Homework Assignment 6 - SOLUTIONS
Due Monday, November 23, 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) Use Matlab’s random number generator to generate a 4x4 image of numbers ranging from 1..16. Use the Matlab commands below to generate the matrix. Construct an approximation pyramid for this image, using 2x2 block neighborhood averaging (don’t round off the results to integer). Then find the corresponding residual pyramid, assuming that the interpolation filter implements pixel replication.

```matlab
rng('default'); % reset random number generator
I0 = randi(16, 4); % create 4x4 matrix, 1..16
```

Solution:
The original “image” (i.e., level 0) is

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>11</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>9</td>
<td>16</td>
<td>3</td>
</tr>
</tbody>
</table>

A mean approximation pyramid is created by forming 2x2 block averages and subsampling. Here is the complete pyramid:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>11</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>9</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.5</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.0</td>
<td>8.75</td>
</tr>
</tbody>
</table>

To calculate the prediction residual at a given level upsample (by replicating) the next level approximation and subtract it from the original at this level. The prediction residual at level 0 is

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>11</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>16</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>9</td>
<td>16</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.5</td>
<td>14.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.0</td>
<td>8.75</td>
<td>10.3125</td>
</tr>
</tbody>
</table>

The prediction residual at level 1 is

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10.5</td>
<td>14.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.0</td>
<td>8.75</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- [10.3125 10.3125] - [10.3125 10.3125] = [0.1875 3.6875]

The complete prediction residual pyramid is

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>0.5</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>4.5</td>
<td>-8.5</td>
<td>2.0</td>
<td>-6.0</td>
</tr>
<tr>
<td>-5.0</td>
<td>-3.0</td>
<td>-5.75</td>
<td>4.25</td>
</tr>
<tr>
<td>7.0</td>
<td>1.0</td>
<td>7.25</td>
<td>-5.75</td>
</tr>
</tbody>
</table>
Matlab code:

```matlab
clear all
close all

rng('default');     % reset random number generator
I0 = randi([16,4]);   % create 4x4 matrix, 1..16

% Create level 1: average 2x2 blocks of level 0 and subsample
I1 = imfilter(I0,ones(2,2)/4);
I1 = I1(1:2:end, 1:2:end)

% Create level 2: average 2x2 blocks of level 1 and subsample
I2 = imfilter(I1,ones(2,2)/4);
I2 = I2(1:2:end, 1:2:end)

% Create prediction at level 1 by replicating 2x2 blocks of level 2
I1p = imresize(I2,2,'nearest')

% Residual at level 1
R1 = I1 - I1p

% Create prediction at level 0 by replicating 2x2 blocks of level 1
I0p = imresize(I1,2,'nearest')

% Residual at level 0
R0 = I0 - I0p
```

2. (25 pts) Construct a Laplacian pyramid for the “cameraman” image, as done in the lecture slides. Find the mean and standard deviation of the image at the lowest level of the Laplacian pyramid (ie., highest resolution image). Compare that with the mean and standard deviation of the original gray level image.

Solution:
I first run the code on the slides to generate the Laplacian pyramid. Here is the code:

```matlab
clear all
close all

G0 = double(imread('cameraman.tif'));   % read image
G1 = imresize(G0, 0.5); % apply lowpass, then scale by 0.5
G2 = imresize(G1, 0.5); % apply lowpass, then scale by 0.5
G3 = imresize(G2, 0.5); % apply lowpass, then scale by 0.5

figure, imshow(G0, []), title('G0');
```
CSCI 510/EENG 510 Image and Multidimensional Signal Processing Fall 2015

```matlab
figure, imshow(G1, []), title('G1');
figure, imshow(G2, []), title('G2');
figure, imshow(G3, []), title('G3');

G31 = imresize(G3, 2, 'nearest'); % scale up by 2
G21 = imresize(G2, 2, 'nearest'); % scale up by 2
G11 = imresize(G1, 2, 'nearest'); % scale up by 2

L0 = G0 - G11; % residual image
L1 = G1 - G21; % residual image
L2 = G2 - G31; % residual image

figure, imshow(L0, []), title('L0');
figure, imshow(L1, []), title('L1');
figure, imshow(L2, []), title('L2');
```

And the pyramid:

```
L0
```

```
L1
```

```
L2
```

```
G3
```

To find the means and standard deviations:

```matlab
% Find means and standard deviations
fprintf('Mean of original image G0: %f\n', mean2(G0));
fprintf('Standard deviation of original image: %f\n', std2(G0));
fprintf('Mean of Laplacian image L0: %f\n', mean2(L0));
fprintf('Standard deviation of Laplacian image: %f\n', std2(L0));
```

This yields:

Mean of original image G0: 118.724487
Standard deviation of original image: 62.341715
Mean of Laplacian image L0: -0.001111
Standard deviation of Laplacian image: 13.913708

