

## Solutions to Homework 2

### 1. Contrast Enhancement

(a) After rescaling so the values range from 0 to 1, we can remap the values using, for example, square and square root functions. As expected, with a square root function, more dynamic range is allocated to the dark parts of the image, and they genuinely look better. Particularly on the right side, we can resolve details that we couldn't before. However, the upper left corner, where things were bright initially, looks worse than before. It now looks overexposed; however, it is a small part of the image. If the region of interest of the image is considered to be the butterfly near the center, the power values less than one are helpful for enhancing the contrast in the dark region that includes the region of interest.

The squaring operation has the opposite effect. The upper left corner now looks better, but everything else looks much worse. Any power value greater than one will make the region of interest, and the majority of the image, look worse.

(b) Global histogram equalization:

Much of the image is dark, so global histogram equalization remaps dark values to be larger. The upper left corner, which was brighter than the rest of the picture, ends up looking washed out. The dark parts end up looking somewhat better. The effect is similar to the square root remapping. Again, this is probably considered a successful enhancement for this particular image, since the majority of the image, including the region of interest, looks better.

(c) Block-adaptive histogram equalization

The block boundaries are least apparent in the lower middle portion of the image, and in the upper left corner. In fact, the block boundaries are not apparent there at all. This must be because adjacent blocks have approximately the same histogram, so the remapping functions match across the boundary.

The block boundaries are apparent in places where the neighboring blocks have very different starting histograms, so the remapping functions are very different. Note that it doesn't matter in general whether a block and its neighbor are high contrast or low contrast— simply if they are *different* in their histograms, then they will produce different remapping functions, hence block boundaries. Two features where block artifacts are noticed are: (1) since the upper left corner had different illumination from the rest of the image, the block boundaries are obvious along that diagonal strip where the illumination changed. (2) the spiky plants located in the middle right of the picture, and in the lower left corner, are brighter than their surroundings, and are large enough to have a significant impact on the overall histograms of the blocks that contain them.

## 2. 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) = mse(orig,nois);
im = medfilt2(nois,[1 2]);    y(2) = mse(orig,im);
im = medfilt2(nois,[1 3]);    y(3) = mse(orig,im);
im = medfilt2(nois,[2 2]);    y(4) = mse(orig,im);
im = medfiltcross5(nois);    y(5) = mse(orig,im);
im = medfilt2(nois,[3 3]);    y(6) = mse(orig,im);
im = medfilt2(nois,[3 4]);    y(7) = mse(orig,im);
im = medfilt2(nois,[4 4]);    y(8) = mse(orig,im);
im = medfilt2(nois,[4 5]);    y(9) = mse(orig,im);
im = medfilt2(nois,[5 5]);    y(10) = mse(orig,im);
im = medfilt2(nois,[7 7]);    y(11) = mse(orig,im);
```

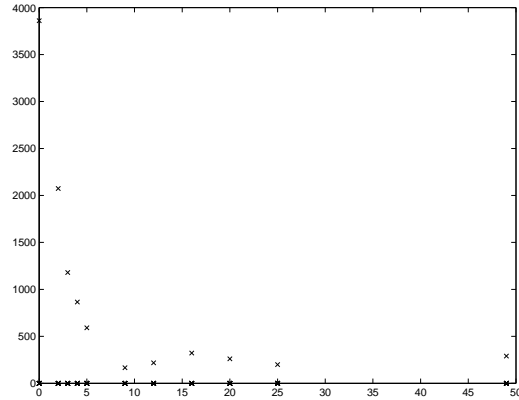
Here, mse is my mean-squared error function:

