Homework Assignment 2 - SOLUTIONS
Due Monday, September 21, 2015

Notes: Please email me your solutions for these problems (in order) as a single Word or PDF document. If you do a problem on paper by hand, please scan it in and paste it into the document (although I would prefer it typed!).

1. (25 pts) A 50x70 image has 3-bit pixels. Its histogram looks like a ramp as shown below. The counts in the histogram follow the formula \( H(r) = kr \), where \( k \) is a constant.

- a) Determine the value of \( k \).
- b) Compute the mean and standard deviation of this image from the histogram.
- c) Compute the transformation function \( s = T(r) \) that will equalize the histogram.
- d) Compute the histogram \( H(s) \) of the resulting image, if it were transformed by \( T \).
- e) Compute the mean and standard deviation of the resulting image.

(Note - a spreadsheet like Excel might be helpful for this problem.)

Solution:

(a) The image has 50x70 = 3500 pixels. Each pixel has values from 0..7. The counts follow the relationship \( H(r) = kr \). To determine \( k \), we know that the sum of the values of \( H(r) \) for all \( r \) must sum to 3500:

\[
\sum_{r=0}^{7} kr = 3500
\]

\[k(0+1+2+\ldots+7) = 3500\]

You can directly evaluate the sum in parentheses, or use the fact that \( 1+2+\ldots+n = n(n+1)/2 \)

Therefore \( k(7)(8)/2 = 3500 \), or \( 28k = 3500 \). Or \( k = 3500/28 = 125 \)

(b) We compute the histogram using \( H(r) = 125r \) in an Excel spreadsheet. We compute the probability density function using \( p(r) = H(r)/3500 \):
<table>
<thead>
<tr>
<th>$r$</th>
<th>$H(r)$</th>
<th>$p(r)$</th>
<th>$r \cdot p(r)$</th>
<th>$(r - \mu)^2 \cdot p(r)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>0.0357</td>
<td>4.225</td>
<td>0.000107</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>0.0714</td>
<td>14.28</td>
<td>0.000174</td>
</tr>
<tr>
<td>3</td>
<td>375</td>
<td>0.1071</td>
<td>37.505</td>
<td>0.000172</td>
</tr>
<tr>
<td>4</td>
<td>500</td>
<td>0.1429</td>
<td>21.07</td>
<td>0.000214</td>
</tr>
<tr>
<td>5</td>
<td>625</td>
<td>0.1786</td>
<td>34.8025</td>
<td>0.000349</td>
</tr>
<tr>
<td>6</td>
<td>750</td>
<td>0.2143</td>
<td>15.855</td>
<td>0.000218</td>
</tr>
<tr>
<td>7</td>
<td>875</td>
<td>0.25</td>
<td>21.875</td>
<td>0</td>
</tr>
</tbody>
</table>

sum 3500 1

\[ \text{mean} = \sum_{r=0}^{7} r \cdot p_r(r) = 5. \]

\[ \text{std dev} = \sqrt{\sum_{r=0}^{7} (r - \mu)^2 \cdot p_r(r) = 1.73.} \]
(c) The transformation function is \( s = T(r) = 7F(r) \) where \( F(r) \) is the cumulative probability distribution function (we also round to the nearest integer).

\[
\begin{array}{cccccc}
 r & H(r) & p(r) & F(r) & T(r) \\
 0 & 0 & 0 & 0 & 0 \\
 1 & 125 & 0.0357 & 0.0357 & 0 \\
 2 & 250 & 0.0714 & 0.1071 & 1 \\
 3 & 375 & 0.1071 & 0.2143 & 2 \\
 4 & 500 & 0.1429 & 0.3571 & 3 \\
 5 & 625 & 0.1786 & 0.5357 & 4 \\
 6 & 750 & 0.2143 & 0.75 & 5 \\
 7 & 875 & 0.25 & 1 & 7 \\
\end{array}
\]

sum 3500 1

Note – Some of the values of \( 7^*F(r) \) are “xxx.”5, so they may round off either up or down.

(d) The histogram of the transformed image is given below (we use the Excel “sumif” formula). Note that the counts in the new histogram still sum to 3500.

\[
\begin{array}{ccccccccc}
 r & H(r) & p(r) & F(r) & T(r) & s & H(s) \\
 0 & 0 & 0 & 0 & 0 & 0 & 125 \\
 1 & 125 & 0.0357 & 0.0357 & 0 & 1 & 250 \\
 2 & 250 & 0.0714 & 0.1071 & 1 & 2 & 375 \\
 3 & 375 & 0.1071 & 0.2143 & 2 & 3 & 500 \\
 4 & 500 & 0.1429 & 0.3571 & 3 & 4 & 625 \\
 5 & 625 & 0.1786 & 0.5357 & 4 & 5 & 750 \\
 6 & 750 & 0.2143 & 0.75 & 5 & 6 & 0 \\
 7 & 875 & 0.25 & 1 & 7 & 7 & 875 \\
\end{array}
\]

sum 3500 1

sum 3500
(e) Use the same equations as in (b) for the mean and standard deviation:

\[ s \quad H(s) \quad p(s) \quad s \quad p(s) \quad p(s)*(s-mean)^2 \]

