SOLUTIONS for HOMEWORK 2

1. **Median Filtering:**

   (a) This is salt and pepper noise.

   (b) Here is my routine for 5-point cross-shaped median filtering, but there are lots of other ways you could have written this:

```
function out = medfiltcross5(in)

[rows,cols] = size( in );
out(rows,cols) = 0;
tmp(rows,cols) = 0;
lx = zeros(5);
for x = 2:rows-1,
  for y = 2:cols-1,
    lx = [in(x,y), in(x-1,y), in(x,y-1), in(x+1,y), in(x,y+1)];
    out(x,y) = median(lx);
  end
end
```

   Here is my routine which does the median filtering using different size filters:

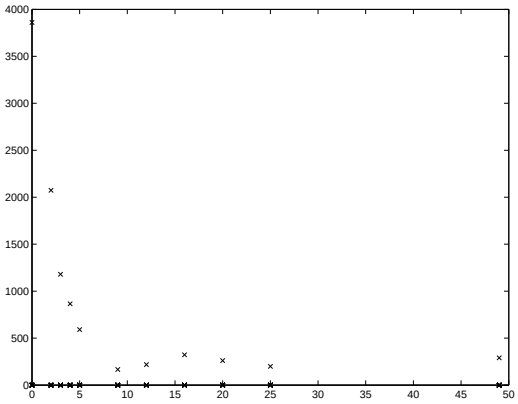
```
function [x,y] = allfilt(orig,nois)

     x = [0 2 3 4 5 9 12 16 20 25 49];
     y = zeros(11);
     y(1) = mymse(orig,nois);
     im = medfilt2(nois,[1 2]);      y(2) = mymse(orig,im);
     im = medfilt2(nois,[1 3]);      y(3) = mymse(orig,im);
     im = medfilt2(nois,[2 2]);      y(4) = mymse(orig,im);
     im = medfiltcross5(nois);       y(5) = mymse(orig,im);
     im = medfilt2(nois,[3 3]);      y(6) = mymse(orig,im);
     im = medfilt2(nois,[3 4]);      y(7) = mymse(orig,im);
     im = medfilt2(nois,[4 4]);      y(8) = mymse(orig,im);
     im = medfilt2(nois,[4 5]);      y(9) = mymse(orig,im);
     im = medfilt2(nois,[5 5]);      y(10) = mymse(orig,im);
     im = medfilt2(nois,[7 7]);      y(11) = mymse(orig,im);
```

   Here, mymse is my mean-squared error function:

```
function out = mymse(im1,im2)
temp = sum(sum((im1 - im2) .* (im1 - im2)));
```

```
[a,b] = size(im1);
c = a*b;
out = temp / c;
```



I call my function and plot the result as follows

```
[x,y] = allfilt(pep,noipep);
plot(x,y,'x')
```

and the result is on the right:

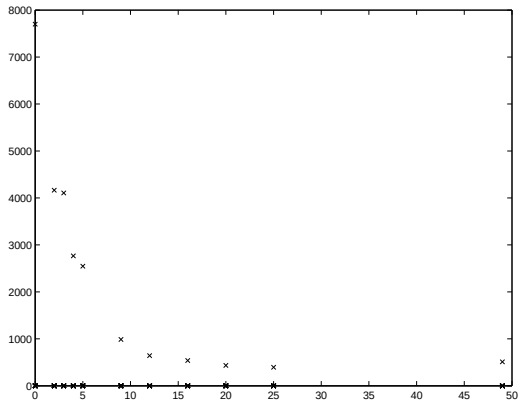
What we can ascertain from the plot is as follows. First, the starting mse is almost 4000, and it plunges for median filtering with the 1x2, 1x3, 2x2, cross-shape, and then reaches a minimum at the 9-point 3x3 filter. The noise level is 20%, so it is understandable that filtering with small filters such as 1x2 and 1x3 will not be sufficiently large. There will be a non-trivial fraction of the filtered pixels where these tiny median filters are getting a majority of pixels in the window being noisy.