```
function out = mse(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(peg,noipep);  
plot(x,y,'x')
```

and the result looks like this:

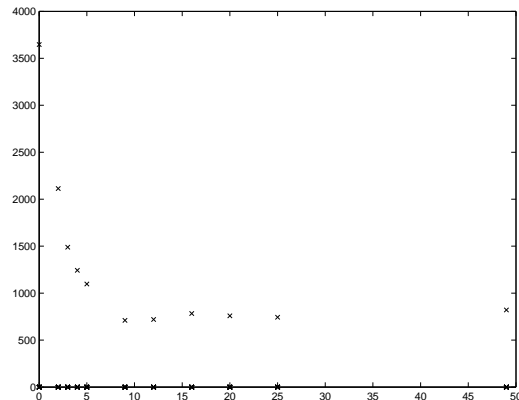
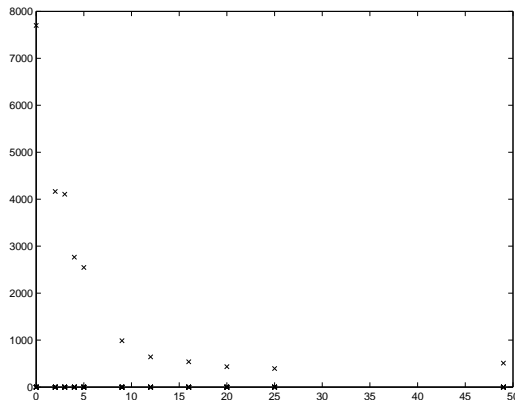


What we can ascertain from the plot is as follows. First, the starting mse is almost 4000, and it plunges for median filtering with the 1x2, 1x3, 2x2, cross-shape, and then reaches a minimum at the 9-point 3x3 filter. The noise level is 20%, so it is understandable that filtering with small filters such 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 above 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.

### 3. Unsharp Masking:

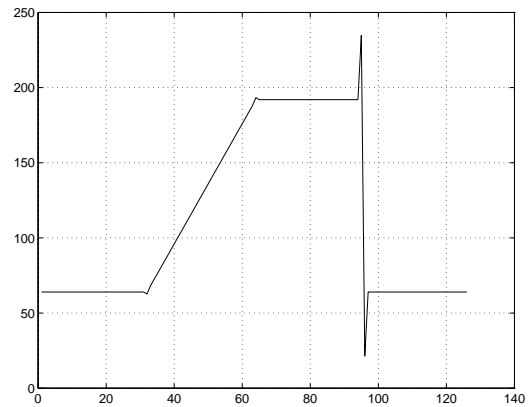
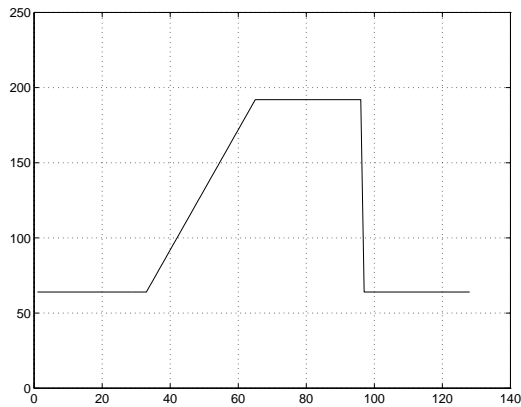
(a) maskB is an identity filter that is the same size as maskA. Filtering with maskB just gives you back the original image. maskC is a highpass filter. MaskD is a highboost filter. Filtering with maskD results in an image which is the sum of the original image plus some weight times a highpass version of it. So this will have the result of boosting the highpass content, thereby sharpening the edges.

(b) We try first a simple unweighted  $3 \times 3$  averaging filter as our lowpass mask, with a fairly large extra weight (1) given to the highpass portion:

```
mask = 1/9 * ones(3);  
tmp = unsharp(tst,mask,1);  
plot(tmp(64,:))
```

(Some people tried VERY large weights, like 5 or 10, in which case the 8-bit range of the image is exceeded, and you have to deal with the truncation and rescaling issues. But there is no need to use a weight that large. Even a weight of 1 produces an enormous edge enhancement effect, as can be seen from the cross section).

After unsharp masking, the test image shows substantial undershoots and overshoots at the step edge, and there are very slight overshoots and undershoots at the beginning and end of the ramp. A cross-section through the test image is shown below left (`plot(tst(64,:))`), and the cross-section for the edge-sharpened version is shown below right:

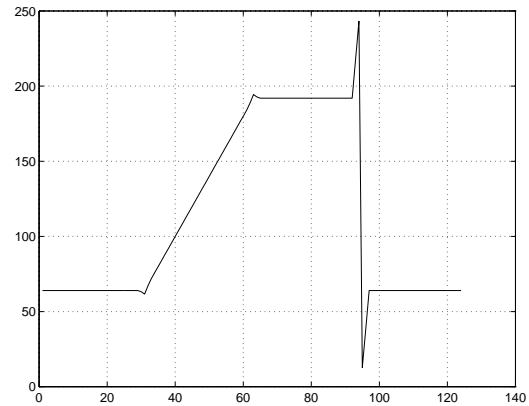
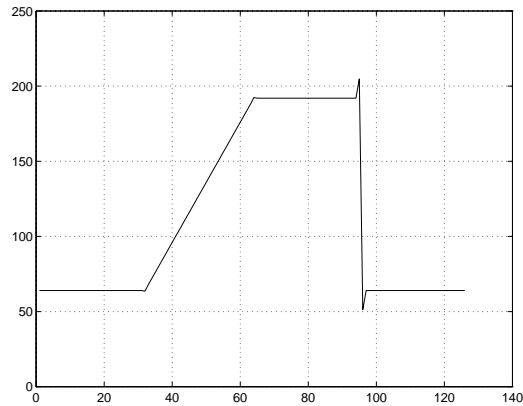


If the extra amount of highpass added in is not so much, the overshoots are less pronounced relative to the ones in the previous plot (graph shown below left):

```
>> tmp = unsharp(tst,mask,0.3);  
>> plot(tmp(64,:))
```

If the lowpass filter is larger, then the overshoots get wider (graph shown below right):

```
>> mask = 1/25 * ones(5);  
>> tmp = unsharp(tst,mask,1);  
>> plot(tmp(64,:))
```



Lastly, if the lowpass filter is strongly center-weighted, then the lowpass filtered image will be close to the original image. In this case, the highpass filtered image (which is the original minus the lowpass filtered) will be close to zero. So we expect that strong center-weighting will produce an unsharp masked image that doesn't look very different from the original.

```
>> mask = 1/18 * [1 1 1; 1 10 1; 1 1 1];  
>> tmp = unsharp(tst,mask,1);
```

This is in fact the case; the overshoots are small. Basically, the various parameters one can play with are changing the heights and widths of the overshoots.

(c) When applying the unsharp masking on the x-ray image, if you use the 3x3 mask with a low weight (1 or 2) you can get edge sharpening without having to deal with truncation or rescaling, since the output image does not exceed the [0,255] range.

With some masks and weights, the range is exceeded. Truncating looks better than rescaling, since for many combinations of masks and weights, a very tiny percentage of pixels exceed 255. The bones are easier to distinguish since the edges appear enhanced. At the same time, however, the unsharp masking increases the noise level. With the right combination of masks and weights, the numbers written on the vertebrae (2-7) become visible.

Trends that can be noticed are: A higher value to the highpass weight results in sharper edges. More boosting to the edges also results in less contrast in the flat (uniform) parts of the image, since the rescaling becomes more severe to compensate for the boosted edges. As the spatial extent of the filter mask is increased (from, say,  $3 \times 3$  to  $5 \times 5$  to  $7 \times 7$ ), the edges seem to get thicker. Increased center weighting in the filter mask reduces the gains in edge sharpness—most people who tried this seemed to agree that center weighted masks didn't produce as nice results as flat (unweighted) masks.