<table>
<thead>
<tr>
<th>s</th>
<th>H(s)</th>
<th>p(s)</th>
<th>s</th>
<th>p(s)</th>
<th>p(s)*(s-mean)^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>125</td>
<td>0.0357</td>
<td>0</td>
<td>0</td>
<td>0.64509</td>
</tr>
<tr>
<td>1</td>
<td>250</td>
<td>0.0714</td>
<td>0</td>
<td>0.0714</td>
<td>0.75446</td>
</tr>
<tr>
<td>2</td>
<td>375</td>
<td>0.1071</td>
<td>0</td>
<td>0.2143</td>
<td>0.54241</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>0.1429</td>
<td>0</td>
<td>0.4286</td>
<td>0.22321</td>
</tr>
<tr>
<td>4</td>
<td>625</td>
<td>0.1786</td>
<td>0</td>
<td>0.7143</td>
<td>0.01116</td>
</tr>
<tr>
<td>5</td>
<td>750</td>
<td>0.2143</td>
<td>0</td>
<td>1.0714</td>
<td>0.12054</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>875</td>
<td>0.25</td>
<td>0</td>
<td>1.75</td>
<td>1.89063</td>
</tr>
</tbody>
</table>

sum 3500 1
mean 4.25
std dev 2.04634

The mean is \( \mu = \sum_{s=0}^{7} s \cdot p_s(s) = 4.25. \)

The standard deviation is \( \sigma = \left( \sum_{s=0}^{7} (s-\mu)^2 \right)^{\frac{1}{2}} p_s(s) = 2.046. \)

The mean of the enhanced image is closer to the middle of the gray level range, and the standard deviation is larger, as we would expect.

2. (25 pts) From probability theory, if you add two independent random variables \( Z=X+Y \), the resulting probability density of \( Z \) is just the convolution of the probability densities of \( X \) and \( Y \); i.e., \( p_Z = p_X * p_Y \). Since a histogram is an estimate of the probability density of the image values, this means that if we add two images point by point, we can estimate the histogram of the resulting image. (For this to be true, the probabilities of the pixels in the two images must be independent of each other.)

Let the histograms of images \( X \) and \( Y \) be given by the following table. Estimate the histogram of the image that is the point-by-point sum of \( X \) and \( Y \). You should be able to do this by hand instead of using Matlab’s “conv” function.

<table>
<thead>
<tr>
<th>Pixel Value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histogram of X</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Histogram of Y</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Solution:
We convert the histograms to probability densities, by dividing by the total number of pixels:

<table>
<thead>
<tr>
<th>Value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>pX</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pY</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Convolving gives

<table>
<thead>
<tr>
<th>Value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>pZ</td>
<td>0.02</td>
<td>0.06</td>
<td>0.12</td>
<td>0.2</td>
<td>0.2</td>
<td>0.18</td>
<td>0.14</td>
<td>0.08</td>
</tr>
</tbody>
</table>

Converting back to histogram counts:

<table>
<thead>
<tr>
<th>Value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>2</td>
<td>6</td>
<td>12</td>
<td>20</td>
<td>20</td>
<td>18</td>
<td>14</td>
<td>8</td>
</tr>
</tbody>
</table>

3. (25 pts) A two dimensional correlation with a separable filter \( w(x,y) \) can be computed by (1) computing a 1D correlation with \( w_y(y) \) along the individual columns of the input image, followed by (2) computing a 1D correlation with \( w_x(x) \) along the rows of the result from (1). Demonstrate this in Matlab using a 2D Gaussian filter on an actual image, and show that the results from the two approaches are identical.

Solution:

```matlab
clear all
close all

f = double(imread('cameraman.tif'));
	sigma = 10;
h = fspecial('gaussian', [6*sigma 6*sigma], sigma);
g2dfull = imfilter(f,h);
imshow(g2dfull, []);

hx = fspecial('gaussian', [1 6*sigma], sigma);
g1d = imfilter(f,hx);
g2d = imfilter(g1d,hx');
figure, imshow(g2d, []);
dg = abs(g2dfull - g2d);     % difference between the results
fprintf('Maximum difference between the two images = %f
', max(dg(:)));
```

On this image, using this size filter, I get:
Maximum difference between the two images = 0.000000

You can also time the two, for a comparison using “tic” and “toc”. Kevin Carper found that for a 31x31 filter, the 2D filter took 0.741752 seconds while the 1D filters took 0.0611719 seconds.
4. (25 pts) Using the method of normalized cross-correlation, find all instances of the letter “a” in the image “textsample.tif”. To do this, first extract a template subimage $w(x,y)$ of an example of the letter “a” (you can use Matlab’s “imcrop” function). Then match this template to the image (you can use Matlab’s “normxcorr2” function). Threshold the correlation scores (you will have to experiment with the threshold) so that you get 1’s where there are “a’s” and nowhere else. Now, you may get a small cluster of 1’s where there is an “a” instead of a single 1. To avoid multiple detections, you can use Matlab’s “imregionalmax” function\(^1\), to get a single 1 for each “a”.

