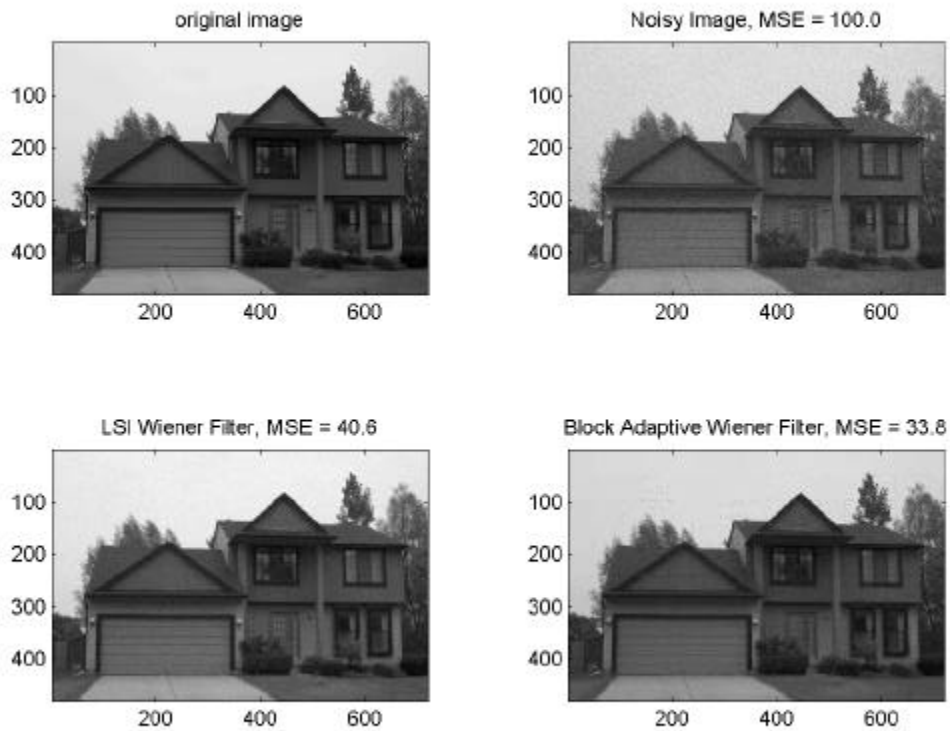
$$\hat{R}_y(0,1) = \hat{\mathbf{s}}_s^2 \hat{a} = 6,135$$

$$\hat{R}_y(0,1) = \hat{\mathbf{s}}_s^2 \hat{a} = 6,138$$

and using the estimate for $\hat{\mathbf{s}}_w^2$ from b., we can determine $\hat{\mathbf{s}}_s^2 = 6,186$, and from that we can determine $\hat{a} = 0.992$.

- See code and figures below. The MSE of the noisy image is 100 (\mathbf{s}_w^2) as expected and the MSE of the filtered image was 40.6.
- MSE of the block adaptive Wiener filtered image was 33.8.
- The block adaptive filtered image had the lowest MSE – this is better, in part, because the WSS assumption is very poor for the whole image, but it is somewhat

better for smaller blocks. There are some blocky artifacts that could be eliminated using overlapping blocks with windowing functions. Smaller block sizes would also help.



Matlab Code:

```
% wiener filtering problem
load hw7image;
[sx sy] = size(ho);
% adding noise
randn('seed',0);
hon = ho + 10*randn([sx sy]);

% autocorrelation and noise estimates
subim = hon(1:100,1:300);
sigw = var(subim(:))

mho = mean(hon(:));
ho2 = hon-mho;
r00 = mean(mean(ho2.^2))
r01 = mean(mean(ho2(1:end-1,:).*ho2(2:end,:)))
r10 = mean(mean(ho2(:,1:end-1).*ho2(:,2:end)))
r1 = (r10+r01)/2;
sigs = r00 - sigw
a = r1/sigs
[nx ny] = ndgrid([-sx/2:sx/2-1],[-sy/2:sy/2-1]);
r = abs(nx + i.*ny);
autocorrmod = sigs.*(a.^r);
PS = abs(fftshift(fft2(fftshift(autocorrmod))));

% derive and apply wiener filter
filt = PS./(PS + sigw);
fim = real(ifft2(fftshift(fftshift(fft2(ho2)).*filt))) + mho;
```

```

% work on 48x48 blocks
afiltim = zeros([sx sy]);
[nx ny] = ndgrid([-24:23],[-24:23]);
r = abs(nx + i.*ny);
for bx = 1:15
    for by = 1:10
        subim = hon((by-1).*48+[1:48],(bx-1).*48+[1:48]);
        % do filtering here
        mho = mean(subim(:));
        ho2 = subim-mho;
        r00 = mean(mean(ho2.^2));
        r01 = mean(mean(ho2(1:end-1,:).*ho2(2:end,:)));
        r10 = mean(mean(ho2(:,1:end-1).*ho2(:,2:end)));
        r1 = (r10+r01)/2;
        sigs = r00 - sigw;
        a = r1/sigs;
        autocorrmod = sigs.*(a.^r);
        PS = abs(fftshift(fft2(fftshift(autocorrmod))));
        filt = PS./(PS + sigw);
        subimfilt = real(ifft2(fftshift(fftshift(fft2(ho2)).*filt))) + mho;
        afiltim((by-1).*48+[1:48],(bx-1).*48+[1:48]) = subimfilt;
    end
end

% determine MSE's
msen = mean((ho(:)-hon(:)).^2);
msefilt = mean((ho(:)-fim(:)).^2);
msea = mean((ho(:)-afiltim(:)).^2);

subplot(221); imagesc(ho); colormap gray; axis('image');
title('original image')
subplot(222); imagesc(hon); colormap gray; axis('image');
str = sprintf('Noisy Image, MSE = %.1f',msen); title(str);
subplot(223); imagesc(fim); colormap gray; axis('image');
str = sprintf('LSI Wiener Filter, MSE = %.1f',msefilt); title(str);
subplot(224); imagesc(afiltim); colormap gray; axis('image');
str = sprintf('Block Adaptive Wiener Filter, MSE = %.1f',msea); title(str);