So the values in the Laplacian image are tightly clustered around zero. We can see that by looking the histograms using:

```matlab
figure, imhist(uint8(G0));
```
3. (25 pts) Consider the simple 4x8, 8-bit image below.

```
21 21 21 95 169 243 243 243
21 21 21 95 169 243 243 243
21 21 21 95 169 243 243 243
21 21 21 95 169 243 243 243
```

a) Compute the entropy of the image.
b) Compress the image using Huffman coding (do this by hand, not using the code from lecture). Compute the compression achieved and compare to the entropy above.
c) Consider a lossless predictor algorithm that predicts the next value \( \hat{f}(x,y) \) as simply the value of the previous pixel to the left \( f(x-1,y) \), and transmits only the error \( e = f(x,y) - \hat{f}(x,y) \). What are the values that are transmitted (note – you also have to transmit the first value on each line)?
d) Compute the entropy of the transmitted values. Now, the entropy of the image that you computed in part (a) is supposed to be the lowest possible value when compressing an image. If the entropy of the transmitted values is lower than that in part (a), explain how this is possible.

Solution:
(a) The entropy of the image is estimated using Eq. (8.1-7) to be
\[
\hat{H} = - \sum_{k=0}^{255} p_r(r_k) \log_2 p_r(r_k)
\]
\[
= - \left[ \frac{12}{32} \log_2 \frac{12}{32} + \frac{4}{32} \log_2 \frac{4}{32} + \frac{4}{32} \log_2 \frac{4}{32} + \frac{12}{32} \log_2 \frac{12}{32} \right]
\]
\[
= -[0.5306 - 0.375 - 0.375 - 0.5306]
\]
\[
= 1.811 \text{ bits/pixel.}
\]

The probabilities used in the computation are given in Table P8.9-1.

<table>
<thead>
<tr>
<th>Intensity</th>
<th>Count</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>12</td>
<td>3/8</td>
</tr>
<tr>
<td>95</td>
<td>4</td>
<td>1/8</td>
</tr>
<tr>
<td>169</td>
<td>4</td>
<td>1/8</td>
</tr>
<tr>
<td>243</td>
<td>12</td>
<td>3/8</td>
</tr>
</tbody>
</table>

(b) Figure P8.9 shows one possible Huffman source reduction and code assignment. Use the procedures described in Section 8.2.1. The intensities are first arranged in order of probability from the top to the bottom (at the left of the source reduction diagram). The least probable symbols are them combined to create a reduced source and the process is repeated from left to right in the diagram. Code words are then assigned to the reduced source symbols from right to left. The codes assigned to each intensity value are read from the left side of the code assignment diagram.

The resulting code is
21 1
95 010
169 011
243 00
(c) Using Eq. (8.1-4), the average number of bits required to represent each pixel in the Huffman coded image (ignoring the storage of the code itself) is

\[
L_{avg} = 1 \left( \frac{3}{8} \right) + 2 \left( \frac{3}{8} \right) + 3 \left( \frac{1}{8} \right) + 3 \left( \frac{1}{8} \right) = \frac{15}{8} = 1.875 \text{ bits/pixel.}
\]

Thus, the compression achieved is

\[
C = \frac{8}{1.875} = 4.27.
\]

Because the theoretical compression resulting from the elimination of all coding redundancy is \( \frac{8}{1.811} = 4.417 \), the Huffman coded image achieves \( \frac{4.27}{4.417} \times 100 \) or 96.67% of the maximum compression possible through the removal of coding redundancy alone.

(d) The transmitted values are

\[
\begin{array}{cccccccc}
21 & 0 & 0 & 74 & 74 & 74 & 0 & 0 \\
21 & 0 & 0 & 74 & 74 & 74 & 0 & 0 \\
21 & 0 & 0 & 74 & 74 & 74 & 0 & 0 \\
21 & 0 & 0 & 74 & 74 & 74 & 0 & 0 \\
\end{array}
\]

The probabilities of its various elements are given in Table 8.9-3.

<table>
<thead>
<tr>
<th>Intensity difference</th>
<th>Count</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>4</td>
<td>1/8</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>1/2</td>
</tr>
<tr>
<td>74</td>
<td>12</td>
<td>3/8</td>
</tr>
</tbody>
</table>

The entropy of the difference image is estimated using Eq. (8.1-7) to be

\[
\tilde{H} = - \left[ \frac{1}{8} \log_2 \frac{1}{8} + \frac{1}{2} \log_2 \frac{1}{2} + \frac{3}{8} \log_2 \frac{3}{8} \right] = 1.41 \text{ bits/pixel.}
\]

This is less than the value found in part (a), which was 1.811 bits per pixel. This is because the entropy in (a) assumes that the pixels are statistically independent. Whereas the predictive coder takes advantage of spatial redundancy.
4. (25 pts) Given a four-symbol source \{a,b,c,d\} with source probabilities \{0.1, 0.4, 0.3, 0.2\}, arithmetically encode the sequence \textit{bbadc}.

Solution:

Any value in the interval \([0.1544, 0.15536)\) at the right side of the figure can be used to code the sequence. For example, the value 0.155.