Take the locations found and draw a box (or some type of marker) overlay on the original image showing the locations of the “a”s. How many “a”s did you detect? Turn in the program, and images of the template subimage and the correlation score image.

Solution: We look at the original image and choose one of the letters of the type we are interested in and find its coordinates via “impixelinfo”. We then crop out this instance and use it as a template for cross-correlation. Here is the cropped image of the letter I used:

![a](image)

Here is the image of the cross-correlation scores:

![Cross-correlation scores](image)

The normalized cross correlation scores range from -1.0 to +1.0. We (by trial and error) choose a threshold to signal the presence of the letter of interest. I choose 0.75, which resulted in the system identifying all the “a”s correctly, and no other letters. Below is the original image with cross hairs drawn at the positions of the “a”s.

\(^1\) Those students who are already familiar with connected component labeling can use “bwlabel” instead.
I found 53 “a”s. Here is the complete Matlab code I used:

```matlab
clear all
close all

I=imread('textsample.tif');
figure(1), imshow(I);%

% Select an instance of the letter "a". To find the coordinates, choose a
% representative letter and use impixelinfo to find its coordinates.
rect = [ 619, 316, 14, 19 ];  % starting col, starting row, width, height
A=imcrop(I,rect);
figure, imshow(A);%

% Use Matlab's cross correlation function (you can also do this yourself in
% the Fourier domain by multiplying the transforms together)
C = normxcorr2(A,I);

% The scores are in an image that is slightly bigger than the original
% image ... it is expanded by half the size of the template in all
% directions. So we will crop out the center portion.
Csub = imcrop(C, [(size(A,2)-1)/2+1 (size(A,1)-1)/2+1 size(I,2)-1 size(I,1)-1]);
figure, imshow(Csub, []), impixelinfo;

% Choose a fixed threshold (scores range from -1.0 to +1.0)
thresh = 0.75;
BW = im2bw(Csub, thresh);
figure, imshow(BW);
```

Krueger: Psychophysical law

 noticeable, was no longer constant (i.e., c) but increased
(i.e., cS) with the base level. S. Brentano’s function
remains dependent on the Weber fraction, k, however.
For Brentano, just as for Fechner (equation 3), the larger
the value of k (the poorer the discriminability or resolving
power on the particular modality), the slower the rise in
subjective magnitude, S, with physical magnitude, I.
Since it can readily be rewritten in logarithmic or relative
units, as in equation 4, the power function can express
ratio invariances even if it seems to do so less simply than
does Fechner’s logarithmic function: therefore, it, too, is
consistent with the perceptual constancies (Yilmaz 1967).
It may seem odd and pointless for the system to add
expansivity and thus to undo part or all of the com-
pressiveness inherent in Weber’s law, but by doing so, it
does not lose access to the ratio invariances, and possibly
transforms them into a more usable form.
% single detection for each letter, we can either use Matlab's
% "imregionalmax" function, or the method of connected components.

%%%%%%%%%%%%%%%%%%%
% Method of "imregionalmax".
RM = imregionalmax(Csub);   % Get points where this is a local maxima
B = RM & BW;               % Keep points that are local max, and > thresh
fprintf('The number of "a"s found is %d\n', sum(B(:)));

% Draw an asterisk at each letter in the original image
figure, imshow(I,[]);
[rows,cols] = find(B);
hold on
plot(cols, rows, 'r*');
hold off
%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%
% Method of connected components
[L,n] = bwlabel(BW);
stats = regionprops(L, 'centroid');
fprintf('The number of "a"s found is %d\n', n);

% Draw an asterisk at each letter in the original image
figure, imshow(I,[]);
centroids = cat(1, stats.Centroid);
hold on
plot(centroids(:,1), centroids(:,2), 'r*');
hold off
%%%%%%%%%%%%%%%%%%%

Note – you can generate an image showing only the a’s, by masking off everything except the
detected points (from Nik Cimino). It’s actually a quick way to make sure you didn’t detect any
non-a’s.

% Create a mask to only show the matches.
mask = uint8( zeros( size( I ) ) );
for i = 1:n
    x0 = round( stats(i).Centroid(1) - size(A,2)/2 );
    y0 = round( stats(i).Centroid(2) - size(A,1)/2 );
    mask(y0:y0+size(A,1)-1, x0:x0+size(A,2)-1) = 255;
end
% Show image with only the matches
Iselected = mask .* ( 255 - I );
figure, imshow( 255 - Iselected );