http://people.sc.fsu.edu/∼jburkardt/presentations/asa_2011_images_homework1.pdf

Homework #10

Algorithms for Science Applications II
Assigned: Friday, 1 April 2011
Due: Friday, 8 April 2011

A picture contains information. But a picture is really just pixels, and removing any single pixel seems to have no effect on the picture. Perhaps the information is in larger patterns made by the pixels. In that case, it might be possible to compress a picture by finding those patterns. We will see a simple example of image compression using the *Haar transform*, that suggests some of the ideas behind more sophisticated compression methods such as the discrete cosine transform.

The files **dewey.jpg**, **imshow_numeric.m**, and **nile.txt** are on the class Blackboard site, and at http://people.sc.fsu.edu/∼jburkardt/latex/asa_2011_images_homework.html.

Problem 1: *Compressing a 1D "Image"*

Suppose we have a vector of data **A(1:N)**; for convenience, we will assume **N** is a power of 2. We wish to replace A by a new vector B, of the same length, which has two parts. The "smooth" or **S** part contains averages of neighboring pairs, while the "detail" or **D** part contains corresponding average differences.

Suppose we start with the vector **A**=[1,2,3,4,5,6,7,8]. Then our smooth vector **S** will be [1.5,3.5,5.5,7.5] and **D** will be [-0.5,-0.5,-0.5,-0.5]. **S** represents our compressed data, and **D** represents information we need only if we want to recover the full data set. To compress further, we simply apply the operation again, but this time just to **S**. Each step of the transform creates a new **S** vector half the size of the previous one. Until the compression becomes too severe, the information in the original data vector seems to get "concentrated" in the **S** vector.

Copy the data file file *nile.txt*. It consists of 570 measurements of the height of the Nile river at flood stage. Set **Y0** to the first 512 entries of this array. In order to make it possible to do comparative plotting, also set up a vector **X0**, to be the integers 1:512.

As long as the data has an even number of entries, the Haar smoothing operation can be written:

```
s = ( ( y(1:2:length(y)) + y(2:2:length(y)) ) / 2;
```

Apply the smoothing operation to both the **X0** and **Y0** vectors to get new data **X1** and **Y1**, each of length 256. Apply the smoothing operation two more times to each vector, to get **X3** and **Y3**, each of length 64.

Use the following plot commands to display your original and smoothed data:

```
plot ( x0, y0, 'b-' )
hold on
plot ( x3, y3, 'r-*' )
```

The smoothed data $(x3, y3)$ does not represent any actual measurements of Nile flood data, so we would like to know whether our compression operation has retained the important information from the original data. Make a short statement in which you compare the graphs of the original and smoothed data; what happens to the spikes (maxima and minima) in the original data? What happens to the general trend of rising and falling?

Turn in:

- your plot of $(x0, y0)$ versus $(x3, y3)$;
- your short statement discussing the plot.

Problem 2: *Compressing a 2D Image*

If we have a 2D image (in particular, a gray scale picture), then it is easy to see how to extend the idea of the Haar transform from a vector to an array. To smooth an array, we smoothing the columns, then the rows. (For simplicity, we will ignore the computation of the "detail" or "difference" component of the transform, which allows us to recover the original image.)

For example, suppose our initial array 4x4 array **A** is:

```
1 3 4 8
2 8 6 4
3 2 5 3
4 5 6 2
```

then the column smoothing results in a matrix 2x4 matrix "C":

```
1.5  5.5  5.0  6.0
3.5  3.5  5.5  2.5
```

and then the row smoothing gives us the final smoothed 2x2 matrix "R":

```
3.5  5.5
3.5  4.0
```

And if our smoothed array has an even number of rows and columns, we can smooth it again.

Copy the grayscale image file *dewey.jpg* into an array called **A**.

To carry out our numeric operations, we need to work with arrays of type **double**. Make a numeric copy of the data using the command **A0=double(A);**. In order to display a numeric array as an image, I have made a command for you called **imshow_numeric()**. Make sure that you can use this command to display the data in **A0**. You might not recognize the person in the image, but you can read the headline on the newspaper. Let us consider the readable headline the *information*.

The column+row Haar smoothing operation for an array **A** can be described as follows:

```
C = ( A(1:2:size(A,1),:) + A(2:2:size(A,1),:) ) / 2;
R = ( C(:,1:2:size(C,2)) + C(:,2:2:size(C,2)) ) / 2;
```

Apply the column+row Haar smoothing to **A0** and call the resulting array **A1**. Use the **imshow_numeric()** program to display the compressed image, and verify that you can still read the headline, even though there is 1/4 of the original data.

Theoretically, you can compress this image 9 times (down to 1x2 pixels!). How many times can you actually carry out the column+row Haar smoothing while still being able to read the headline? You are allowed to use MATLAB's zoom feature to try to magnify the image, which will be shrinking with each compression.

Turn in:

- your plot of the image after **two steps** of Haar compression, that is, the image that is 128x256 pixels in size;
- your report on how many times you were able to shrink the image before the newspaper became unreadable.