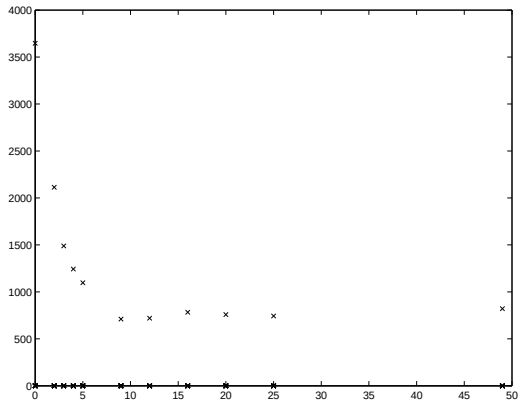
We reach a minimum at 3x3 because evidently this is large enough that the median operation is working consistently to clean up the noise. Looking out at 5x5, we see that this also performs well, but not as well as the 3x3, because evidently the 5x5 is larger than needed for this noise level.

In between the 3x3 and 5x5, the level rises a bit, presumably because Matlab is doing some kind of asymmetrical filtering here (3x4, for example) which is not centered on the pixel in question. So it doesn't do as well.

(c) We do the same thing for the n4 image and we get the following plot (below left). Here we see that the 5x5 outperforms the 3x3. Because twice as many pixels are corrupted, it makes sense that we will need a larger filter to remove the noise. So the location of the minimum has changed. Note also that the mean-squared error, after cleaning, is higher as well.

(d) The plot for the mandrill image is shown below right. Even though this image has the same noise level as n2, it has mean squared errors which are much higher. This is because this image has less spatial correlation in the regular (non-noisy) image components, and so the median filtering is going to change the image more.





2. **Filtering to Fix a Corrupted Image:**

(a) Horizontal 3-point median filter: Here we take the median of the corrupted center pixel (value 0) and the two good pixels on either side. When put in sorted order, the corrupted pixel will always be lowest. So the median equals whichever of the two adjacent neighbors is smaller. Let us call these neighbors X and Y, where $X \leq Y$. So the filtered value here is X.

(b) Horizontal 3-point mean filter: Here the corrupted center pixel is actually getting averaged in with the two adjacent good neighbors. This will be worse in MSE than (a). The filtered value is $(X + Y + 0)/3$. Ranking so far: $MSE(b) > MSE(a)$.

(c) Horizontal 3-point midpoint filter: The mid-point filter takes the average of the maximum and minimum of the set of 3 pixels. The minimum will be the corrupted center pixel. The maximum is Y. So this gives us a filtered value of $Y/2$. This will in general be worse MSE than (b). Ranking so far: $MSE(c) > MSE(b) > MSE(a)$

(d) 6-point sparse median filter, as shown. The output, corresponding to the center point of the filter, is the median of the 6 points in the filter marked with an X. Note that the pixel itself is not used in the median computation. This one doesn't use any corrupted pixels in the computations. So this is probably the best MSE of the ones we've seen for far. Ranking so far: $MSE(c) > MSE(b) > MSE(a) > MSE(d)$

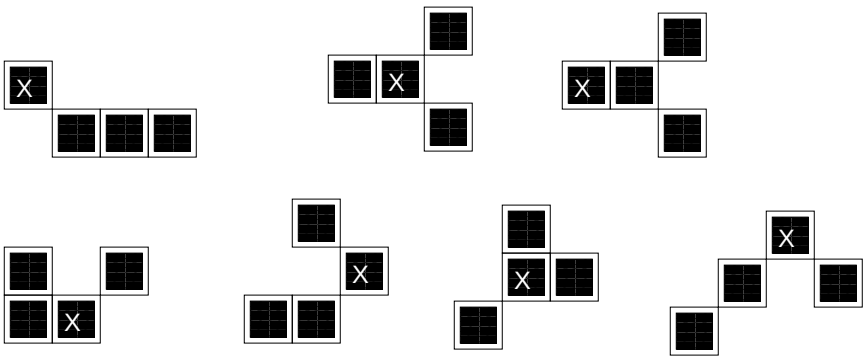
(e) 5-point plus-shaped median filter. This will have an output value of 0, because 3 of the corrupted pixels (value 0) get included in the median computation. So this is really bad. The corrupted picture will have no improvement.
Ranking so far: $MSE(e) > MSE(c) > MSE(b) > MSE(a) > MSE(d)$

(f) 3x3 square 67% order-statistic filter. The way this is defined, it comes out *exactly* the same as the 6-point sparse median filter. Ranking so far: $MSE(e) > MSE(c) > MSE(b) > MSE(a) > MSE(d) = MSE(f)$