```

3. Noise and deblurring. In order to derive an L.S.I. filter, we must assume that the input image, f , is WSS with autocorrelation and power spectrum: $R_f(n, m)$ and $P_f(\mathbf{w}_x, \mathbf{w}_y)$, respectively. For convenience, we will also assume zero-mean. For the solutions below, we will simplify the problem by pulling the w to the other side of the blurring kernel:

$$\begin{aligned}
 g(n, m) &= f(n, m) * b(n, m) + w(n, m) * b(n, m) + v(n, m) \\
 &= f(n, m) * b(n, m) + u(n, m) \\
 \text{where } \dots u(n, m) &= w(n, m) * b(n, m) + v(n, m)
 \end{aligned}$$

- a. Using the deblurring-Wiener formulation that we derived in class

$$H(\mathbf{w}_x, \mathbf{w}_y) = \frac{P_f(\mathbf{w}_x, \mathbf{w}_y) B^*(\mathbf{w}_x, \mathbf{w}_y)}{P_f(\mathbf{w}_x, \mathbf{w}_y) |B(\mathbf{w}_x, \mathbf{w}_y)|^2 + P_u(\mathbf{w}_x, \mathbf{w}_y)}$$

where $B(\mathbf{w}_x, \mathbf{w}_y) = F\{b(n, m)\}$ and $P_u(\mathbf{w}_x, \mathbf{w}_y) = \mathbf{s}_w^2 |B(\mathbf{w}_x, \mathbf{w}_y)|^2 + \mathbf{s}_v^2$, so

$$H(\mathbf{w}_x, \mathbf{w}_y) = \frac{P_f(\mathbf{w}_x, \mathbf{w}_y) B^*(\mathbf{w}_x, \mathbf{w}_y)}{P_f(\mathbf{w}_x, \mathbf{w}_y) |B(\mathbf{w}_x, \mathbf{w}_y)|^2 + \mathbf{s}_w^2 |B(\mathbf{w}_x, \mathbf{w}_y)|^2 + \mathbf{s}_v^2}$$

- b. Assume that the variances for z_1 and z_2 are \mathbf{s}_{z_1} and \mathbf{s}_{z_2} .

$$R_w(n, m) = A^2 \mathbf{s}_{z_1}^2 \mathbf{d}(n, m)$$

$$R_v(n, m) = \mathbf{s}_{z_1}^2 [a(n, m) ** a^*(-n - m)] + B^2 \mathbf{s}_{z_2}^2 \mathbf{d}(n, m)$$

$$R_{wv}(n, m) = A \mathbf{s}_{z_1}^2 a^*(-n - m)$$

$$P_w(\mathbf{w}_x, \mathbf{w}_y) = A^2 \mathbf{s}_{z_1}^2$$

$$P_v(\mathbf{w}_x, \mathbf{w}_y) = \mathbf{s}_{z_1}^2 |A(\mathbf{w}_x, \mathbf{w}_y)|^2 + B^2 \mathbf{s}_{z_2}^2$$

$$P_{wv}(\mathbf{w}_x, \mathbf{w}_y) = A \mathbf{s}_{z_1}^2 A^*(\mathbf{w}_x, \mathbf{w}_y)$$

$$\mathbf{s}_w^2 = A^2 \mathbf{s}_{z_1}^2$$

$$\mathbf{s}_v^2 = \mathbf{s}_{z_1}^2 \sum_n \sum_m |a(n, m)|^2 + B^2 \mathbf{s}_{z_2}^2$$

- c. We need to recalculate $P_u(\mathbf{w}_x, \mathbf{w}_y)$ for

$$\begin{aligned} u(n, m) &= A \cdot z_1(n, m) ** b(n, m) + z_1(n, m) ** a(n, m) + B \cdot z_2(n, m) \\ &= z_1(n, m) ** [Ab(n, m) + a(n, m)] + B \cdot z_2(n, m) \end{aligned}$$

which yields

$$P_u(\mathbf{w}_x, \mathbf{w}_y) = \mathbf{s}_{z_1}^2 |AB(\mathbf{w}_x, \mathbf{w}_y) + A(\mathbf{w}_x, \mathbf{w}_y)|^2 + B^2 \mathbf{s}_{z_2}^2$$

and filter function

$$H(\mathbf{w}_x, \mathbf{w}_y) = \frac{P_f(\mathbf{w}_x, \mathbf{w}_y) B^*(\mathbf{w}_x, \mathbf{w}_y)}{P_f(\mathbf{w}_x, \mathbf{w}_y) |B(\mathbf{w}_x, \mathbf{w}_y)|^2 + \mathbf{s}_{z_1}^2 |AB(\mathbf{w}_x, \mathbf{w}_y) + A(\mathbf{w}_x, \mathbf{w}_y)|^2 + B^2 \mathbf{s}_{z_2}^2}$$