(g) 3x3 weighted median filter which uses the following weighting. This is a stupid weighting. It will mean that the output will have value 0, and there is no improvement of the corrupted picture at all. Final Ranking: $MSE(g) = MSE(e) > MSE(c) > MSE(b) > MSE(a) > MSE(d) = MSE(f)$

3. **Binary Morphological Operators:**

(a) The smallest 8-connected structuring element B which does the job has four pixels. There are many sets B with 4 pixels which would do the job. Here are a few of them:

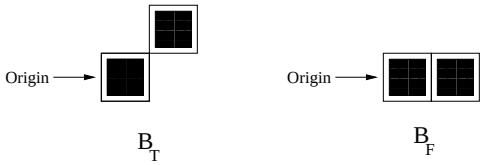


In each case, the origin of the set is marked with an X.

The smallest 4-connected set would have 5 pixels, so that is not the correct answer.

(b) The closing operation consists of dilation followed by erosion. If we use the structuring element B_T shown below, the little set A will pick up two new pixels (going diagonally upwards to the right) in the dilation step. But it will lose them again in the erosion step. So $A \bullet B = A$. All four rotated versions of this little structuring elements are equally valid answers.

If we use the structuring element B_F shown below, the little set A will pick up four new pixels (which are the four pixels immediately to the right of the four pixels already in A) in the dilation step. But it will only lose three of them in the erosion step. So $A \bullet B \neq A$. All four rotated versions of this little structuring elements are equally valid answers.



Because I didn't specify that the set has to be 8-connected, and I didn't specify that the origin of the set has to be located in one of the pixels in the set, there are other sets which are valid answers too. For example, a valid B_T is:

4. **Histograms and Averaging:**

The input histograms have only two values. Let us take them to be 0 and 180, just for simplicity (since there will be some divisions by 9 coming up). When we filter the left hand image, the output histogram will now have 4 values: 0 and 180 as well as 60 and 120. The latter two occur when the 3x3 filter is centered on the dark side of the edge (6 dark pixels and 3 light ones), and centered on the light side of the edge (3 dark pixels and 6 light ones):

$$\frac{6 \times 0 + 3 \times 180}{9} = 60$$

$$\frac{3 \times 0 + 6 \times 180}{9} = 120$$

The exact heights of the bins can be calculated as: 200x99 = 19800 pixels have value 0 (and the same for 180) and 200 pixels have value 60 (and the same for 120).

On the other hand, when we filter the checkerboard, we can have these 4 bin locations, as well as 2 others:

$$\frac{5 \times 0 + 4 \times 180}{9} = 80$$

$$\frac{4 \times 0 + 5 \times 180}{9} = 100$$

because the filter can be centered at one of the black/white corners, in which case there are 5 of one color and 4 of the other in the filter.

The exact heights of the bins can be calculated as follows: There are 64 squares total, so 32 dark squares. The squares are of size 25 by 25, so the interior portion of 23x23 will produce all dark output values.

$$32 \times 23 \times 23 = 16928$$

So there are 16928 dark pixels and the same for light pixels.

A pixel of value 60 occurs along the edges (but not the corners). There are 8 square edges vertically x 8 columns x 23 pixels in each x 2 because of horizontal edges. So there are 2944 pixels of value 60 (and the same for 120).

Lastly there are 64 corners, and each one has two pixels that will have a filtered output value of 80. So 128 pixels have a value 80, and 128 have a value 100.

As a check, (16928 + 2944 + 128) x 2 = 40,000.

NOTE: The heights of the bins that I gave here make the assumption that both images are padded outwards with the same type of pattern that they have on the inside. In particular, this means that the right-hand image continues with alternating blocks, and so the first and last row/column of blocks are no different from the interior blocks. If you make a different assumption about the image boundary, then you will get a different final answer, but we counted all such boundary assumptions as being correct.

5. **Median Filtering**

   The first one is 15 by 15. The second one is 31 by 31, and the 3rd one is 5x5. The fast way to see that is by looking at the rounding of the corners. A square median filter does not preserve a two-dimensional corner. In the third picture, the bars have corners that are very slightly rounded. In the first picture, they are more rounded. And the middle one has the biggest bites taken out of the corners.

   Another way to arrive at the answer is to look at the stripes in the bars themselves.